Surveying the Twin Peaks

Rich Hilliard

http://softsysarchitect.net
richh@mit.edu

Surveying is essential “in the planning and execution of nearly every form of construction”*

* Wikipedia

I prefer surveying for a week to spending a week in fashionable society even of the best class.
— Ellen Henrietta Swallow Richards

Preliminaries

- Invited talks are intended as provocation
- These are thoughts-in-progress*
- The usual talk is "functionally decomposed"

  * e.g., background → problem → solution

* See also my: In search of the Higgs, or What's wrong with SEMAT?

My Prejudices

- Skeptical of "process"
- "NFRs" are dysfunctional
- Architecture is architecture
- Requirements, Architecture, ... are interest-relative

Process skeptic

- Method follows concept
- Tools follow from method
- Process doesn’t follow

NFRs are dysfunctional

- "NFR" is a non-category*
- Usually treated as after-thought

* *Framing Stakeholders' Concerns*, Guest editors' introduction, *IEEE Software*. Nov-Dec 2010

Architecture is Architecture*

- Enterprise = System = Software

* Architecture Description in-the-large (WICSA 2009)

Interest-relativity

- Multiple stakeholders
- Diverse interests
- I'm instantly skeptical of approaches that do not take diverse stakeholders and varying interests into consideration

(My prejudices will come back to haunt us for the rest of the talk.)

Why do they intertwine?

Twin Peaks, San Francisco. elevation 922′ (281 m)

Why do they intertwine?

Twin Peaks, San Francisco. elevation 922′ (281 m)

Why do they intertwine?

- "*Concerns* are what we care about in software."*
- Dijkstra's phrase, "separation of *concerns*"

* S.M. Sutton Jr. and I. Rouvellou, Concern Space Modeling in COSMOS, OOPSLA 2001

*"separation of concerns"**

Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day, so to speak. In another mood we may ask ourselves whether, and if so: why, the program is desirable. But nothing is gained—on the contrary!—by tackling these various aspects simultaneously. It is what I sometimes have called **"the separation of concerns"**, which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. This is what I mean by "focussing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.

* E.W. Dijkstra, On the role of scientific thought, 1974

A brief history of concerns

Early Days

algorithm + data = program

Dijkstra
"separation of concerns"
1974

Ross *et al.*, Software engineering: process, principles, and goals

Software Qualities and Specialties

*Rise of "process"**

1980s

Era of (almost) 1st-class concerns

1992
ViewPoints
(Nusibeh, Finkelstein *et al.*)

1995
architecture views: structure and behavior, 4+1, etc.

IEEE 1471:2000

1996
Aspect-oriented programming

1999
Multi-Dimensional Separation of Concerns

* the Lost Decade?

Concerns in Architecture Description (AD)

IEEE 1471 asked the question,
Where do views come from?

and suggested this answer,
Views address the concerns of the stakeholders

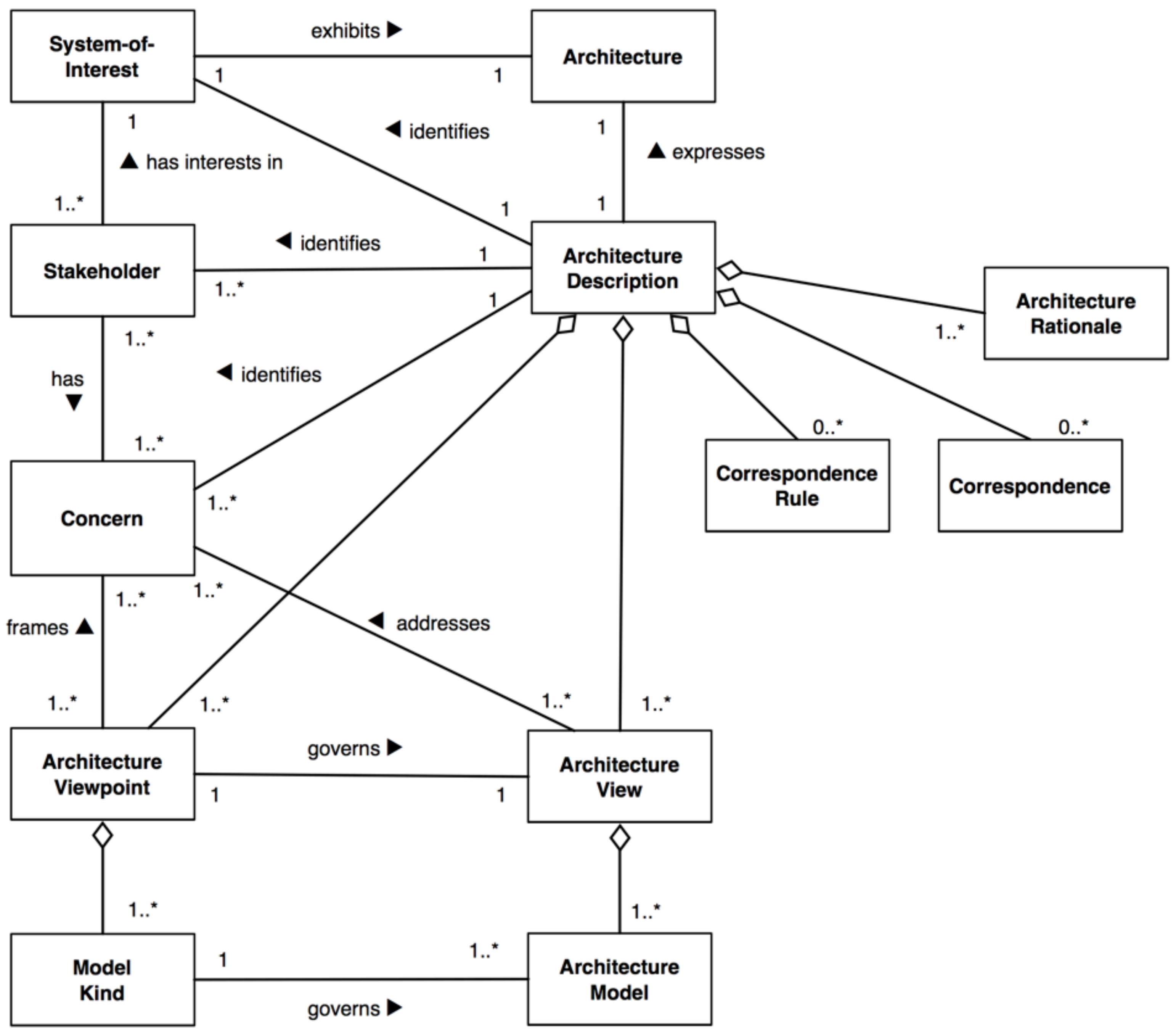
* IEEE 1471:2000, *Recommended Practice for Architectural Description of Software-Intensive Systems*

concern, defined*

- ***concern***: interest in a system relevant to one or more of its stakeholders
  A concern pertains to any influence on a system in its environment; including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences

* ISO/IEC/IEEE 42010:2011, *System and software engineering — Architecture description*

Concerns in AD



* ISO/IEC/IEEE 42010:2011, *System and software engineering — Architecture description*

Role of Concerns

- name subjects, (i.e., “areas of interest”)
- stakeholders and concerns reify “the environment”
- establish ‘minimum’ requirements on models (“representation schemes”)
- select viewpoints, and then checked in views
- use in decision recording

Examples

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

What are Concerns (any) good for?

- Original IEEE 1471 case:
  Is view V relevant to stakeholder S?
- Traceability:
  Should these elements be related?
- Trade-offs:
  Does decision A affect decision B?

Concerns help to make the implicit (into the) explicit

Summiting the Twin Peaks

- Can we do better?
- Concerns should be modeled and managed as first-class entities
- Our current process models do not support this

Misunderstanding Concerns

- “quality concerns”
- “risks”
- “functions”
- “pervasive cross-cutting concerns”

Empty process models

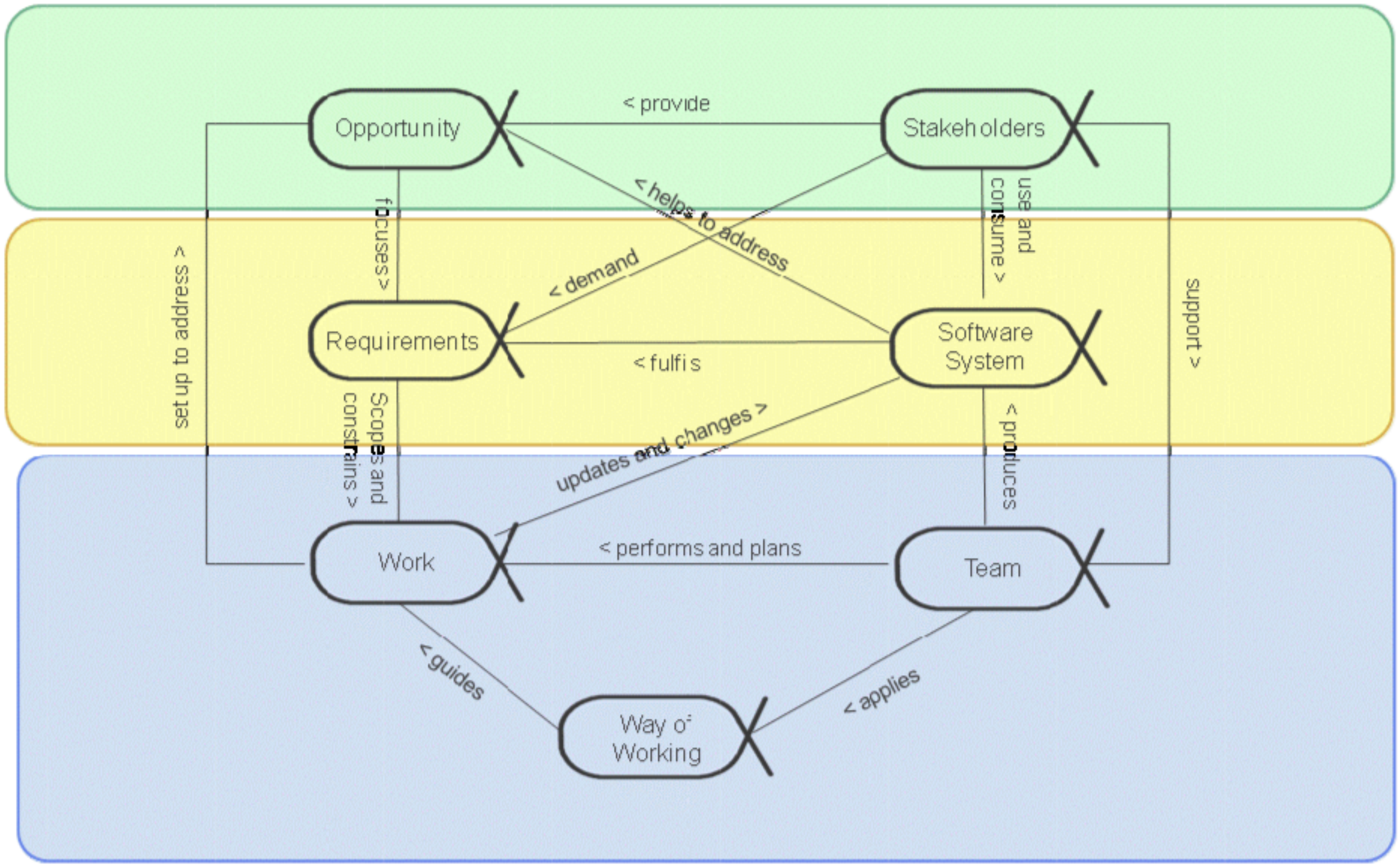


Figure 3 – The Kernel Alphas

Essence, Version 1.0

Empty process models

- What links a Requirement on this System with a Work Item? with a Team?
- Does the Team have the right skills to successfully deliver this Work Item?
- Is this Work Item affected by a change to this Requirement?
- What risks might befall this System? Are they being mitigated, managed and solved?

Concerns *bind*

- Concerns underwrite the reasons for work (processes and tasks):
  - e.g., We perform this work item because it yields an understanding of system deployment
- Concerns give work products their meanings:
  - e.g., This work product explains how reliability is managed during system development
- Concerns are the things we are interested in; they bind together processes, artifacts, people, in terms of their relevance

Perhaps "semantic traceability" is a better phrase?

Concerning Concerns

- Atomic?
- Closed set?
- “Relatable”?

Precursors

- Goal-oriented requirements
- Patterns and tactics
- Break down problem | solution "peaks" into manageable-sized piece
- Suggest ways of refining, relating atoms

Do Concerns help?

- What software architectures (or architectural styles) are stable in the presence of changing requirements, and how do we select them?
- What classes of requirements are more stable than others, and how do we identify them?
- What kinds of changes are systems likely to experience in their lifetime, and how do we manage requirements and architectures (and their development processes) in order to minimize the impact of these changes?

* From the original paper.