All About IEEE Std 1471

Rich Hilliard
r.hilliard@computer.org
June 2007

Some of these materials derive from courses given with David Emery and Mark Maier since 1998.

Outline

- ***Historical preamble***:
  - **What we "knew" about Architecture in 1998**

Architecture in 1998
is in a separate file

- **IEEE 1471**
- **A Brief History**
- **Goals and Motivations**
- **Concepts**
- **Usage**
- **Future**

- **Some Topics and Applications**
  - **Architecture Frameworks**
  - **Processes of Architecting**
  - **Viewpoint modeling**

Viewpoint Modeling
is in a separate file

IEEE 1471: What Is It?

- IEEE Standard 1471 *Recommended Practice for Architectural Description of Software-Intensive Systems*
  - Developed by the IEEE Computer Society
- IEEE 1471 is a *recommended practice*
  - A "recommended practice" is one kind of IEEE standard
  - A using organization must decide whether to, and how to, employ IEEE 1471
- IEEE 1471 applies to Architectural Descriptions
  - Architectural Descriptions can conform to the standard
  - systems, projects, processes or organizations cannot
  - Think of it as a standard about "blueprints" not about "buildings"
- ISO adopted it as ISO/IEC 42010 in July 2007

A Brief History

- IEEE Architecture Planning Group:
  - First met August 1995, in Montréal
  - Final report to IEEE Software Engineering Standards Committee, April 1996
  - 6 Participants, 80 reviewers
- IEEE Architecture Working Group: May 1996 to 2000
  - Bi-monthly meetings
  - 29 participants, 150 reviewers
- First IEEE Ballot, October 1998
- IEEE 1471 approved for use, September 2000
- Adopted as an ANSI (US) standard, August 2001
- Adopted by ISO through a fast-track ballot, March 2006
  - Published as ISO/IEC 42010, Systems & Software Engineering — Architecture Description
  - Joint revision by ISO and IEEE under ISO/IEC JTC 1/SC7 WG 42

A Brief History: IEEE Architecture Planning Group

Chartered by IEEE *Software Engineering Standards Committee* to:

- **Define direction for incorporating architectural thinking into IEEE standards**
- **Develop framework (terms, concepts and principles) for software systems architectures**
- **Examine IEEE standards for architectural relevance**
- **Produce an Action Plan for future IEEE activities in this area**

Motivation: Why Architecture?

- **Why do some systems "succeed"?**
- **Explicitly "architected" systems seem to turn out "faster, better and cheaper"**
  - **All successful, unprecedented systems have been explicitly architected (Rechtin, 1992 and Maier and Rechtin, 2000)**
- **Architecture is recognized as a critical element in the successful development and evolution of software-intensive systems**

Scope of IEEE 1471

- ***Software-intensive systems*** **are those complex systems where software contributes essential influences to the design, construction, deployment and evolution of the system as a whole**
- **There is a growing body of knowledge in the application of architectural concepts to these systems to attain the benefits of reduced costs and increased quality, such as usability, flexibility, reliability, interoperability and other system qualities**

IEEE Architecture Working Group: Goals and Objectives

- **Take a "wide scope" interpretation of architecture as applicable to software-intensive systems**
- **Establish a conceptual framework and vocabulary for talking about architectural issues of systems**
- **Identify and promulgate sound architectural practices**
- **Allow for the evolution of those practices as relevant technologies mature**

IEEE Architecture Working Group: Work Activities

Initiated *Recommended Practice for Architectural Description* to address:

- **Architectural representation**
- **Role of architecture in life cycle**
- **Identification of key stakeholders**
- **Candidate architectural methods and processes**
- **Techniques for architectural review and analysis**

Organization of IEEE 1471

1 Overview
2 References
3 Definitions
4 Conceptual Framework
5 Architectural Description Practices (normative)
A Bibliography
B On The Definition Of Architecture
C Views And Viewpoints
D Examples Of Viewpoints
E Relationship To Other Standards

Using IEEE 1471

- IEEE 1471 is intended for use in a variety of life cycle contexts, e.g.:
- Architecture of Single Systems
  - Whether applications, systems, products, systems of systems, product lines, product families…
- Iterative Architecture for Evolutionary Systems
- Discovering the Architecture of Existing Systems

Defining "Architecture" before IEEE 1471 IEEE (vintage 1990)

- **What is an "architecture"?**
  Architecture. The organizational structure of a system or component
  - **IEEE *Glossary of Software Engineering Terminology*, 610.12–1990**

- **Nice definition, but nothing in it distinguishes an architecture from a "make file".**

Defining "Architecture" in IEEE 1471

- ***architecture*: the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.**

where:

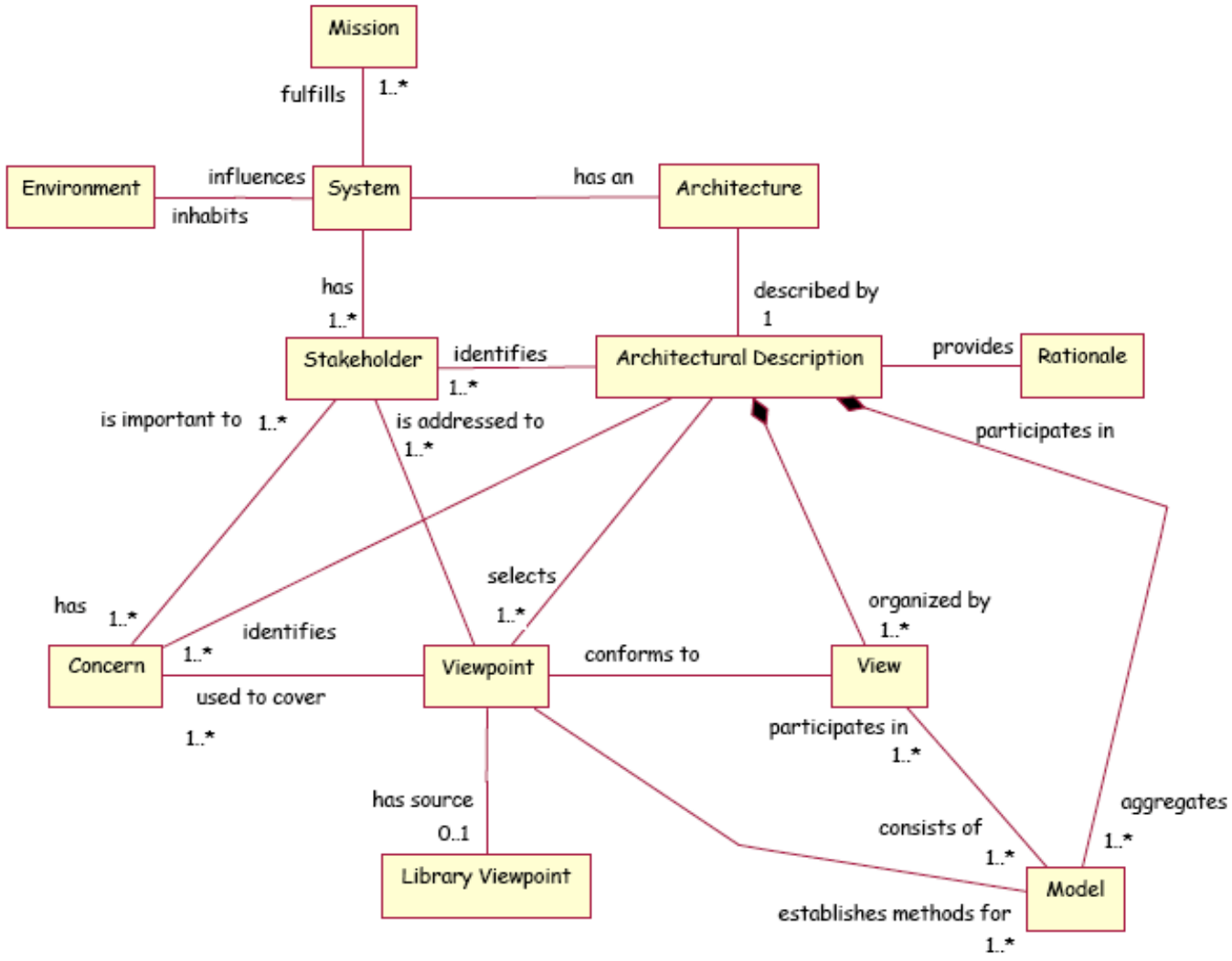
  - **fundamental = essential, unifying concepts and principles**
  - **system = application, system, platform, system-of-systems, enterprise, product line, ...**
  - **environment = developmental, operational, programmatic, … context**

- **Every system has an architecture**

Role of the Conceptual Framework

- **To establish terms and concepts for architectural thinking**
- **To provide a means to talk about Architectural Descriptions within the context of**
  - **System Stakeholders**
  - **Life Cycle**
  - **Uses of Architectural Description**
- **To serve as a basis for evolution of knowledge in a field where little common terminology existed**

The IEEE 1471 Conceptual Framework



The IEE 1471 Conceptual Framework: Architectural Description

- ***architectural description*** **(AD): a collection of products to document an architecture**
- **An AD is addressed to the system's stakeholders to answer their architectural concerns about the system**
- **An AD is organized into one or more views of the system**
- **Each view addresses one or more concerns of the stakeholders**

The IEEE 1471 Conceptual Framework: Stakeholders and Concerns

- ***stakeholder***: **an individual, team, or organization (or collections thereof) with interests in, or concerns relative to, a system**
- ***concerns***: **those stakeholders' interests which pertain to the development, operation, or other key characteristics of the system (e.g., performance, reliability, security, evolvability, distribution, …)**

Some Typical System (Architecture) Stakeholders

- **Client**
- **Acquirer**
- **Owner**
- **User**
- **Operator**
- **Architect**
- **System Engineer**
- **Developer**
- **Designer**
- **Builder**
- **Maintainer**
- **Service Provider**
- **Vendor**
- **Subcontractor**
- **Planner**

The IEEE 1471 Conceptual Framework: Stakeholders and Concerns

- **ADs are interest-relative:**
  - **An AD identifies the system's stakeholders and their concerns**
- **Concerns form the basis for completeness:**
  - **An AD addresses all identified stakeholders' concerns**
    - **If not, it is by definition, incomplete**

Rationale: Stakeholders & Concerns

- **Concept of 'stakeholder' established in the requirements analysis community**
  - **Reflecting the reality that many different people are involved in complex systems, and each person has a different perspective**
  - **Particularly true where client (e.g. acquisition agency) /= end user**
- **Many non-functional requirements/aspects based on specific stakeholders**
  - **Affordability for acquisition, maintainability for maintainers**
  - **Users just want a system that works now**
- **Stakeholder concerns used to establish/justify multiple views**
  - **"Proof by contradiction:" If a view doesn't answer some stakeholder concerns, why bother?**

The IEEE 1471 Conceptual Framework: Architectural Views

- **An AD consists of one or more views**
- ***view*: a representation of a whole system from the perspective of a related set of concerns**
  - **The architectural views are the actual description of the system**
- **Support multiple audiences each with their own concerns**
- **Reduce perceived complexity through separation of concerns**

IEEE 1471 Conceptual Framework: Architectural Views

- **Views are not "orthogonal" but each view generally contains new information**
- ***Views are modular:***
  - **A view may contain one or more architectural models, allowing**

  (1) a view to utilize multiple notations, and

  (2) a model to be shared between multiple views
- ***Consistency between views in an AD:***
  - **An AD documents any known inconsistencies among the views it contains**

views : architectural description :: chapters : book

IEEE 1471 Conceptual Framework: Architectural Viewpoints

- ***Views should be well-formed*:**
  - **Each view corresponds to exactly one viewpoint**
  - **Viewpoints define the resources and rules for constructing views**
- ***Concerns drive the selection of the viewpoints to be used:***
  - **A viewpoint establishes the purposes and audience for a view and the techniques or methods employed in constructing a view**
  - **Each concern is addressed by some architectural view**
- ***Viewpoints are first-class*:**
  - **Each viewpoint used in an AD is "declared" before use**
- ***No fixed set of viewpoints:***
  - **IEEE 1471 is "agnostic" about where viewpoints come from**
  - **Enterprises will evolve a viewpoint "library" or framework**

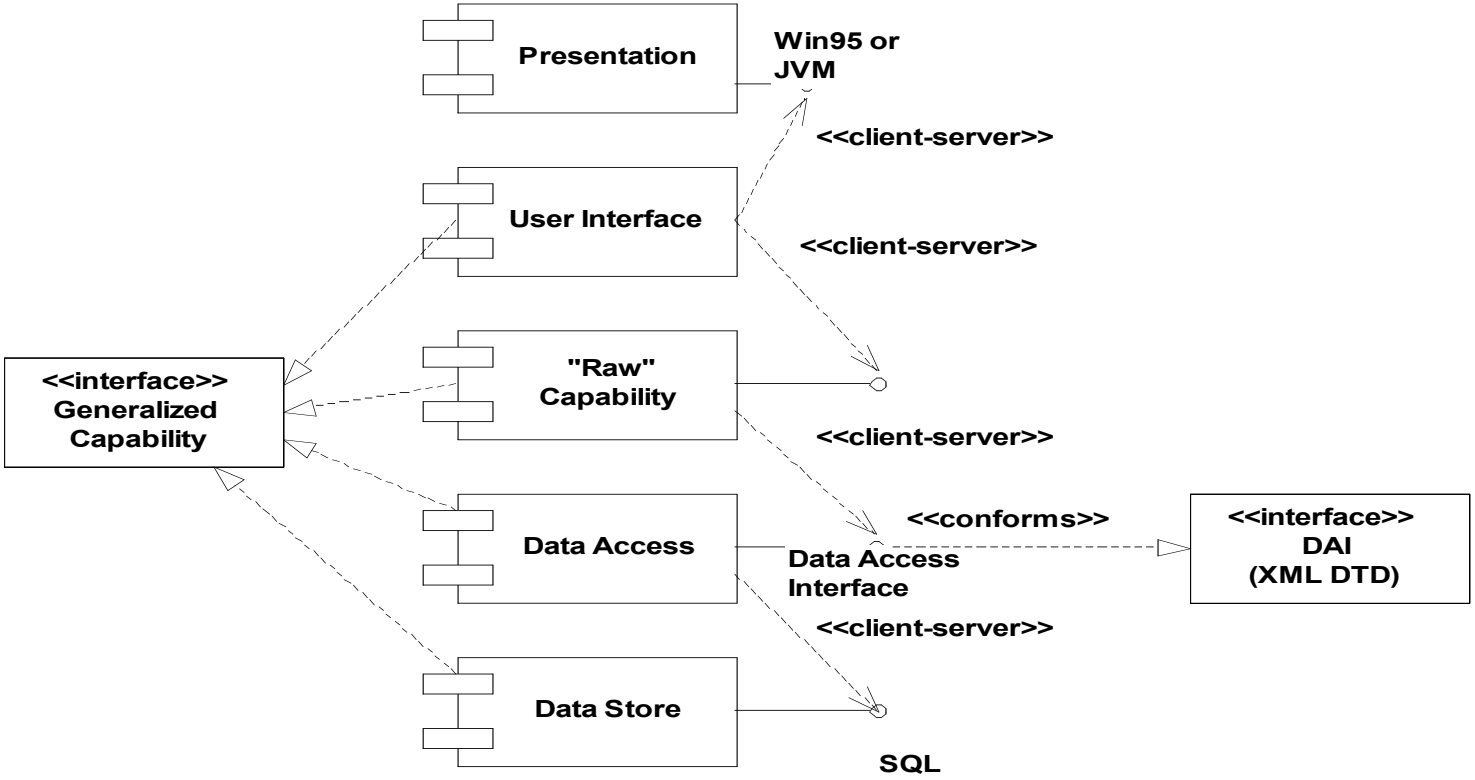
How to declare a Viewpoint

- ***Each architectural viewpoint is determined by*:**
  - **Viewpoint name**
  - **The stakeholders addressed by the viewpoint**
  - **The architectural concerns "framed" by the viewpoint**
  - **The viewpoint language, or modeling techniques, or analytical methods used to construct, depict and analyze the resulting view**
  - **The source, if any, of the viewpoint (e.g., author, literature citation)**
- ***A viewpoint may optionally include*:**
  - **Consistency or completeness checks associated with the underlying method to be applied to models within the view**
  - **Evaluation or analysis techniques to be applied to models within the view**
  - **Heuristics, patterns, or other guidelines which aid in the synthesis of an associated view or its models**

A Viewpoint Example: Capability

- **Its name: Capability**
- **Its Stakeholders:**
  - **producers, developers and integrators**
- **The architectural concerns it frames:**
  - **How is functionality packaged?**
  - **How is it fielded?**
  - **What interfaces are managed?**
- **The viewpoint language it uses:**
  - **Components and their dependencies (UML component diagrams)**
  - **Interfaces and their attributes (UML class diagrams)**
- **Its source: also known as Static, Application, Structural viewpoints**

Example: Capability View



- The Capability View covers all system functionality for operating on data
- Capabilities are fielded using a 5-tier layered organization with interfaces between pairs of layers
  - Each layer is a capability
  - Entire stack is a deployable capability
- Capabilities can serve other capabilities

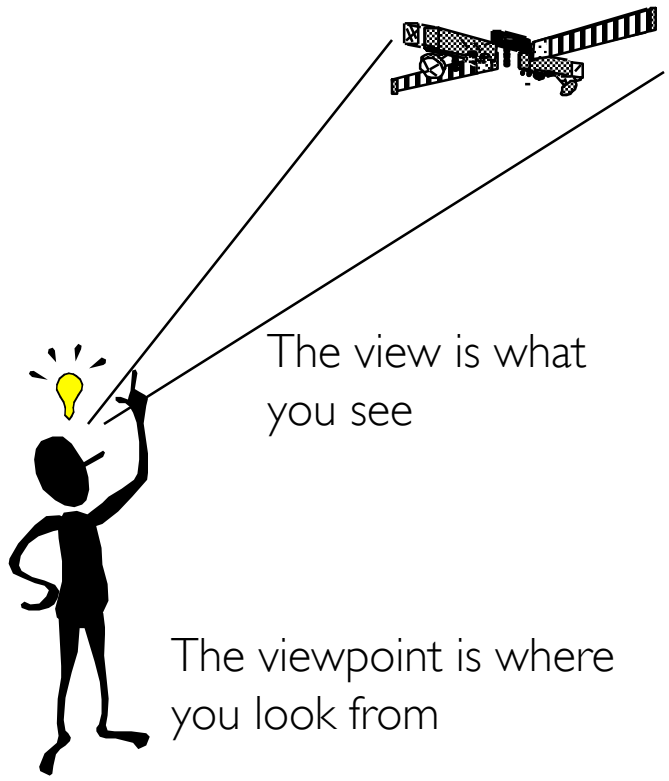
Why separate View and Viewpoint?

- **Originally derived from experience**
  - **Noticed 'patterns' in good architectural descriptions, addressing the same issues on different systems**
    - **E.g. Security, performance, structure, data, etc**
  - **Capturing the 'pattern' made it easier to do the next architecture, by providing a known starting point**
- **Literature survey produced several approaches with well-defined views**
  - **RM-ODP, Zachman, Kruchten's 4+1, etc**
  - **But no separate name for view and viewpoint concepts**
- **Also influenced by programming language design and also the software patterns movement**
  - **Explicitly capturing and declaring the pattern before use considered to be good engineering**

view : viewpoint :: program : programming language

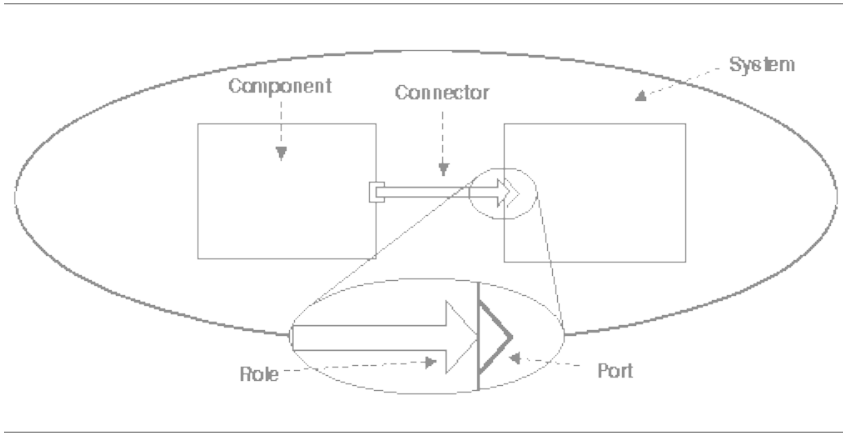
Understanding Views and Viewpoints

- **Definitions:**
  - **A view is a description of the entire system from the perspective of a set of related concerns. A view is composed of one or more models.**
  - **A viewpoint determines the resources and rules for constructing a view**
- **Example:**
  - **A chair and a table have different front views, but the concept of a front view is the same**



Another Example: Components & Connectors Viewpoint

- **Concerns:**
  - **What are the computational elements of a system and their organization?**
  - **What element comprise the system?**
  - **What are their interfaces?**
  - **How do they interconnect?**
  - **What are the mechanisms for interconnection?**
- **Viewpoint language:**
  - **Components, connectors, ports and roles, attributes**
- **Analytic Methods:**
  - **Attachment, type consistency**



Based on: Acme: An Architecture Description Interchange Language, Garlan, Monroe, Wile, Proceedings of CASCON'97, November 1997

Viewpoint Considerations

- IEEE 1471 says "1-to-1 mapping of views and viewpoints in an AD"
  - Some asked "can I have more than 1 view for a given viewpoint?"
- Reason for 1-1 mapping is conceptual consistency
  - A view is supposed to cover the entire system for a given set of stakeholders/concerns
  - Two views for the same viewpoint would cover the same concerns, raising consistency issues between the two views
- Viewpoints can (and should) allow for multiple techniques, languages, etc.
  - Focus on viewpoint is concerns, not representations
- IEEE 1471 envisions "libraries of viewpoints"
  - Architect selects those useful for system at hand

Library (Pre-defined, Reusable) Viewpoints

- **Viewpoints are not system specific, unlike the stakeholders and views**
- **Hence, the architect may be able to reuse viewpoint descriptions between projects**
- **Therefore, viewpoints may be included by reference in an AD**

IEEE 1471 Requirements

- IEEE 1471 is written in terms of “shall”, “should” and “may”
  - To facilitate conformance checking
- An Architectural Description (AD) must contain at least the following:
  - Identification of stakeholders and concerns
  - Selection and declaration of the viewpoints used
  - Architectural views, each conforming to a viewpoint
  - Any known inconsistencies
  - Architectural rationale

IEEE 1471 is agnostic about…

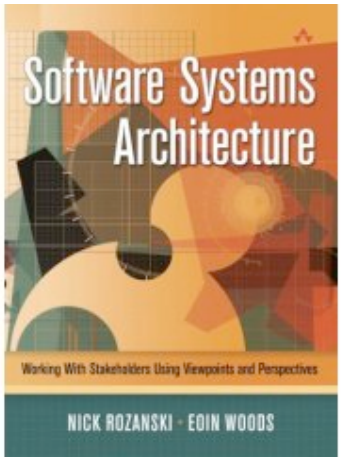
- **The format or media for an architectural description**
- **The notations or architecture description languages used**
- **The processes or methods use to produce (or evaluate) the architectural description**

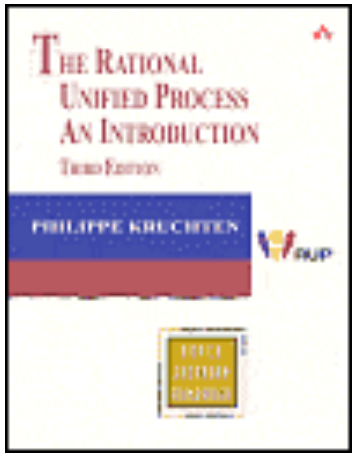
Influences on the Development of IEEE 1471

- **Emerging practices: Philippe Kruchten's 4+1 view model, Siemens', MITRE's architecture methods**
- **Rechtin and Maier's Systems Architecting: multi-disciplinary nature of architecture**
- **Ross' Structured Analysis: making viewpoints first-class**
- **Dijkstra's separation of concerns in software engineering**
- **Barry Boehm: explicit identification of system's stakeholders**

Usage

- **Google hits on +IEEE +1471": 349,000!**
- **Academic work building upon conceptual framework**
- **Books for practitioners**
- **Architecture frameworks**





See: IEEE 1471 bibliography on website

Applications of IEEE 1471 (circa 2001)

- **Architecture of Software-Intensive Systems Curriculum**
- **The Open Group Architecture Framework**
- **Software Architecture Review and Assessment (SARA) Industry Group**
- **Hewlett-Packard**
- **Rational/IBM**
- **Air Force Command and Control System Target Architecture**

Insights (or, things we got right)

- **Nothing unique to software-intensive systems in the standard**
- **The discipline of identifying stakeholders and concerns has value far beyond Architecting**
- **Separating Architecture from Architecture Description**
- **Separating View from Viewpoint**
- **Indirections**
  - **Not prescribing a fixed set of viewpoints**
  - **Stakeholders and concerns**

"All problems in computer science can be solved by another level of indirection"

— Butler Lampson (or David Wheeler)

<Adjective> Architectures

- Application Architectures
- Data Architectures
- Enterprise Architectures
- Logical Architecture
- Makefile Architectures
- Operational Architectures
- Physical Architectures
- Security Architectures
- Systems Architectures
- Technical Architectures
- Occupant Architectures
- Heating and Lighting Architectures
- Building Code Architectures

Future

- Joint revision: ISO Working Group 42 and IEEE
- Expand scope from software-intensive systems to general systems
- Align with ISO life cycle models: 15288 (general systems), 12207 (software)
- Harmonize with other ISO architecture-related standards
- Definitely some “fixes” and cleanup
- Maybe some new stuff!
  - Architecture frameworks
  - Rules for consistency between views

More About IEEE 1471

- **Web site**
- **Users group (see website)**
- **Bibliography (see website)**

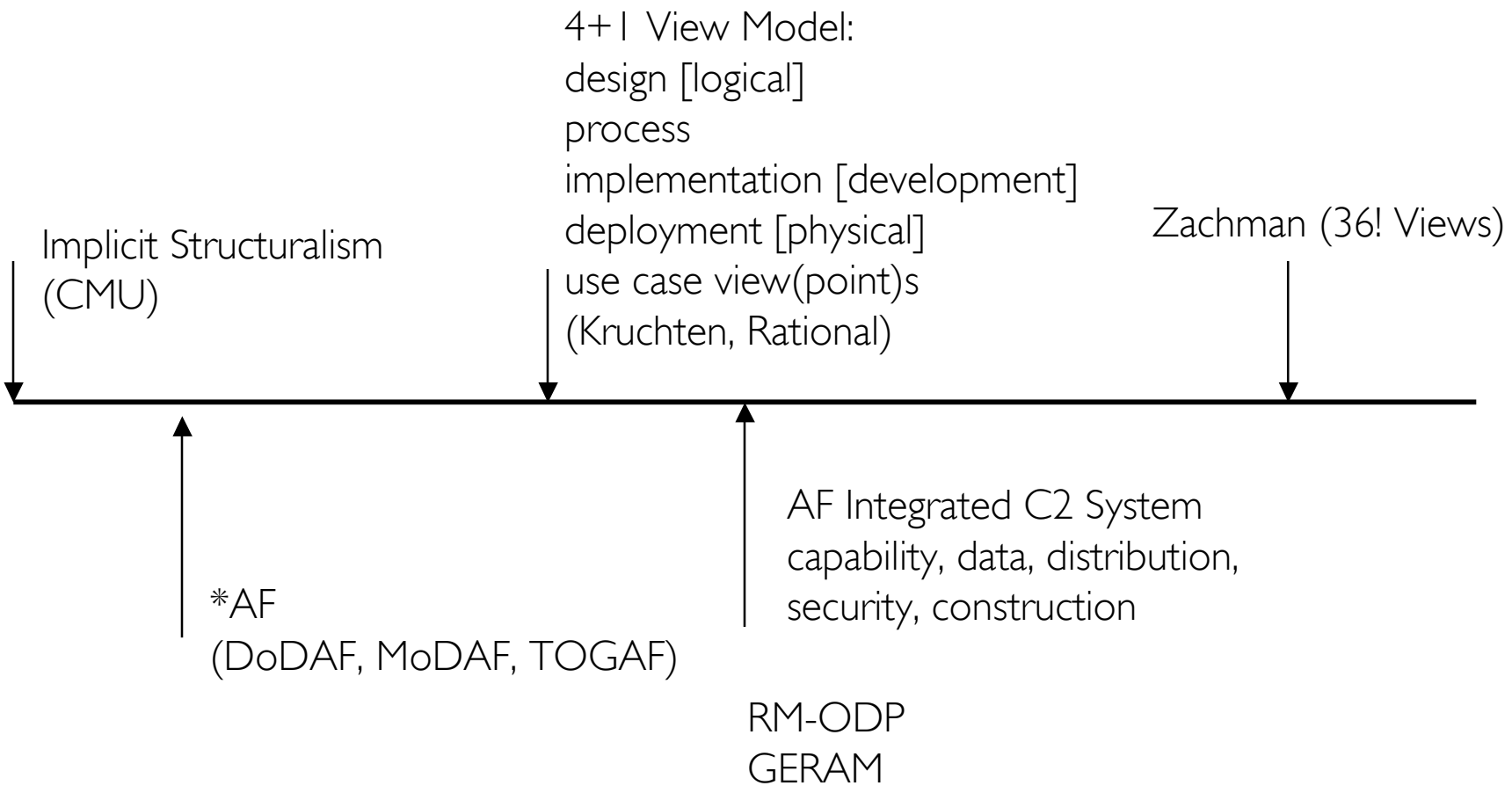
http://www.iso-architecture.org/ieee-1471/

Architecture Frameworks

Example of Multi-View(pointed) Approaches

- **Kruchten's "4+1 Model", Siemens, Rozanski & Woods,…**
- **ISO Reference Model - Open Distributed Computing**
- **ISO GERAM**
- **Zachman Framework**
- **DoD Architecture Framework, MoDAF, …**
- **The Open Group's Architecture Framework**

IEEE 1471 is intended to encompass...



ISO/IEC DIS 10746: RM-ODP

- Reference Model for Open Distributed Processing
- DIS 10746-3 specifies "Architecture", using 5 viewpoints:
  - Enterprise: purpose, scope and polices
  - Information: semantics of information and information processing
  - Computational: functional decomposition into objects and their interfaces
  - Engineering: mechanisms and functions for distributed interaction
  - Technology: choices of technology
- DIS 10746-3 also specifies consistency rules across viewpoints

Using RM-ODP

- **RM-ODP specifies a required set of viewpoints for use**
  - **But other viewpoints may also be necessary to completely describe an architecture**
- **RM-ODP does not address the 'front-end' activities, i.e. Goals, Vision, Needs**
- **Nor does it provide a process for populating the views specified by the required viewpoints**

The US DOD Architecture Framework

- **DODAF " provides the rules, guidance, and product descriptions for developing and presenting architecture descriptions that ensure a common denominator for understanding, comparing and integrating architectures."**
- **"There are three major perspectives, i.e., views, that logically combine to describe an architecture ... the operational, systems and technical views."**
  - **Operational: Functional description of how systems work together**
  - **Systems: Physical components and their relationships**
  - **Technical: Standards and conventions for interoperability and commonality**

Using the DoDAF

- **Framework concentrates on required deliverables, not architectural process**
  - **Deliverables presume an already fleshed-out architecture**
  - **Framework specifies 7 mandatory products and 19 optional products**
- **Thus the Method should be used to produce a complete architecture, with Framework documents as stakeholder requirements**
  - **C4ISR product requirements should be included as part of the viewpoints selected for the architecture**
  - **Then Framework becomes specification of how to present an architecture**

Defining Architecture Framework

- ***architecture framework*: A set of predefined viewpoints, concerns, generic stakeholders and viewpoint correspondence rules, used to capture common practice for architecture descriptions in specific domains or user communities.**
- ***view correspondence:* A connection or mapping between elements of views in an architectural description, used to establish consistency or similar relationships that apply to the architecture being described.**

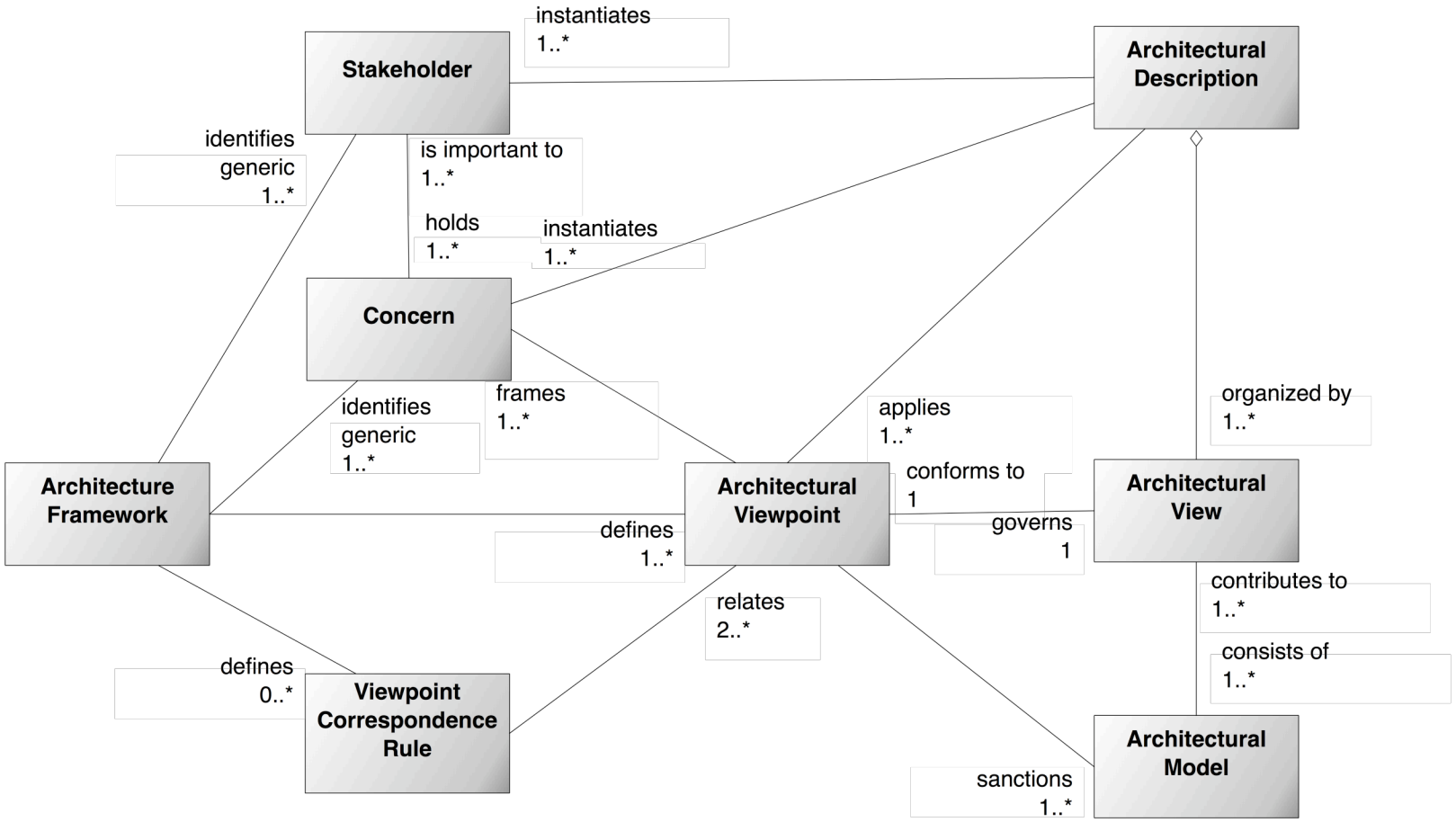
New work by WG 42

Defining Architecture Framework

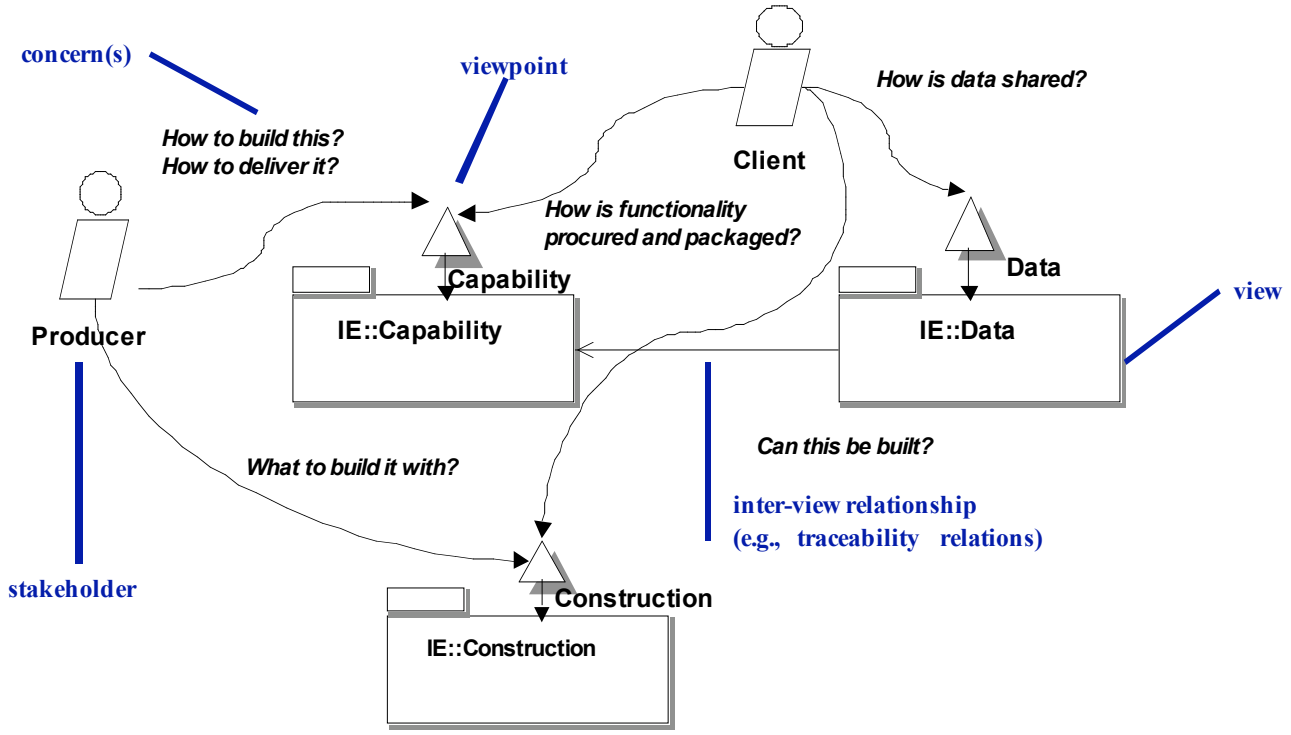
- *view correspondence rule:* A declaration of a mapping between elements identified in multiple viewpoints. A view correspondence rule may be part of an architectural framework, or specified within an individual architectural description.
- New kinds of conformance:
  - An architecture framework to ISO 42010
  - An AD to an architecture framework

New work by WG 42

Architecture Frameworks



Organizing Architecture Frameworks



Processes of Architecting

Why no Process Requirements in IEEE 1471?

- **Focus of IEEE AWG on capturing existing consensus**
  - **Much more consensus on "what" than on "how"**
- **Explicit process for architecture still emerging**
  - **Example: Rational Unified Process to generate "4+1" Architecture Descriptions**
- **Current practice in specifying process also considers "quality" or "effectiveness"**
  - **E.g. SEI CMM 5-level models**
  - **No clear consensus on what constitutes "good" architectures or even "good" architectural descriptions**

An Implicit Process

- How does the Architect do the job?
  - Frame and Understand problem
  - Vision, Goals and Needs: Customer buy-in
  - Identify Stakeholders
  - Select Viewpoints and Model Views
  - Integrate Views
  - Oversee Construction/Production
  - Maintain/Evolve Architecture
    - Bottom up (from construction), outside-in (from environment),
    - Variances, Interpretation, Modification consensus process

A Little Philosophy

Building metaphor

- **Construction/Civil Works**
  - **5000 year history (Imhotep, (2635-2595 B.C.) is first recorded "architect")**
  - **First writings on architecture date to Roman times (Vitruvius (70bc- 20bc), *De Architectura*)**
  - **Distinction between "Architect" and "Civil Engineer" developed with Industrial Revolution**
    - **Based on "Mechanics of Materials" science**
- **"Architect" and "Civil Engineer" now have different training, roles, responsibilities**

"Architecture" is (now) distinct from "Engineering"

- **Architect responsible for the suitability of the building**
  - **Churches should generate a feeling of space, as well as having good acoustics**
  - **Structure must match its intended use and its environment**
    - **Imagine Sydney Opera House in the Outback**
- **Engineer responsible for execution of architecture**
  - **Building must not fall down**
  - **Constructed using appropriate (cost-effective) materials**
- **Engineering assumes architecture**
  - **Not engineer's role to decide what makes a Church a Church…**
- **Software Systems architect responsible for the system in its environment**
  - **Software/Systems Engineers execute the architecture**

The Practice Continuum

Characteristic	Architecting	A & E	Engineering
Situation/Goals	Ill-Structured	Constrained	Understood
	Satisfaction	Compliance	Optimization
Methods	Heuristics	⟷	Equations
	Synthesis	⟷	Analysis
	Art and Science	Art **and** Science	**Science** and Art
Interfaces	Focus on "Mis-Fits"	Critical	Completeness
System Integrity Maintained Through	"Single Mind"	Clear Objectives	Disciplined Methodology and Process
Management Issues	Working for Client	Working with Client	Working for Builder
	Conceptualization and Certification	Whole Waterfall	Meeting Project Requirements
	Confidentiality	Conflict of Interest	Profit versus Cost

Maier and Rechtin, The Art of Systems Architecting, 2000

What's Architecture? The Definition Debate

- Defining “Architecture” was most contentious part of IEEE 1471 ballot
- Several alternative definitions exist
- IEEE 1471 definition reflects a compromise to achieve maximum consensus
  - One alternative was to drop the definition from the standard, since the standard is about “Architectural Descriptions” and not "Architecture" per se
  - But, working consensus was eventually reached

Alternate Definitions for the "A-Word"

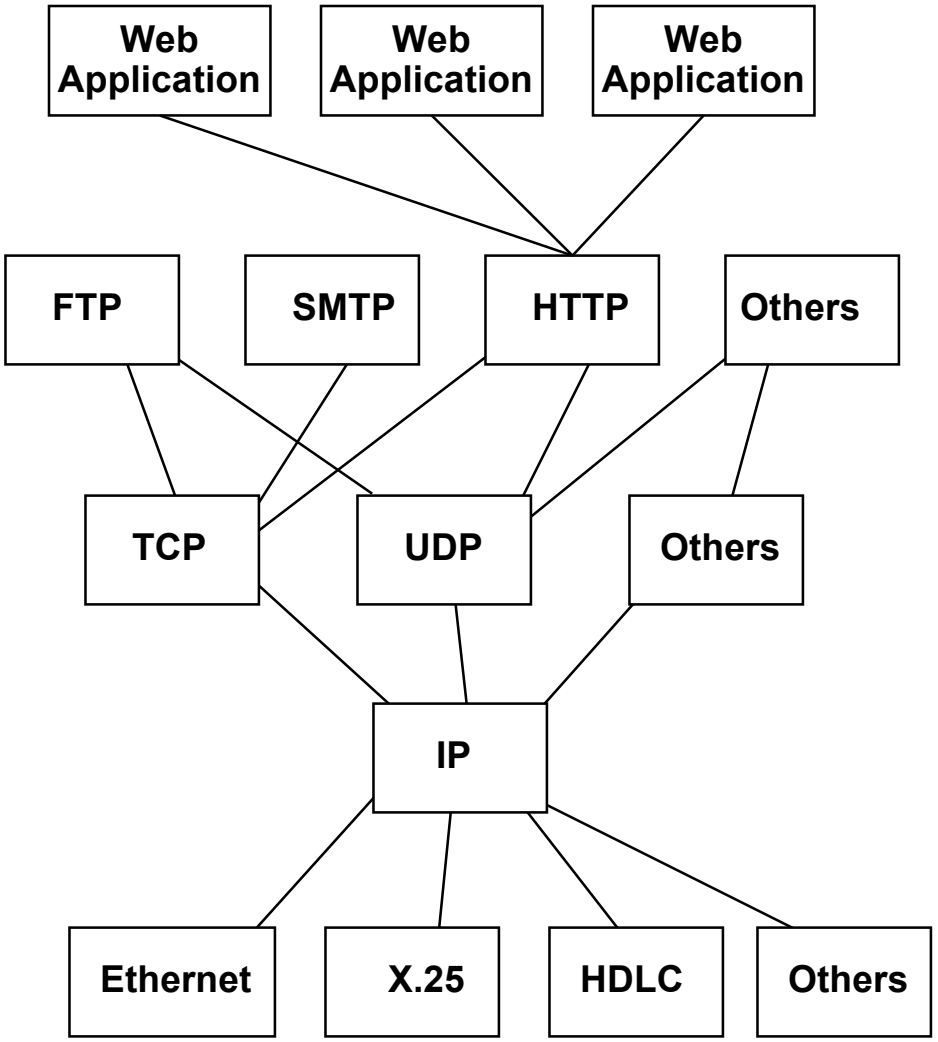
- **INCOSE SAWG**
  - **Systems Architecture: Fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints and behaviors**
- **IEEE 610.12**
  - **Architecture: organizational structure of a system or component. Architecture is the highest-level concept of a system in its environment**
- **Rechtin & Maier**
  - **Systems Architecting is that part of systems engineering most concerned with purpose determination, concept formulation, and certification for use**
- **Perry & Garlan**
  - **The structure of the components of a system, their interrelationships, and principles and guidelines governing their design and evolution over time**

The "ism" debate: Is Architecture more than "structure"?

- **"Structuralists": No. Structure is the key issue for architecture**
  - **Most 'software architecture' approaches focus solely on structure**
- **"Contextualists": Yes. Structure is "design"; fitness for use is the key issue**
  - **Architecture should be different from Top-Level Design**
- **But …Structure of what?**
- **IEEE 1471 comes down on the side of "contextualists"**
  - **Components, … relationships to each other, and to the environment ...**

When Structure isn't Physical

- Here is a diagram of the relationship of protocols on the Internet
- Protocols (not physical things) are the most important "structures" on the Internet
- If the architecture of the Internet isn't protocols, what is it?
  - Clearly the current physical organization of the Internet is not very organizing



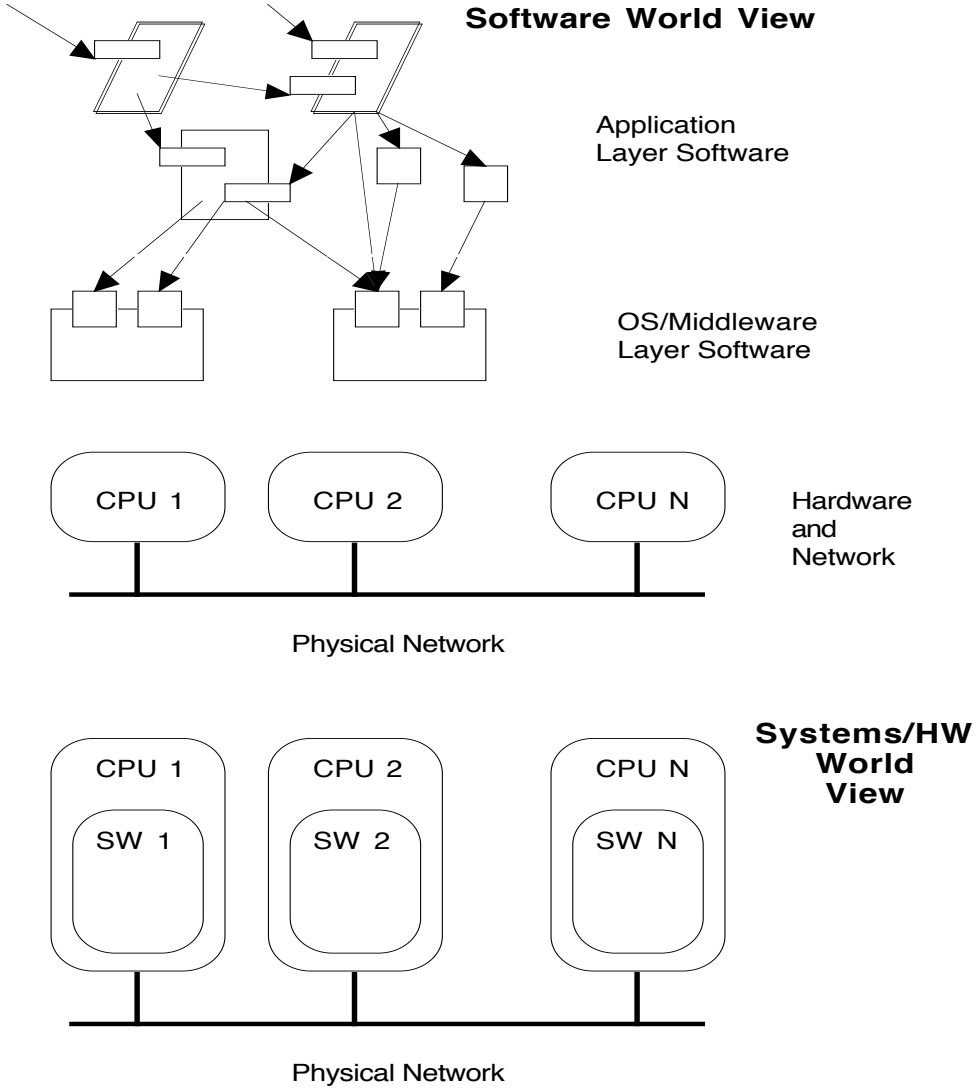
Is Structure sufficient?

- IEEE 1471 authors argued “no”
- Many aspects of a system are not “structural”
  - “ilities”: reliability, maintainability, flexibility, etc.
  - Security
- Need to ensure system “fit for intended use”
  - Many systems meet all stated requirements and are not usable
  - From “building metaphor”, Architect ensures fitness for use, engineer assumes Architecture
    - A gothic cathedral is much more than flying buttresses…
- Architecture as trade-off space for requirements
  - Architecture must be able to represent all key requirements
- Some in Software Architecture community starting to look outside of structure...

Structure, Concerns, and Multiple Views

- Suppose a customer wants an information systems to perform some specified functions, with specified performance, and needs its security to be certified by an external group
- What information does an architect need to provide?
  - Clearly functional and physical descriptions are called for
  - Can performance be derived from the previous descriptions, with no additional information?
  - Can security certifiability be determined, with no additional information?
  - In general, the answer to the last two is NO
- We need descriptions in terms meaningful to actual concerns, not everything is derivable from structure (at least in practice)

Isn't One View Enough?



- **To a hardware-systems engineering software may appear to be encapsulated in hardware**
- **But in a distributed software system, it looks the reverse from the software perspective**
- **What are the costs or drawbacks of each perspective?**

Lesson: One view isn't enough, the single hierarchy of components doesn't describe the real world.