

Requirements Themes

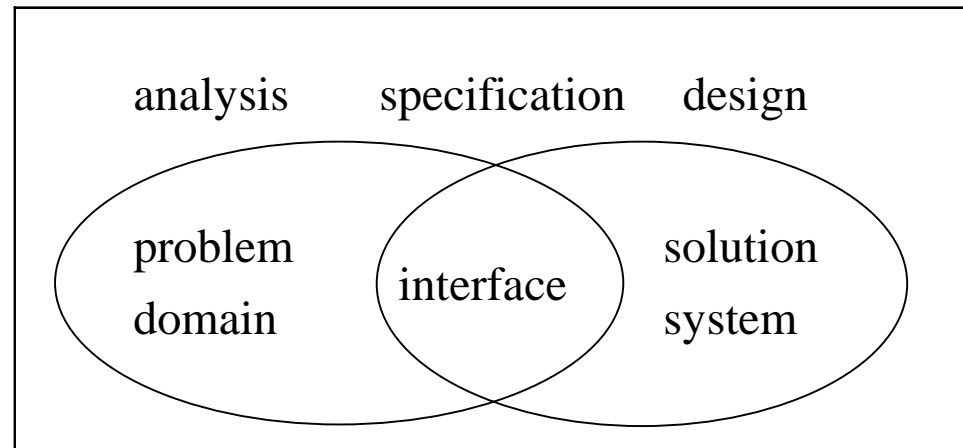
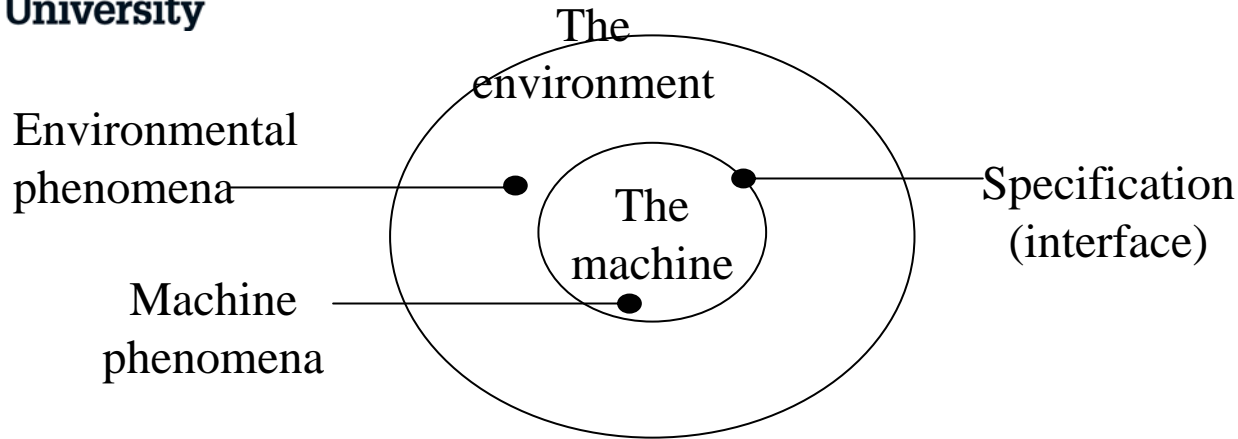
- The nature of analysis and requirements
 - Jackson's view – three descriptions.
 - Analysis in Current Methods (e.g., SA, OOA, PDOA). Strengths and weaknesses of such methods.
- Process modelling
 - How process modelling fits
 - Choice of notation: RADs
- Use Cases
 - Strengths and weaknesses, guidelines, dependencies and extensions.
- Alignment (Overview of research papers).

Three descriptions

- Jackson suggests the need to produce three separate, and distinct descriptions.

“In principle you really need three descriptions: the common description; the description that’s true only of the machine and the description that’s true only of the application domain. If you make all of those descriptions, and separate them carefully, you’ll be all right.”

Descriptions and phases



Tasks in Requirements

- **Analysis** (Requirements) - concerns study of the problem domain and the problem(s) in it
 - Much misunderstood – as we shall see
- **Specification** - defining the interaction between the solution system and the problem domain
- **Design** - (**not** part of requirements engineering) *largely* concerns inventing the internal workings of the solution system.

Analysis (SA & OOA)

- *“There’s a big temptation to believe that you can describe the application domain and the machine all together, in one combined description.*
- *But if you only make one description, you’ll surely be tempted to put things into it that describe only the machine, and to leave out things that describe only the application domain. After all, you have to describe the machine sooner or later, don’t you ?*
- *You can see the results clearly in many object-oriented modelling descriptions. Often they are accompanied by fine words about modelling the real world. But when you look closely you see that they are really descriptions of programming objects, pure and simple. **Any similarity to real-world objects, living or dead, is purely coincidental.**”*

Michael Jackson [JACK95]

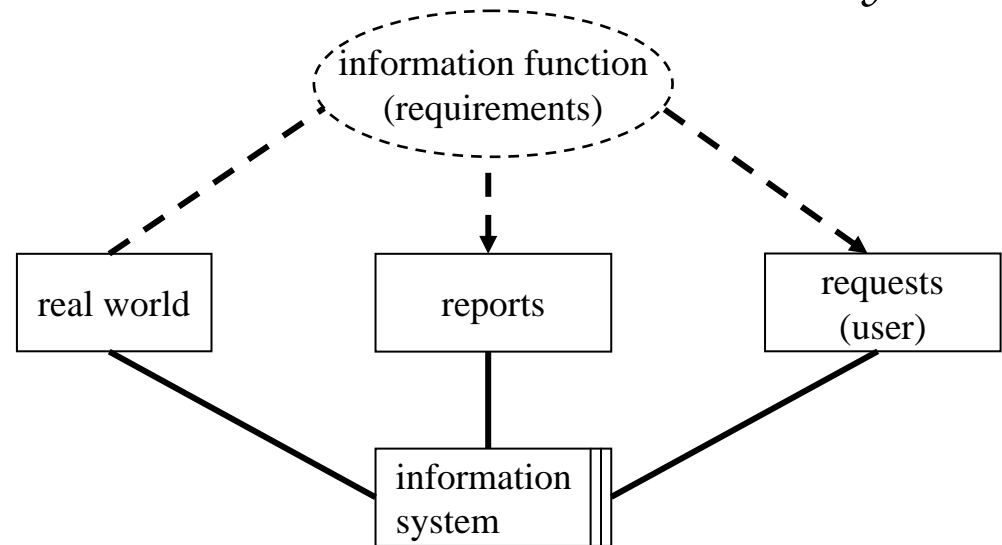
Story So Far

- SA/SD – concentration on system context.
- Tendency to model existing system (rather than the ‘essence’ or ‘business need’, or even the process.
- Still provides useful notational tool-kit
- OOA. Real world objects? Not really.
- Use cases: Oh dear, oh dear oh dear...
- So what else is there?

- “...*knowing the type of problem should help guide the analysis and specification:*
 - *what questions to ask*
 - *what aspects of the PD to describe*
 - *the nature of the requirements*
 - *which modelling techniques to use etc..*
 - *(and, ultimately, give guidance on the idesign solution)”*.

Bray – RE Slides

- Continuous (automatic) reporting: automatically provides information about some part of reality
- Requested (upon demand) reporting: provides information about some part of reality upon request



- Information function requirements refer to the *real world*.
- Information function constrains *user* (and requests), and *reports*.
- The information system interacts with the real world, user and reports (which it produces).

| requirements document content | (some) relevant techniques |
|---|---|
| Things in the real world and their attributes and relationships (data model) | Entity Relationship Diagram (ERD) and Data Dictionary (BNF) |
| All real world events that change the results of queries and the sequences in which those events can occur | Event list and Entity Life History |
| How the system can access the real world state and events? | Text |
| Requirements – the queries to be supported and the corresponding reports | Text |
| File formats for any existing files that the system needs to access | File maps, structure charts, BNF |
| Distortions and delays introduced by any connection domain | Text |

Process Oriented Approaches

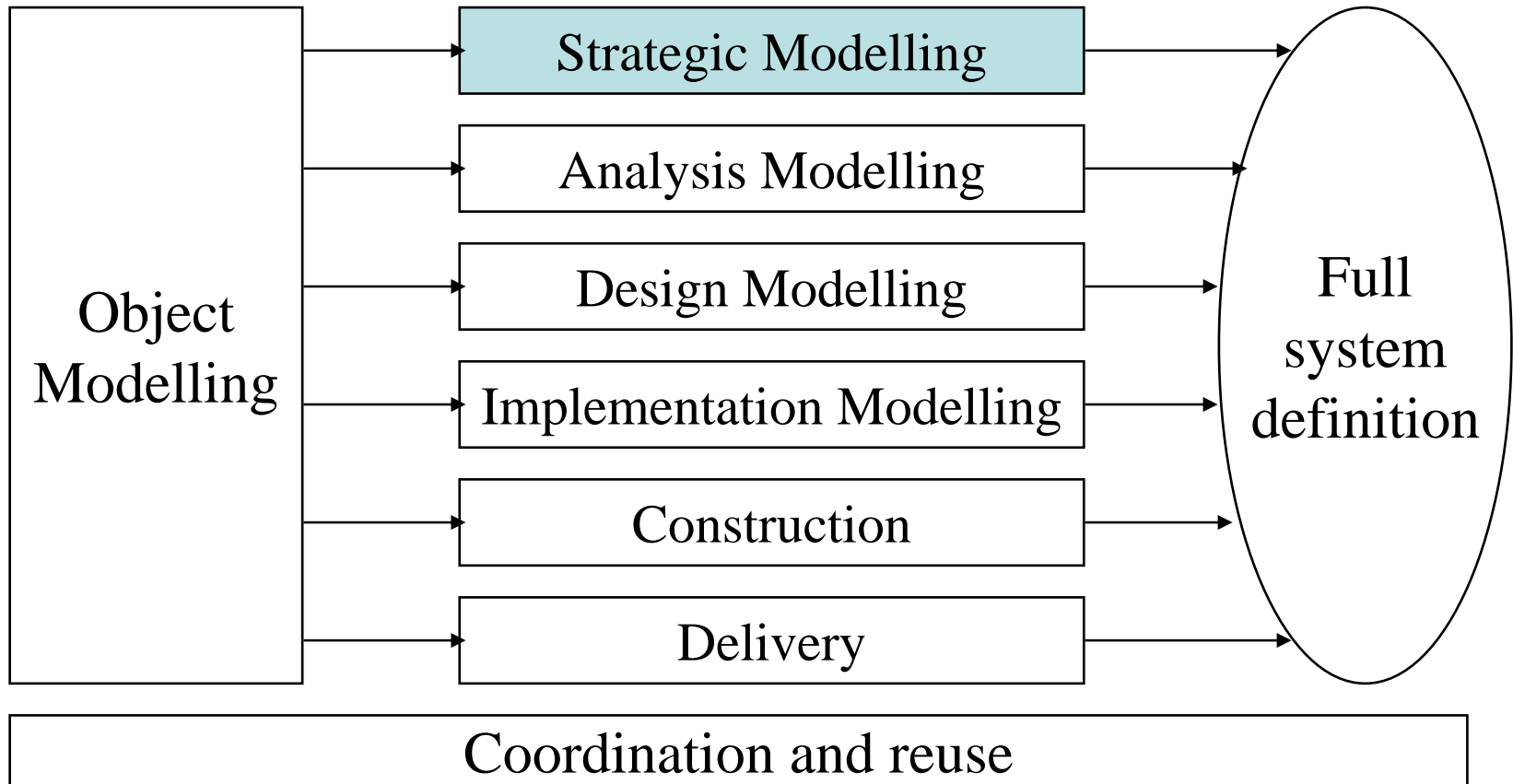
- Ideas originate in 1990s (but are taken up...)
- Follow process modelling ideas (BPM)
- Early methods tend to adopt proprietary approaches (e.g., Warboys et al.)
- Later (pragmatic) attempts to fit to popular approaches, notably the UML (EARTH)
- Recent innovations move upstream to business goals and business strategy (e.g., B-SCP)
- Recent moves back to tool support
 - Simple mappings BPM to UC or
 - Complex tool-sets (VIDE)
- Also consider legacy and direct impact to architecture

- Models of the Software Development Process.
 - Key ideas: Process programming, process maturity, study of people, tools, measurement, notations.
- Models of Client (business) processes.
 - Business process modelling.
 - Strategic modelling, requirements and analysis.
 - Alignment of IT with strategy and operation.
- Software development as a business process.

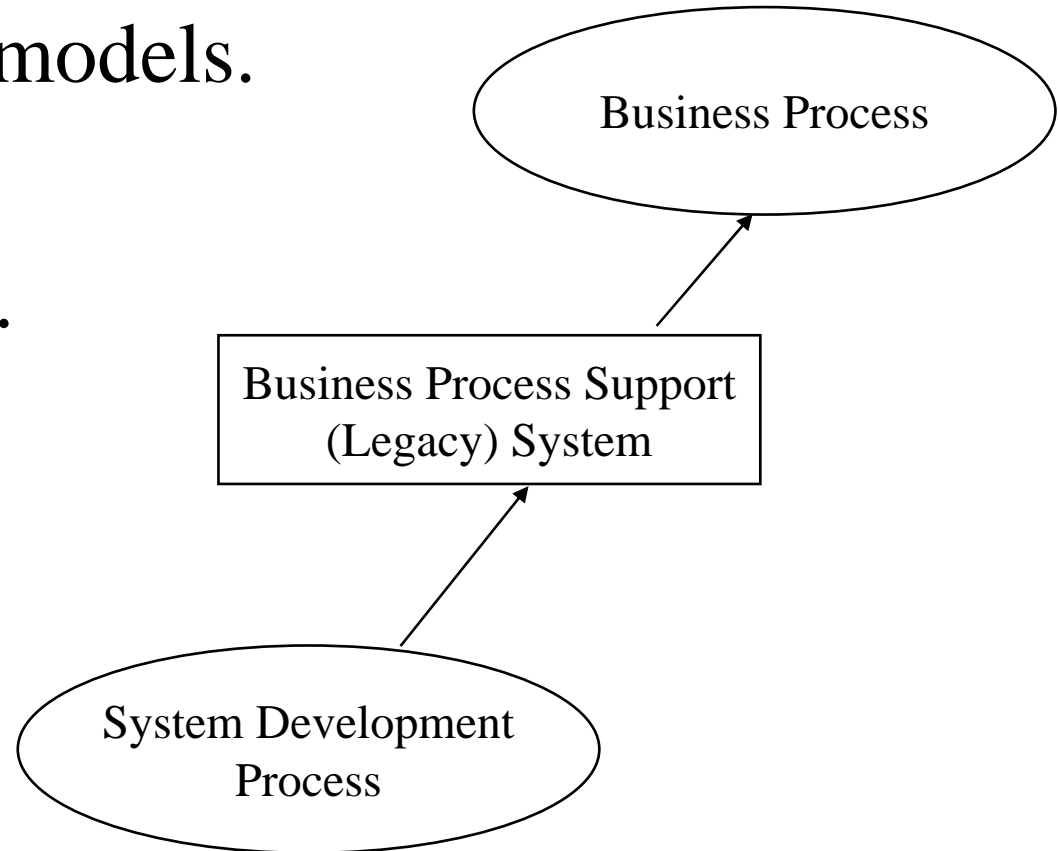
The OMG OO Lifecycle



Lifecycle

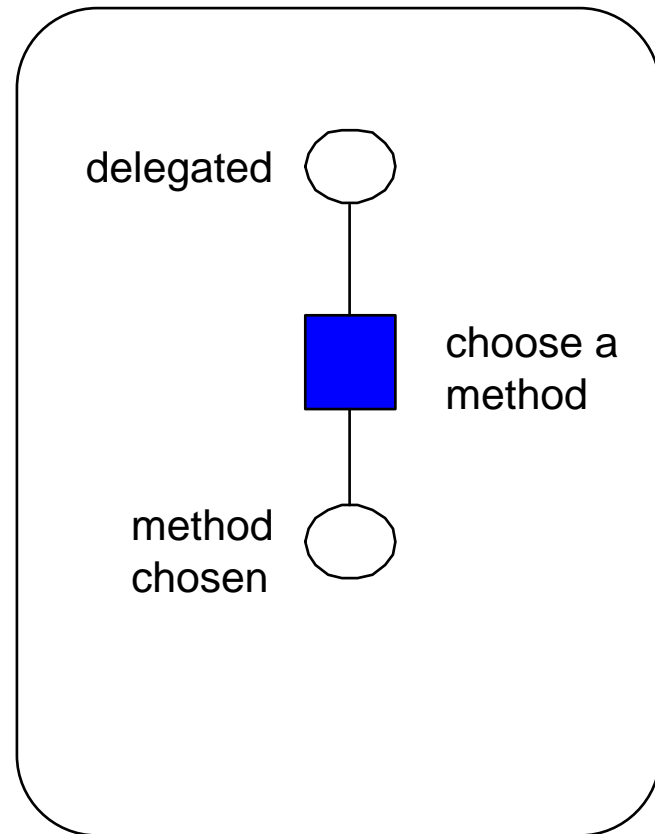


- Different kinds of models.
- Different goals.
- Different audience.
- Impact of change?
- Mapping problem
 - orthogonal?



- Role: Key concept. Group activities into roles (intuitive and matches people).
- Actions
- Interactions (Synchronous)
- States (sometimes explicit)
- Case refinement (choice)
- Part refinement (parallel)

- An action is an activity which the role carries out in isolation.
- Carrying out an action moves the role from its present state to the next state.



- An activity carried out at the same point as another activity (or other activities) in another role (or roles). A shared event.
- The consequence of an interaction is that all of the roles involved move from their current state to their next state.
- Interaction must be initiated by some (driving) role.
- Interactions are synchronous.

- Are interactions in the real world synchronous?
- Are interactions in the real asynchronous?
 - Conceptually some are, though many hidden interactions. Also depends on perspective.
 - As an example (email).
- If interactions are often asynchronous, then why would we have a synchronous model?
- How can we use synchronous models to model both sorts of interactions?

- Thread of control in a role need not proceed sequentially.
- Choice or case-refinement. There may be any number of alternative threads but only one of the threads (or cases) may be chosen.
- Concurrent threads or part-refinement. Each thread represents part of the path. The threads all join together again after the split denoting that all paths have been completed.

- Not required to explicitly label the states of a role, though some authors prefer to do so.
- Labelling states (with circles or ellipses) helps the semantics of the role become clearer.
 - Labels make explicit the pre-conditions, pre-actions and consequences (post-conditions) of each activity.
 - Sometimes need to separate parallel threads into separate (or main and sub) roles..
- Diagram becomes larger and this may hamper understanding.

- Title (which describes the UC goal)
- Actor(s)
- Context
- Pre-condition and Post-condition
- Main Flow
- 1.
- 2. etc
- Alternative flow(s)
- Exceptional flow(s)

Use Case:

Actors:

Purpose:

Precondition:

Flow of events:

Postcondition:

Place Order

Customer

Capture an order from a regular customer

The customer has logged in to the on-line system and selects Place Order

1. The use case begins when a Customer selects place order.
2. The Customer enters their name and e-mail address.
3. **Use Validate User**
4. The Customer enters product codes for the products to be ordered.
5. The system provides a product description and a price for each item.
6. The system provides user payment details for confirmation.
7. The system provides delivery address for confirmation.
8. The Customer presses submit.
9. The system gives the customer an order number and confirms the order.

The order is posted to the inventory and accounting systems

- These are in two parts: General Style Rules and Specific Structure Rules.
- The Style Rules are applicable across the description. There are 7 Style Rules.
- The Structure Rules are specific to individual sentences in the description.
- There are 2 Structure Rules.

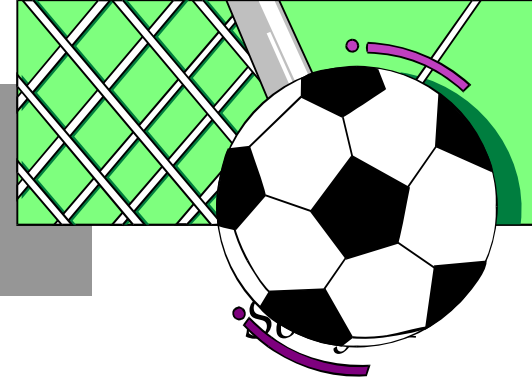
- **Structure 1:** Subject verb object.
 - *Operator presses the button.*
- Note that **verb** refers to present tense as described in **Style 2**.

- **Style 3:** Avoid using adverbs and adjectives, these add unnecessary clutter to the description and give values that are difficult to quantify. Only use negatives in alternative and exceptional flows of events. Avoid using pronouns (E.g. he, she, it, we, their etc) and synonyms. Examples:
 - *Doctor writes the prescription slowly.*
 - *slowly* is an adverb - we don't need to know how the doctor writes the prescription, just that *the doctor writes the prescription*.
 - *Patient swallows the big pill*
 - *big* is an adjective and is unnecessary; you should write *the patient swallows the pill*.

- **Style 3 (contd.)**

- *The patient stands next to the doctor.*
- *He puts the prescription in his pocket.*
- Who is “he”? Whose pocket is “his”? Write proper nouns / names instead:
 - *The doctor puts the prescription in the patient’s pocket.*
 - *The GP puts the prescription in the customer’s pocket.*
- This sentence is at fault because it uses synonyms (GP for doctor and customer for patient). Only use the agreed language of the domain since a synonym does not convey the same meaning.

Two sporting use cases



1. The match reached full-time
2. The referee blew his/her whistle
3. The ball crossed the goal-line
4. The goal was not given

Alternatives

4. The goal was given

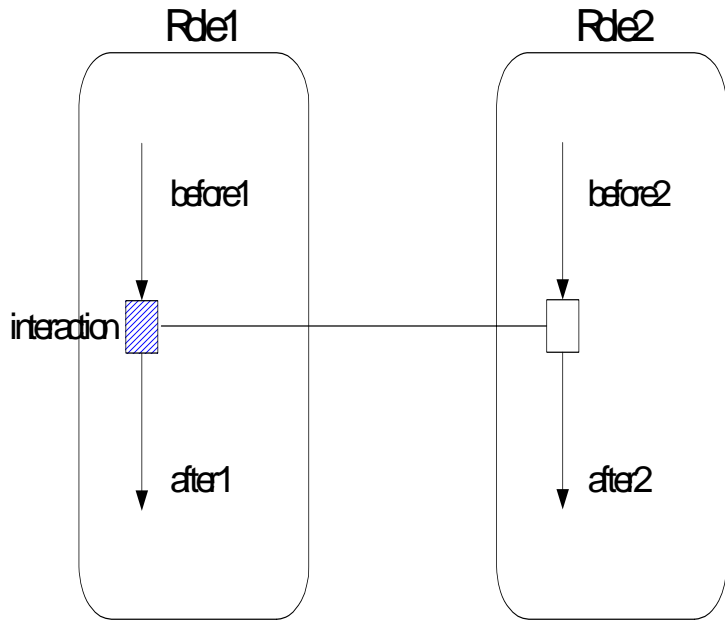
1. The match reached full-time
2. The referee blew his/her whistle
3. The ball crossed the goal-line
4. The goal was given

Alternatives

4. The goal was not given

Validation & Context. Someone who ‘knows the the game’.

Three Notations



```
Interaction Role1.Interaction
Me(before1 → after1)
Role2(before2 → after2)
End
```

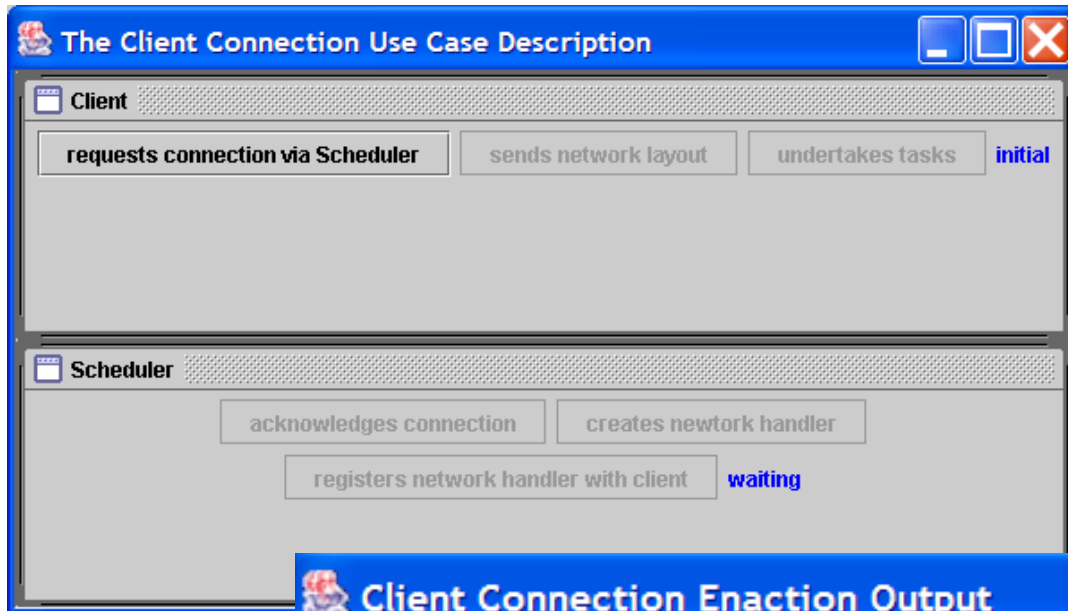
```
Interaction Keith.gives_pen
  Me (has_pen -> no_pen)
  Karl (no_pen -> has_pen)
End
```

Actor
Keith

Event
gives pen

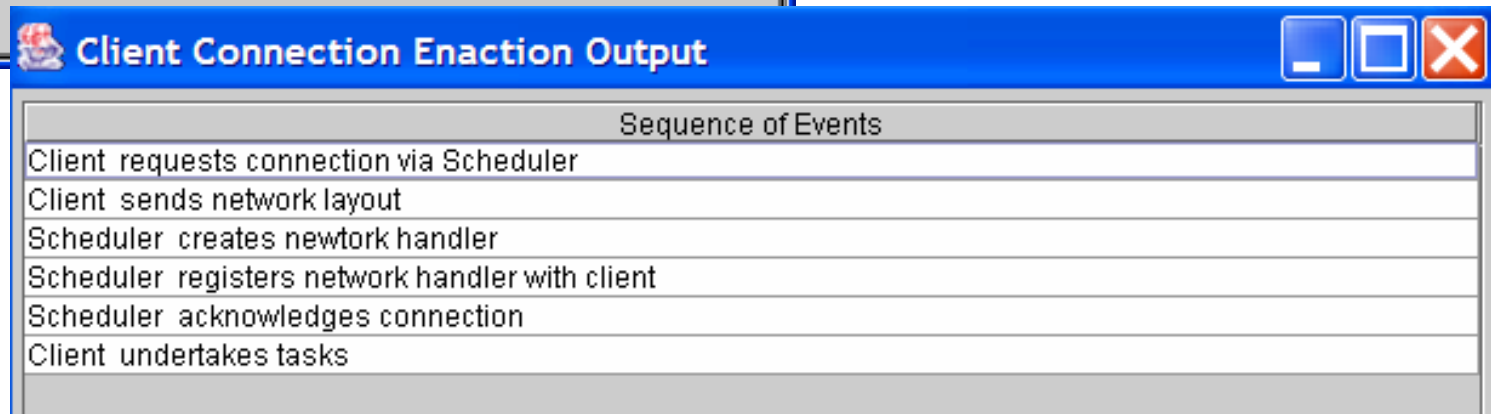
| | | | | |
|------------|-------------|----------------|------------|-------------|
| pre | post | Actor 2 | pre | post |
| has pen | no pen | Karl | no pen | has pen |

- Add pre-post to event (typically each line)
 - Interactions involve synchronisation of multiple actors.
 - Supports intra and inter-use case dependencies
 - Option to enact (order of enaction) being controlled by the pre / post *states* of events.
 - Forces consideration of dependencies amongst events.
- Allow greater stakeholder involvement.
 - Minimal (extra) effort for modeller.
- Allow traceability through from process model to use case (and beyond...)
 - Hence, don't lose the benefits.



Events re-ordered. New order is in effect: 1, 3, 4, 5, 2, 6

Of course, states not written order really control invocation of events.



Recap: Themes

- The nature of analysis and requirements
 - Requirements and Specification
 - Analysis in Current Methods.
- Process modelling
 - Where in process.
 - Using RADs
- Use Cases
 - Strengths and weaknesses, guidelines, dependencies and extensions.
- Alignment (Self directed. Links to papers).