Using Aspects in Architectural Description

Rich Hilliard
r.hilliard@computer.org

13 March 2007

(The 10 minute remix)

What the Viewer brings to the Viewed in the Viewing, yields the View.

— Douglas T. Ross

Creator of plex and SADT

(21 December 1929 – 31 January 2007)

Questions

- Is the aspects metaphor of any use in the Architectural Description of software-intensive systems?
- How might this differ from its use in programming?
- Should aspects be supported with(in) existing architectural standards and practices?

Architecting vs. Programming

- Architects routinely practice sophisticated separation of concerns
- Non-functional concerns prevail in architecting. (Functionality is easy)
- Current architecting practices are based on multiple viewpoints

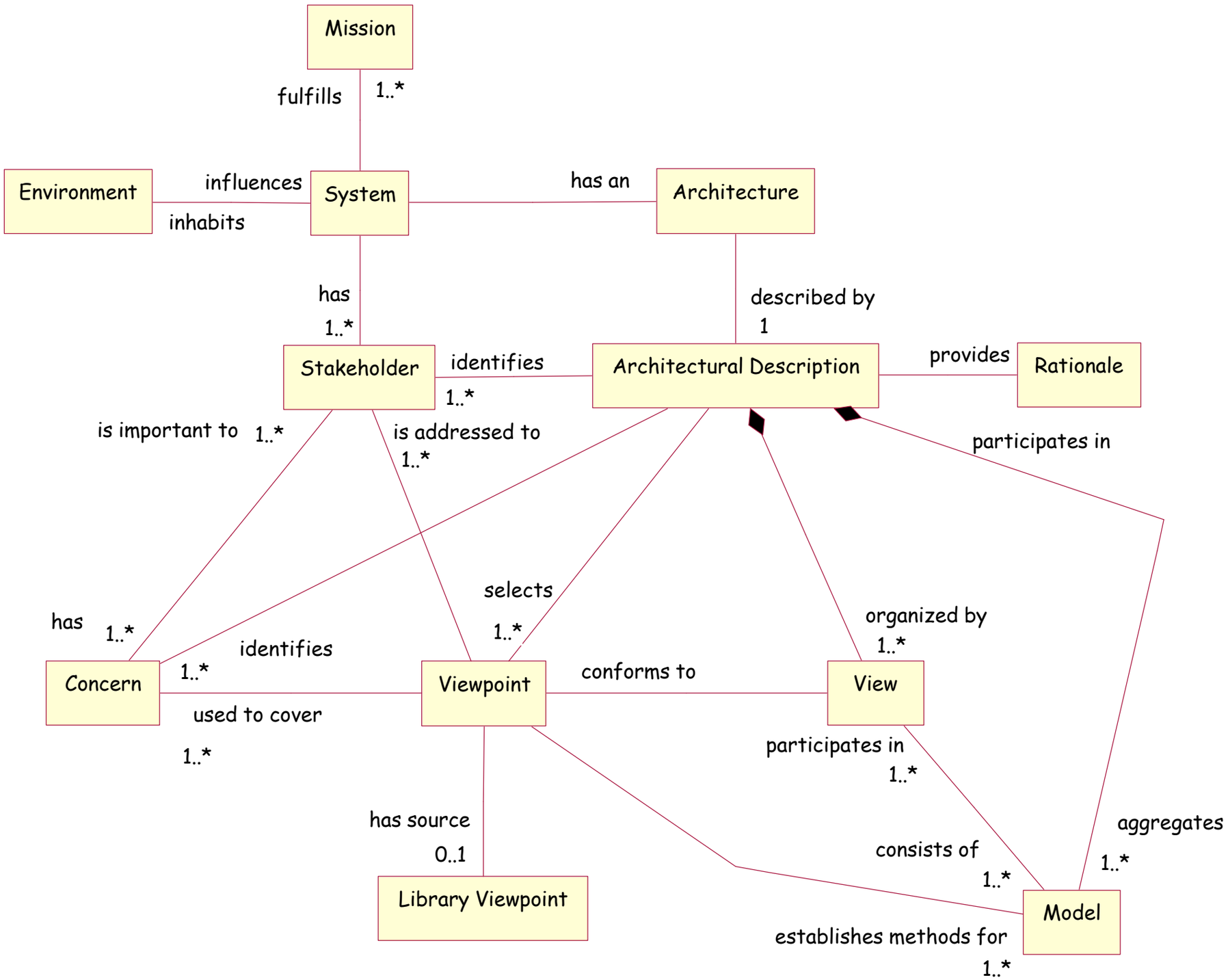
  ∴ no dominant decomposition in a single language

- Is there a role for aspects in this setting?

Context

- Focus on one portion of architectural practices: **Architectural Description**
- Using the conceptual framework of IEEE Std 1471 (2000), *Recommended Practice for Architectural Description of Software-Intensive Systems*
  - Currently undergoing joint revision with ISO as ISO/IEC 42010

IEEE 1471 Conceptual Framework



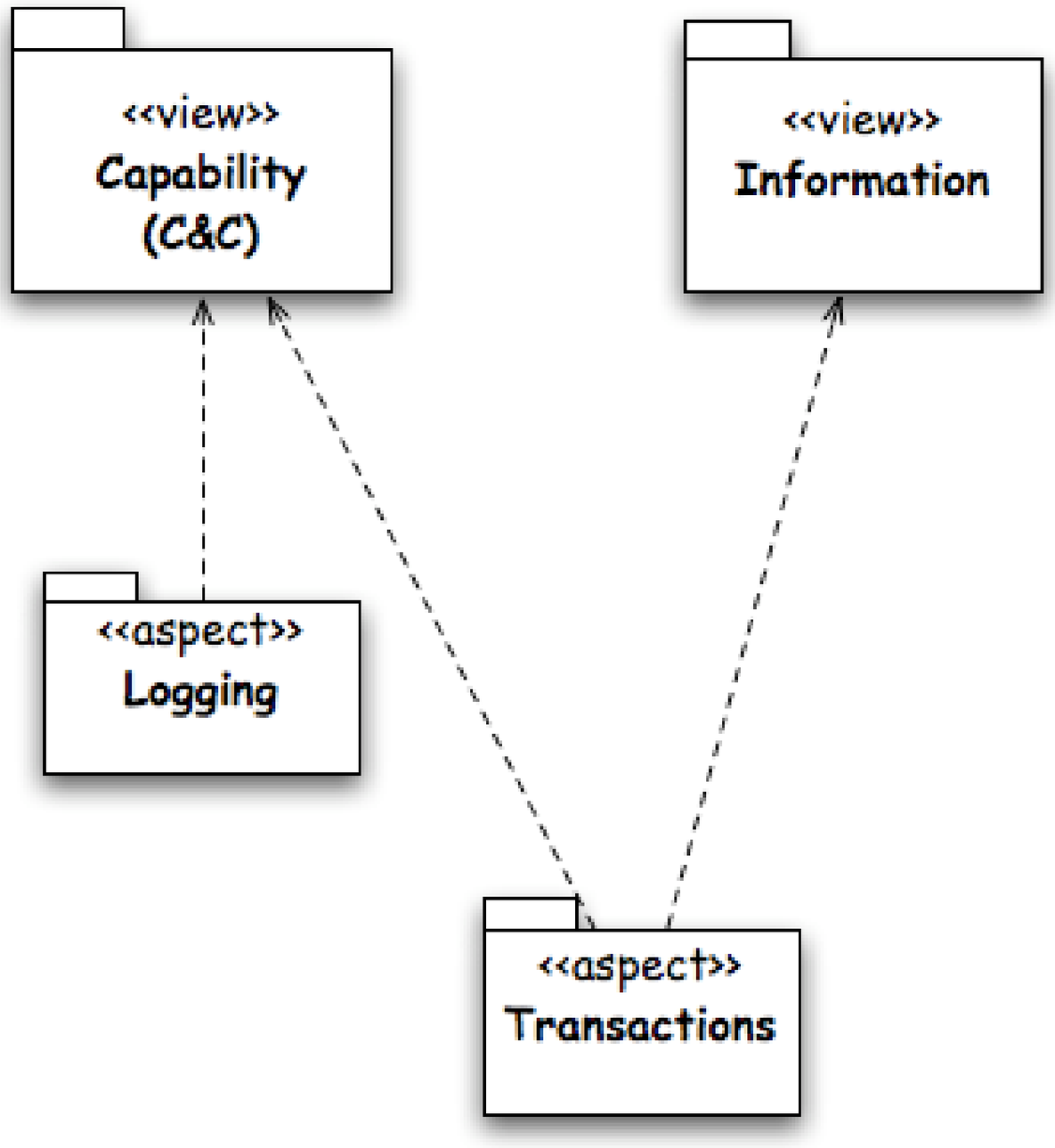
Architectural Models

- A view consists of one or more architectural models
- Architectural models are used to:
  - facilitate the use of multiple notations (viewpoint languages) within a view
    - e.g. Kruchten's 4+1 Logical Viewpoint uses both UML class diagrams and component diagrams
  - modularize architectural details which apply to more than one view
    - a model may be shared among views
- Viewpoints establish what models may appear in associated view

Using Aspects in AD

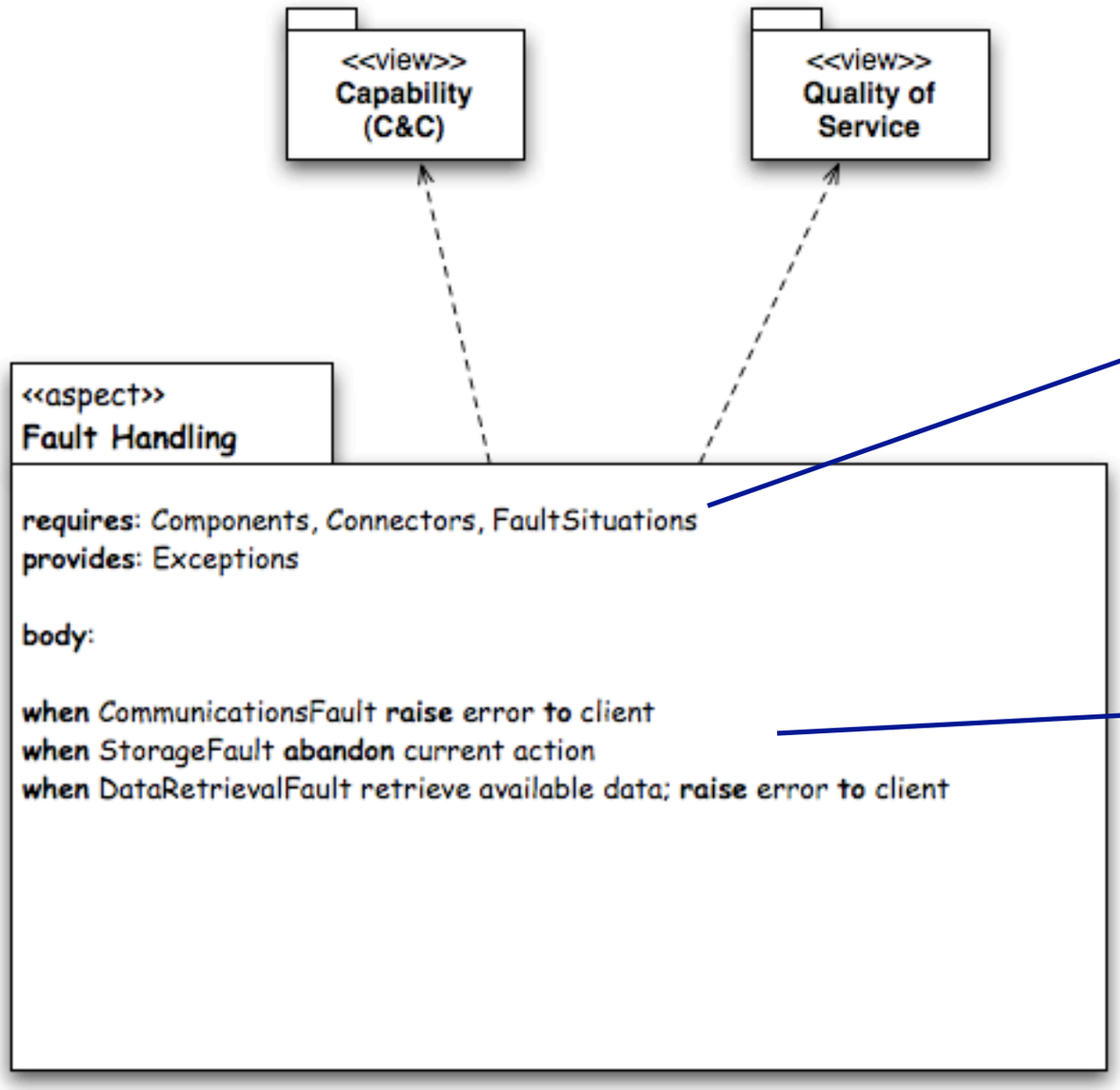
- Definition
  - An **architectural aspect** is a shared architectural model addressing exactly one architectural concern
- Consequences of Definition
  - architectural aspects cross-cut architectural views (and their models)
  - an aspect is the finest-grained solution element expressible

Further Consequences of the Definition



- There are two kinds of aspects: **intra-view aspects** (shared within a view) and **cross-view aspects** (shared among views)
- When a model is not shared (i.e, applies in only one place) it is not an aspect—just a constituent of a view
- Asymmetric: base remains the views and viewpoint languages

Cartoon Example



Refers to elements from "base" viewpoint languages —join points

Uses elements of "aspect" viewpoint languages

Fixing Architectural Models

- Models minimally specified in IEEE 1471:2000, no rules applying to them
  - Can we do better?
- Define model ownership, or separate definitions from viewpoints
- Enhancing models with provides and requires clauses gives us
  - Join points
  - signatures: model types
  - open question what kind of quantification is needed. type? instance? Hypothesis: “style” in most ADs is universally quantified statements over types
- Basis for view integration (which was left open in IEEE 1471:2000)

Related Work: several themes

- Component & connector-based, single viewpoint (i.e., “ADLs”)
  - aspects cut across components and connectors (view elements), not across views; e.g., AspectualACME, TranSAT, FuseJ, Fractal Aspect Component, ...Special mention: Katara-Katz
- Concern-oriented: the future is better modeling of concerns (Kandé, Sutton-Tarr)
- Existing cross-cutting architectural constructs, nicely align with cross-view and intra-view aspects, respectively
  - Rozanski & Woods’ **architectural perspectives**
  - Ran’s **architectural textures**

Workshop on Aspects in Architectural Description
http://aosd.net/workshops/aarch/2007/

Future Work

- Refine an approach to making the provides/requires language(s) more rigorous in the face of an open set of viewpoint languages
- Community example using the Health-Watcher case study