RolEnact Process Modelling Language

Keith Phalp

Research Talk to the Declarative Systems and Software Engineering Group (DSSE)
Department of Electronics and Computer Science
University of Southampton, February 1997
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

Formal Paradigms

Enact Engine

CSP

RolEnact

CTS

POSD

MBaSS

RolEnact Windows

PWI

Presentation

Enaction / Support

RADs

IDEF

DFD

POSD
Divisional Director:
- new project approved
- start new project manager
- Agree TOR for project

Designer:
- choose a method
- prepare an estimate
- obtain estimate
- give plan to designer

Project Manager:
- start new designer
- write TOR for designer
- prepare a plan
- produce project debrief report

Decision Points:
- design OK? yes: produce design, carry out design, quality check, deliver design
- design OK? no: produce an estimate, choose a method, prepare a plan
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
RolEnact for Designer

Divisional_Directo

initial

do
agree_TOR
newProject_Manager <=

Designer0

delegated

do
accept_design
check_design
choose_method <=
design

Project_Manager0

delegated

do
agree_delegate
debrief
newDesigners
prepare_plan

Designer_Estimate

delegated

do
obtain_estimate
prepare_estimate <=
Extras: RolEnact for Designer
A Role: Director

Create Divisional_Director.newProject_Manager
    me(initial → manager_started)
    new Project_Manager
End

Interaction Divisional_Director.agree_TOR
    me(manager_started → initial)
    Project_Manager(initial → agreed_TOR)
End
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

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

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

Action Designer.reject_design
   me(assessing_design → able_to_design)
End

Interaction    Designer.deliver_design
   me(accepted_design → design_sent)
   Project_Manager(plan_sent → 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