Coming Full Circle

Observations on the history of process modelling

OR

How requirements models became process models and then became requirements models again

Keith Phalp

Invited talk: Computing Research Seminar Series,
Bournemouth University, October 2001

Motivation

- Can you do something about process modelling for SEM4?
  - Focus on business process models.
  - Consider the relationship with requirements engineering and with legacy.
  - Pick a few things that have interested me.
    - E.g., measurement, requirements.
  - Consider the notational jungle.
  - Illustrate ideas with one type of notation.

Process Modelling

Capturing and **describing** a process for some **purpose**

Key feature of process modelling:
“many of the phenomena encountered must be enacted by a human rather than a machine”.
Curtis, B., M.I. Kellner, and J. Over, Process Modelling, Communications of the ACM, 35(9), 1992.

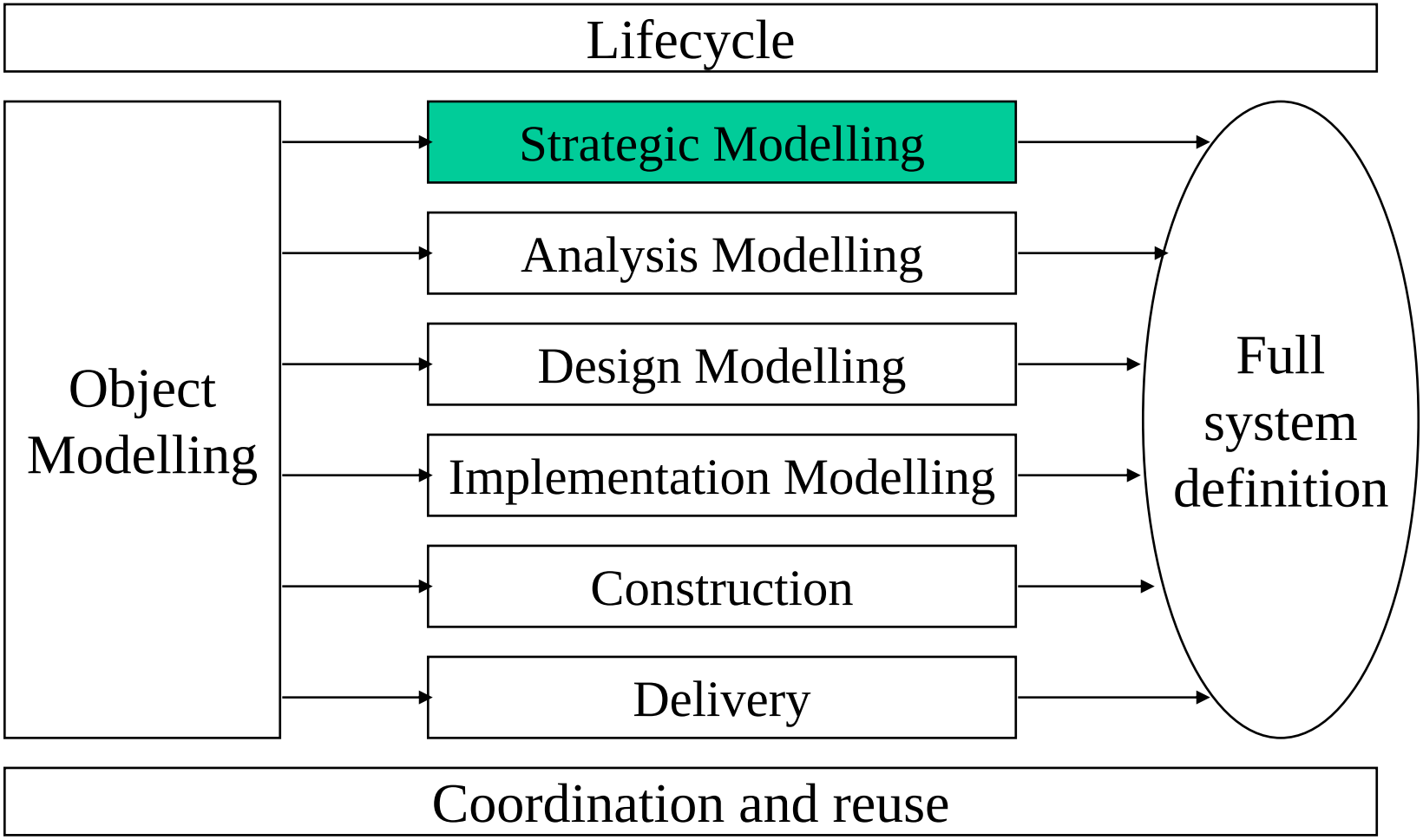
Why Business Processes?

- History of Tool Support. IPSEs
- Software as a Business Process Support System.
  - Strategic Modelling.
  - Legacy.
- Software development business process.
- Business Process Reengineering.
- Consultancy led selling. MMBP

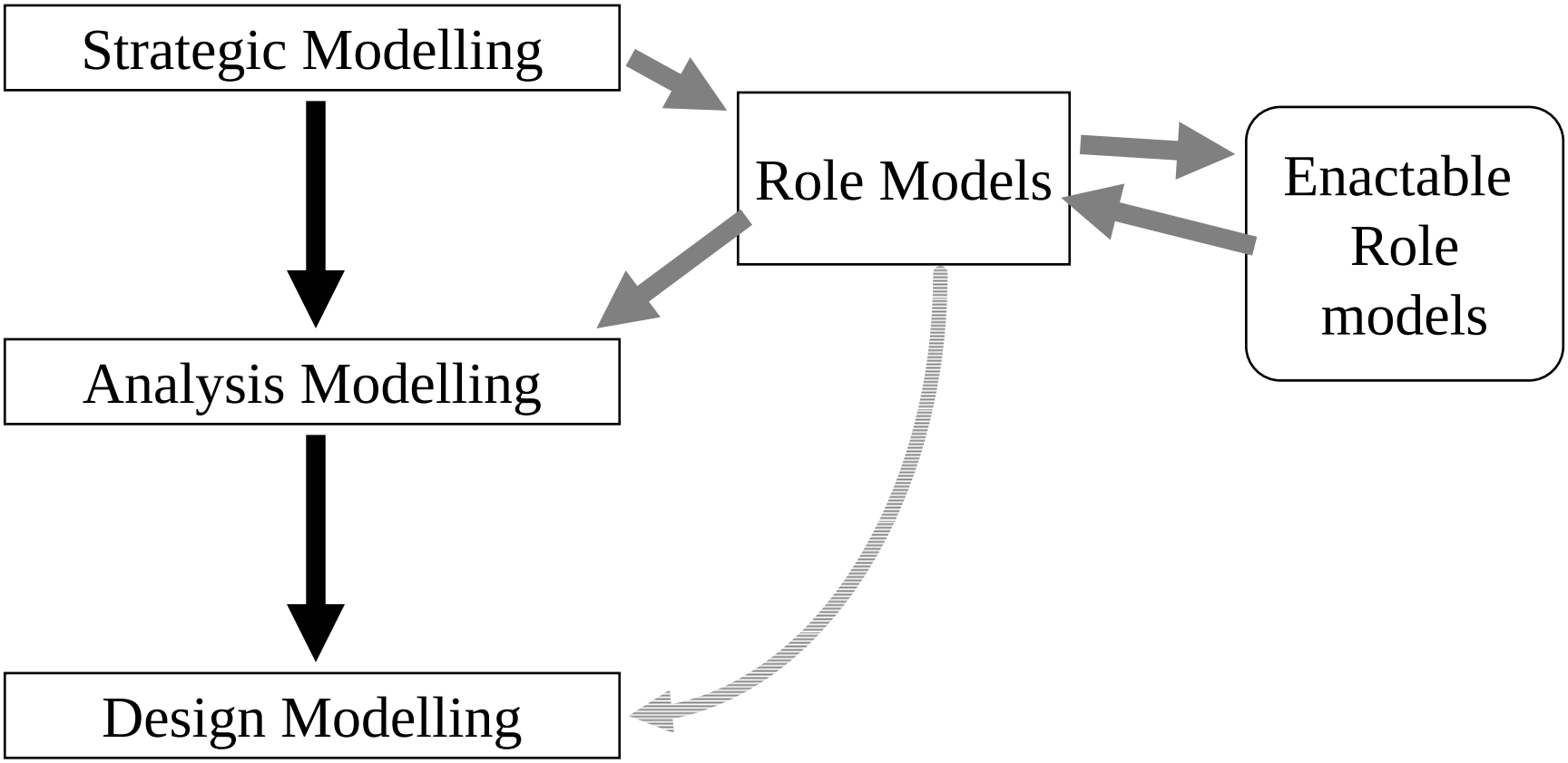
Strategic (Business) Modelling

- Modelling the business goals.
  - Understand, describe & validate business needs.
- Constraints of existing practices & support. (Legacy systems). SEBPC.
- ID weaknesses and suggest changes (BPR).
- Getting the specification right.
  - Cost of errors in coding said to be at least 100 times more than if they were spotted in specification.

The OMG OO Lifecycle

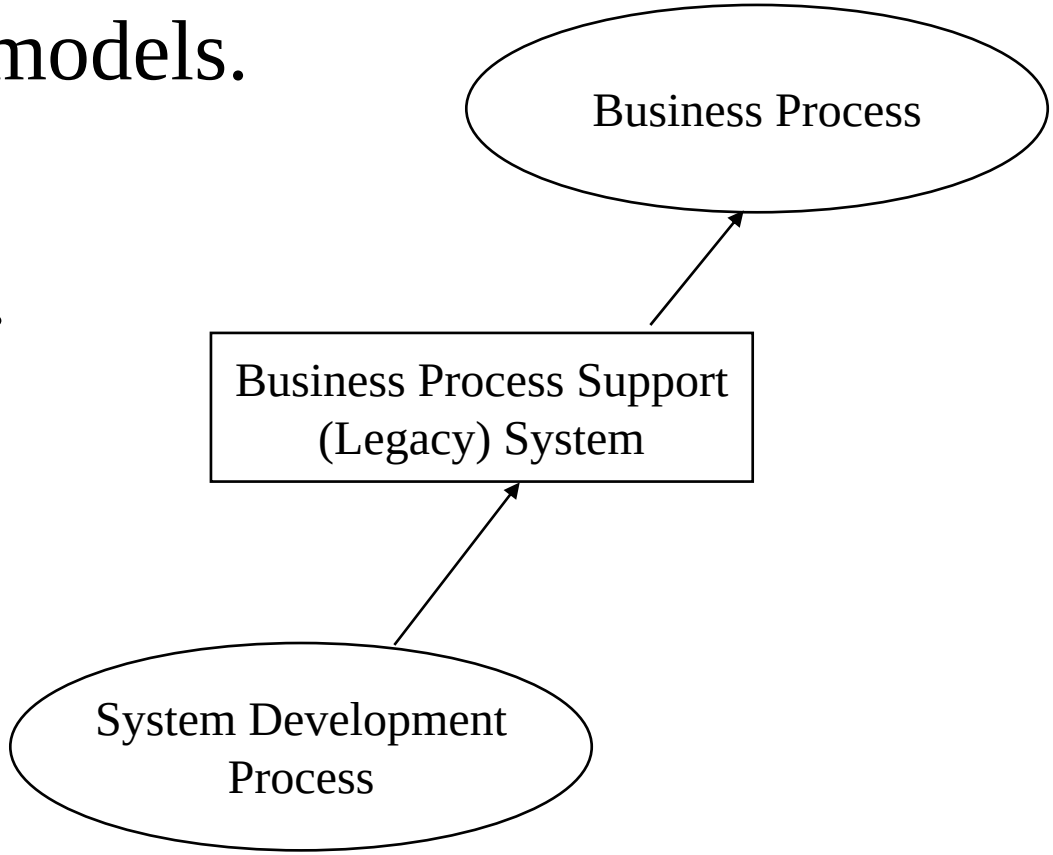


A Strategic Iterative Model



The Legacy (SEBPC) Argument ..the other BP approach

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



What do process actors need to know?

‘For an individual (or group) in the organisation to carry out their activities, they need to know what activities they must take part in, in what order those activities must take place, what other individuals or groups they must interact with, and which actions are dependent upon those interactions’.

Handy, C. (1976), ‘Understanding Organisations’, Penguin.

Role Based Models

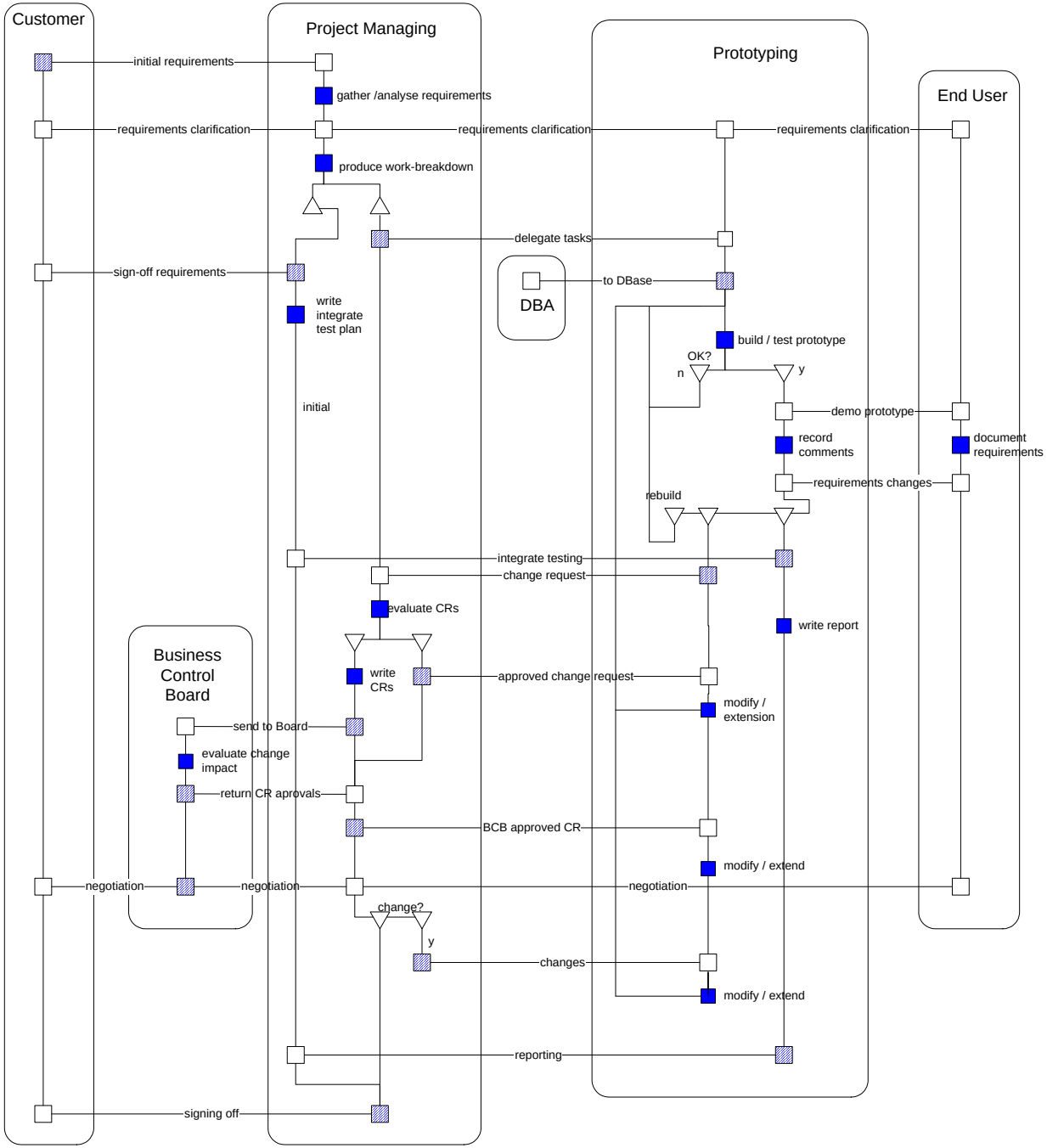
- ‘Role based models satisfy these requirements by grouping activities into ‘roles’, which describe the desired behaviour of individual groups, or systems’.

  Ould, M.A. (1992), An introduction to Process Modelling using RADs, in IOPTCLUB Practical Process Modelling, Mountbatten Hotel, Monmouth Street, Covent Garden, London.

- ‘A role involves a set of activities which, taken together, carry out a particular responsibility or set of responsibilities’.

  Ould, M.A. (1995), Business Processes modelling and Analysis for Re-engineering and Improvement, John Wiley & Sons.

A RAD


Customer
Project Managing
Prototyping
End User
initial requirements
gather /analyse requirements
requirements clarification
requirements clarification
requirements clarification
produce work-breakdown
delegate tasks
sign-off requirements
write integrate test plan
DBA
to DBase
build / test prototype
OK?
n
y
initial
demo prototype
record comments
document requirements
requirements changes
rebuild
integrate testing
change request
evaluate CRs
write report
Business Control Board
write CRs
approved change request
modify / extension
send to Board
evaluate change impact
return CR aprovals
BCB approved CR
modify / extend
negotiation
negotiation
negotiation
change?
y
changes
modify / extend
reporting
signing off

Heuristics to Counts

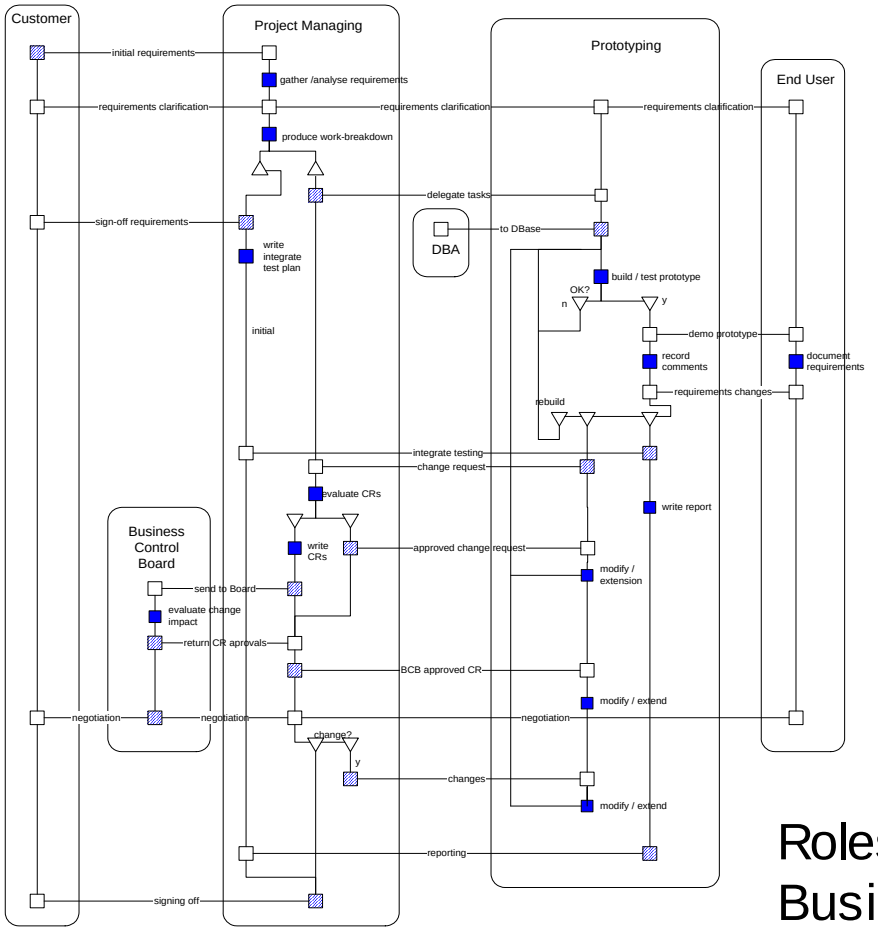
- *'As a set, the roles should be loosely coupled, i.e. we should expect few interactions between them'.*
- Coupling Factors:

  $CpF_X = (I_X) / (A_X + I_X)$.

  $CpF_{Sys} = (I_{sys}) / (A_{sys} + I_{sys})$.

  – No "magic number" nor optimisation of one heuristic to the exclusion of all others.

A Simple Analysis

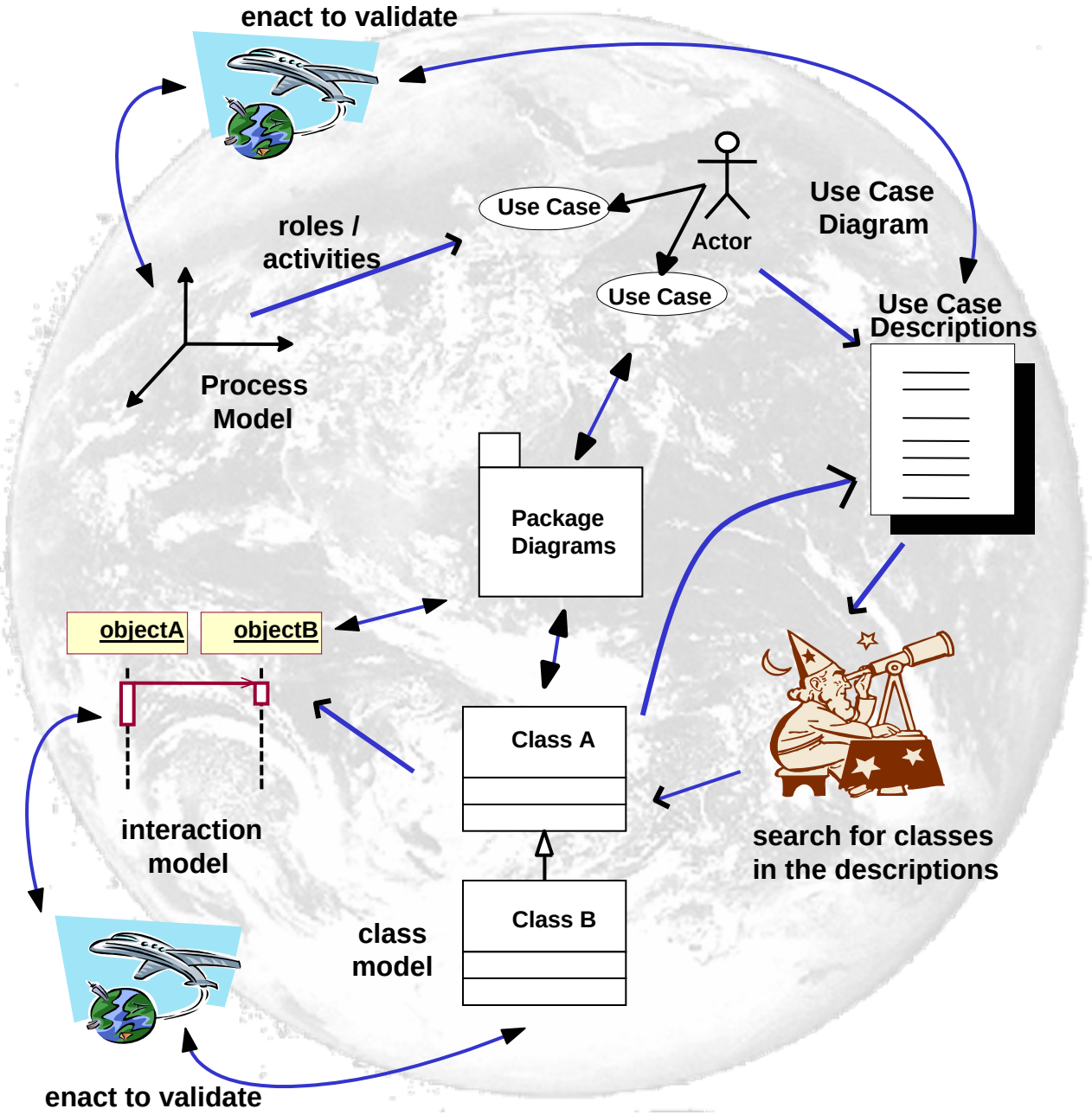


Roles	A_X	I_X	CpF
Business Control Board	1	3	0.75
Project managing	5	13	0.72
Prototyping	6	11	0.65
DBA	0	1	1.00
Customer	0	5	1.00
End user	1	4	0.80

Moving Towards Enaction: RolEnact

- A language for process modelling: analysis and presentation.
- Based upon a condition-action paradigm.
- Primitives match those of role-based models (such as RADs).
- Processes described in terms of roles, the states of these roles, and the activities or events in which each role may take part.

EARTH method

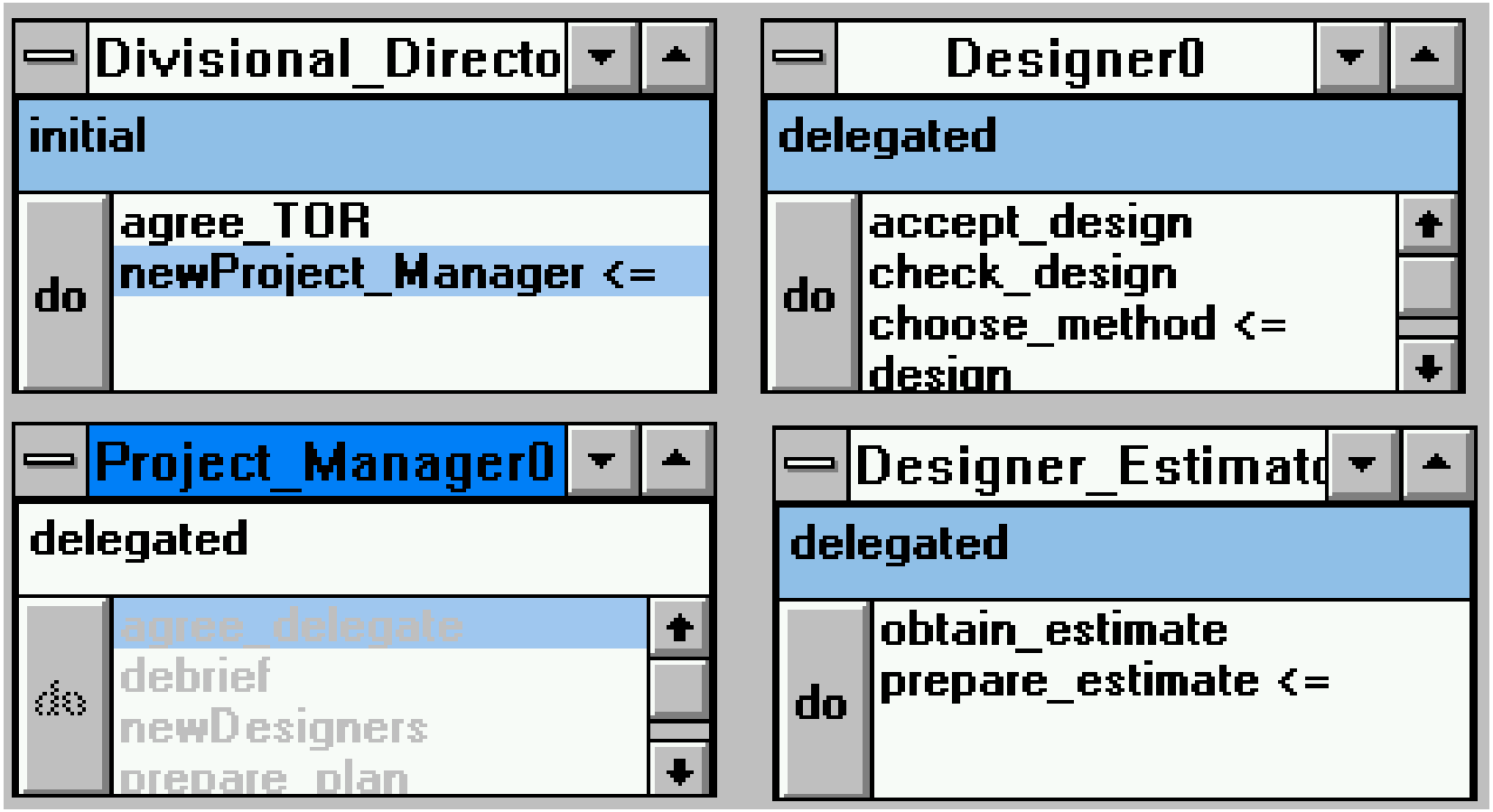


What goes around...

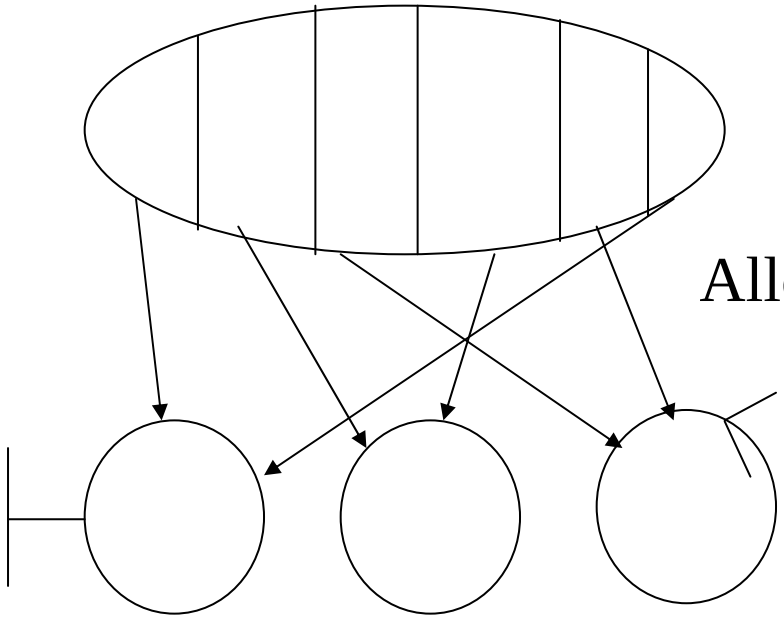
- Considered the move to and some reasons for emergence of business models.
  - Illustrated some changes (refinements) using the example of role based models.
- So have things changed (for the better)?
  - Requirements -> business models (with enaction).
  - Back to requirements in mapping to use cases.
  - Force design issues by enaction of use case descriptions.

Additional Slides

RolEnact for Designer



Classes from Use Cases: OOSE



Interface objects

Entity objects

Control objects

Functionality of a use case

Allocation to object responsibilities...

...but doesn’t say how.

Jacobson’s OOSE Approach

Questions

- What Questions should we ask of a description.
  - Ones that allow us to specify / design?
  - Some to identify the objects, attributes, services we need. (More intuitive - next).
  - Some to test the behaviour / dependencies of actions. (Which is usually implicit).
    - May use other models to aid this process.
  - Some to check the communicability.
  - Some to assess the 4Cs.

Consider an event or action

***Pre**: Driver not at machine (initial).*

Both machine and driver in this (initial) state.

How did driver know there were spaces? (Sign + Object that knows count)

- The *Driver* drives to the ticket machine.

***Post**: Driver (at Ticket Machine).*

Both machine and driver in this new (at machine) state.

Ticket machine notified of this by what object? (Sensors)

Hidden action(s)

1.1 The *Driver* drives over the entry pad (sensor).

Pre and post as for ticket machine

Me (initial -> overPad)

EntryPad(initial -> overPad)

1.2 The *Sensor* notifies the ticket machine [of what…?]

```
Selection Driver.driveOverPad
   Me( initial -> DriverAtMachine )
   EntryPad( initial -> overPad )
End

Selection EntryPad.PadNotify
   Me( overPad -> initial )
   TicketMachine( initial -> CarAtMachine)
End

Selection Driver.PressForTicket
   Me( DriverAtMachine  -> ticketRequested )
   TicketMachine( CarAtMachine -> ticketRequested )
End

Selection TicketMachine.Dispense
   Me( ticketRequest -> ticketDispensed )
   Ticket ( initial -> date_stamped )
End

Interaction Driver.TakeTicket
   Me( ticketRequested -> ticketTaken)
   TicketMachine( ticketDispensed -> ticketTaken )
```

Object States: Formal

- Dependencies for 1 to 4.
- States act as pre / post conditions.
- E.,g., for driver to take ***ticket*** it must have been dispensed.
- Ticket not from behaviour, but a data object.
- and so on...