The Open University

Crosscutting Requirements

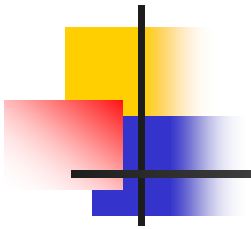
Bashar Nuseibeh

The Open University, UK

B.A.Nuseibeh@open.ac.uk

http://mcs.open.ac.uk/ban25/

And thanks to colleagues Charles Haley, Michael Jackson, and Robin Laney



The Bottom Line

"If you build software without [requirements and] specifications, it can never be incorrect – it can only be surprising."

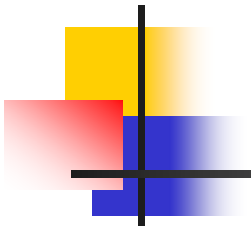
B. Kernighan

Warning: this talk contains no explicit mention of aspect-oriented programs, which some members of the audience may find disturbing. Viewer discretion is advised.

The "voice of the customer"

How I will spend the next hour

- General remarks about requirements engineering (RE)
- General remarks about "early aspects"
- Reflections in Viewpoints in RE
- Overlapping viewpoints and crosscutting requirements
- So, now what?

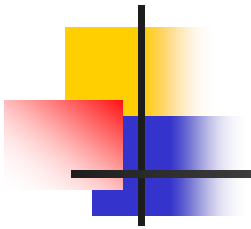


Managing your expectations

1. I will ask some questions to which I have no answer.

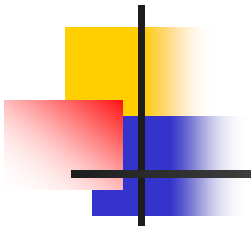
2. I will make some assertions with which you will inevitably disagree.

3. I will largely leave it to you to judge if and how what I say is useful for aspect-oriented development.



Some assumptions

- **Aspect = crosscutting concern (CC)**
- **Concern**
  - Is a property of interest to a stakeholder
- **Crosscutting**
  - Intertwining, Interdependent, Interacting, Overlapping
- **I make no assumptions about the nature of the implementation (aspect-oriented or otherwise).**



Where do CC's come from?

- **From the need to (re-)organise code**
  - To reduce tangling and scattering, and to promote maintainability as the software evolves
- **From the problem world**
  - Inhabited by stakeholders such as customers and users.
  - Fertile ground for identifying stakeholder concerns and exploring their interaction.

The problem world

- Is the world of requirements

- Requirements are:
  - expressions of stakeholder needs to achieve particular goals.
  - expressed in the vocabulary of the problem domain, rather than the solution domain.
  - They describe the world as we would like it to be.

Requirements Engineering (RE)

- **Is about:**

1. **Discovering** stakeholder goals, needs, and expectations
   - **Adjusting** stakeholder expectations

2. **Communicating** these to system implementers
   - **Adjusting** implementer expectations

Orientation

- Context and Groundwork
- **Eliciting** Requirements
- **Modelling** and **Analysing** Requirements
- **Communicating** Requirements
- **Agreeing** Requirements
- **Evolving** Requirements

Based on: B. Nuseibeh and S. Easterbrook, Requirements Engineering: A Roadmap, Proceedings of International Conference on Software Engineering (ICSE-2000), The Future of Software Engineering, A. Finkelstein (ed.), 4-11 June 2000, Limerick, Ireland, ACM Press.

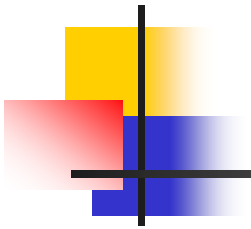
Orientation

- Context and Groundwork
- **Eliciting** Concerns
- **Modelling** and **Analysing** Concerns
- **Communicating** Concerns
- **Agreeing** Concerns
- **Evolving** Concerns

Based on: B. Nuseibeh and S. Easterbrook, Requirements Engineering: A Roadmap, Proceedings of International Conference on Software Engineering (ICSE-2000), The Future of Software Engineering, A. Finkelstein (ed.), 4-11 June 2000, Limerick, Ireland, ACM Press.

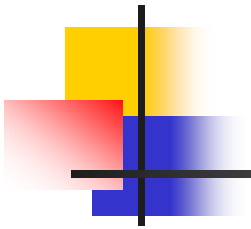
Some difficult questions

- What is a requirements engineer?
  - A software architect?
  - A systems engineer?
  - An anthropologist?
  - … ?
- Why is RE changing?
  - Refinement – not realistic?
  - Documentation – not necessary?
  - Time scales – too long?



"Early Aspects"

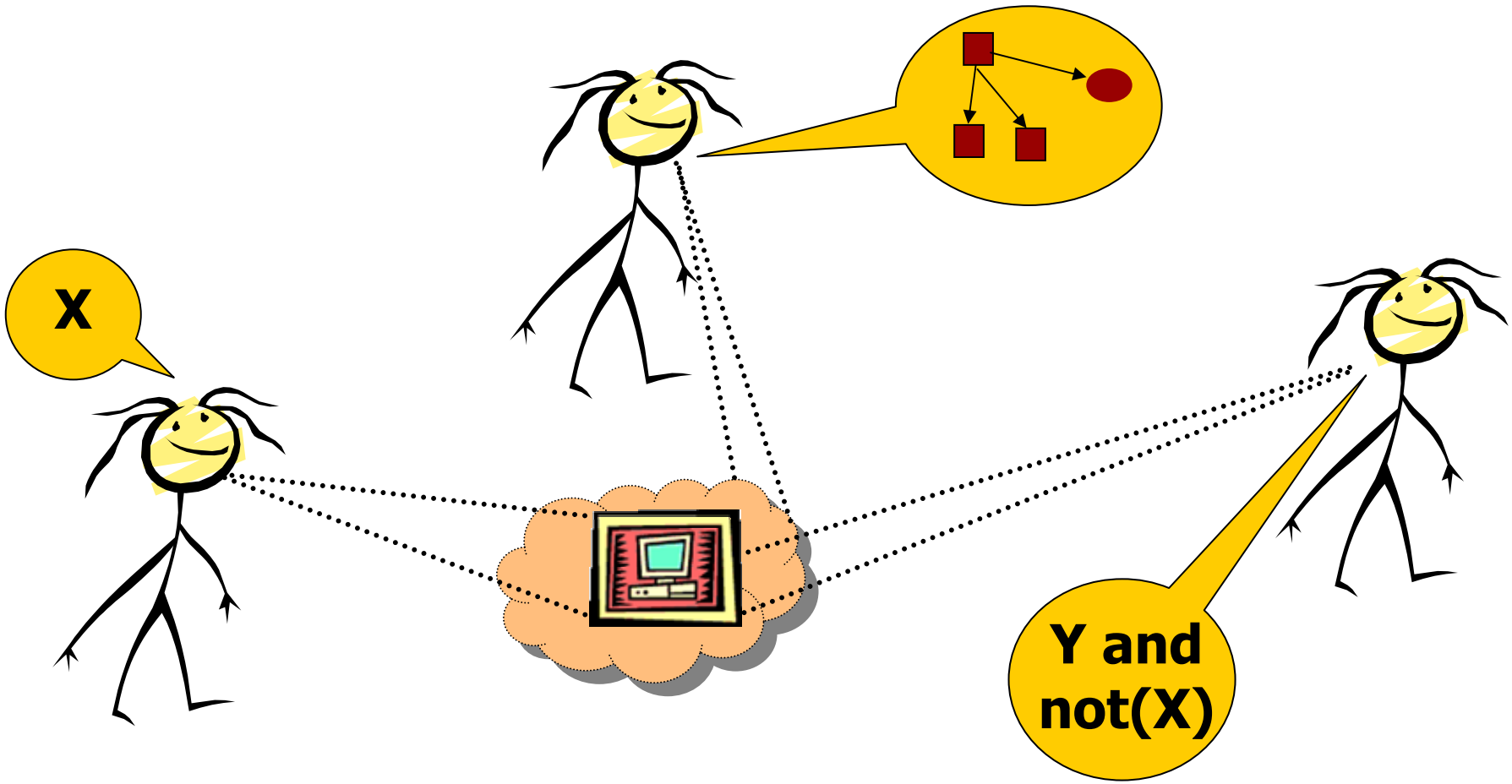
- **Are crosscutting concerns**
  - in requirements and design
  - that have a broadly-scoped effect on other requirements or architectural components.



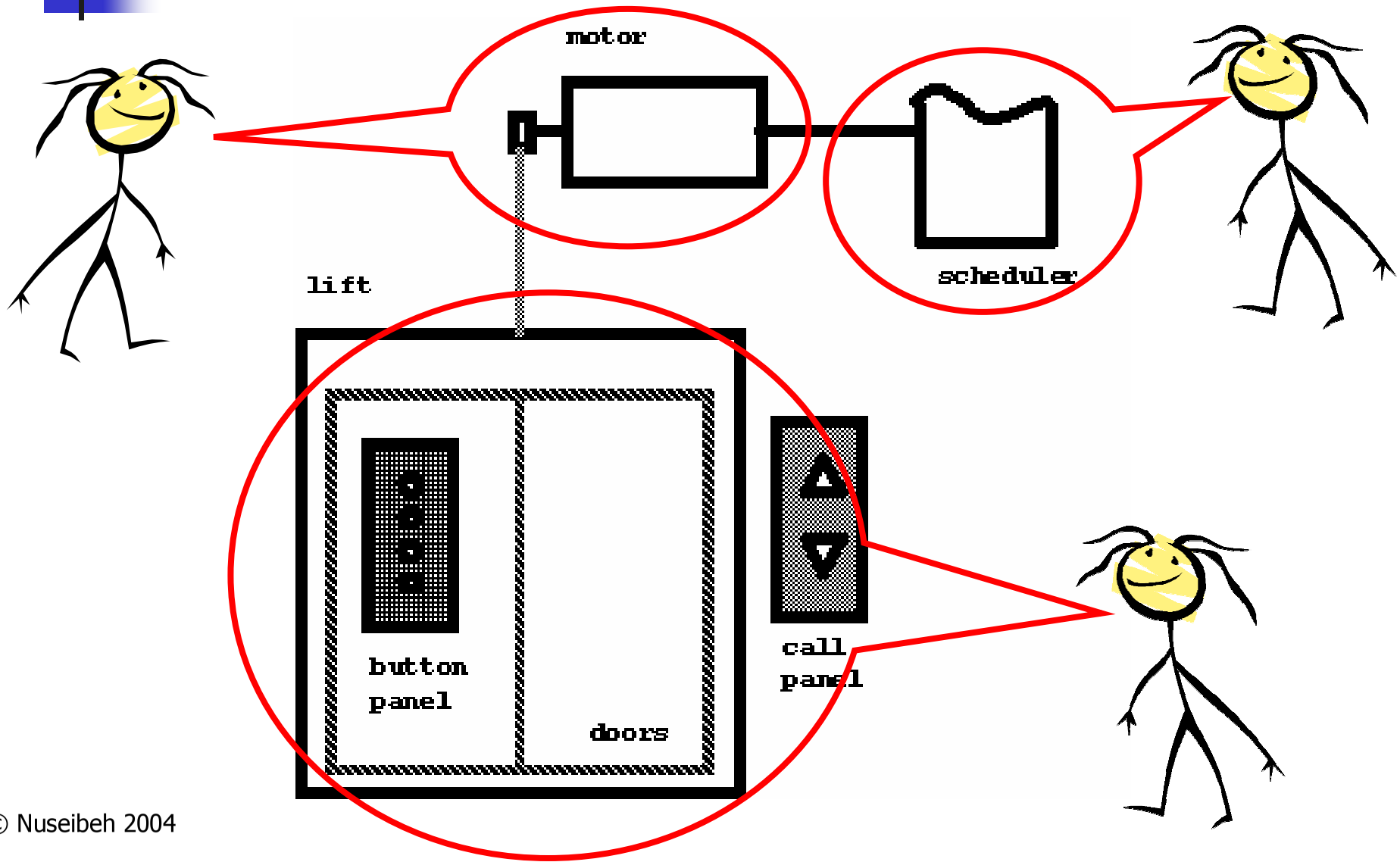
Concerns are in the eyes of the beholder

- **Concern**
  - Is a property of interest
  - to a stakeholder
- **Enter "viewpoints" …**

The multiple perspectives problem

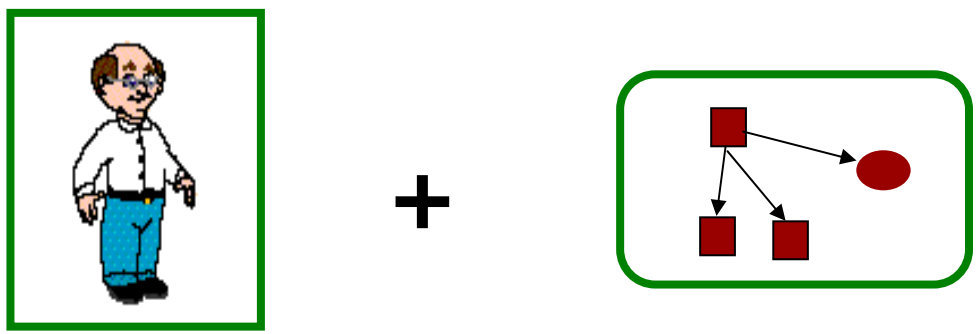


An old example



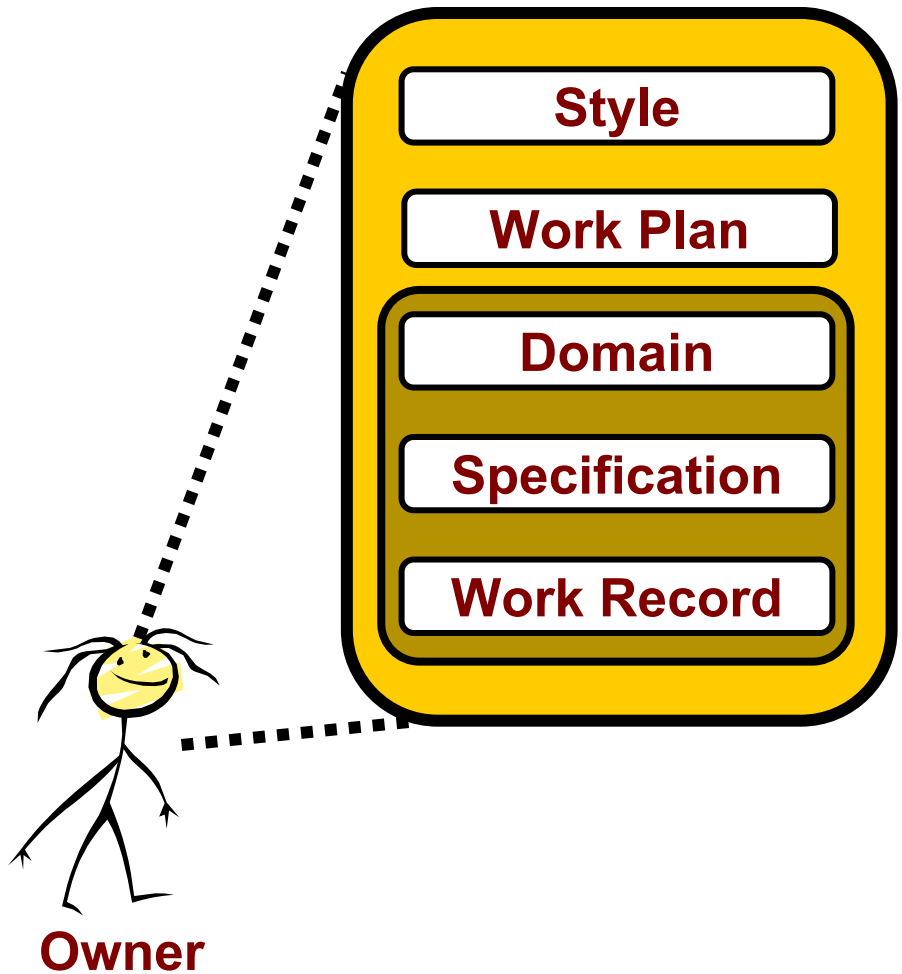
The ViewPoints™ framework

- **A framework for organising SE knowledge**
  - Collecting and partitioning knowledge about representations, processes, products of software development
  - With each ViewPoint combining notion of "**development participant**" with "**view**".

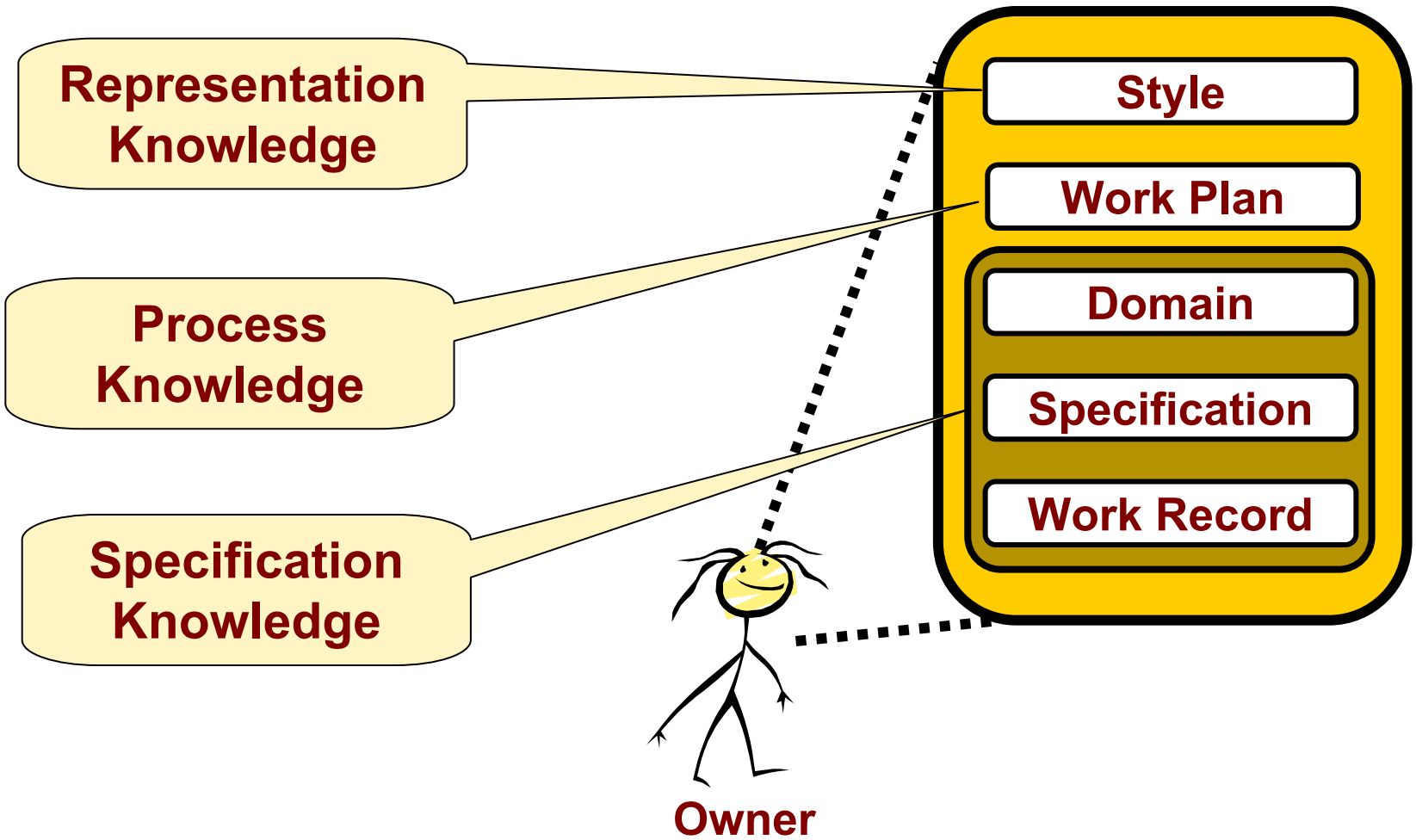


ViewPoints are …

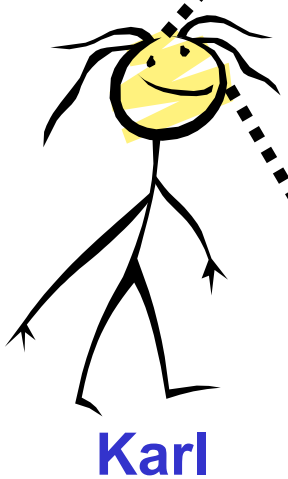
- Loosely coupled, locally managed, distributable objects;
- encapsulating cross-cutting and partial knowledge;
- about notation, process, and domain of discourse;
- from the perspective of a particular stakeholder, or group of stakeholders, in the development process.



ViewPoints are …



A Sample ViewPoint



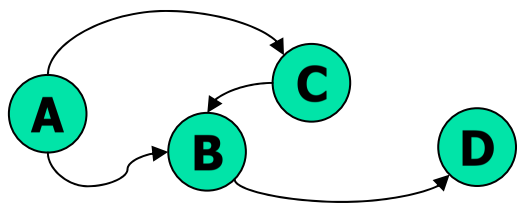
Karl

Style:
Labelled Transition System (LTS)

Work Plan:
Assembly actions (add-state, add-transition),
Check actions (no unconnected states),
Heuristics(no more than 20 states per LTS)

Domain: Bank Cash Withdrawal

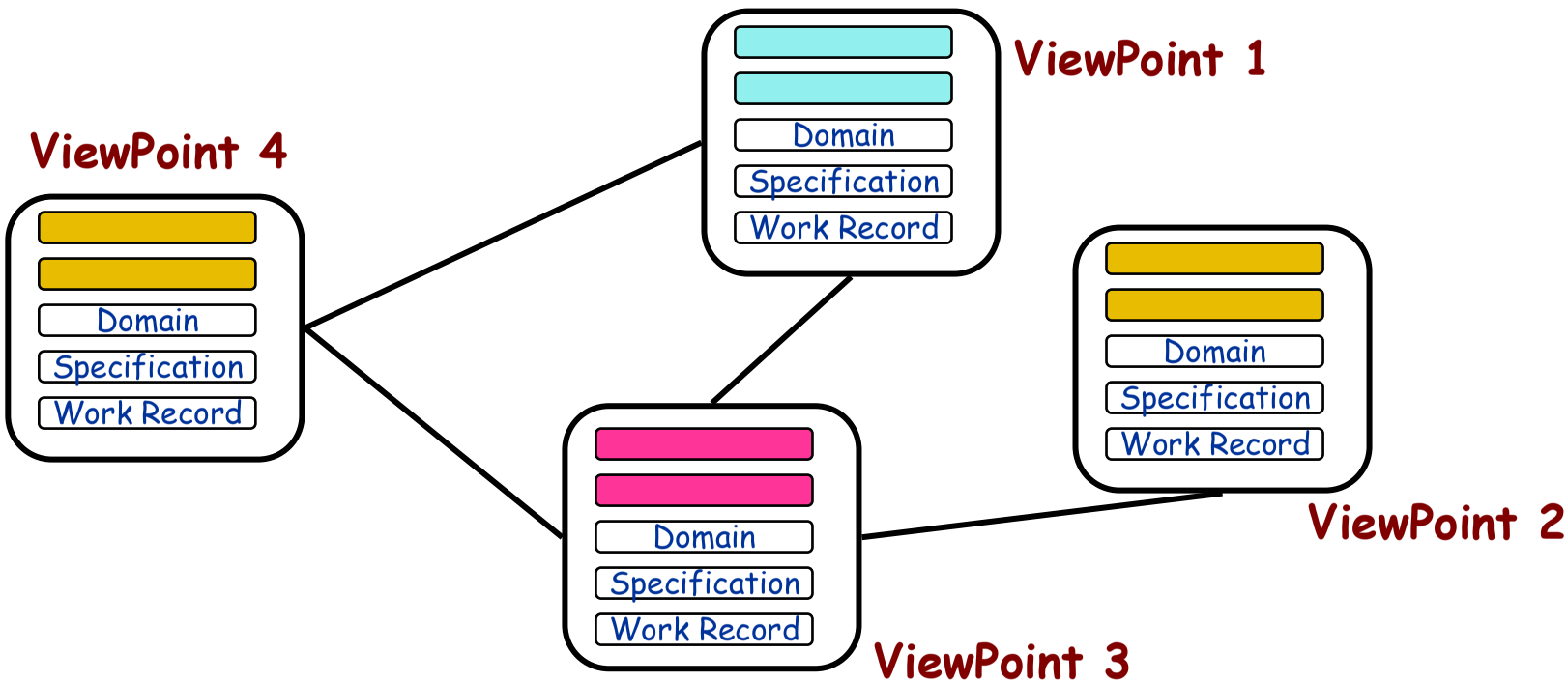
Specification:



Work Record:
Add-state(A), Add-state(B), Add-state(C),
Add-state(D), Add-transition(A,B),
Add-transition(A,C), Add-transition(C,B),
Add-transition(B,D)

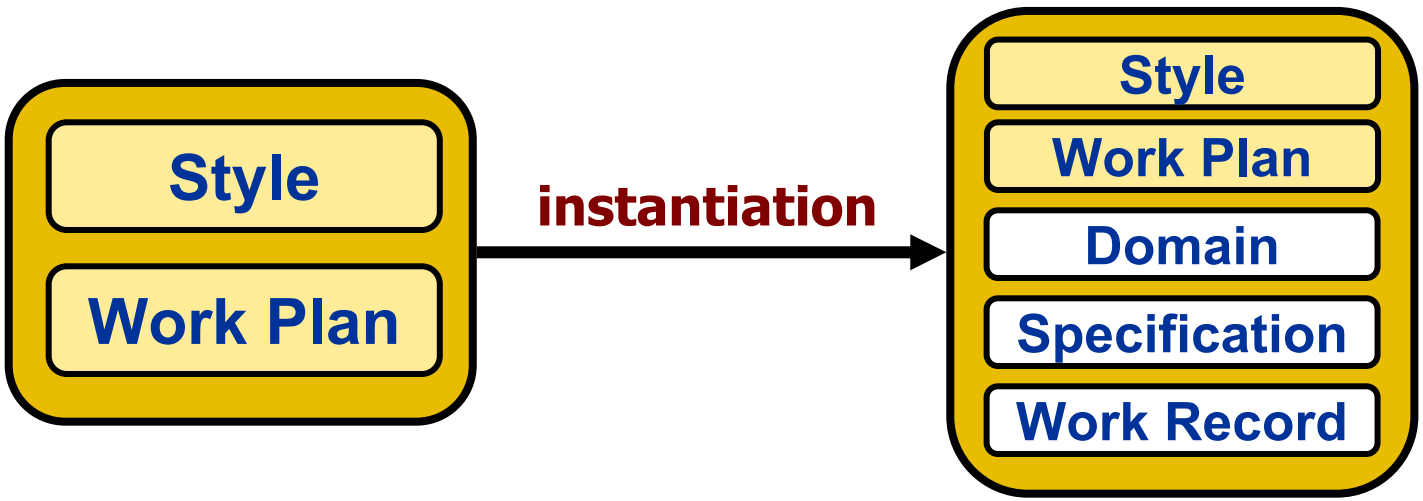
System Specifications with ViewPoints

- **A System Specification**
  - Is a **configuration of ViewPoints**;
  - that is, **a structured collection of related ViewPoints**



ViewPoint Templates

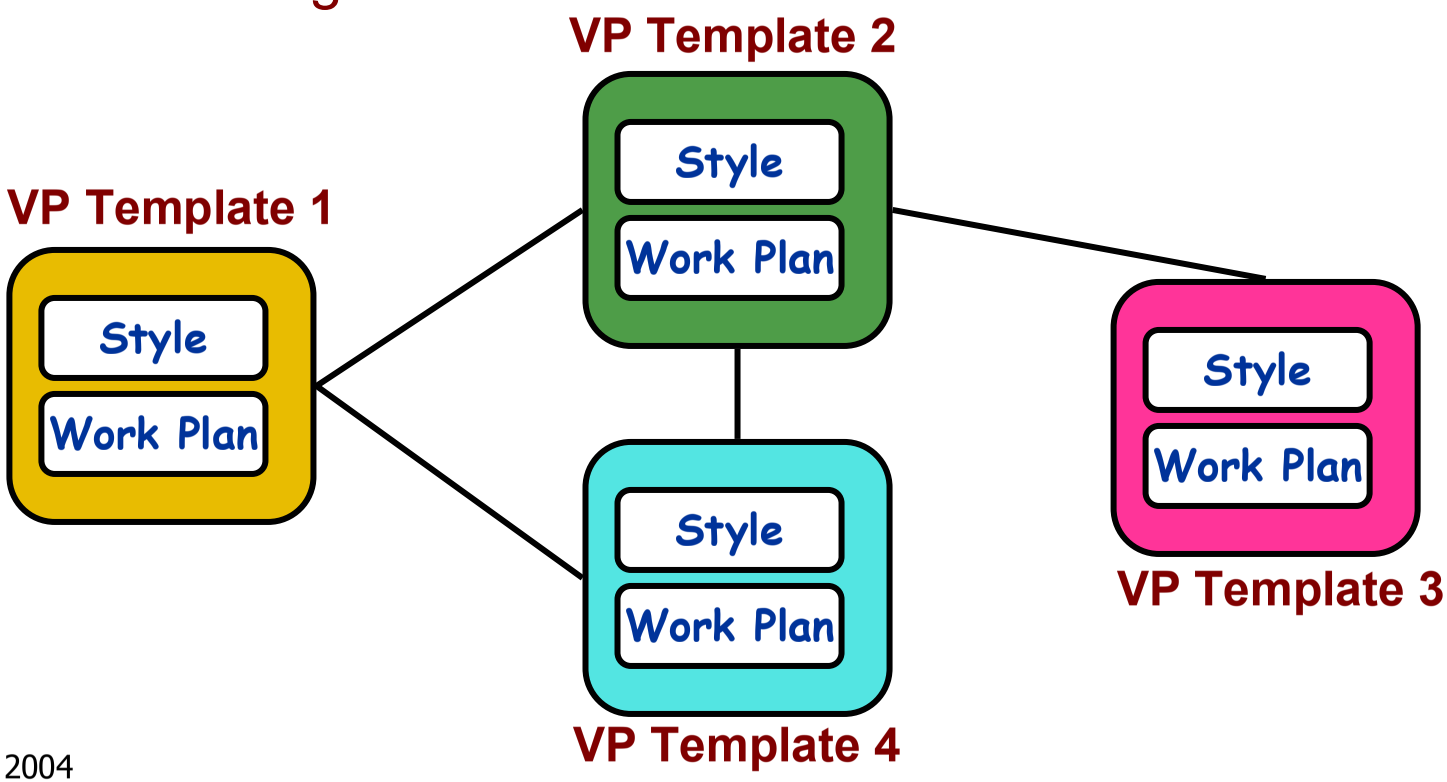
A ViewPoint Template is a ViewPoint type in which only the Style and Work Plan slots have been filled.



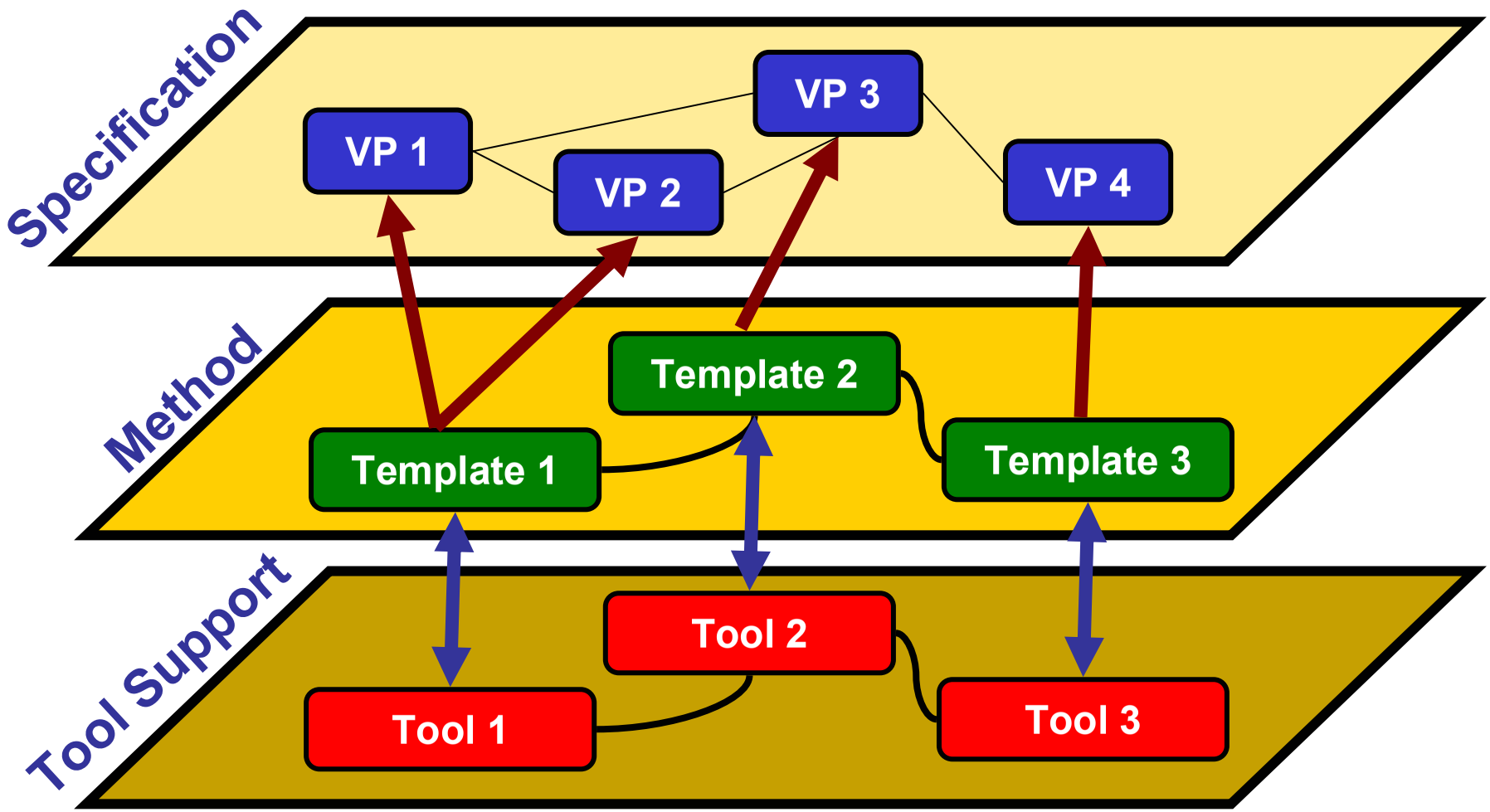
A ViewPoint Template may be "instantiated" to create a ViewPoint, in which the remaining three slots are filled

Method Engineering with VP Templates

- **Method = Configuration of ViewPoint Templates**
  - A method provides a structured set of templates designed to be used together



Tool Support for ViewPoints

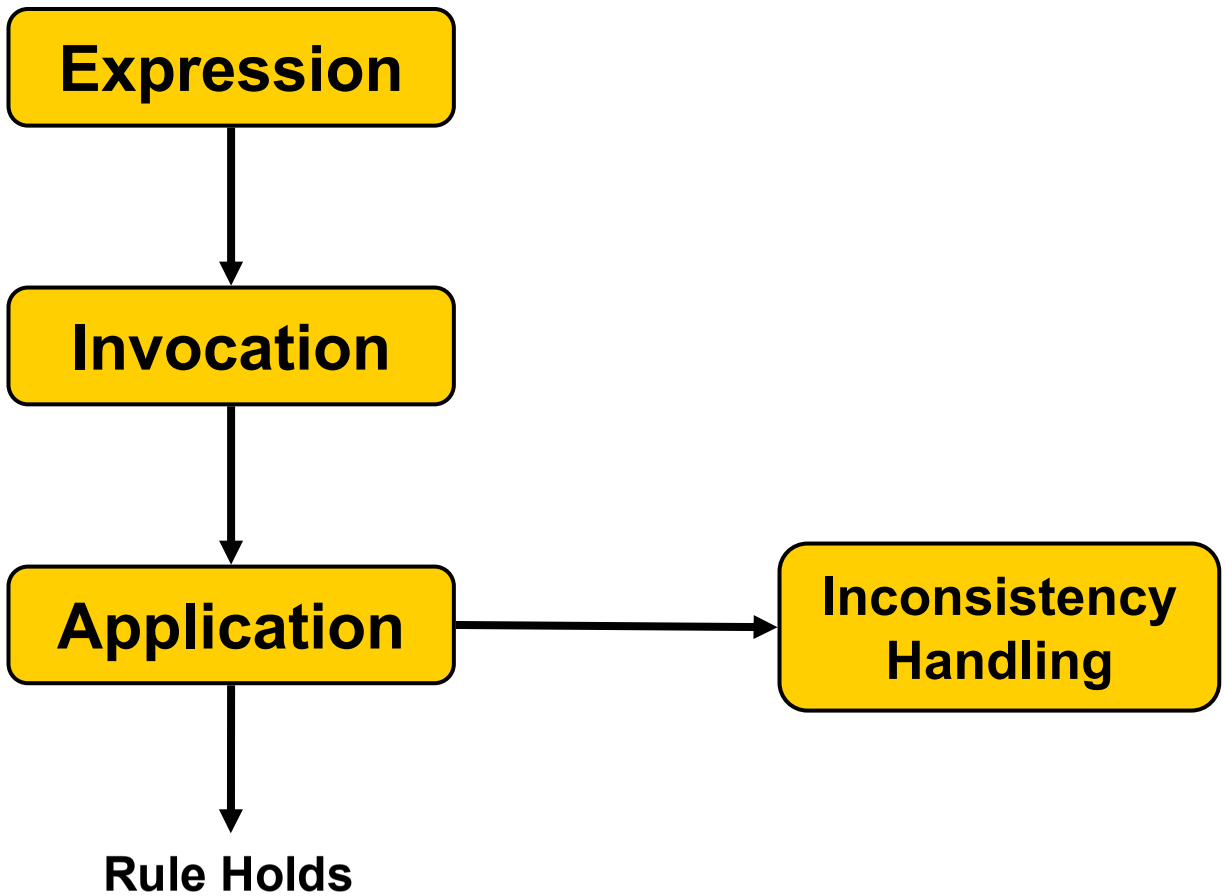


Relating ViewPoints

- **Inter-ViewPoint Relationships**
  - (Syntactic) descriptions of overlap between ViewPoints
  - The framework's integration glue

- **Inter-ViewPoint Rules**
  - Defined by method engineers and domain specialists
  - Checked by analysis and reasoning tools
    - Consistency means that the rules hold
    - Inconsistency means that the rules have been broken
  - Used to generate additional ViewPoints and/or information

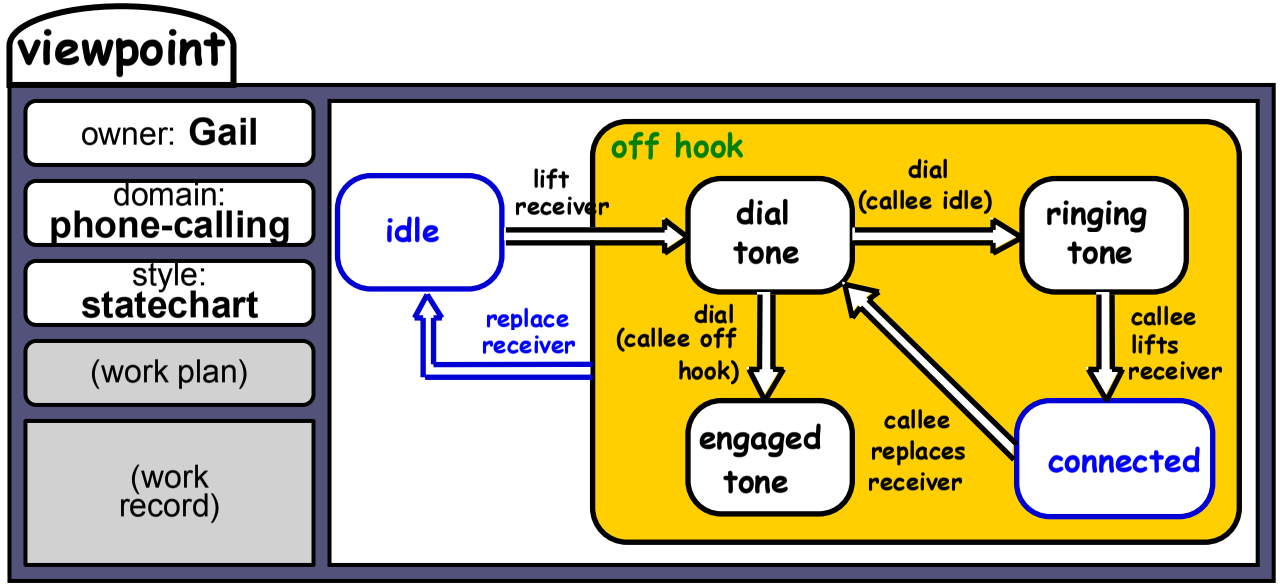
Managing Inter-ViewPoint Rules

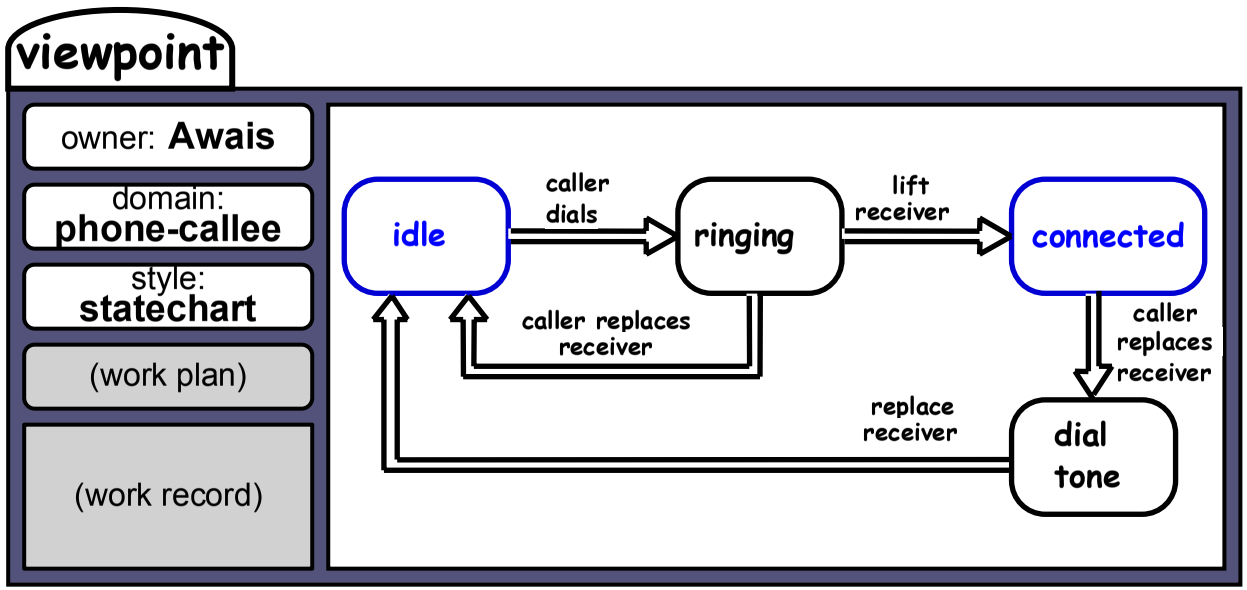


Example

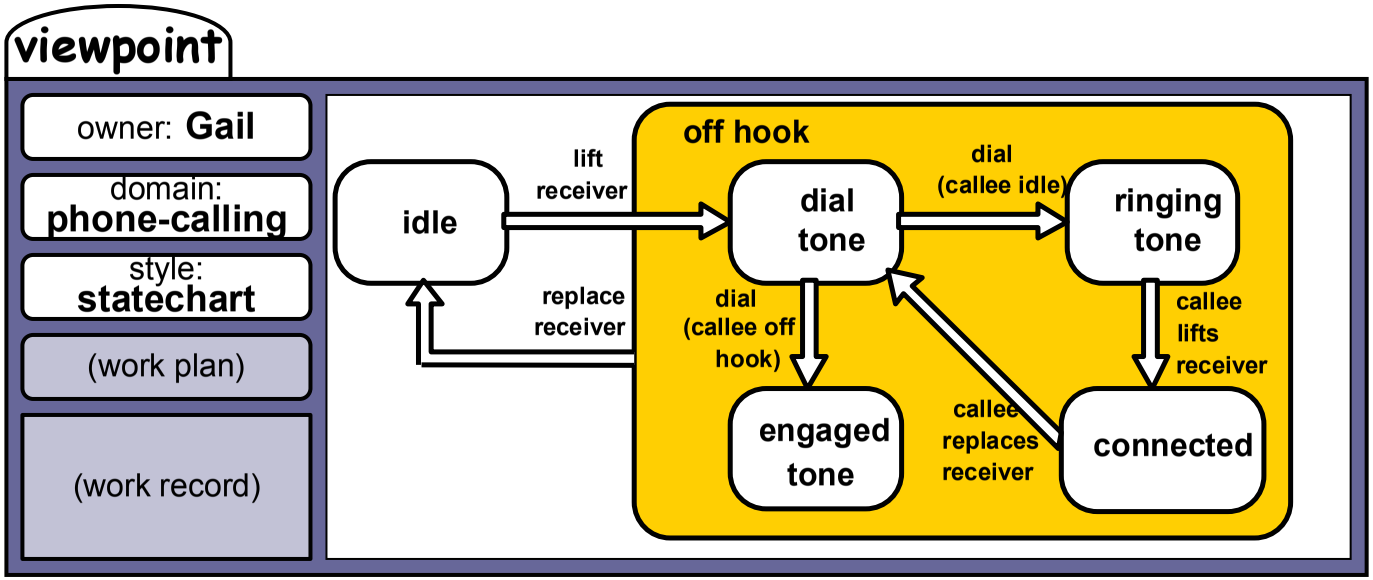
Rule 1:
Balancing transitions

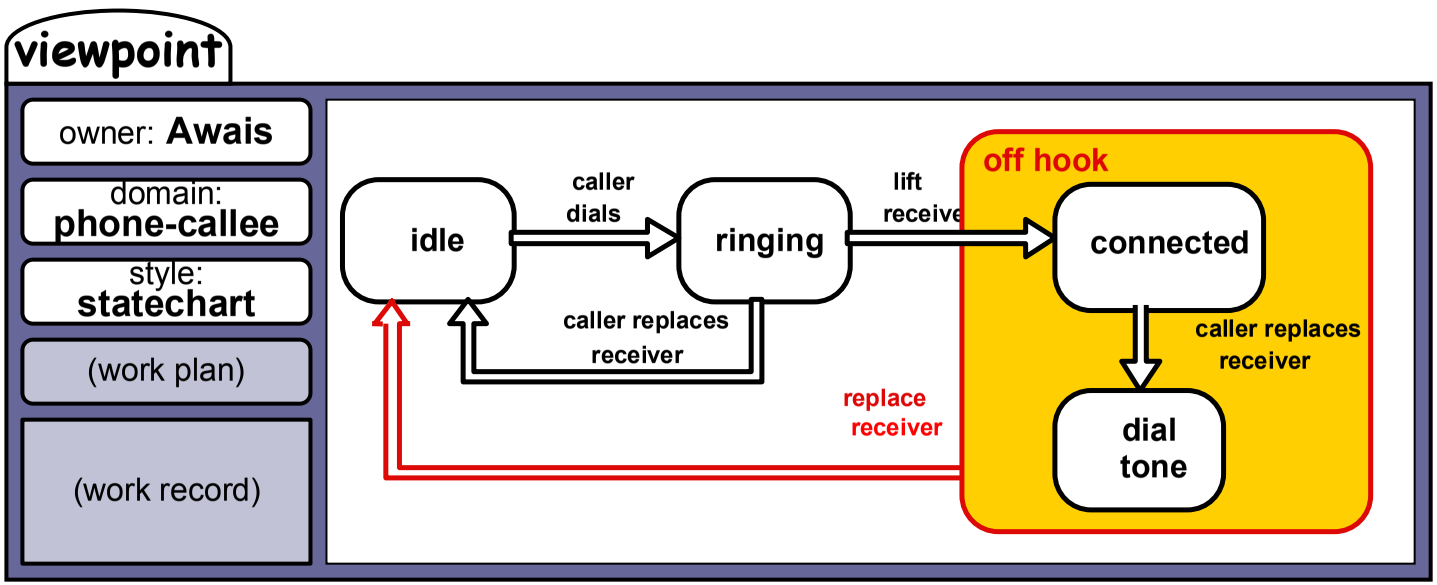
Rule 2:
Balancing super-states



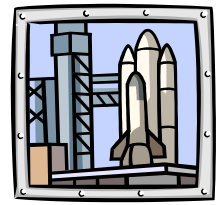


After resolution



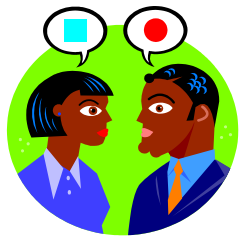


Experiences



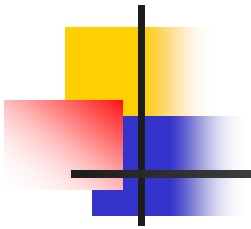
- **Case studies using a variety of methods**
  - including CORE, SSADM, HOOD, SCR.
- **NASA case studies**
  - Mixture of NL text, tables, flowcharts, etc
  - Evolved during the case studies
  - Restructured fragments of existing spec using ViewPoints
  - Identified (implicit and explicit) relationships between them
  - Checked relationships to detect inconsistency
  - Re-checked relationships as specification evolved
- **Results and lessons learned**
  - ViewPoints
  - Inconsistency

ViewPoints are often inconsistent



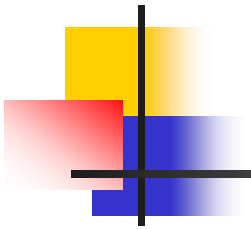
Inconsistency is a fact of life in real requirements.

Humans are very capable of tolerating inconsistency.



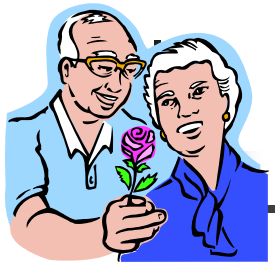
Making Inconsistency Respectable

- "A foolish consistency is the hobgoblin of little minds adored by little statesmen and philosophers and divines. With consistency a great soul has simply nothing to do ... speak what you think today in words as hard as cannonballs and tomorrow speak what tomorrow thinks in hard words again though it contradict everything you said today."



Why ViewPoints are relevant

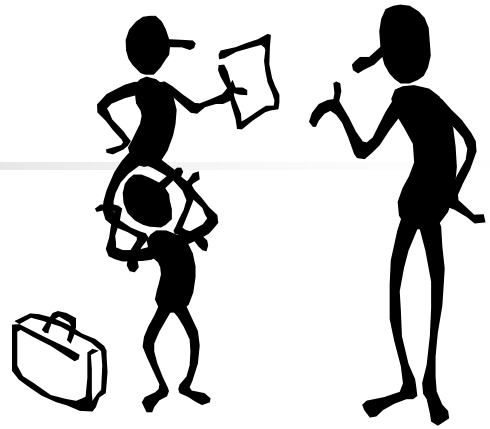
- They allow explicit capture and representation of concerns, crosscutting otherwise
- They allow explicit capture and representation of relationships between concerns
  - through identification of **overlaps** between ViewPoints,
  - expression of inter-ViewPoint relationships
- They provide a framework for **checking and reasoning** about inter-ViewPoint relationships



Meaningful relationships are difficult!

- ViewPoints provide effective separation of concerns
- However
  - "having divided to conquer, we must reunite to rule" [Jackson]
- So, the challenge is:
  - Composition and coordination
  - Reasoning and analysis across multiple ViewPoints

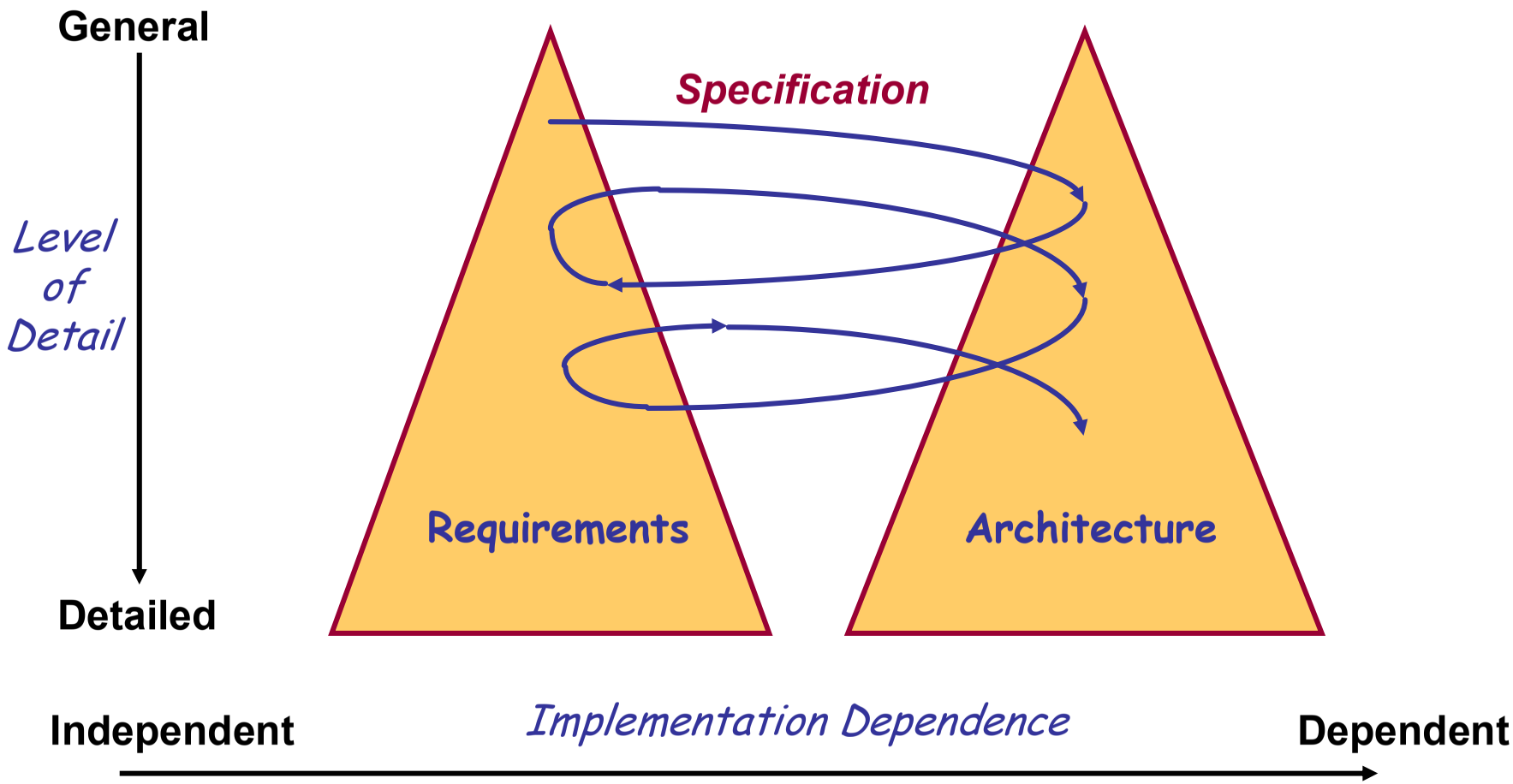
Current work



- Scalable tool support
  - xlinkit
- Composition and coordination
  - Coordination contracts, Darwin ADL
- Weaving requirements and architectures
  - Twin Peaks

Relating requirements and architectures

Twin Peaks:



Of hammers and nails

- It's great to see that there is a 'pull' for AOSD technology.
- I have argued today that the pull from the problem world is at least equally important.
- A warning to us all, whether we are 'selling' ViewPoints or Aspects, if all we have is a hammer, everything will look like a nail!


THANK
YOU!