

INTERNATIONAL  
STANDARD

ISO/IEC/  
IEEE  
42010

First edition  
2011-12-01

---

Systems and software engineering —  
Architecture description

*Ingénierie des systèmes et des logiciels — Description de l'architecture*



Reference number  
ISO/IEC/IEEE 42010:2011(E)

© ISO/IEC 2011  
© IEEE 2011

# *All About ISO/IEC/IEEE 42010 version 2014-r5*

**Rich Hilliard**  
**r.hilliard@computer.org**  
**richh@mit.edu**  
**18 April 2014**

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

## *Acknowledgements:*

**Some materials based on previous presentations by  
Johan Bendz, David Emery, Rich Hilliard and Mark Maier**

## *Comments, suggestions?*

**This is a work in progress, the first major update since *All about IEEE Std 1471* in 2007.**

**Please send comments, corrections, suggestions to  
[richh@mit.edu](mailto:richh@mit.edu)**

**When commenting, please refer to the version id on front  
page**

## ***Executive Summary***

- **This is a presentation about the international standard known as**
- **ISO/IEC/IEEE 42010:2011, *Systems and software engineering—Architecture description***
- **including its history, core concepts, use and application**

# Overview

- **Introduction**
- **History, IEEE 1471**
- **Current standard**
- **Conceptual foundations**
- **Conformance and requirements**
- **Architecture frameworks and architecture description languages**
- **Future**
- **Resources**
- **Additional topics and examples**

# Introduction

- What is ISO/IEC/IEEE 42010?
- An international standard, entitled  
*Systems and software engineering—Architecture description*
- Published December 2011
- Based on IEEE Std 1471:2000
- Specifies *best practices* for documenting enterprise, system and software architectures



## *Introduction: Key ideas*

- An *architecture description* expresses an architecture of a system of interest
- An architecture description contains of *multiple views*
- Each view adheres to a *viewpoint*
- Each view consists of *models*
- ISO/IEC/IEEE 42010 specifies minimal requirements for:
  - architecture descriptions
  - architecture frameworks
  - architecture description languages
  - architecture viewpoints

# *A Brief History: Motivation*

- **1990s: Increasing interest in:**
  - software architecture
  - system architecture
  - enterprise architecture

## **Key References**

Dewayne E. Perry and Alexander L. Wolf. “Foundations for the study of Software Architecture”. In: ACM SIGSOFT Software Engineering Notes 17.4 (1992), pp. 40–52

Philippe B. Kruchten. “The “4+1” View Model of architecture”. In: IEEE Software 12.6 (1995), pp. 42–50

Eberhardt Rechtin. Systems architecting: creating and building complex systems. Prentice Hall, 1991

J. A. Zachman. “A framework for information systems architecture”. In: IBM Systems Journal 26.3 (1987), pp. 276–292

# ***A Brief History: Influences on IEEE 1471***

- **Emerging practices:**
  - Kruchten's 4+1 view model, RM-ODP
- **Rechtin and Maier:**
  - multi-disciplinary nature of architecture
- **Ross' Structured Analysis (SADT):**
  - making viewpoints *first-class*
- **Dijkstra:**
  - separation of concerns in software engineering
- **Boehm:**
  - explicit identification of system's stakeholders



# ***A Brief History: IEEE Std 1471***

- **IEEE Architecture **Planning** Group**
  - August 1995: first meeting, Montréal
  - 6 participants, 80 reviewers
  - April 1996: final report to IEEE Software Engineering Standards Committee
- **May 1996 to 2000: IEEE Architecture **Working** Group**
  - Bi-monthly meetings
  - 29 participants, 150 reviewers
- **October 1998: first IEEE ballot**
- **September 2000: IEEE Std 1471 approved for use**
- **August 2001: Adopted as an ANSI (US) standard**
  - ANSI/IEEE Std 1471

# *A Brief History: IEEE Architecture Planning Group, 1995*

**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 architecture**
- **Examine IEEE standards for architectural relevance**
- **Produce an Action Plan for future IEEE activities in this area**
  - **Led to creation of Architecture **Working** Group to produce a standard**

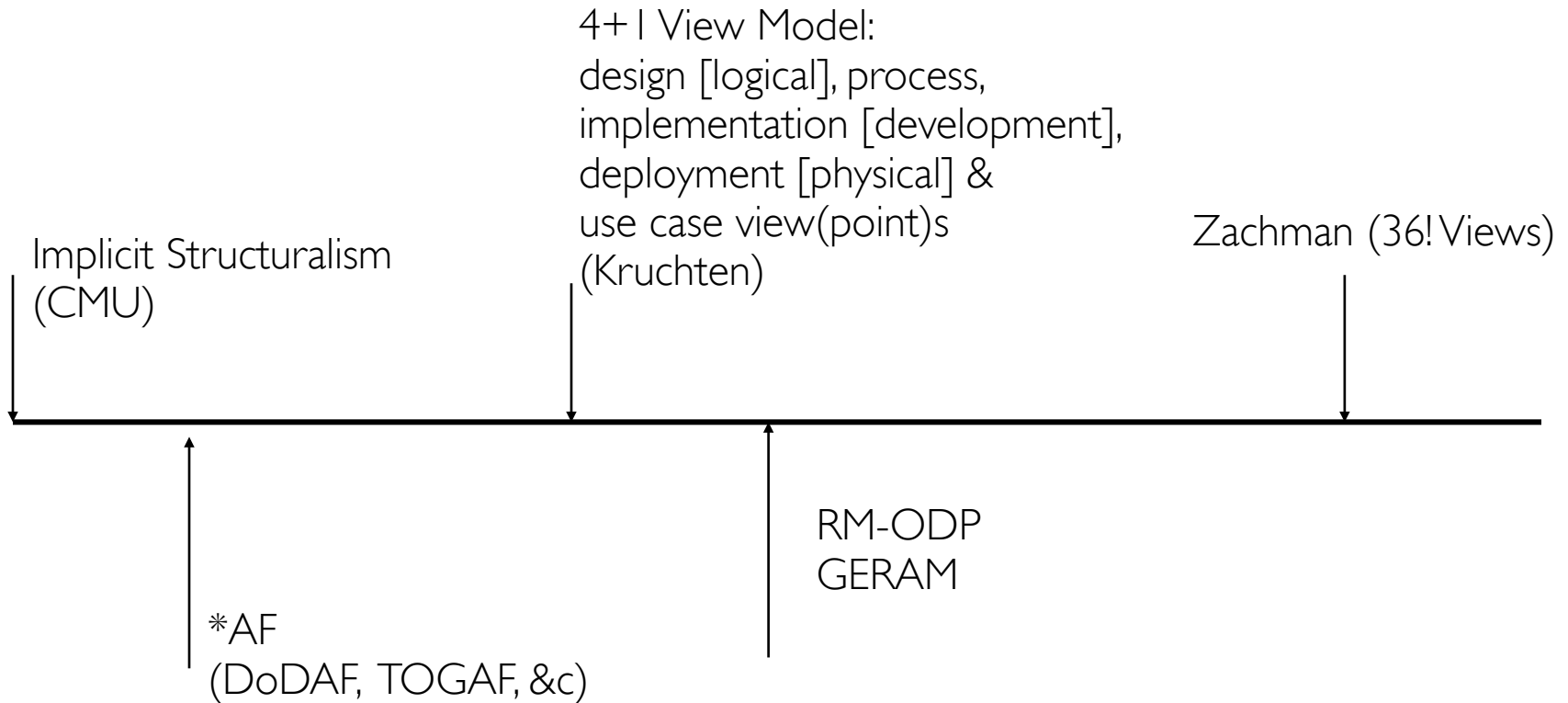
# *Architecture Working Group: Goals and Objectives*

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

# IEEE 1471:2000

- IEEE Standard 1471 *Recommended Practice for Architectural Description of Software-Intensive Systems*
  - Sponsored by the IEEE Computer Society
- IEEE 1471 was a *recommended practice*
  - “recommended practice”: one kind of IEEE standard
  - user decides whether to and how to employ IEEE 1471
- IEEE 1471 applied to *Architecture Descriptions*
  - Architecture Descriptions could conform to the standard
  - Systems, projects, processes or organizations could not conform
  - Analogy: a standard about “blueprints” not about “buildings”

# *Circa 2000: IEEE 1471 was intended to encompass...*



# ***A Brief History: Internationalization***

- **2005: ISO JTC 1/SC 7 formed an Architecture Working Group**
- **2006: ISO fast-track ballot of IEEE 1471 with immediate coordinated update**
- **2007: Publication of ISO/IEC 42010:2007**
  - ISO cover page on IEEE 1471:2000
- **2007–2009: Drafted revision**
  - Joint ISO and IEEE revision under ISO/IEC JTC 1/SC7 **WG 42**
- **2009–2010: Balloted revision**
- **2011: Publication of ISO/IEC/IEEE 42010**

## ***A Brief History: Revision Areas***

- **Joint revision by ISO JTC 1/SC 7 Working Group 42 and IEEE**
- **Expanded scope from software-intensive systems to general systems**
- **Aligned with ISO life cycle models: ISO 15288 (general systems), ISO 12207 (software)**
- **Harmonization with other ISO architecture-related standards**
- **Fixes, corrections and cleanup**
- **Extension to new areas:**
  - **Architecture Frameworks**
  - **Architecture Description Languages**

## *«Sidebar»*

# *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 (Maier and Rechtin, 2000)
- **Architecture is recognized as a critical element in the successful development and evolution of software-intensive systems**



## *«Sidebar» Motivation: Why a standard?*

- **Architecture is abstract**
- **Architecture works best when written down:**
  - to review, analyze, build to, and govern it
- **Best practices have focused on describing good architectures**
  - blueprints rather than buildings
  - without prescribing a process or method of architecting
- **Focus on a foundation “ontology” of architecture description**
- **Minimal set of conformance requirements, allow for exploration, evolution of the practice**

# *What is ISO/IEC/IEEE 42010?*

- **Formal international standard**
- **Unlike “market standards” such as OMG or Open Group**

**(IEEE Std 1471 now “retired”)**



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# ***What's in the Standard?***

## ***Contents***

- **Scope**
- **Conformance**
- **Terms and definitions**
- **Conceptual foundations**
- **Architecture descriptions**
- **Architecture frameworks and architecture description languages**
- **Architecture viewpoints**
- **Annexes**

# *What's in the Standard?*

- **Terms and definitions (10 terms, ~1 page)**
- **Conceptual foundations (~7 pages)**
- **4 Conformance Cases and their requirements (“shalls”):**
  - on architecture descriptions (24r\*, 4 pages)
  - on architecture frameworks (2r\*, 1 page)
  - on architecture description languages (1r\*, .5 page)
  - on architecture viewpoints (1r\*, 2 pages)
- **3 Annexes and a Bibliography**

**\* r = *number of requirements***

# *Scope of ISO/IEC/IEEE 42010*

- *General systems, software products and services, enterprises*  
“systems that are man-made and may be configured with one or more of the following: hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g. operator instructions), facilities, materials and naturally occurring entities”
- **including:**  
“individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, whole enterprises, and other aggregations of interest” [IEEE 1471:2000]
- *In the Standard, “system” is used to refer to any of these*

## ***“Out of Scope”***

*The Standard neither assumes nor prescribes the following:*

- **system theory**
  - What constitutes a system?
- **system life cycle**
  - How are systems developed, operated, &c
- **architecture methods (or tools)**
  - How are architectures devised, managed, &c.
- **Users of the Standard are free to choose their own!**

## *Terms and definitions*

- **The Standard defines and uses these terms as “reserved words”**

architecting  
architecture (of a system)  
environment (of a system)  
(system) stakeholder  
(system) concern

architecture description  
architecture framework  
architecture view  
architecture viewpoint  
model kind

# *Terms and definitions*

- **architecting**

**process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle**

- **architecture ⟨of a system⟩**

**fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution**



# *Terms and definitions*

- **environment** **〈of a system〉**

context determining the setting and circumstances of *all influences* upon a system

The environment of a system includes developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.

- **〈system〉 stakeholder**

individual, team, organization, or classes thereof, having an interest in a system

- **〈system〉 concern**

interest in a system relevant to one or more of its stakeholders

# *Terms and definitions*

- **architecture description (AD)**  
work product used to express an architecture
- **architecture framework (AF)**  
conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders

# *Terms and definitions*

- **architecture view**  
work product expressing the architecture of a system from the perspective of specific system concerns
- **architecture viewpoint**  
work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns
- **model\* kind**  
conventions for a type of modelling

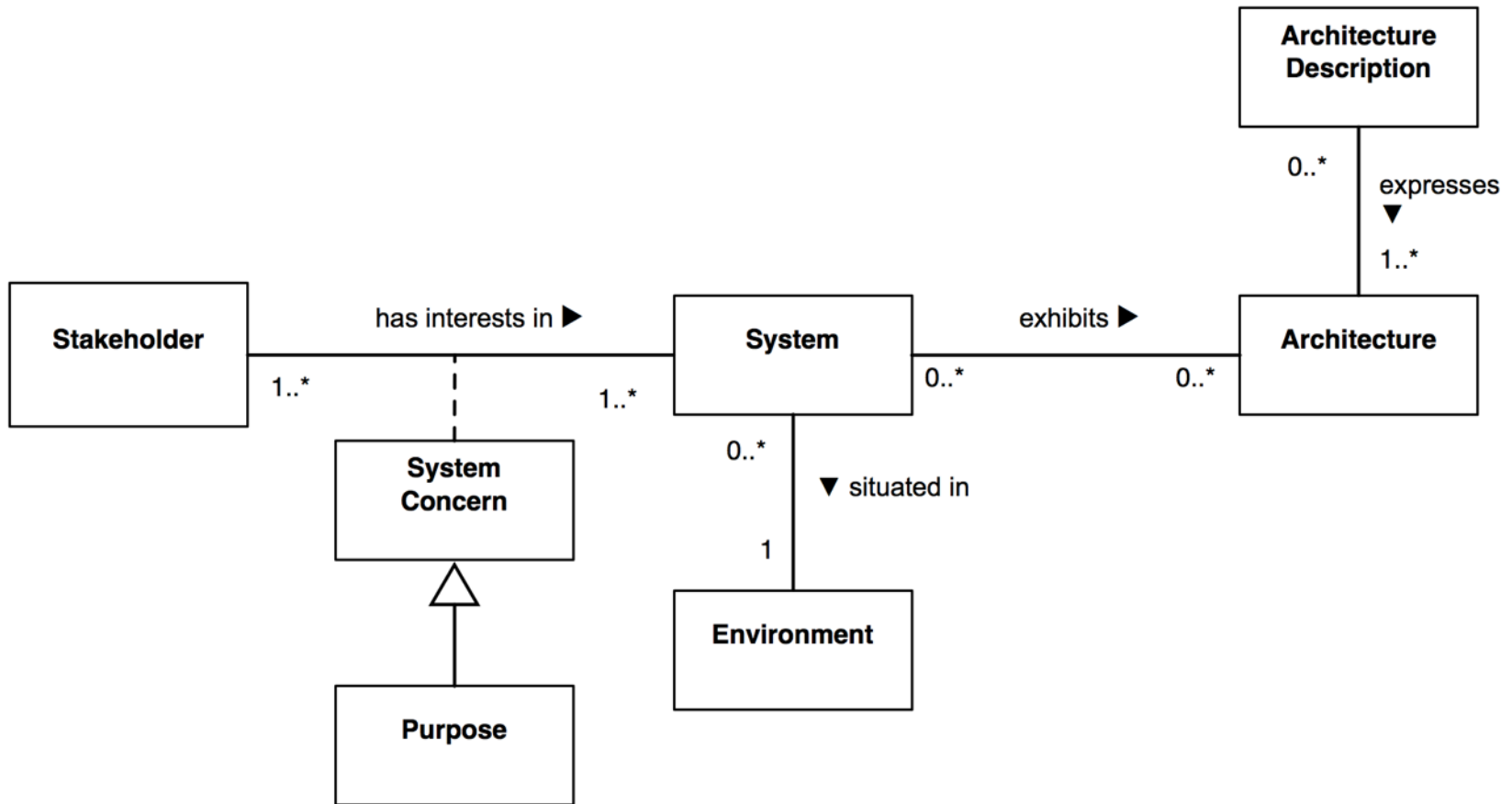
*\*What is a model?*

M is a *model* of S if M can be used to answer questions about S.

# *Conceptual foundations*

- To establish terms and concepts for architectural thinking
- To talk about **architectural descriptions** in context of
  - stakeholders and their concerns
  - system life cycle
  - “use cases” of architecture descriptions
- Basis to express the requirements on best practices
  - Expressed through text and an ontology  
“42010 meta model”
- Basis for evolution of the practice
  - where little commonality has existed

# Conceptual foundations: Context



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *Conceptual foundations: Context (in words)*

- **Systems have Architectures**
- **Architecture is what is fundamental about a System**  
(cf. the definition)
- **Systems have multiple Stakeholders with diverse Concerns**
- **Every System inhabits an Environment**
- ***Architecture Descriptions*** are used to express Architectures
- ***The Standard is about best practices for Architecture Description (not about Architectures)***

# *Conceptual foundations:*

## *Context*

- What's a *system*?
  - not defined by this Standard
- Architects need to identify the systems they are working on
- Many “system theories” compatible with the Standard

### Examples

**Systems Engineering (ISO 15288), General Systems Theory, Viable System Theory, Action Theory, Cybernetics, ...**

#### **List of systems theories**

[http://en.wikipedia.org/wiki/List\\_of\\_types\\_of\\_systems\\_theory](http://en.wikipedia.org/wiki/List_of_types_of_systems_theory)

# ***Conceptual foundations: Environment***

- **environment** *⟨of a system⟩*  
context determining the setting and circumstances of *all influences* upon a system
- **Every system inhabits an environment**
  - the totality of influences upon the system
  - throughout its life cycle
  - its interactions with that environment, including other systems
- **The environment of a system includes developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences**
- ***To understand a system's environment, understand its stakeholders and concerns***



# *Conceptual foundations: Stakeholders*

- **〈system〉 stakeholder**  
individual, team, organization, or classes thereof, having an interest in a system
- **Stakeholders have diverse interests, needs and requirements (often conflicting!) for a system**
  - key influences on the architecture throughout the system life cycle
  - classified as “**concerns**”

# *Conceptual foundations: Example stakeholders*

- clients
- acquirers
- owners
- users
- operators
- architects
- system engineers
- Quality Assurance
- customers
- power users
- analysts
- consumers
- developers
- designers
- maintainers
- service providers
- vendors
- suppliers
- subcontractors
- planners
- IV&V contractor
- Joe, the CTO

# *Conceptual foundations:*

## *Concerns*

- **〈system〉 concern**

interest in a system relevant to one or more of its stakeholders

- **Example concerns**

functionality, feasibility, usage, system purposes, system features, system properties, known limitations, structure, behavior, performance, resource utilization, reliability, security, information assurance, complexity, evolvability, openness, concurrency, autonomy, cost, schedule, quality of service, flexibility, agility, modifiability, modularity, control, inter-process communication, deadlock, state change, subsystem integration, data accessibility, privacy, compliance to regulation, assurance, business goals and strategies, customer experience, maintainability, affordability and disposability

# *Conceptual foundations: Concerns*

- **Concerns “frame” the key topics of an architecture**
- **Concerns arise throughout the system life cycle**
- **Concerns appear in many forms, in:**
  - **stakeholder needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks or other issues, ...**
- **The Standard does not specify how to name, interrelate or manage concerns**

# *Conceptual foundations: Motivation*

- Concept of *stakeholder* established in Requirements Engineering
  - Reflecting that many parties are involved in complex systems
  - Often having different perspectives
- Diverse *concerns* motivate multiple views
- Many “extra-functional requirements” derive from specific stakeholders
  - Affordability concerns owners, acquirers
  - Maintainability concerns maintainers and operators
  - Users just want a system that works now!

# *Conceptual foundations: Defining “architecture”*

- **architecture** ⟨**of a system**⟩

**fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution**

**where:**

**fundamental** = essential, unifying characteristics and ideas

**system** = application, system, platform, system-of-systems, enterprise, product line, . . .

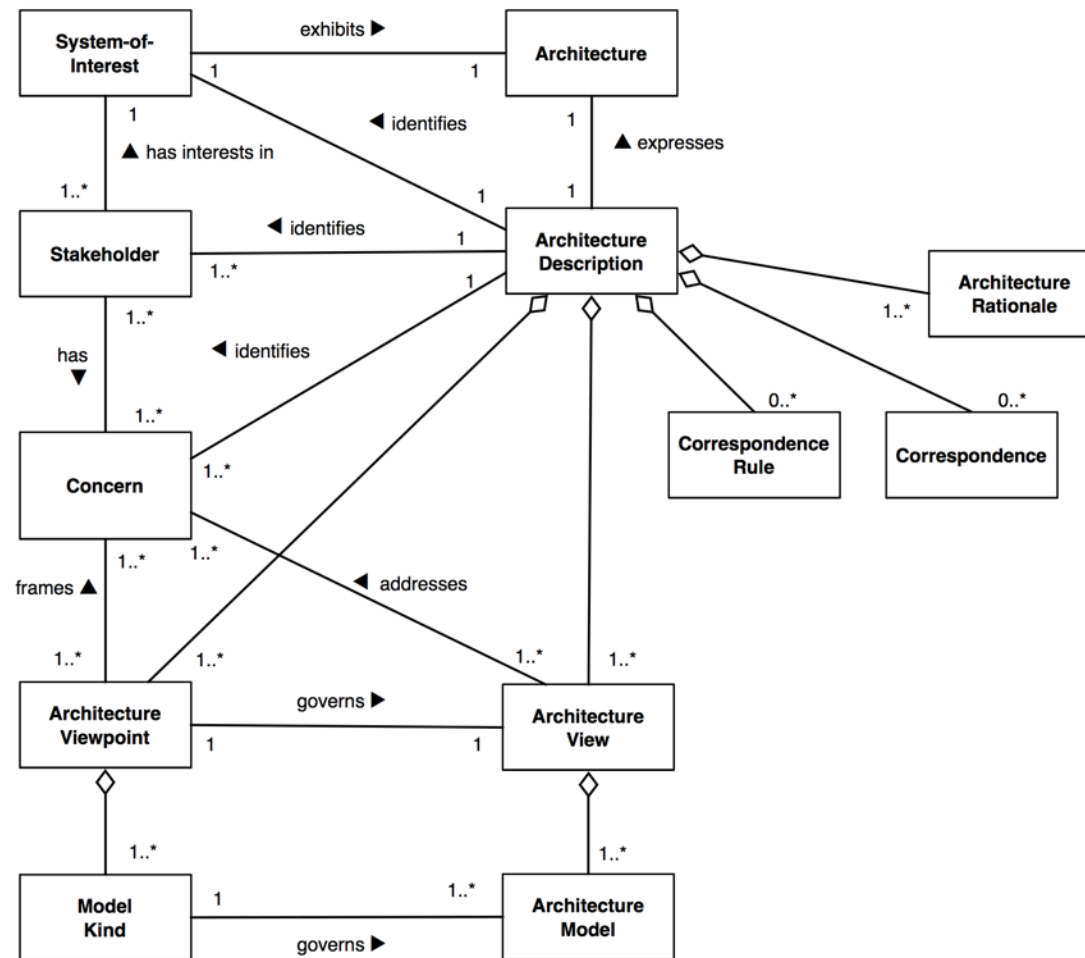
**in its environment** = developmental, operational, programmatic, ... context

# *Conceptual foundations: Architectures vs. architecture descriptions*

- **architecture description (AD)**  
work product used to express an architecture
- Architectures are *abstract notions*
- Architecture descriptions are *concrete artifacts* that can be shared and analyzed

**The Standard is about architecture descriptions, their contents and organization**

# Conceptual foundations: Architecture descriptions



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>



# *Conceptual foundations: Architecture descriptions (ADs, in words)*

- **An AD expresses an Architecture**
- **An AD addresses system Stakeholders and their architecture-related Concerns**
- **An AD is organized into Architecture Views**

views : architectural description :: chapters : book

# *Conceptual foundations: Architecture views*

- **architecture view**

work product expressing the architecture of a system from the perspective of specific system concerns

- **Each Architecture View addresses some Concerns of some Stakeholders**

- Support multiple audiences
- Reduce perceived complexity through *separation of concerns*

- **Each Architecture View consists of Architecture Models**

- **Each Architecture View adheres to the modeling conventions of its *Architecture Viewpoint***

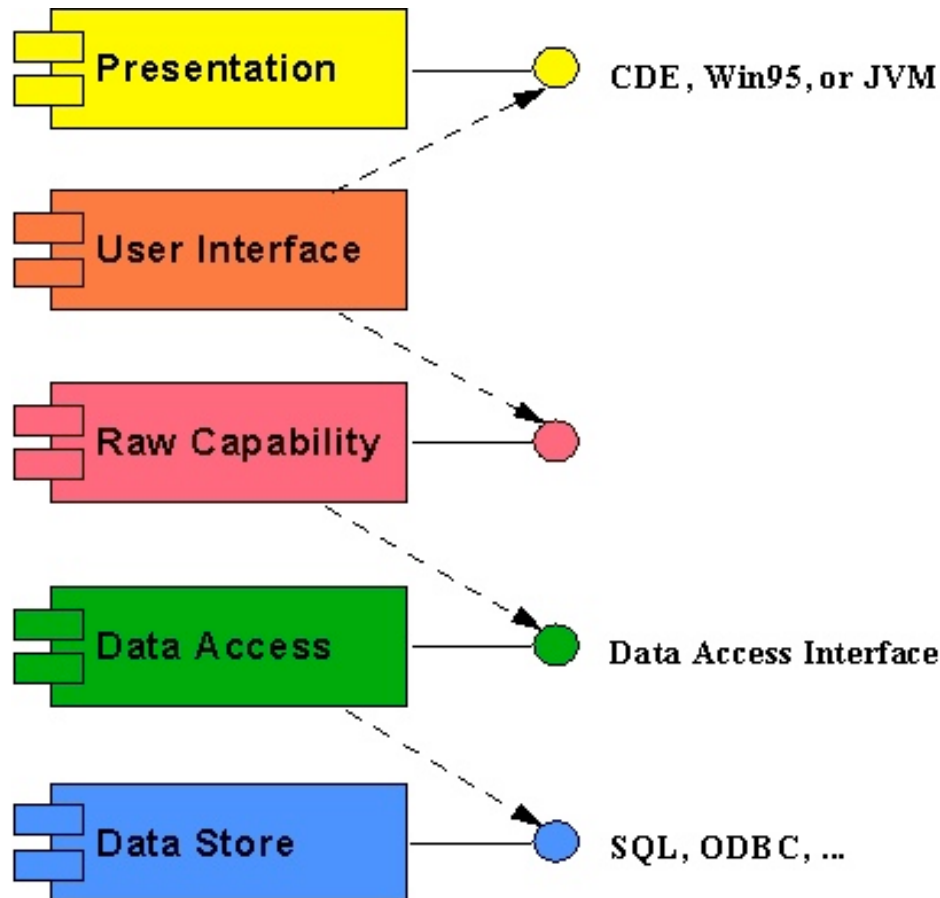
**Dijkstra, 1974: “separation of concerns”**

# *Conceptual framework: Architecture viewpoints*

- **architecture viewpoint**  
work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns
- **Views should be well-formed:**
  - Each view in an AD adheres to exactly one viewpoint
- **Viewpoints are *first-class*:**
  - Each viewpoint used in an AD is “declared” before use
- **Concerns determine viewpoint selection in an AD**
- **No fixed set of viewpoints:**
  - Standard is “agnostic” about where viewpoints come from
  - Organizations will evolve a viewpoint library or framework

## *(Old!) Example: Capability viewpoint*

- **name: Capability**
- **stakeholders:**
  - producers, developers and integrators
- **framed concerns:**
  - How is functionality packaged?
  - How is it fielded?
  - What interfaces are managed?
- **modeling conventions used:**
  - Components and their dependencies (e.g., UML component diagrams)
  - Interfaces and their attributes (e.g., UML class diagrams)
- **sources: also known as Static, Application, Structural viewpoints**



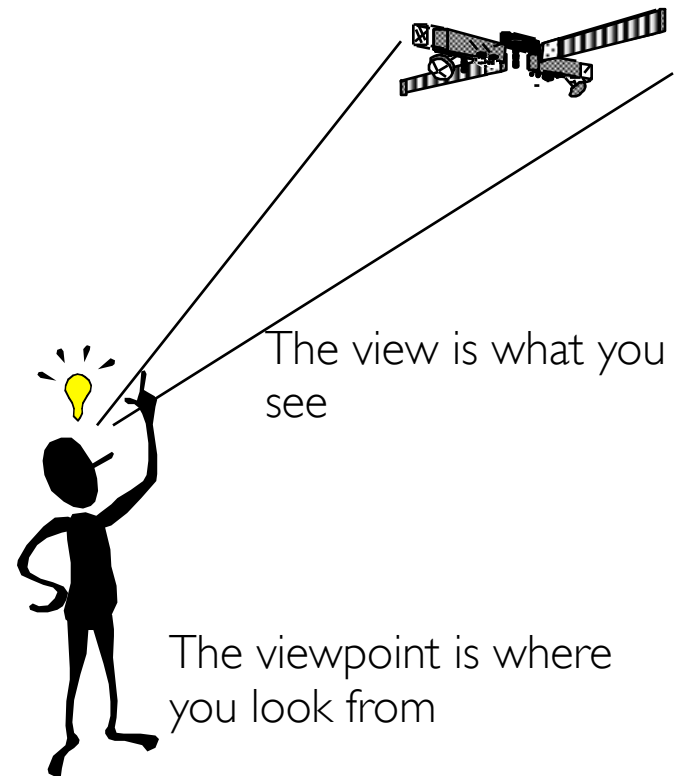
## *(Old!) Example: Capability view*

- The Capability view covers all system functionality for operating on data in System X
- Capabilities are fielded using a 5-tier layered organization with interfaces restricted to adjacent layers
  - Each layer instance is a capability
  - Entire stack instance is a deployable capability
- Capabilities can serve other capabilities

Copyright © 1998–2013 :: <http://www.iso-architecture.org/42010/>

# Conceptual foundations: Understanding views and viewpoints

- A **view** is a description of the whole system architecture from the perspective of a set of concerns
- A **viewpoint** defines the rules for constructing views
- Example:
  - A chair and a table have **different front views**, but the idea of a **front view(point)** is the same



# *Conceptual foundations: Why separate View and Viewpoint?*

- Originally derived from experience
  - Noticed “patterns” in good architecture descriptions, addressing the same issues on different systems
    - security, performance, structure, data, ...
  - Capturing the “pattern” made the next architecture easier, by providing a known starting point
- Literature showed several approaches with well-defined views
  - Kruchten’s 4+1, RM-ODP, Zachman, ...
  - Without separate names for **view** and **viewpoint** concepts
- Programming language influence:
  - Explicitly capture and declare a pattern before use
- Viewpoints are (re)usable across many systems

view : viewpoint :: program : programming language

# Examples of Current Viewpoints

Accessibility  
 Actor Co-operation  
 Application  
 Application Behavior  
 Application Co-operation  
 Application Structure  
 Application Usage  
 Assurance  
 Availability  
 Availability and Resilience  
 Behavior  
 Business Function  
 Business Goals  
 Business Process  
 Commerce  
 Component and Connector  
 Computational  
 Concurrency  
 Context  
 Decision  
 Decision Chronology  
 Decision Detail

Decision Forces  
 Decision Relationship  
 Deployment  
 Development  
 Engineering  
 Enterprise  
 Evolution  
 Execution  
 Functional  
 Implementation  
 Information  
 Information Structure  
 Infrastructure  
 Infrastructure Usage  
 Instructional  
 Internationalization  
 Landscape Map  
 Layered  
 Logical  
 Module  
 Networked Organizations  
 Operational  
 Organization  
 Parallelism  
 Performance  
 Platform

Process  
 Process Co-operation  
 Product  
 Regulation  
 Requirements  
 Safety  
 Scalability  
 Scenarios  
 Security  
 Service Data  
 Service Decision  
 Service Degree  
 Service Realization  
 Stakeholder Involvement  
 Systems Management  
 Technology  
 Trust  
 Usability  
 Validation  
 Variability

## 42010 Bibliography:

<http://www.iso-architecture.org/42010/reading-room.html>

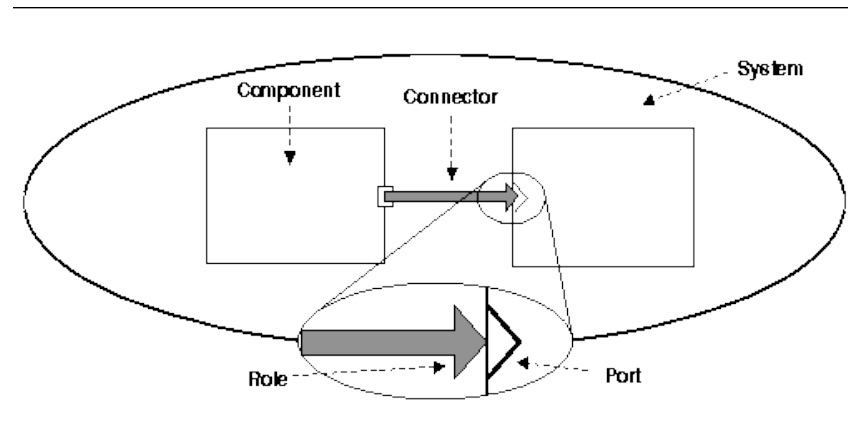
## Architecture Viewpoint Library (under development):

<http://www.iso-architecture.org/viewpoints/>



# 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?
- **C&C meta model:**
  - Components, connectors, ports and roles, attributes
- **Analytic Methods:**
  - Attachment, type consistency



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

# *Conceptual foundations: Architecture views*

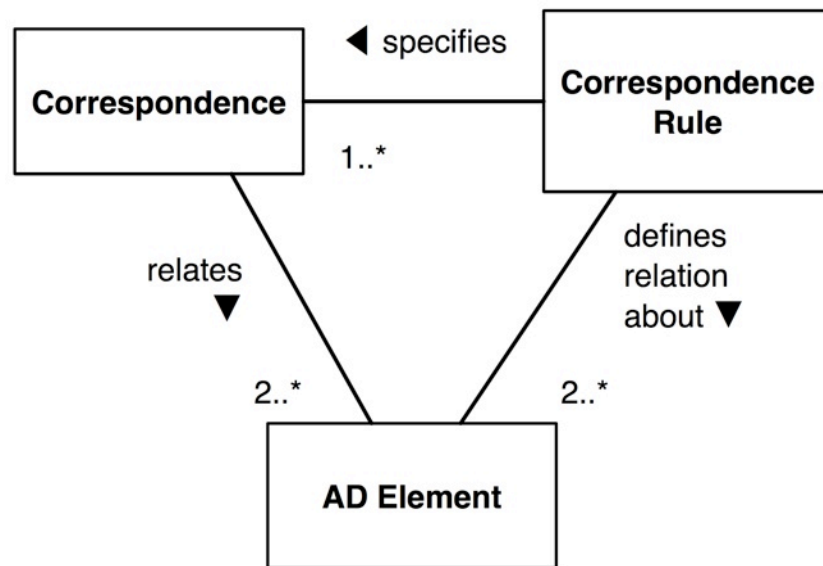
- **Views are independent, not “orthogonal”**
  - each view generally contains new information
- **Views are “modular”**
  - views consist of (one or more) *architecture models*
- **Consistency between views in an AD**
  - *Correspondences* used for consistency, traceability and other relations

# *Conceptual foundations: Models and model kinds*

- **model kind**  
conventions for a type of modeling
- Each model in a view within an AD has a *Model Kind*
- **Models enable:**
  - views to utilize multiple notations, and
  - sharing models across multiple views

*All models are wrong, some are useful.*  
— George E. P. Box

# Conceptual foundations: AD Elements and Correspondences



- **AD element**: any construct in an AD including
- **Stakeholders, Concerns, Views, Viewpoints, Models, Model Kinds, and their constructs (entities, relations, etc.)**

# *Conceptual foundations: AD Elements and Correspondences*

- **Correspondences relate elements of an AD to each other**
  - **Correspondence Rules exist in the “viewpoint layer” (M2)\***
    - **what correspondences should exist in an AD**
    - **Example: Every software element in a Logical View should be allocated to at least one processing node in the Deployment View**
  - **Correspondences exist in the “view layer” (M1)**
    - **what connections exist within this architecture**
    - **Example: A table showing individual logical elements allocated to particular processors**
- **Correspondences can be useful even without a rule**
- **Correspondences/Rules *across ADs* are also useful**

**\*Meta model layers will be discussed below (~slide 71).**

# *Conceptual foundations: Architecting in the System Life Cycle*

**The Standard is intended for use in a variety of life cycle contexts, e.g.:**

- **Architecting single systems or**
  - applications, systems, products, systems of systems, product lines, product families...
- **Iterative architecting for evolutionary systems**
  - “Rearchitecting” i.e. discovery of existing system architectures
- **“Steady state” enterprise projects**
- **Agile projects**

# *Conceptual foundations: Uses of Architecture Descriptions*

- **system design and development activities**
- **development and maintenance documentation**
- **evaluation of implementation alternatives**
- **analysis and evaluation alternative architectures**
- **documenting essential aspects of a system**
- **guide to future change**
- **specifying variation points or limitations**
- **recording architecture decisions, rationale and implications**
- **input to automated tools for simulation, system generation, analysis**

# *Conceptual foundations: Uses of Architecture Descriptions*

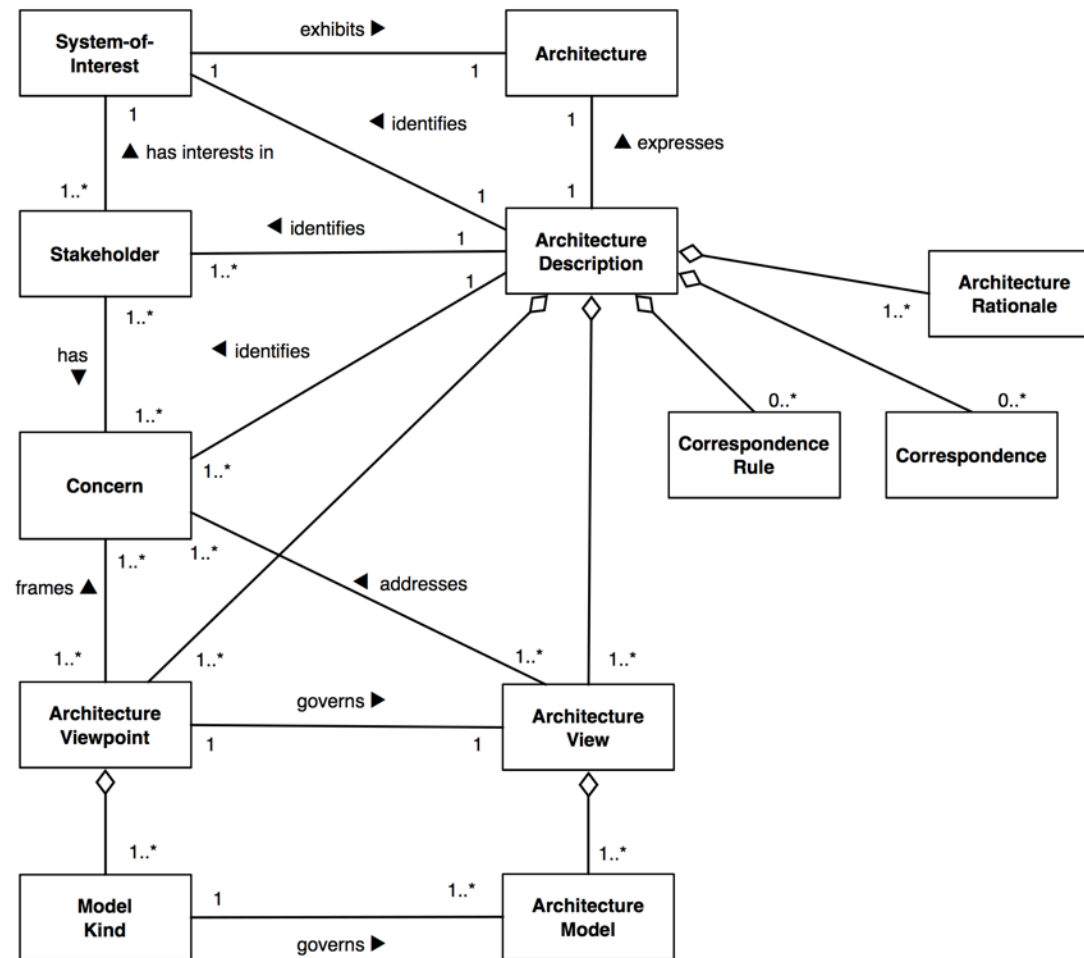
- **specifying a collection of system's common features (via architectural style, reference architecture or product line architecture)**
- **stakeholder communication in development, production, deployment, operation or maintenance**
- **acquisition via requests for proposal or statements of work**
- **negotiation among clients, acquirers, suppliers and developers**
- **sharing lessons learned and reusing architectural knowledge through viewpoints, patterns and styles**
- **training and education on best practices in architecting**



# *Conceptual foundations: Uses of Architecture Descriptions*

- **life cycle review, analysis, and evaluation of the system**
- **transition planning from legacy to new architecture**
- **guide to operational and infrastructure support, configuration management**
- **in system planning, scheduling and budgeting activities**
- **criteria for certifying implementation compliance**
- **conformance mechanism to external, project and/or organization policies**

# Wrap up: Conceptual foundations



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *Requirements and Conformance*

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# Requirements and Conformance

- The Standard defines 4 conformance cases
  - Architecture Description
  - Architecture Framework
  - Architecture Description Language
  - Architecture Viewpoint\*
- An item *conforms to the Standard* if it meets the relevant requirements (“shalls”) for that case

\*Architecture viewpoints can be documented either “stand alone” or as part of any of the other items.

# *Requirements on Architecture Descriptions*

*An AD conforming to the Standard has these properties*

- Includes\* information identifying the system of interest, and any supplementary information
- Identifies the Stakeholders with interests in the architecture
- Identifies the Concerns fundamental to the architecture
- continued ...

\* “includes” can be physical inclusion or logical reference.

**A template for writing Architecture Descriptions:**

<http://www.iso-architecture.org/templates/>

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *Requirements on Architecture Descriptions*

*An AD conforming to the Standard has these properties*

- **Includes Viewpoints framing all identified Concerns**
  - rationale for viewpoint selection
- **Each viewpoint fully specified (see below)**
- **Includes 1 view for each viewpoint, with**
  - identifying and supplementary information
  - architecture models covering the whole system and addressing all concerns of its governing viewpoint
  - record of any outstanding issues
- **Each model with identifying information and associated model kind**

## *All about multiple views*

- **Architecture Descriptions are inherently multi-View**
  - No single View addresses all concerns
  - (This was controversial in the 1990s but is generally accepted now)
- **A View should **cover** the entire system (with respect to the concerns of interest)**
  - Each View addresses a specific set of Concerns
  - The View can be composed of multiple models, to **cover** the system
    - e.g. a network queuing Model and a CPU processing model to address end-to-end performance concerns
    - The View aggregates Models/Model contents sufficient to ensure completeness

# *Requirements on Architecture Descriptions*

## *A viewpoint is determined by—*

- Viewpoint name
- Stakeholders addressed by the viewpoint
- Architecture concerns **framed** by the viewpoint
- Modeling conventions, language, notations, modeling techniques, analytical methods used to construct, depict and analyze resulting views
- Consistency, completeness checks
- Evaluation, analysis techniques to be applied
- Heuristics, patterns or other guidelines
- Sources, if any (e.g., author, literature citation, prior art)

**Viewpoint template:**

<http://www.iso-architecture.org/42010/templates/>



## ***The Standard is agnostic toward...***

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

# *Stepping Back*

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

## *Stepping back*

- **Nothing unique to software-intensive systems in the Standard**
- **Architectures exist to satisfy known concerns from stakeholders**
- **Architecture Descriptions are inherently multi-viewpointed**
  - No single view addresses all concerns
- **The discipline of identifying stakeholders and concerns has value far beyond Architecting**

# *Stepping back: The “Big” Ideas from the Standard*

- **Separate Architecture from Architecture Description**
  - *The map is not the territory*
- **Separate View from Viewpoint**
  - *The map is not the legend*
- **Use indirection**
  - 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)

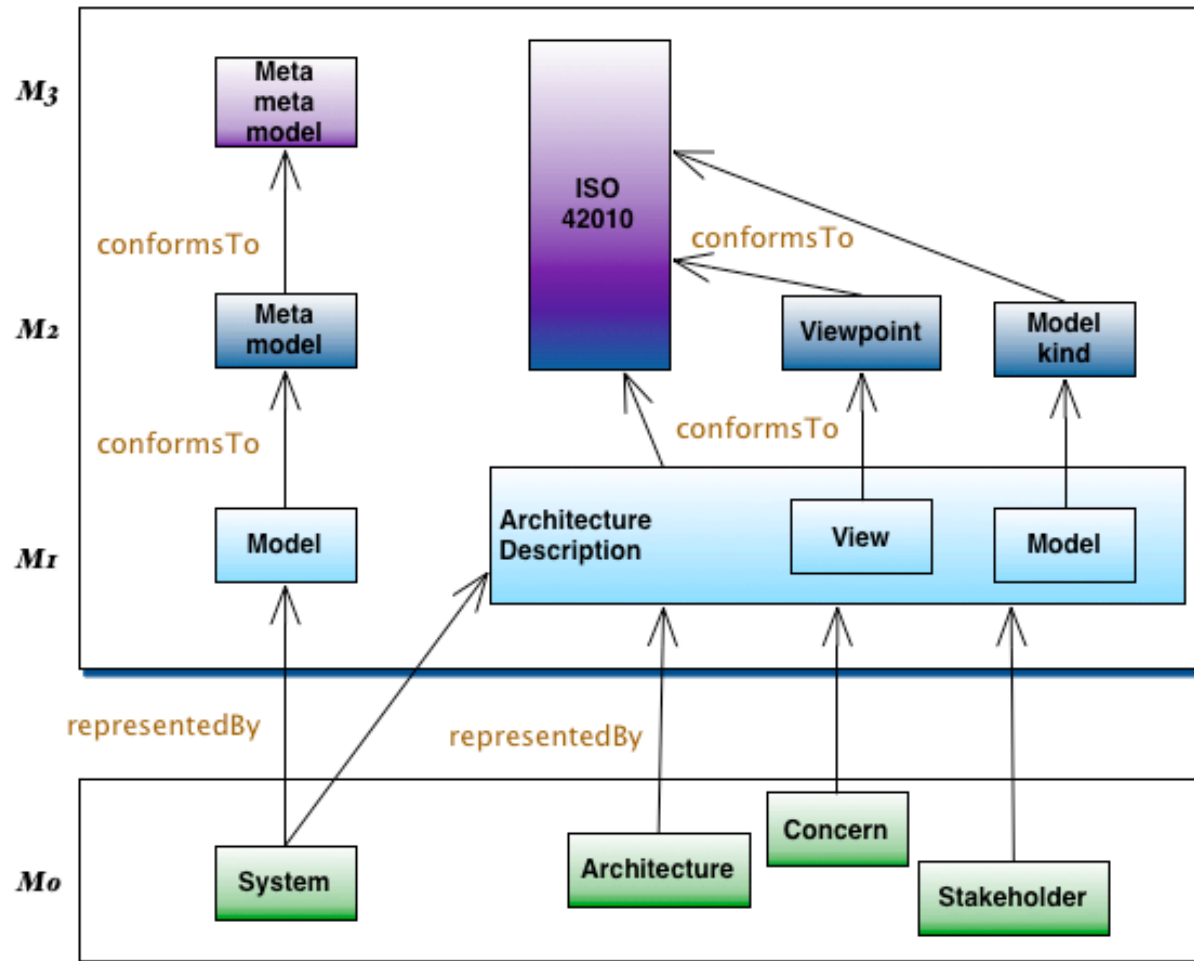
# *Stepping back: Stakeholders and Concerns*

- **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

## *Stepping back: Views vs Viewpoints*

- **Viewpoints (“what to describe”) are distinct from Views (“this description”)**
  - Views contain the content specific to this Architecture (Description) of this particular system of interest
  - Viewpoints defines “how to construct the view”
  - Viewpoints ensure stakeholder concerns are identified, allocated and covered
- **The Standard requires a 1-1 relationship between Viewpoint and View in a given AD**
  - consequence of “whole system” requirement on views

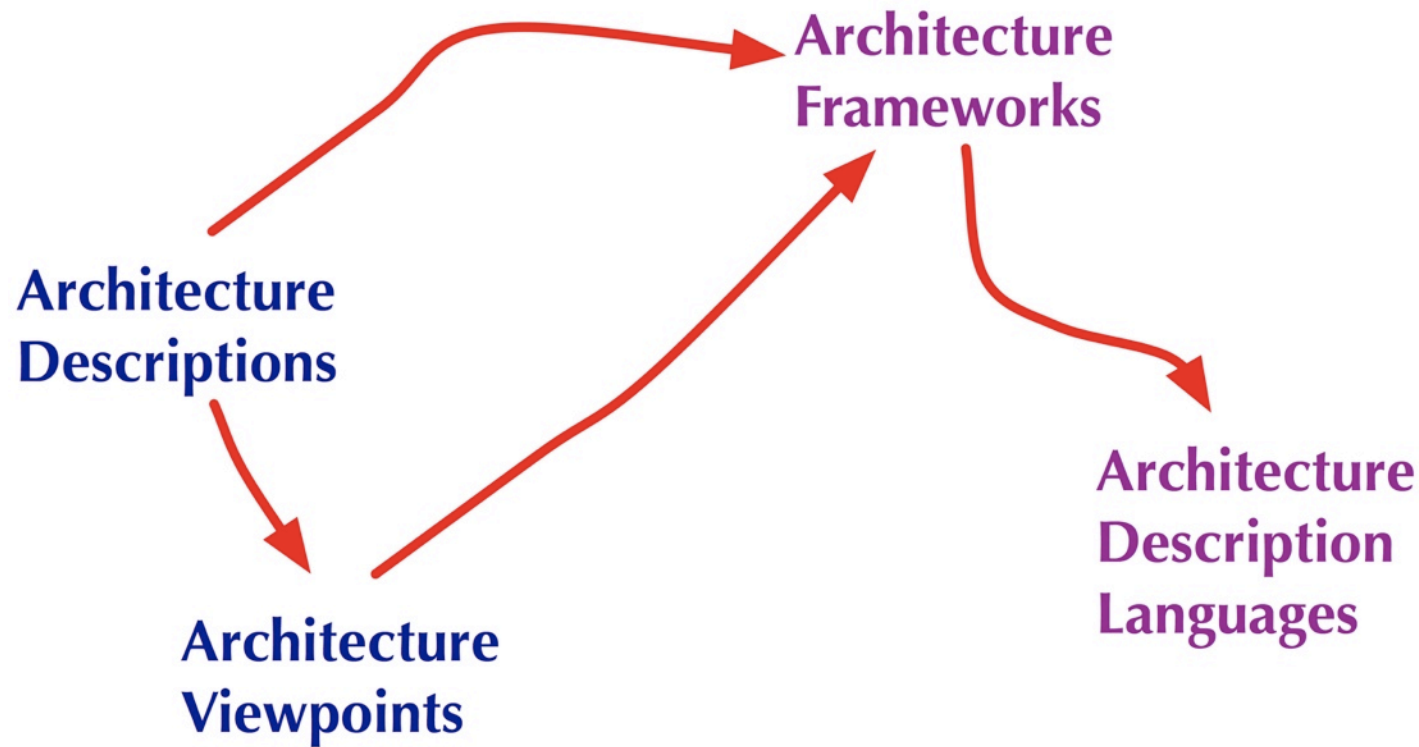
# Stepping back: 3+1 MDA



Jean Bezivin, "On the unification power of models", 2005.

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

## *Stepping back: Logical progression*





# *Architecture viewpoints*

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *Viewpoints:*

## *Why care about viewpoints?*

Routine design involves solving familiar problems, reusing large portions of prior solutions. ... Most engineering disciplines capture, organize, and share design knowledge to make routine design simpler. Handbooks and manuals are often the carriers of this organized information.

— Mary Shaw, “Prospects for an engineering discipline of software” *IEEE Software*, November 1990

In the Bibliography (of the Standard):

[42] Viewpoints repository, <http://www.iso-architecture.org/viewpoints>

# ***Viewpoints: Motivation***

- **Viewpoints “bind together”**  
Problem – Representation – Method
- **Other approaches have been less than successful...**
- **Possible Analogy: Evolution of software design methods**
  - Top-down functional refinement
  - to object-oriented design
  - to aspect-oriented design

# *Architecture Frameworks*

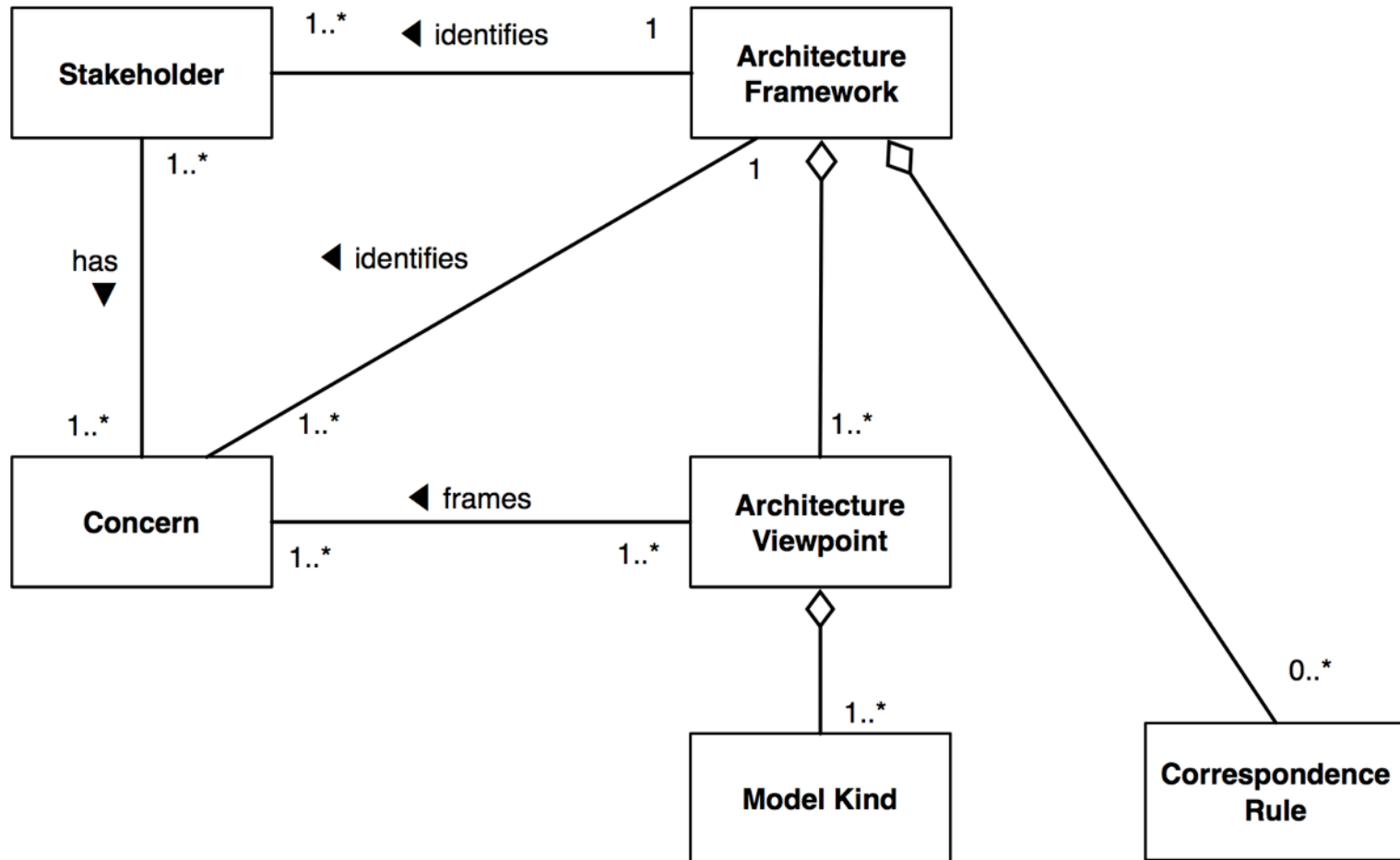
Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# Architecture frameworks

- **architecture framework (AF)**  
conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders
- **Generalizes notion of an AD**
  - includes stakeholders, concerns, viewpoints and model kinds
  - no views or models
- **Formalizes idea from the enterprise, systems and software communities:**  
“An enterprise architecture framework, or architecture framework for short, is a prefabricated structure that you can use *to organize your enterprise architecture into complementary views*”

# Conceptual foundation

## Architecture framework



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *Requirements on Architecture Frameworks*

**An architecture framework is determined by**

- **identified stakeholders**
  - **identified concerns**
  - **viewpoints and model kinds framing those concerns**
  - **any correspondence rules**
  - **conditions of applicability**
- 
- **Defines conformance of a framework to the Standard, and**
  - **Adherence of ADs to the framework**

# *Architecture Description Languages (ADLs)*

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>



# Architecture Description Languages

- *Insight: an ADL is “like” an architecture framework*
- **Requirement:**
  - ADL identifies, stakeholders, concerns, at least one model kind
  - no requirement for viewpoints!
- **1st generation ADLs:**
  - Wright, Rapide, Acme, ...
- **2nd+ generation ADLs**
  - UML (and profiles), AADL, ArchiMate, SysML, ...

# *Future*

- **Current work of ISO/IEC JTC 1/SC 7 WG 42**
- **ISO/IEC 42030 standard for Architecture evaluation:**
  - building on conceptual foundation of ISO/IEC/IEEE 42010, and body of knowledge in architecture assessment and evaluation (ATAM, SARA, &c.)



<http://www.iso-architecture.org/iso-archeval/>

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

## ***Future: Open Issues and Opportunities***

- **Concern modeling, naming and relationships**
- **Relating AK mechanisms:**
  - **viewpoints, styles, patterns..**
- **Viewpoints and frameworks for specific domains of application**
- **Viewpoint Interoperability, Model Integration**
- **Architecting methods exploiting viewpoints**
- **Automated tool support**

## *For more information...*

### *Resources*

- **Web site which includes:**
- **Users group**
- **Bibliography**
- **Templates, presentations and other resources**

<http://www.iso-architecture.org/42010/>

## *Some Additional Topics*

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# Topics: *Defining “architecture” before IEEE 1471*

- 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 *makefile*

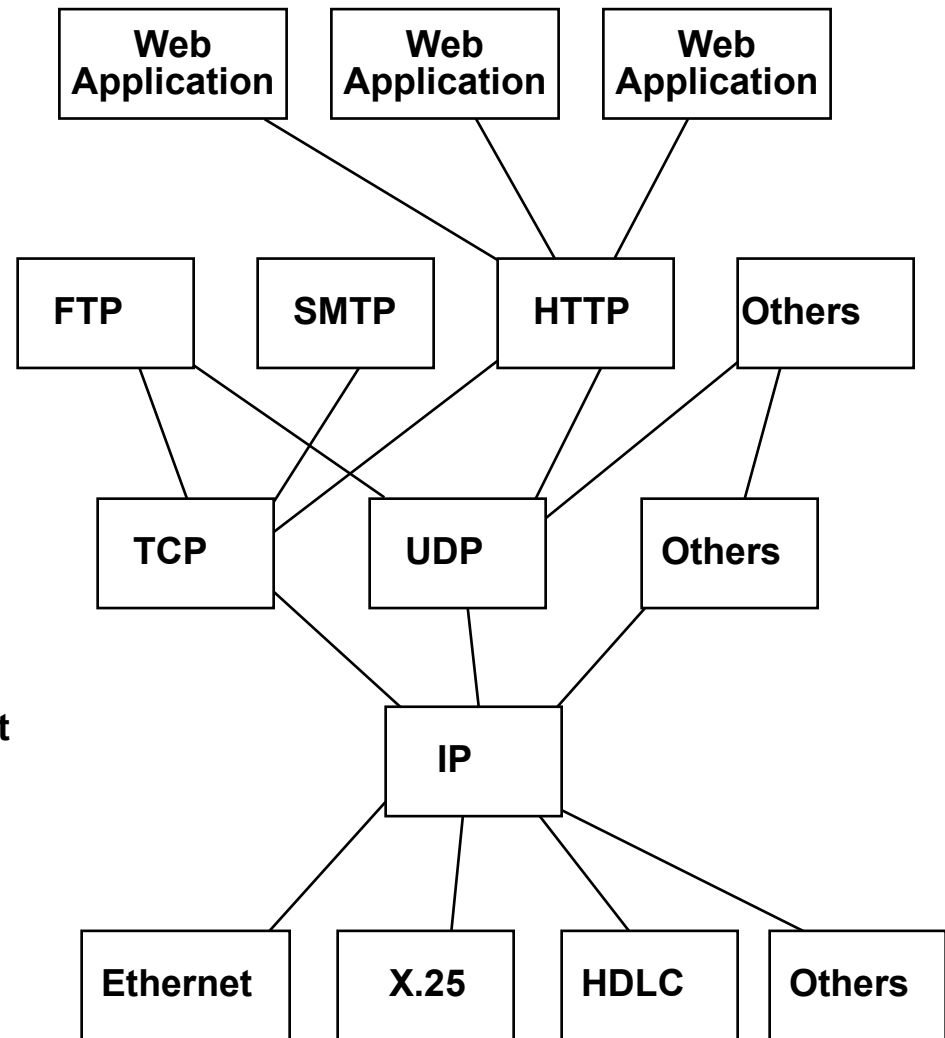
# *Topics:*

## *Is architecture more than “structure”?*

- “Structuralists” say,
  - No, **Structure** *is the key to architecture*
- “Contextualists” say,
  - Yes, *fitness for use is the key to architecture*
  - Structure is “design”
  - Architecture is different from Top-Level Design
- **But ... Structure of what?**
- IEEE 1471 encouraged the contextualist stance
  - Components, ... relationships to each other, and to the environment ...
  - *but could still be applied by structuralists without prejudice*

## Topics: *When Structure isn't Physical*

- The diagram shows 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 interesting



Copyright © 1998–2013 :: <http://www.iso-architecture.org/42010/>



## *Topics:* *Is Structure sufficient?*

- IEEE 1471 authors argued “no, many aspects of a system are not structural”
  - “ilities”, reliability, maintainability, flexibility, security, etc.
  - hence the focus on Concerns!
- Need to ensure system “fit for intended use”
  - Many systems meet all stated requirements and are not usable
  - Following *the building metaphor*:
    - Architect ensures fitness for use
    - Engineer assumes the Architecture, works within the lines
    - A gothic cathedral is much more than flying buttresses...
- Architecture as trade-off space for requirements
  - Architecture must be able to achieve all key requirements

# *Why no Process Requirements in IEEE 1471?*

- **Focus of IEEE AWG on capturing existing consensus**
  - Significant consensus on "what" vs. "how"
- **Explicit process for architecture still emerging**
  - e.g., 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" architecture descriptions

# *An Implied 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 Kinds**
- **Integrate Views**
- **Oversee Construction/Production**
- **Maintain/Evolve Architecture**
  - **Bottom up (from construction), outside-in (from environment),**
  - **Variances, Interpretation, Modification consensus process**

# *Building metaphor*


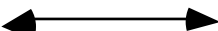
- **Construction/Civil Works**
  - 5000 year history (Imhotep, (2635-2595 B.C.) is first recorded architect)
  - First writings on architecture date to Roman times, Vitruvius (70bc–20bc), *De Architectura*
  - Distinction between Architect and Civil Engineer developed with Industrial Revolution
    - 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
	<b>Art</b> and Science	Art <b>and</b> Science	<b>Science</b> 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

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

INTERNATIONAL  
STANDARD

ISO/IEC/  
IEEE  
42010

First edition  
2011-12-01

---

Systems and software engineering —  
Architecture description

*Ingénierie des systèmes et des logiciels — Description de l'architecture*



Reference number  
ISO/IEC/IEEE 42010:2011(E)

© ISO/IEC 2011  
© IEEE 2011

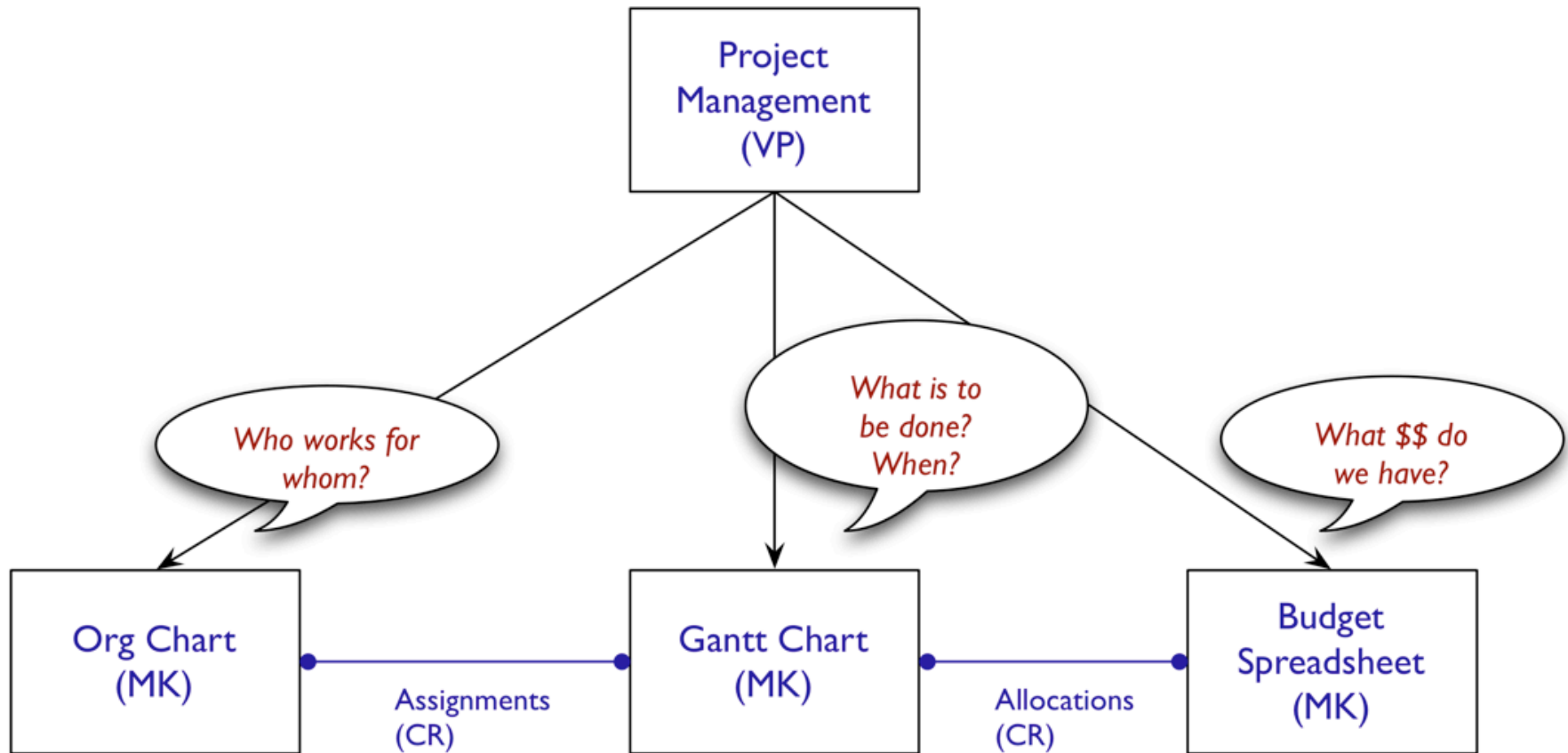
# *Studies in AD Style*

**Rich Hilliard**  
**[richh@mit.edu](mailto:richh@mit.edu)**

Inspired by the classic:  
Hibbard, Hisgen, Rosenberg, Shaw & Sherman,  
*Studies in Ada Style*,  
Springer-Verlag, 1983

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *Project Management Viewpoint (multiple model kinds)*



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>



# Relating to Zachman's framework schema

(Concerns)

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>	PEOPLE <i>Who</i>	TIME <i>When</i>	MOTIVATION <i>Why</i>
Objective/Scope (contextual) <i>Role: Planner</i>	List of things important in the business	List of Business Processes	List of Business Locations	List of important Organizations	List of Events	List of Business Goal & Strategies
Enterprise Model (conceptual) <i>Role: Owner</i>	Conceptual Data/ Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (logical) <i>Role: Designer</i>	Logical Data Model	System Architecture Model	Distributed Systems Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Technology Model (physical) <i>Role: Builder</i>	Physical Data/Class Model	Technology Design Model	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Representation (out of context) <i>Role: Programmer</i>	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Speculation
Functioning Enterprise <i>Role: User</i>	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

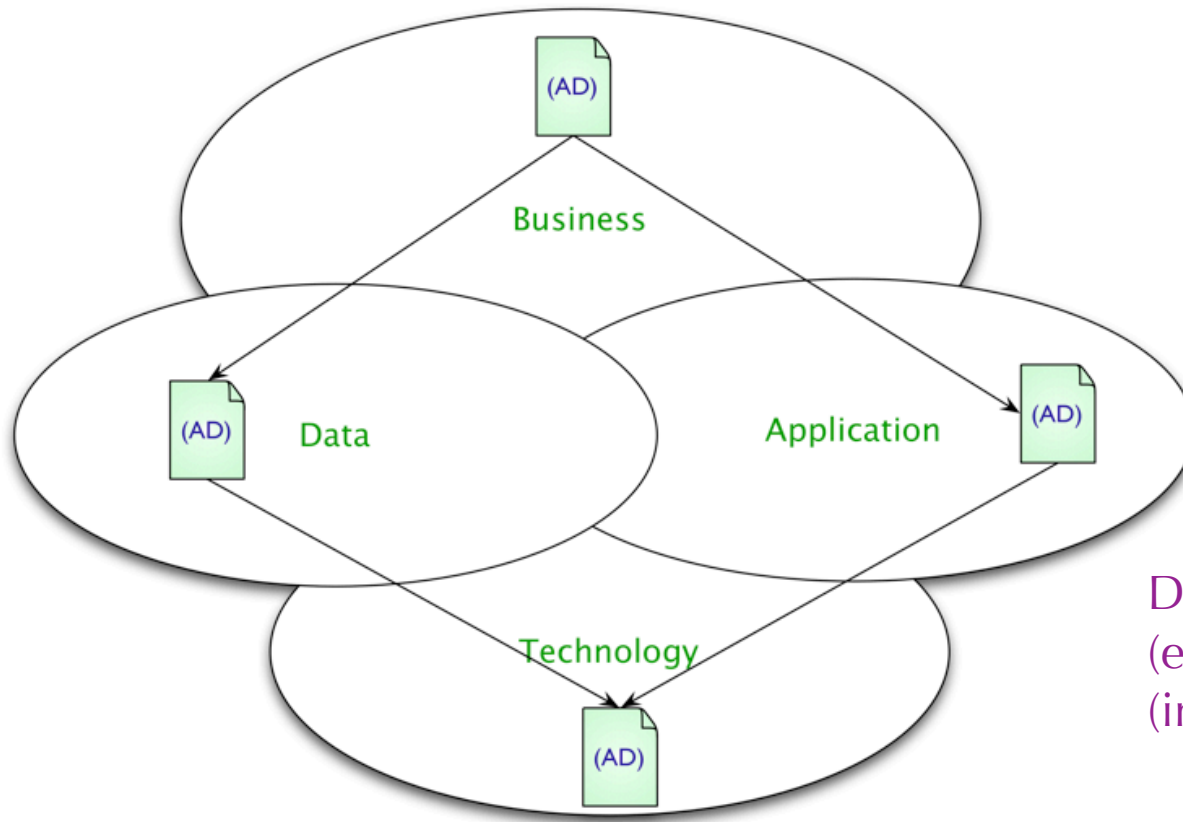
(MKs)

(Stakeholders)

Zachman, circa 1992.

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *“Domains”*

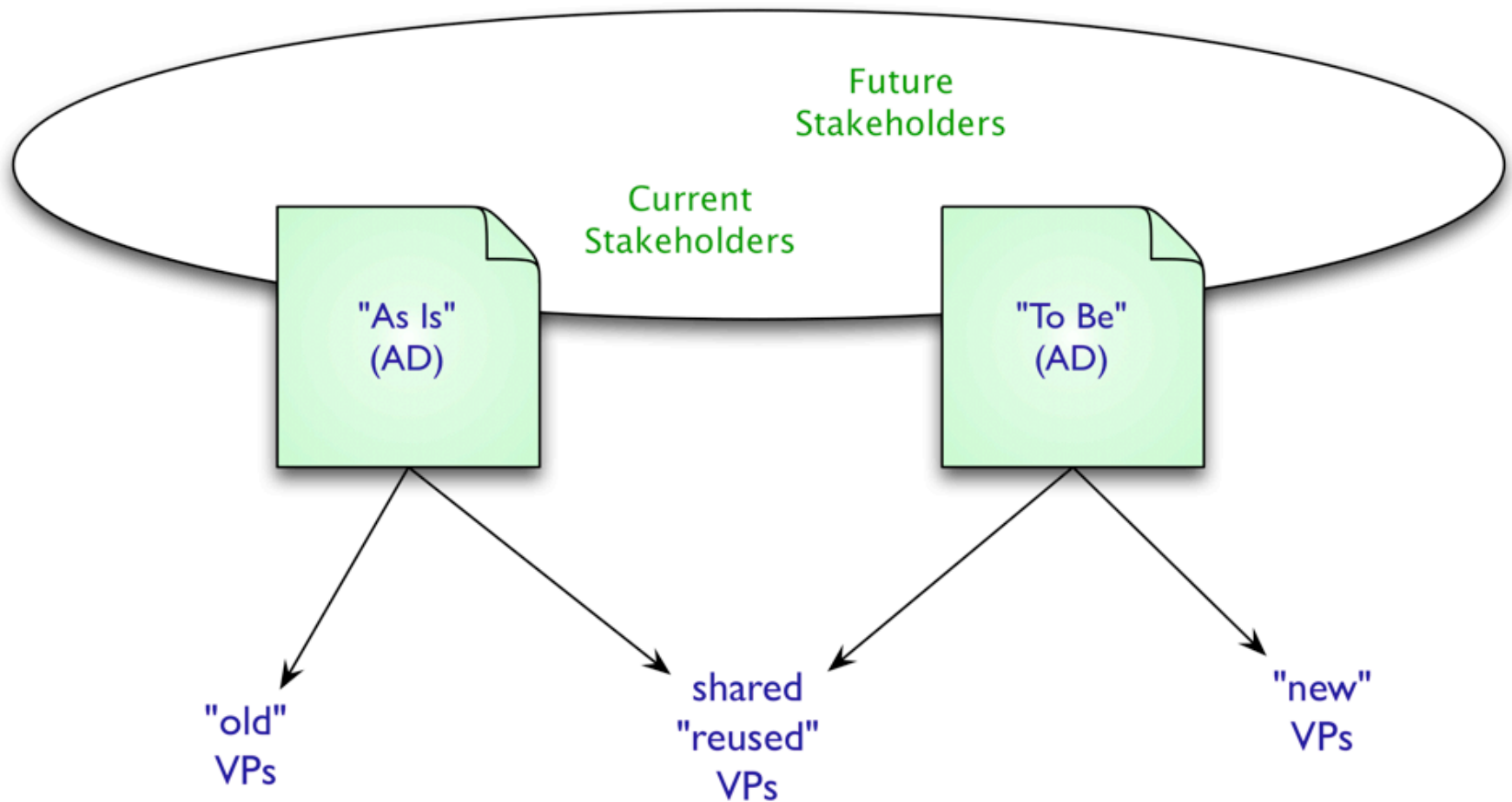


Domains: could result in 1 (evolving) AD or multiple (interconnected) ADs.

TOGAF et al.

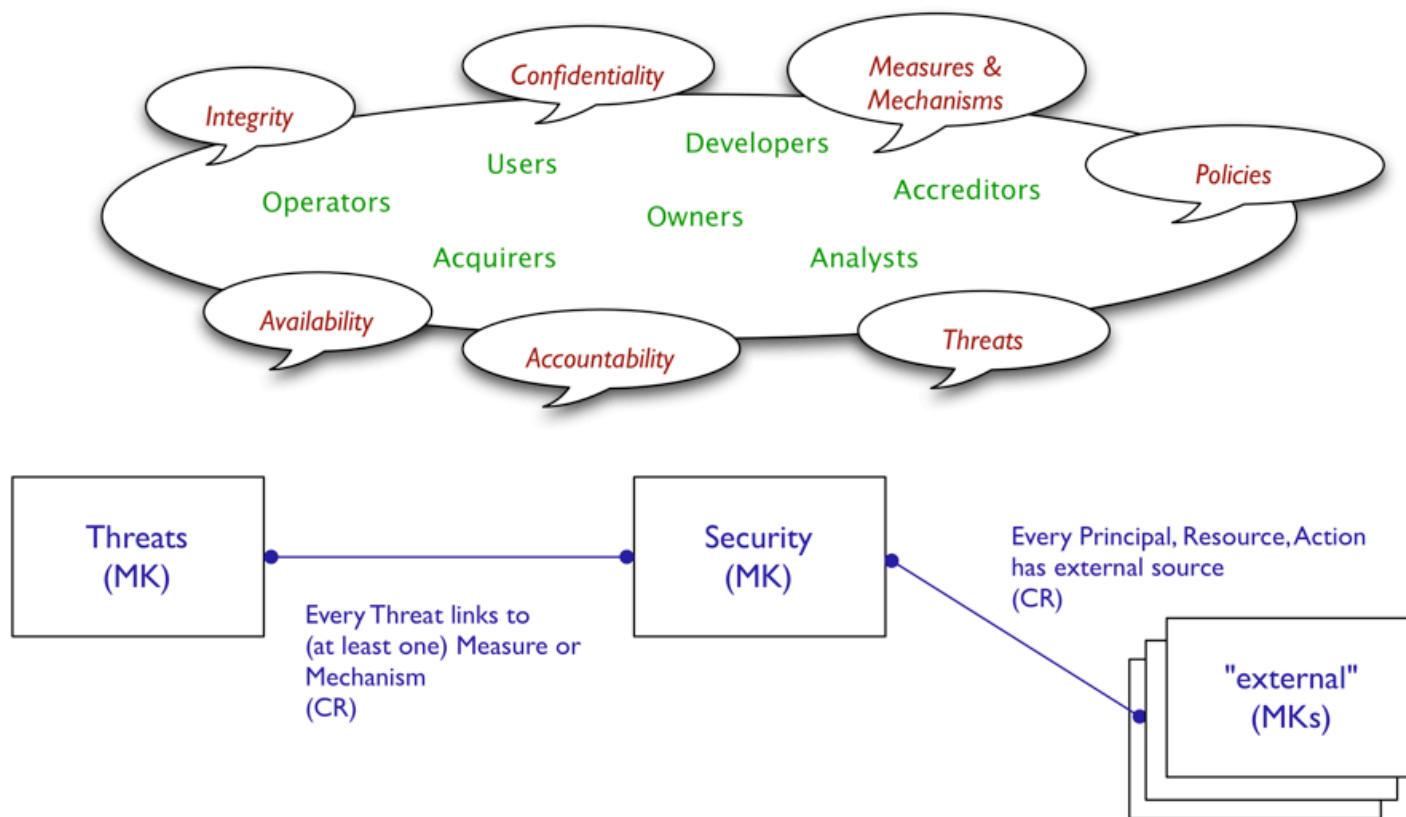
Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *As-Is (Baseline) and To-Be (Target) Architecture Descriptions*



Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# Trust Viewpoint

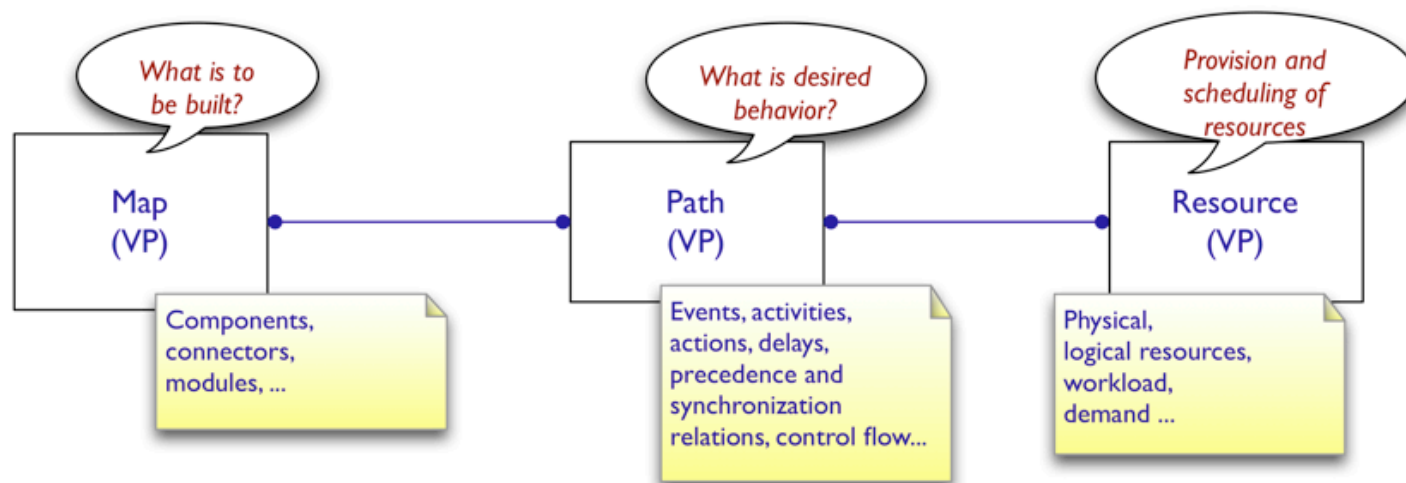


Hilliard, 2009

<http://web.mit.edu/richh/www/writings/hilliard-TrustVP-r1.pdf>

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# A Performance Framework: Maps, Paths and Resources



CRs include:

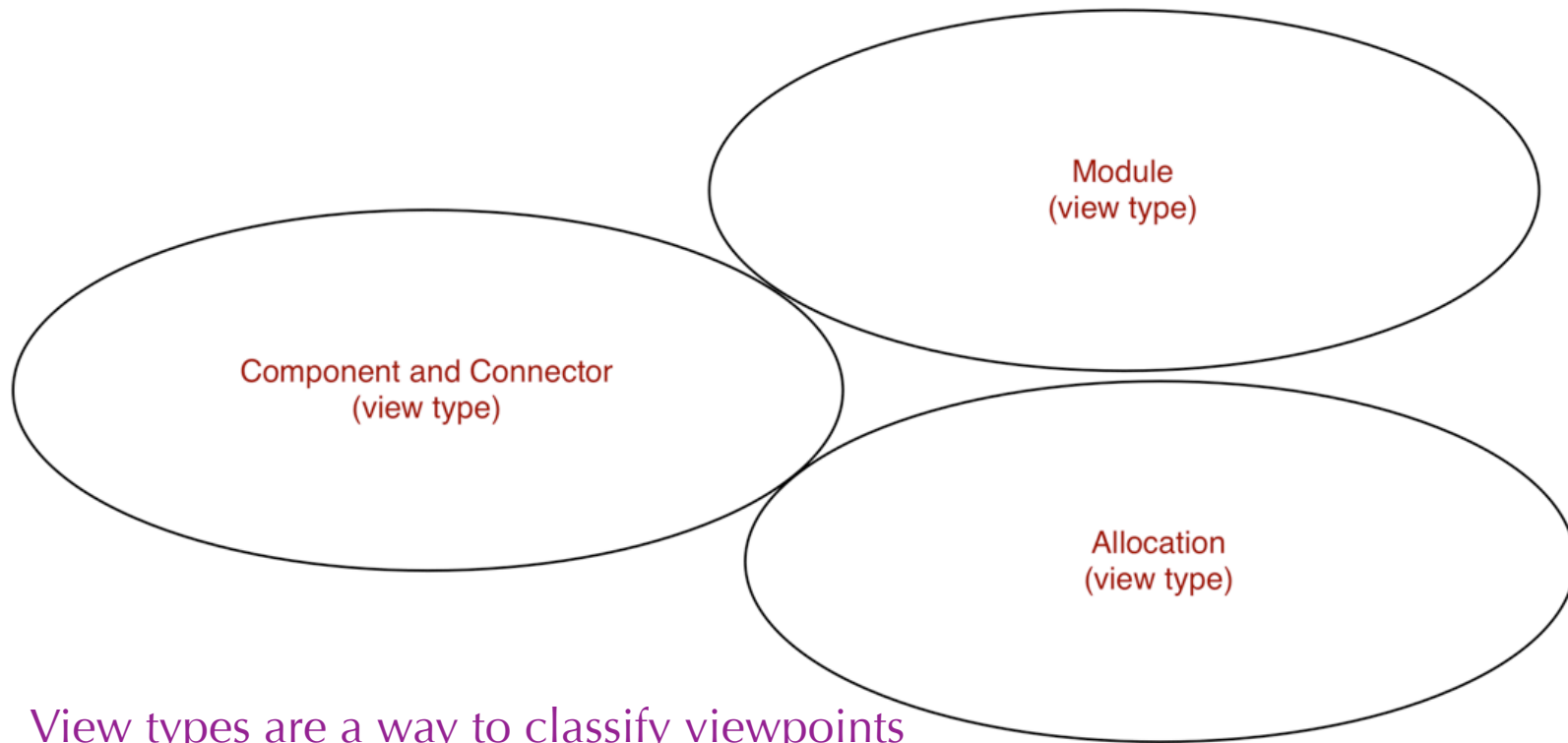
- Resource requirements per Activity.
- Activities allocated to Components.

Constituent viewpoints may be "native" or imported from existing viewpoints.  
The framework supports several kinds of performance analysis: congestion, contention, throughput, ...

C.M. Woodside, 1995, "A three-view model for performance engineering of concurrent software" IEEE Trans. Soft. Eng 21(9), 1995

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# Viewpoints vs. View types

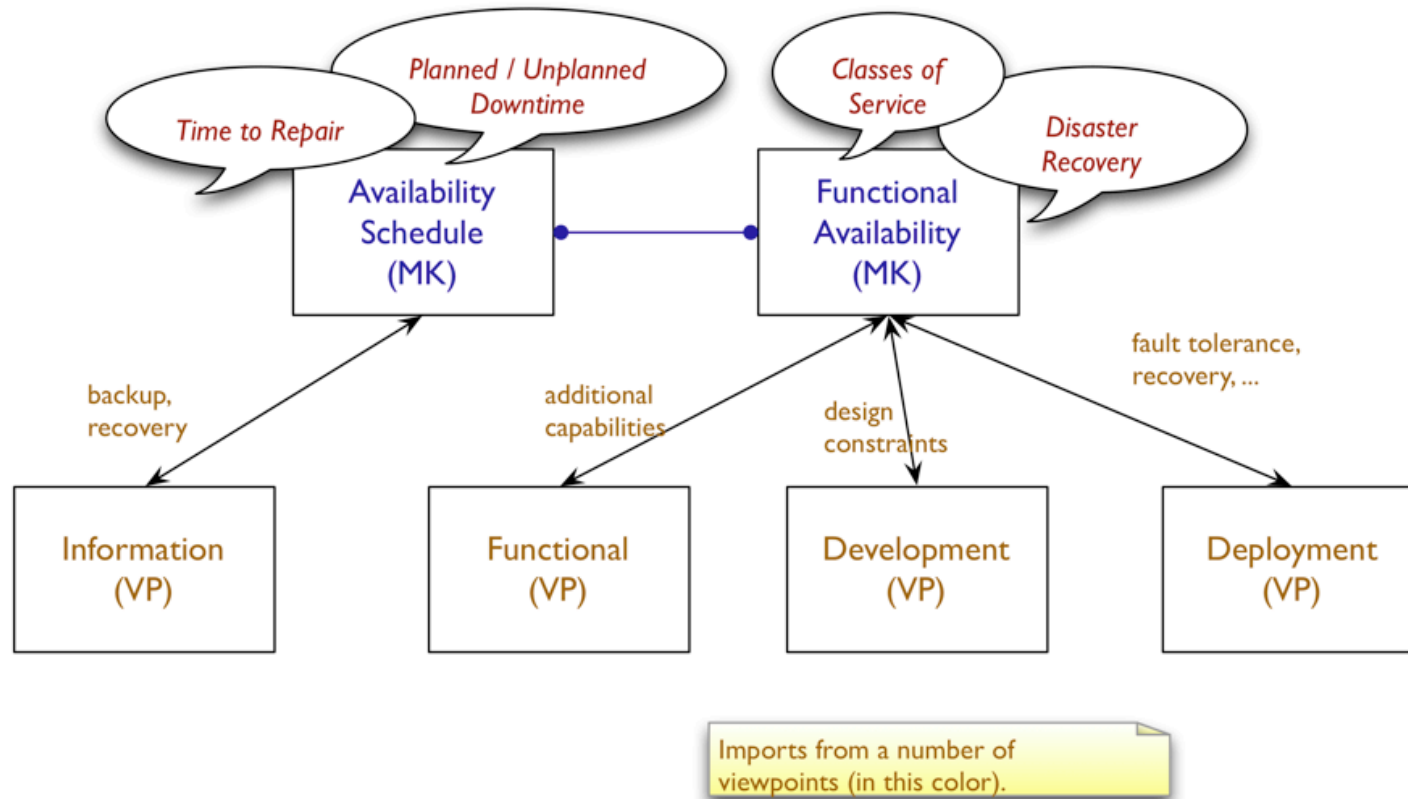


View types are a way to classify viewpoints based on their AD elements and concerns.

P.C. Clements et al. *Documenting Software Architectures: views and beyond*. 2nd. Addison Wesley, 2010

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

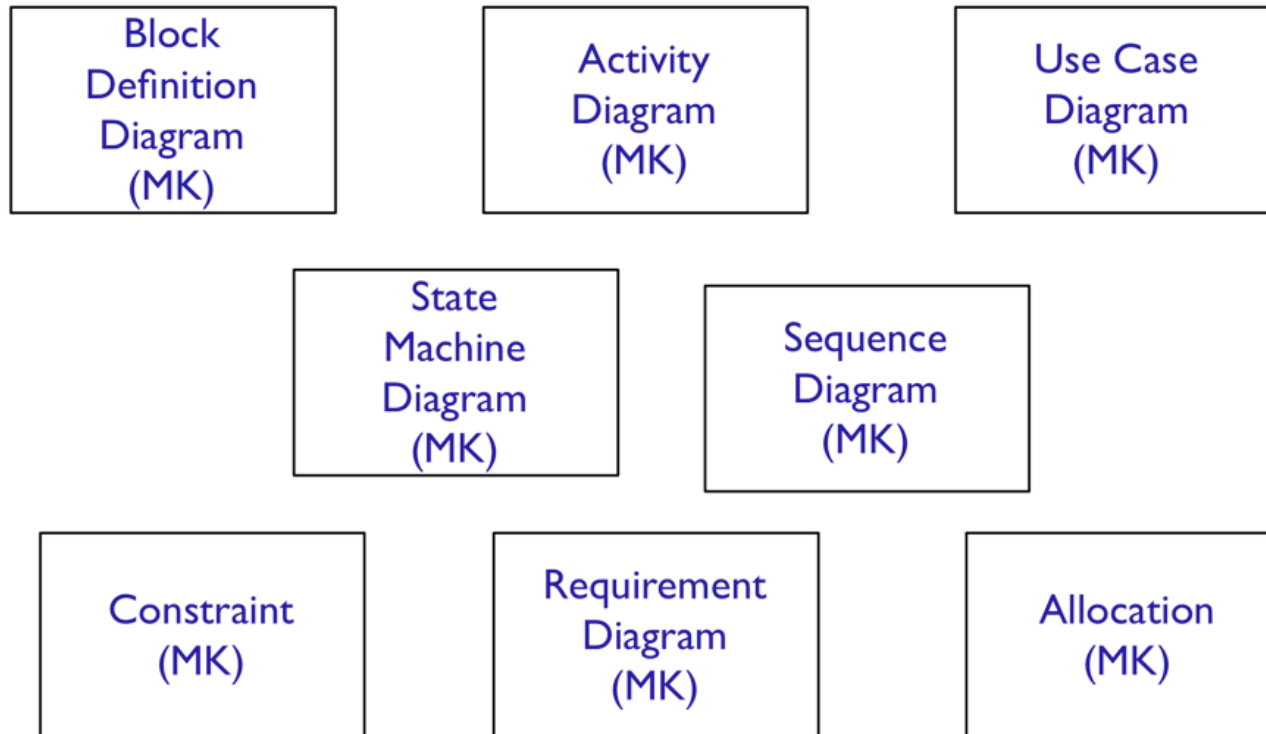
# Cross-cutting Perspectives: Availability and Resilience



N. Rozanski and E. Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. 2nd edition. Addison Wesley, 2011.

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>

# *SysML: as an architecture description language*



OMG System Modeling Language (SysML), formal/2010-06-01

Copyright © 1998–2014 by Rich Hilliard :: <http://www.iso-architecture.org/42010/>