Process Modelling

Notations, Methods and Enaction

Keith Phalp
Department of Electronics and Computer Science
University of Southampton
kp@ecs.soton.ac.uk

Invited talk at Bournemouth University, January 1997

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.

Overview

- Context. Importance of notation?
  - Many different approaches. (Choice).
  - Knowing the model is right.
- Modelling experience, and impact on methods.
- Using enactable models. An example.

Faith, Maths, Validation and Debugging

- Formal: Rigour: Need checking by experts
- Pragmatic: Typically diagrams:
  - Sacrifice rigour for understandability.
  - Users validate. Suffer from multiple interpretation
- Enactable. Visual. Try it out.
- Combinations.
  - Understandability and rigour.
  - Flexiblity and familiarity.
  - Separation of concerns.

Business Processes: Experience

- **Initial modelling must start with an easy to understand approach (diagrams).**
- Even simple diagrammatic notations lead to complex models
  - which users find difficult to comprehend.
- Mechanisms to add structure to detailed models can help.
- **Enactable notations do help**
  - **but users need to be shielded from them.**

Levels of Notation

Elicitation / Validation

RADs

IDEF

DFD

POSD

Formal Paradigms

Enact Engine

CSP

CTS

RolEnact

POSD

MBaSS

RolEnact Windows

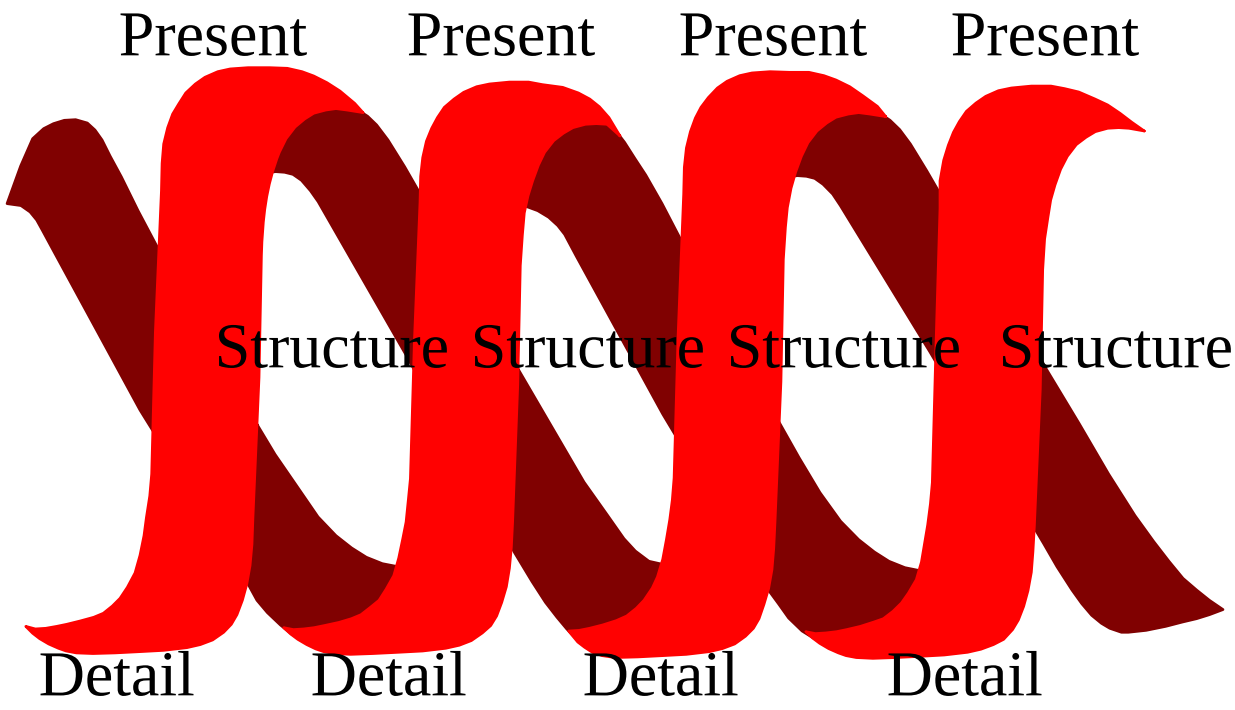
PWI

Presentation Enaction / Support

Putting it together

- Elicitation: Capture & describe processes using notations which users will understand (validate and feedback).
- Experimentation: Further understand process: Experiment with executable models / scenarios.
- Presentation / Discussion / Education - with understandable static and executable models.

A Spiral Method



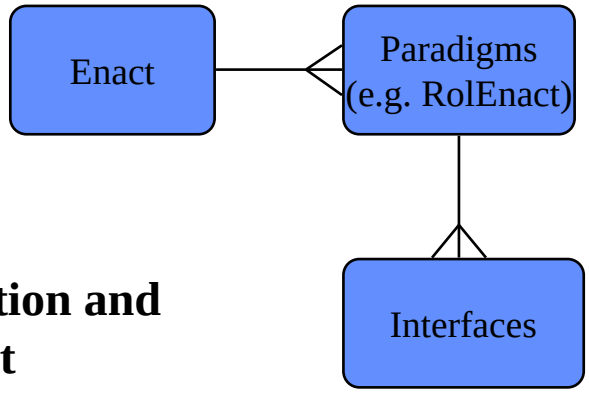
Detail, Structure, Present

- Establish detailed notation(s)
- Gather evidence (multiple sources).
- Produce model based on documentary understandable evidence.
- Analysis of detailed Modelling:
- Run process scenarios (analyse)
- include process data
- Augment model with findings
- Produce alternative process scenarios.
- Validate process understanding

- Establish structured notation
- Impose structure
- Produce structured (understandable) models
- Highlight findings.
- Produce multiple viewpoints

- Choose presentation mechanism
- Present inconsistencies, findings, process scenarios.
- Use models to facilitate discussion.
- Use structured models, e.g., POSD models, models as guideline for reports.

A Route Through the Notations



User Facing

Enactable Models

Simulation and Support

DFDs

CSP

An illustrative route

RADs

Map
Translate

RolEnact

Enact

RolEnact
Windows

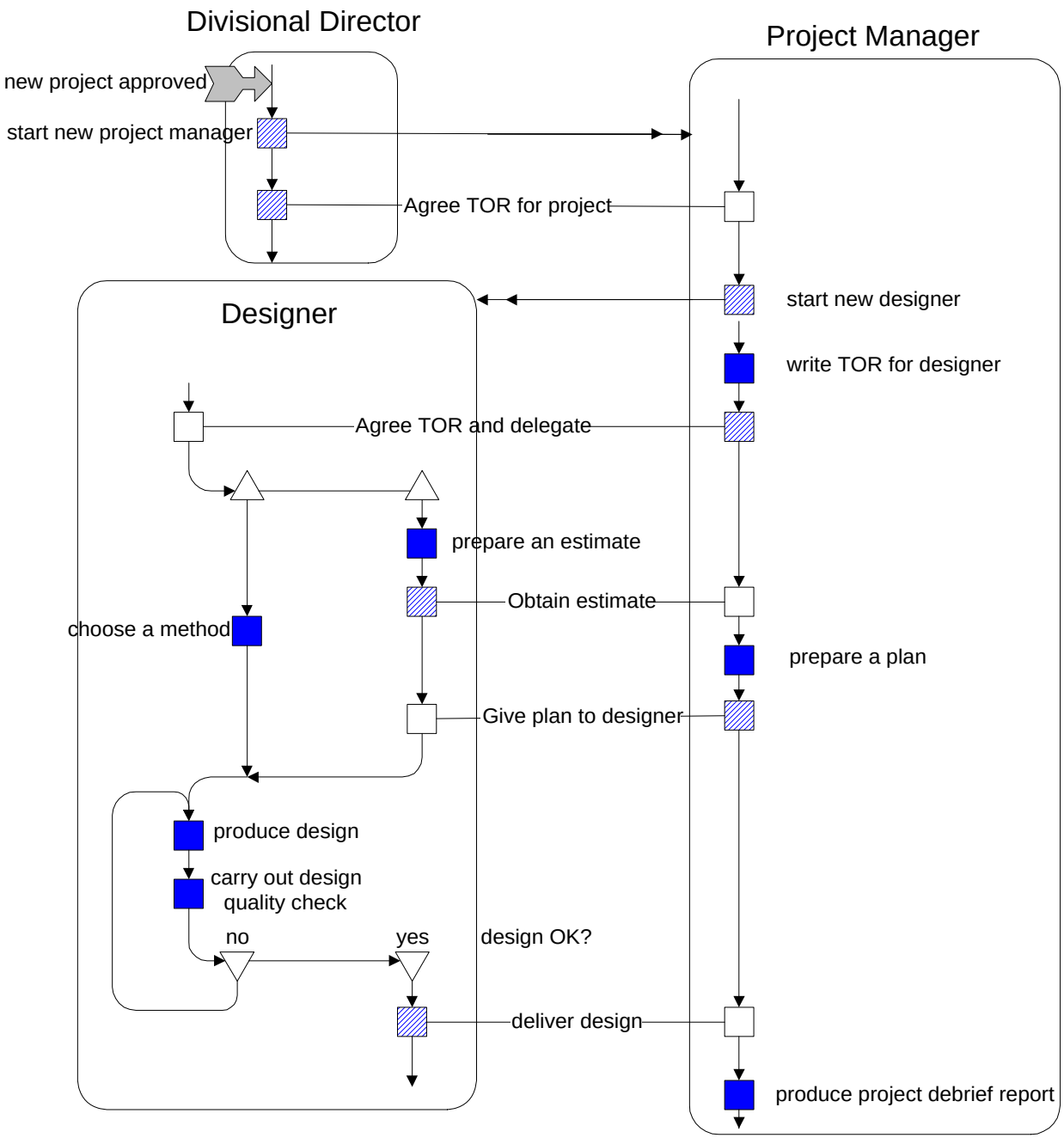
IDEF

CTS

Model Based
Slide Shows

PWBS

Other Support
e.g. PWI

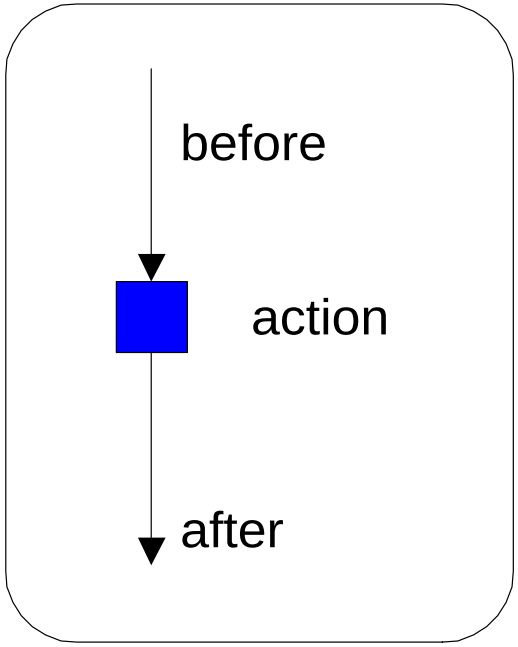

Divisional Director
Project Manager
new project approved
start new project manager
Agree TOR for project
Designer
start new designer
write TOR for designer
Agree TOR and delegate
prepare an estimate
Obtain estimate
choose a method
prepare a plan
Give plan to designer
produce design
carry out design quality check
no
yes
design OK?
deliver design
produce project debrief report

Action

Action Role.Action

Me(before → after)

End



Action Project_Manager.prepare_a_plan

Me(estimate_received → plan_prepared)

End

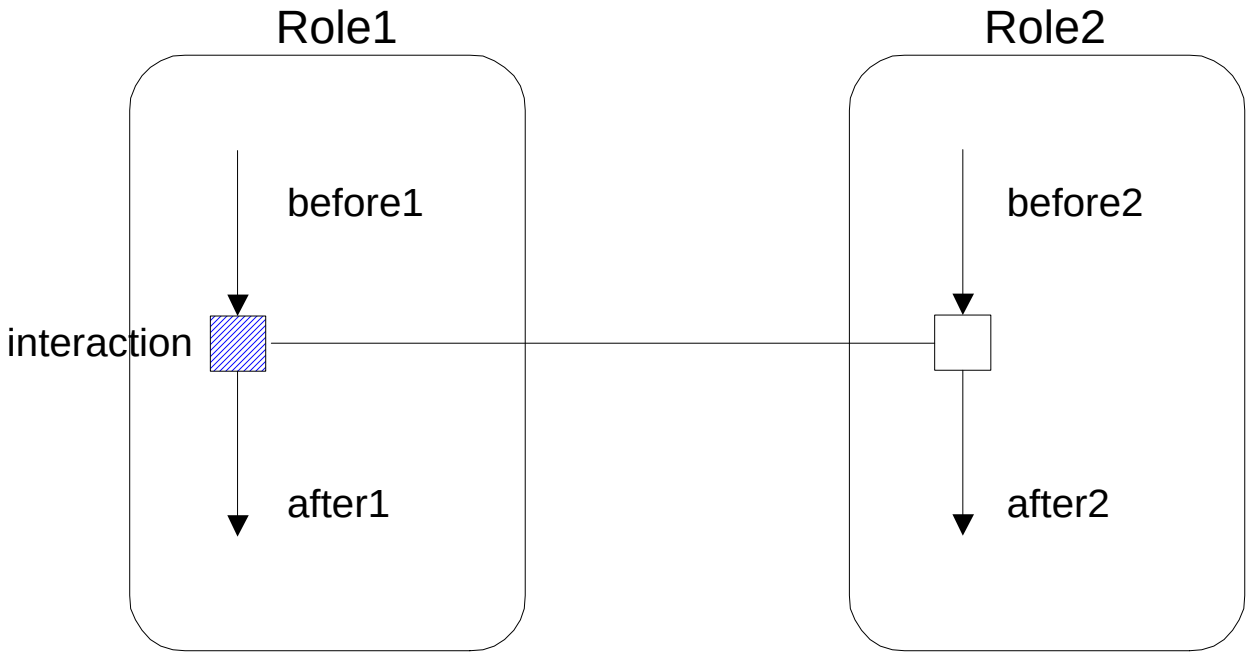
Interaction

Interaction Role1.Interaction

Me(before1 → after1)

Role2(before2 → after2)

End



Selection

Selection Role1.Selection

Me(before1 → after1)

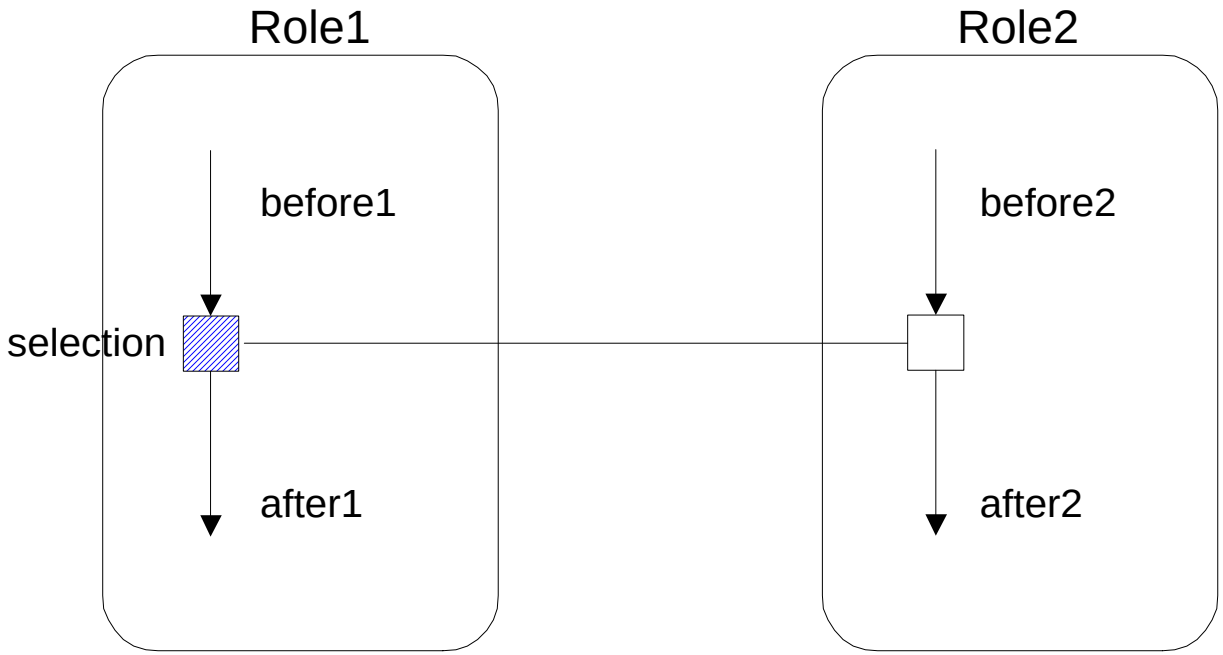
Role2(before2 → after2)

End

Automatically creates:

Me.Role2:=r,

r.Role1:=Me



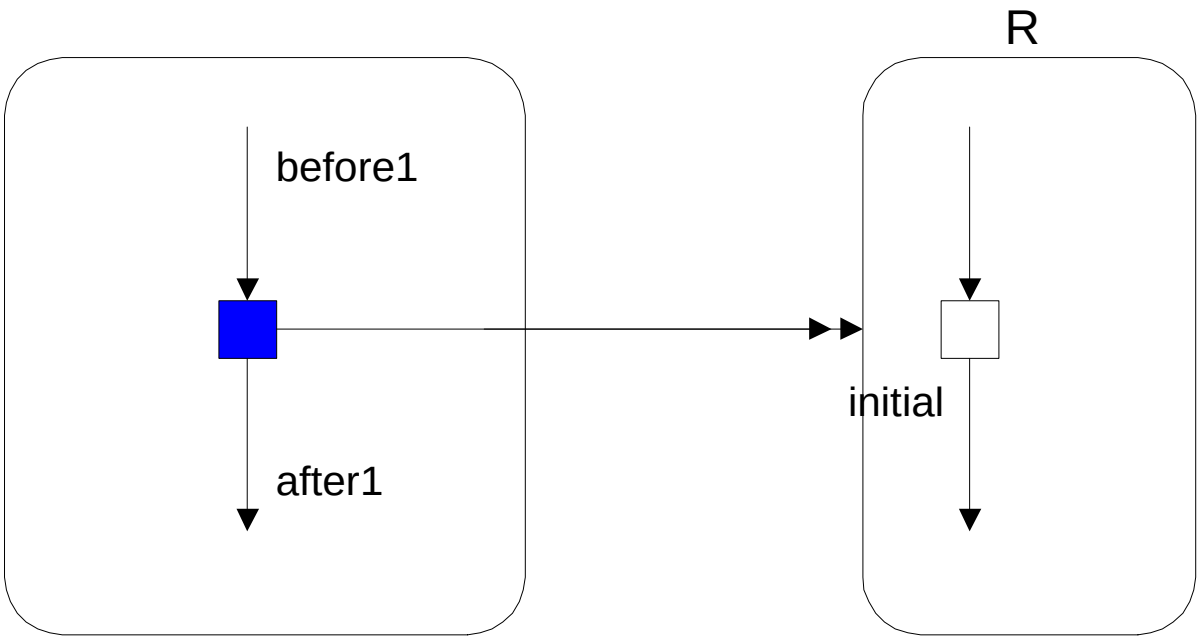
Creation

Create Role1.Create

Me(before1 → after1)

new Role2

End



A Role: Director

Create Divisional_Director.newProject_Manager

me(initial $\rightarrow$ manager_started)

new Project_Manager

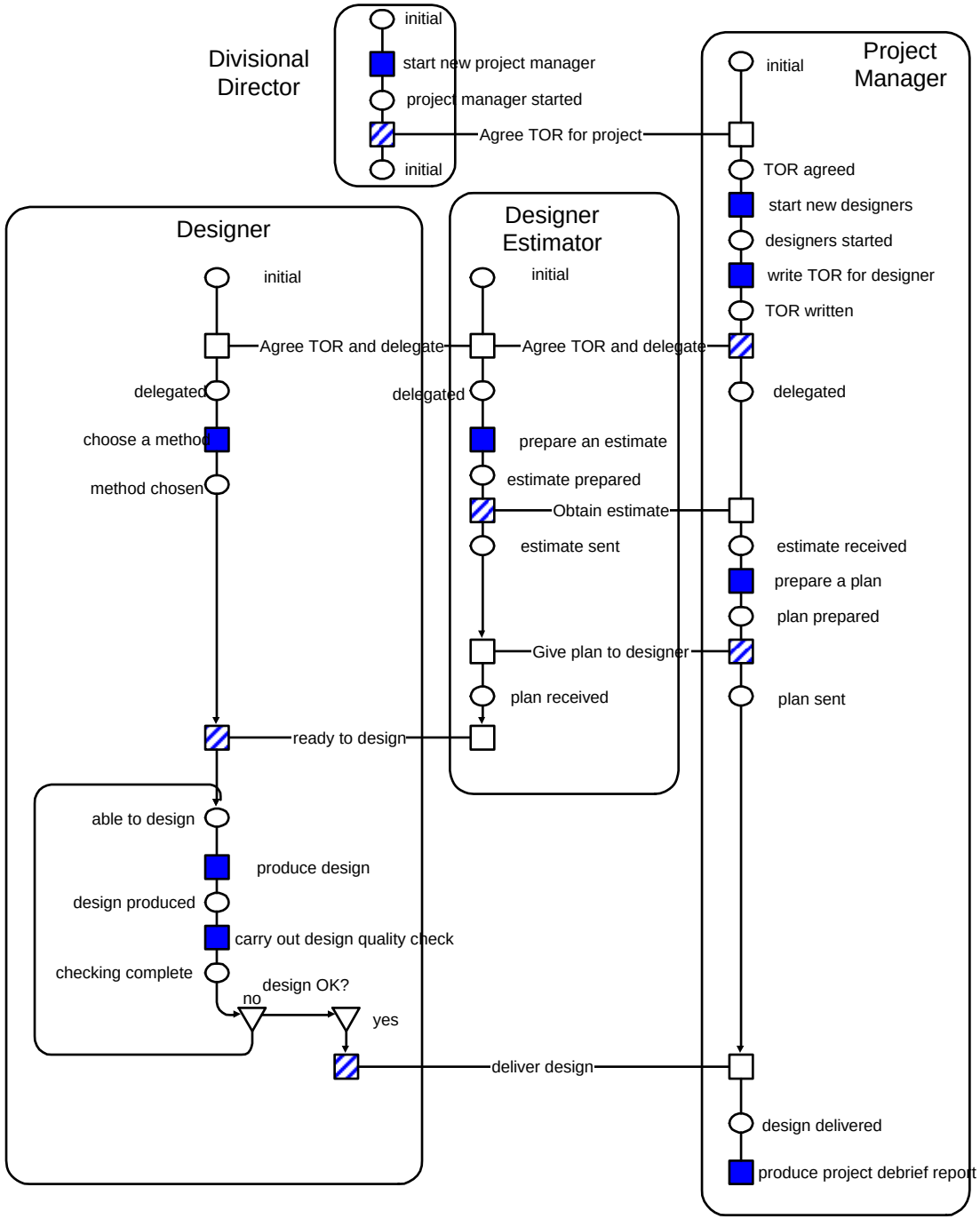
End

Interaction Divisional_Director.agree_TOR

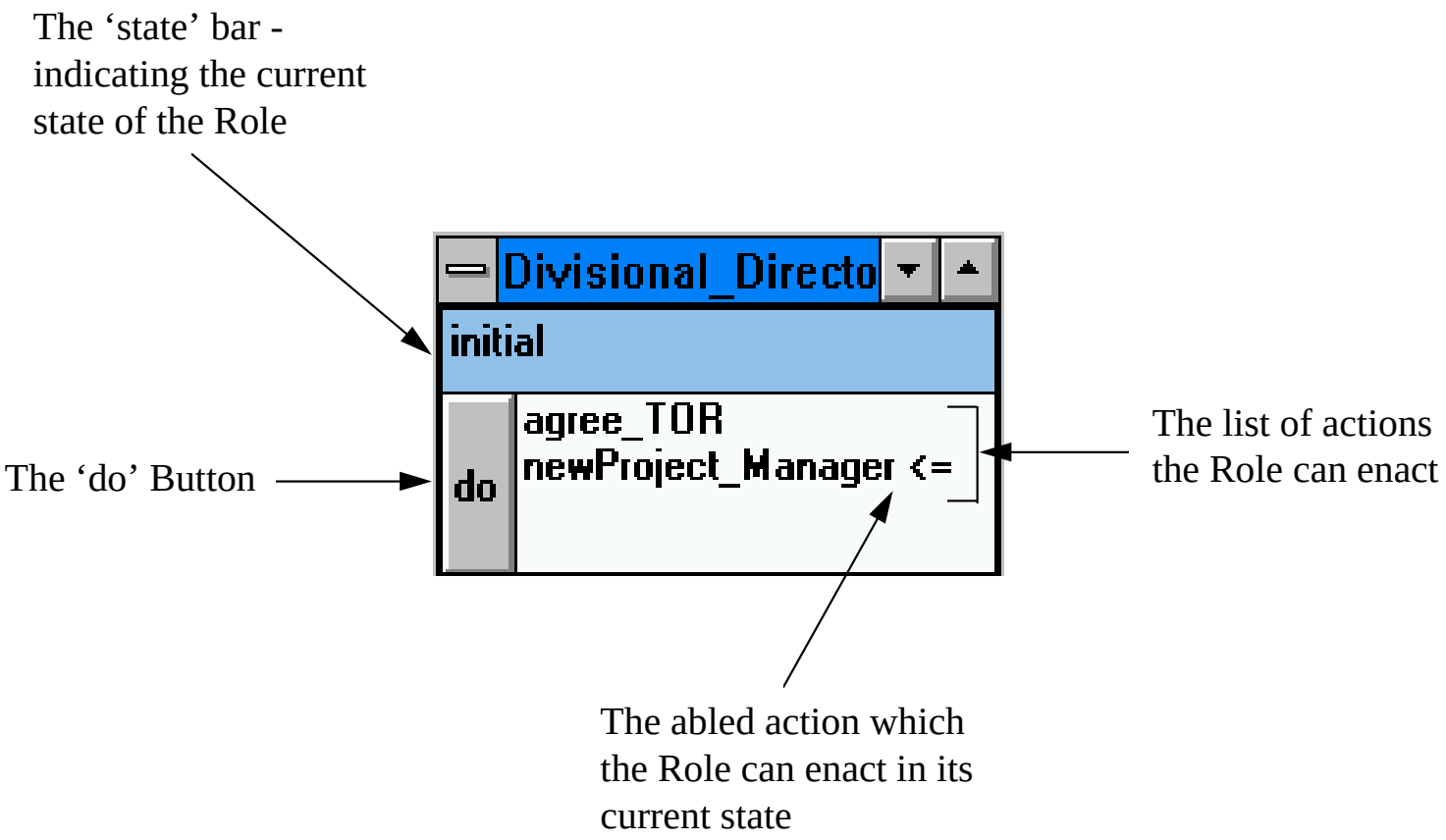
me(manager_started $\rightarrow$ initial)

Project_Manager(initial $\rightarrow$ agreed_TOR)

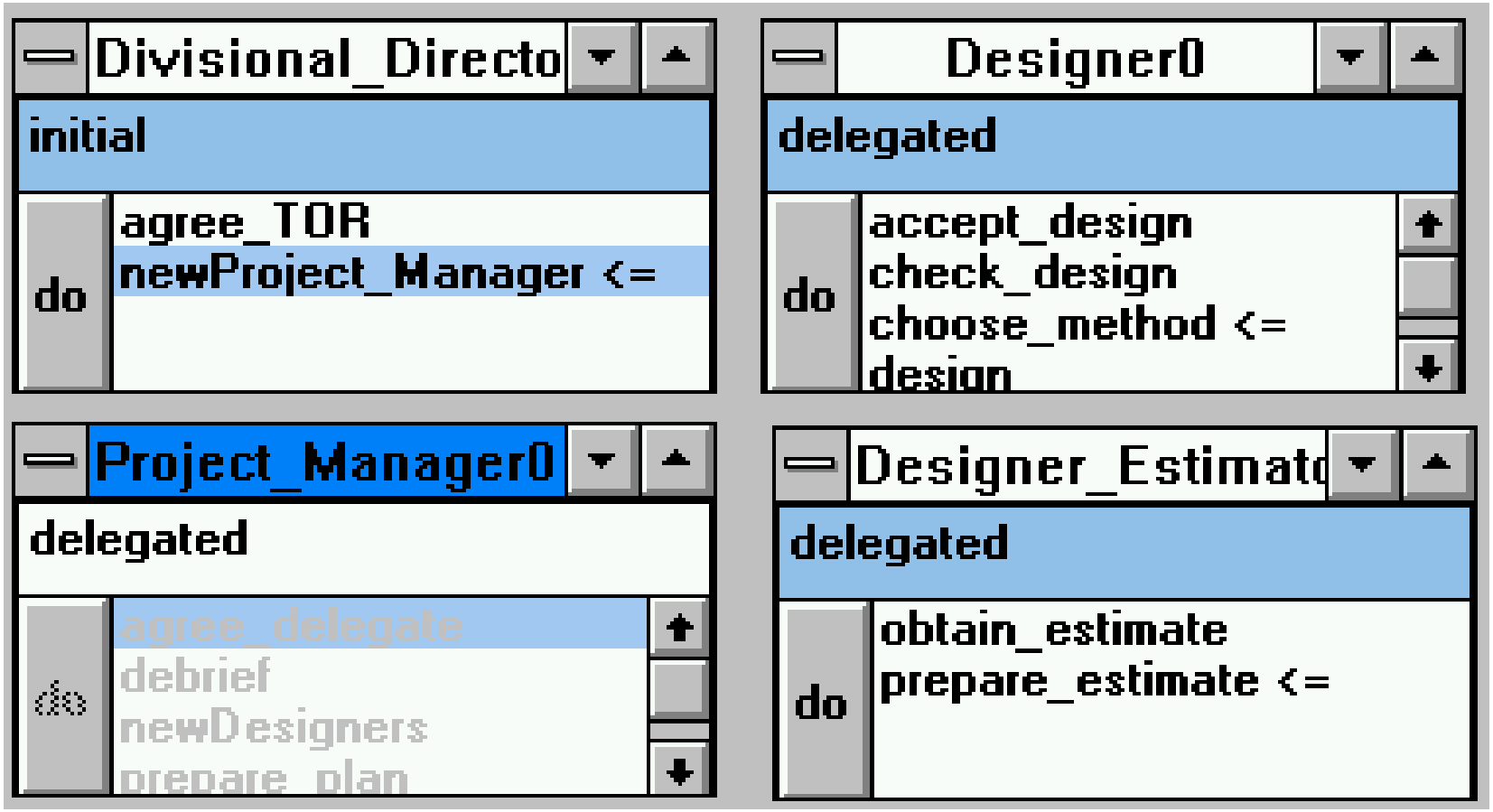
End


Divisional Director
initial
start new project manager
project manager started
Agree TOR for project
initial
Project Manager
initial
TOR agreed
start new designers
designers started
write TOR for designer
TOR written
delegated
Designer
initial
Agree TOR and delegate
delegated
choose a method
method chosen
ready to design
able to design
produce design
design produced
carry out design quality check
checking complete
design OK?
no
yes
Designer Estimator
initial
Agree TOR and delegate
delegated
prepare an estimate
estimate prepared
Obtain estimate
estimate sent
Give plan to designer
plan received
estimate received
prepare a plan
plan prepared
plan sent
deliver design
design delivered
produce project debrief report

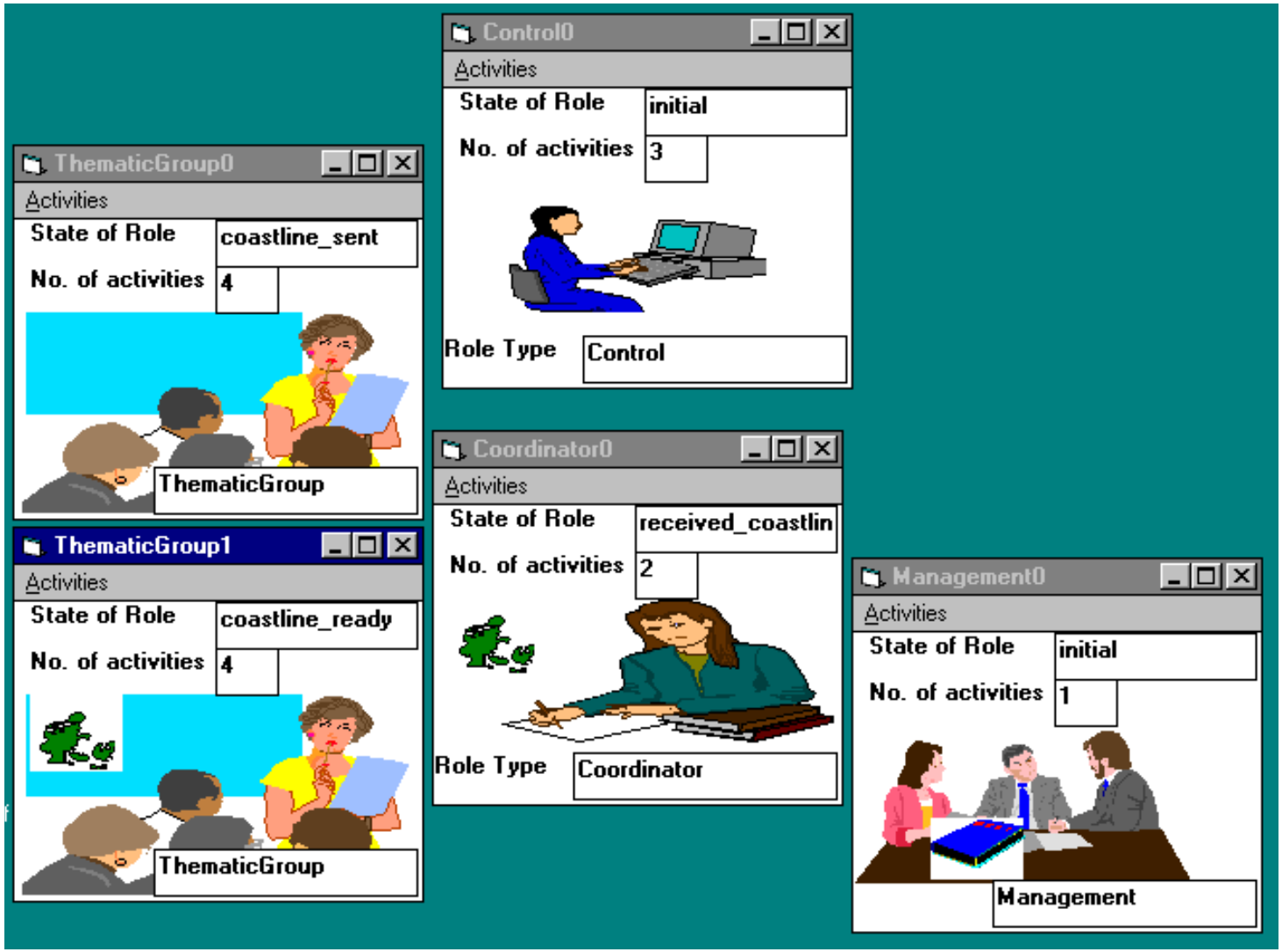
RolEnact Windows Interface



RolEnact for Designer



Other Interfaces for RolEnact



Recap

- Modelling Story: Commonly used notations for elicitation and validation.
- Coherent route from user facing models to experimentation and simulation.
- Further Work: Paradigms, Interfaces, Code generation. Development of Methods. Application to real processes.

Extra Slides

RolEnact detail

Project Manager

Create Project_Manager.newDesigners
me(agreed_TOR → designers_started)
new Designer
new Designer_Estimator
End

Action Project_Manager.write_TOR
me(designers_started → TOR_written)
End

Interaction Project_Manager.agree_delegate
me(TOR_written → delegated)
Designer(initial → delegated)
Designer_Estimator(initial → delegated)
End

Action Project_Manager.Prepare_plan
me(estimate_received → plan_prepared)
End

Interaction Project_Manager.send_plan
me(plan_prepared → plan_sent)
Designer_Estimator(sent_estimate → received_plan)
End

Action Project_Manager.debrief
me(design_received → project_completed)
End

Designer

Action Designer.choose_method

me(delegated → method_chosen)

End

Interaction Designer.ready_for_design

me(method_chosen → able_to_design)

Project_Manager.Designer_Estimator(received_plan → ended)

End

Action Designer.design

me(able_to_design → design_produced)

End

Action Designer.check_design

me(design_produced → assessing_design)

End

Designer

Action Designer.accept_design

me.(assessing_design $\rightarrow$ accepted_design)

End

Action Designer.reject_design

me(assessing_design $\rightarrow$ able_to_design)

End

Interaction Designer.deliver_design

me(accepted_design $\rightarrow$ design_sent)

Project_Manager(plan_sent $\rightarrow$ design_received)

Designer_Estimator

Action Designer_Estimator.prepare_estimate

me(delegated → estimated)

End

Interaction Designer_Estimator.obtain_estimate

me(estimated → sent_estimate)

Project_Manager(delegated → estimate_received)

End