

Using the UML for Architectural Description

Rich Hilliard
rh@isis2000.com

Outline

- **What is IEEE P1471?**
- **The IEEE P1471 Conceptual Framework**
- **Requirements on Architectural Descriptions**
- **Using the UML in the Context of P1471**
- **Wrap Up**

What is IEEE P1471?

- **IEEE Draft *Recommended Practice for Architectural Description***
- **Begun in August 1995 by the IEEE Architecture Working Group**
 - 30+ participants
 - 150+ international reviewers
- **Recently completed balloting by IEEE**
 - to be issued by IEEE Computer Society

- <http://www.pithecanthropus.com/~awg>

IEEE Goals for P1471

- **Focus on the architecture of 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**

Ingredients of IEEE P1471

- **A normative set of definitions for terms like “architecture”, “architectural description”, “architectural view”**
- **A conceptual framework establishing these concepts in the context of use of architectural descriptions for system construction, analysis and evolution**
- **A set of requirements on an architectural description of a system**

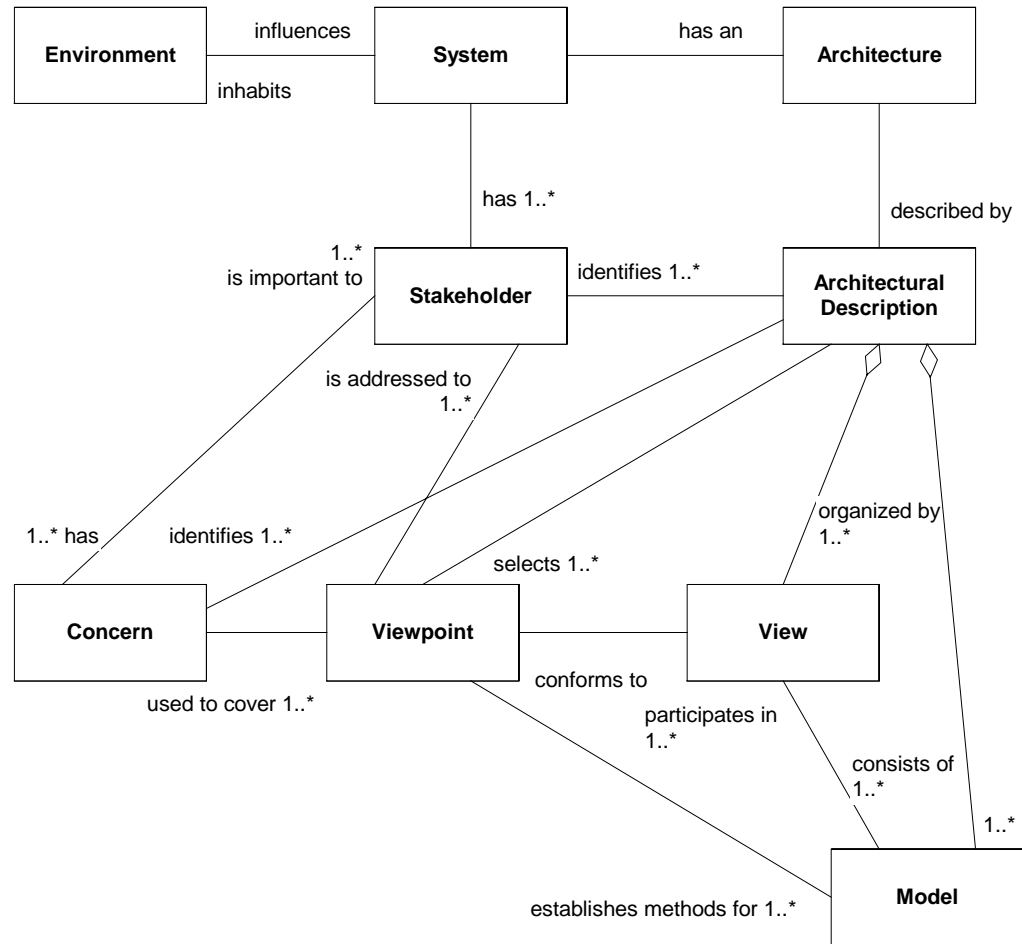
Using P1471

- **IEEE P1471 is a *recommended practice***
 - One kind of IEEE standard
- **P1471 applies to *Architectural Descriptions***
 - How to describe an architecture
 - *Not* a standard architecture, or architectural process, or method
- **P1471 is written in terms of “shall,” “should” and “may”**
 - ADs may be checked for conformance to the recommended practice
 - P1471 does not define any conformance of systems, projects, organizations, processes, methods, or tools

What is an Architectural Description?

- In P1471, an *architectural description* is a collection of products to document an architecture
- P1471 does not specify the format or media for an architectural description
 - Notation-independent
- P1471 *does* specify certain (minimal) required content of an AD reflecting current practices and consensus

The P1471 Conceptual Framework (excerpt)



P1471 Requirements: 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 stakeholders' concerns

P1471 Requirements: Views

- ***Multiple views:***
 - An AD consists of one or more views
 - A *view* is a representation of a whole system from the perspective of a set of concerns
- ***Views are themselves modular:***
 - A view may contain one or more architectural models, allowing a view to utilize multiple notations
- ***Inter-view consistency:***
 - An AD documents any known inconsistencies among the views it contains

P1471 Requirements: Viewpoints

- ***Views are well-formed:***
 - Each view corresponds to exactly one viewpoint
 - A viewpoint is a pattern for constructing views
 - Viewpoints define the rules on views
- ***No fixed set of viewpoints:***
 - P1471 is “agnostic” about where viewpoints come from
- ***Concerns drive viewpoint selection:***
 - Each concern is addressed by an architectural view
- ***Viewpoints are first-class:***
 - Each viewpoint used in an AD is “declared” before use

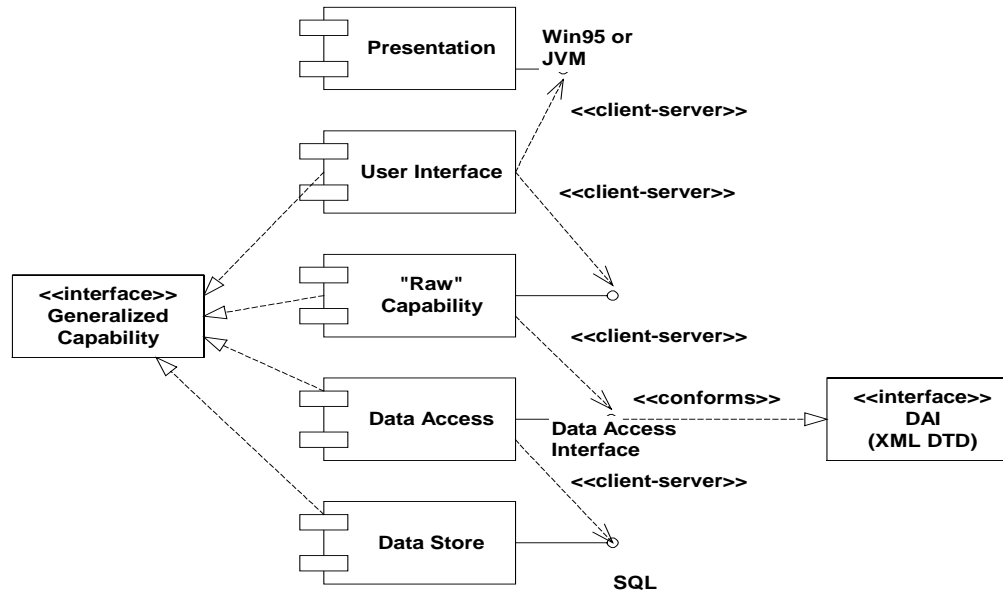
Declaring a Viewpoint

- ***Each viewpoint is specified by:***
 - Viewpoint name
 - The stakeholders addressed by the viewpoint
 - The stakeholder concerns to be addressed by the viewpoint
 - The viewpoint language, modeling techniques, or analytical methods used
 - The source, if any, of the viewpoint (e.g., author, literature citation)
- ***A viewpoint may also include:***
 - Any consistency or completeness checks associated with the underlying method to be applied to models within the view
 - Any evaluation or analysis techniques to be applied to models within the view
 - Any heuristics, patterns, or other guidelines which aid in the synthesis of an associated view or its models

A Viewpoint Example

- **Viewpoint name: Capability**
- **Stakeholders:**
 - The client, producers, developers and integrators
- **Concerns:**
 - How is functionality packaged?
 - How is it fielded?
 - What interfaces are managed?
- **Viewpoint language**
 - Components and their dependencies (UML component diagrams)
 - Interfaces and their attributes (UML class diagrams)
- **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

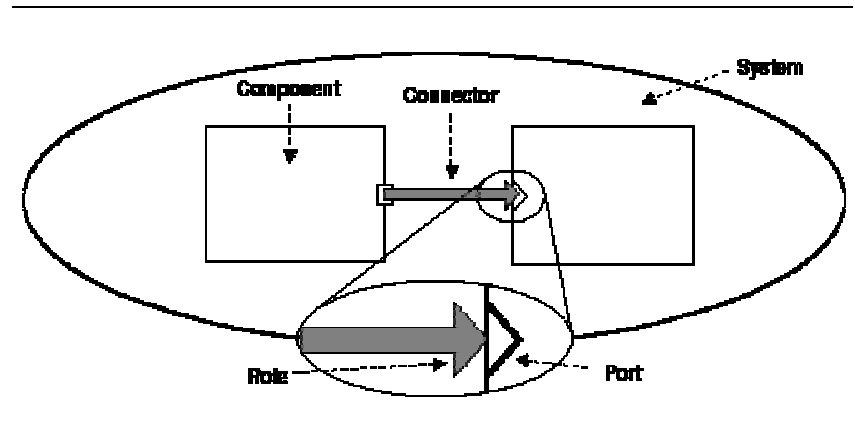
Library Viewpoints

- **Viewpoints are not specific to a system; unlike the stakeholders and views**
- **Hence, an architect ought to be able to reuse viewpoint declarations**
 - Equivalently, viewpoints can be included by reference
- **It would be nice to have a standard way to document viewpoint declarations in UML**

Example:

Viewpoint Name: Structural

- **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

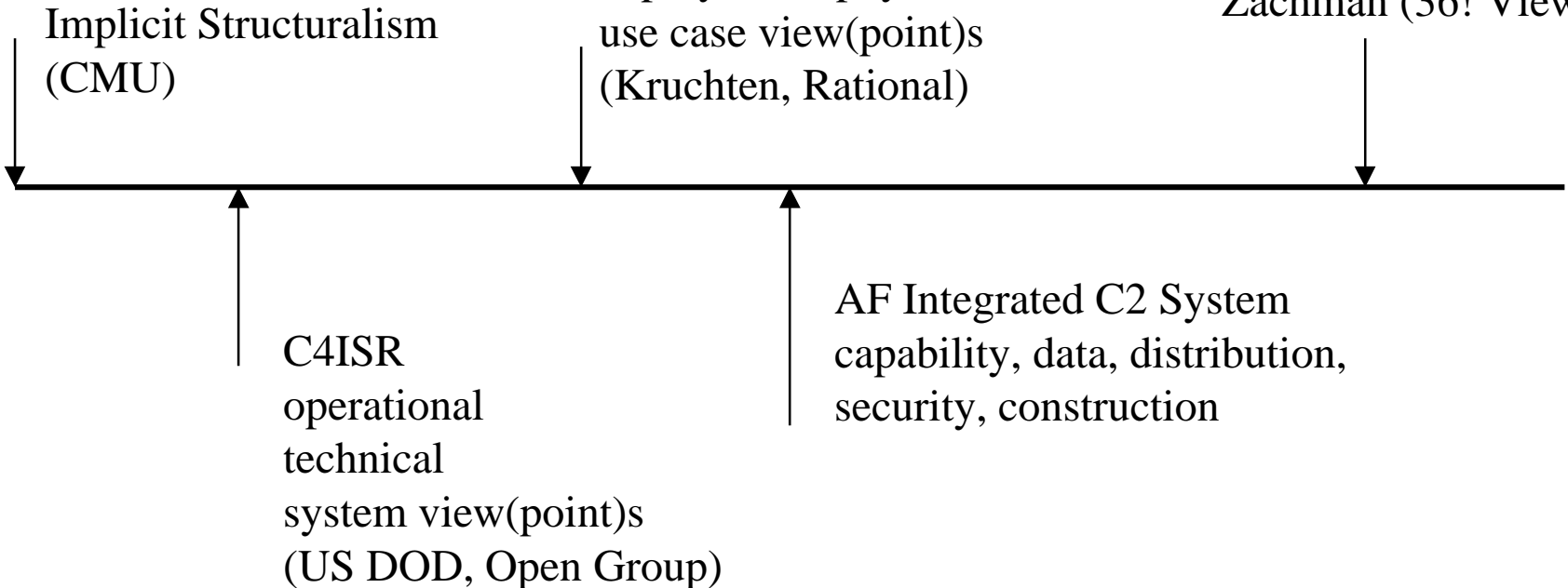


Based on: Acme: An Architecture Description Interchange Language, Garlan, Monroe, Wile, Proceedings of CASCON'97, November 1997

“Viewpoint Scale”: *P1471 is intended to encompass...*

4+1 View Model:
design [logical]
process
implementation [development]
deployment [physical]
use case view(point)s
(Kruchten, Rational)

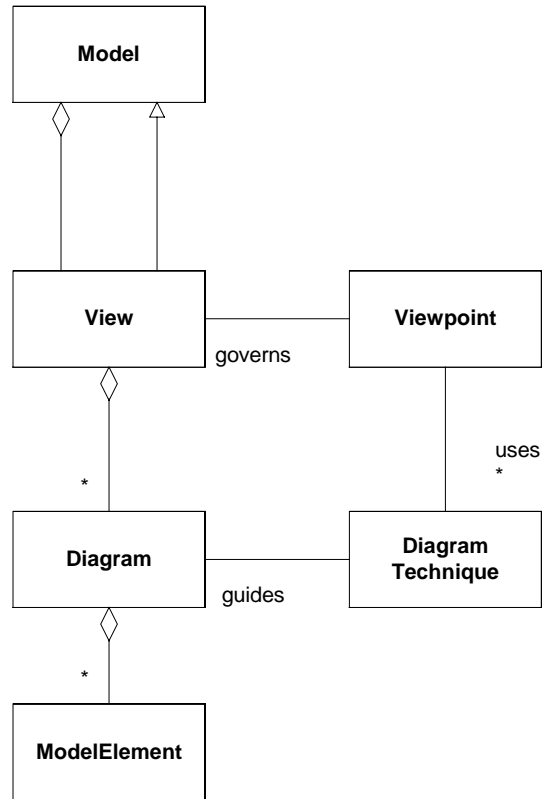
Zachman (36! Views)



Approaches to Using the UML in the Context of P1471

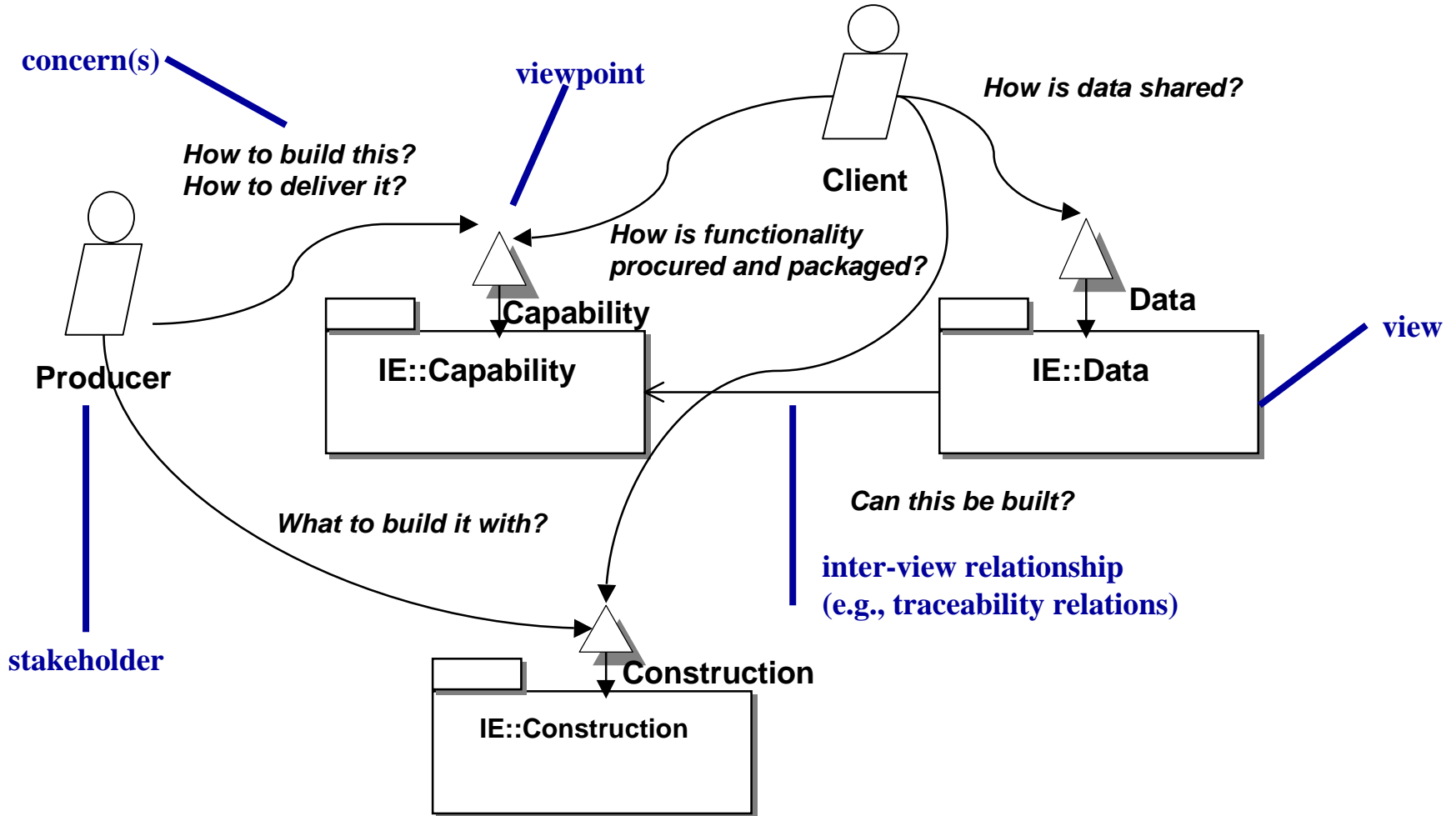
- **“Out of the Box”**
 - Pre-existing diagram types are all applicable to architectural description
- **Lightweight Extensions**
 - Tagged values, stereotypes, constraints are typically used “per viewpoint”
 - View and viewpoint notions are between extension and profile in their granularity
- **UML as Integration Framework**
 - View and viewpoint are nascent concepts in UML but not reflected in the metamodel
- **Outside the UML Ontology**
 - Some concerns architects worry about fall outside UML ontology
 - Security, Reliability, Commerce, Management, ...

Enhanced Metamodel



Wrap Up

Fragment of a Big Picture: Internet Engine



General Issues (Outside of just P1471)

- **Are components (e.g. ACME) UML components?**
- **Traceability mechanisms between views (within and between viewpoints)**
- **Variation á la product lines v. individual systems**
- **First-class constraints (rather than constraints as annotations) {see ISAW-3 paper}**
- **Finer control over things like Visibility, Inheritance**
 - the built-ins are fine for design, programming level
- **Declaring new diagram types**
- **Capture of fragments (i.e., patterns and styles)**