Models to Support Changes to Local Government Processes

Keith Thomas Phalp,
Process Group,
Declarative Systems and Software Engineering,
Department of Electronics and Computer Science,
University of Southampton

Abstract: This report details the modelling undertaken by the Process Group of the Department of Electronics and Computer Science at the University of Southampton, as part of the GISIP[1] project. This modelling has enabled the Process Group to provide modellers with the advantages of understandable user-facing static models, as well as user-facing process simulations, whilst retaining the rigour of traditional executable process modelling languages. The methods outlined have been applied to a number of European business processes to support part of their process improvement programmes.

1. Introduction: User-Facing Process Models and Executable Process Models

Process modelling work, specifically the arguments for the adoption of particular modelling paradigms, can be characterised along an axis from rigorous formal descriptions to looser usually diagrammatic descriptions. At the extremes of this axis one may discern two distinct camps. These can be described as the process programming camp, and the pragmatic modellers.

Process Programmers

The first camp contains those who argue for rigorous program-like process models in order to be able to accurately describe, and experiment with, the logic of the process (Osterweil 86). An extension of this argument has been the development of executable process models. Executable models allow for process simulation, and experimentation with the logic of the process. For example, they enable the discovery of bottlenecks in the process, or in extreme cases the discovery of process deadlocks. Indeed, earlier work within the Process Group has used an executable stepper (Henderson 92, Henderson, 93) in order to be able to run models written in the language CSP (Hoare, 85).

More sophisticated executable models (like RolEnact (Henderson 95)) which we describe below) may also be used to provide mock-ups of process support. This use of executable

[1] The Geographical Information Systems Integration Process - project home page can be found at: http://www.dsse.ecs.soton.ac.uk/~kp/gisip.html

models as a step towards providing computer-based process support is becoming an increasingly important aspect of process modelling. Allied with the use of other enabling technology, such as GIS, (on which the process examples in this report are based) such process support for IT enabled change is becoming increasingly common.

Pragmatic Modellers

The opposite view might be considered that of the pragmatists (Phalp and Shepperd 94), who argue that it is more important for modelling notations to be understandable than rigorous. Typically, graphical models have been used to describe processes, so that users can more easily understand the models. Indeed, much process modelling in practice has so far taken such an approach (Phalp, 95).

A Combination Approach

Both of the approaches outlined above have their limitations. Notably, the pragmatic approach leads to the production of models based on looser semantics which therefore contain more ambiguity, and thus rely upon the naming rules and other heuristics to convey their meaning clearly. However, the more rigorous models usually rely on highly mathematical or program-like notations, which thus means that it is difficult to present them to users, and hence, validate that they are representative of the process under scrutiny.

Ideally one would wish to have an understandable graphical notation which had clear semantics underpinning it. In addition, one would still wish to have a notation which was executable so that process models could be run, in order to simulate the process, and experiment with process behaviour. One approach, which has been taken within the Process Group is to map commonly used user-facing process modelling notations to more rigorous notations; for example mapping a core subset of Role Activity Diagrams (RADs) to CSP communications Sequential Processes (Abeysinghe and Phalp 95). However, though this approach allows the user to have understandable models, and the modellers to be able to experiment with process simulations, this is only a partial solution to the problem.

2. Process Modelling: Aims of the Work

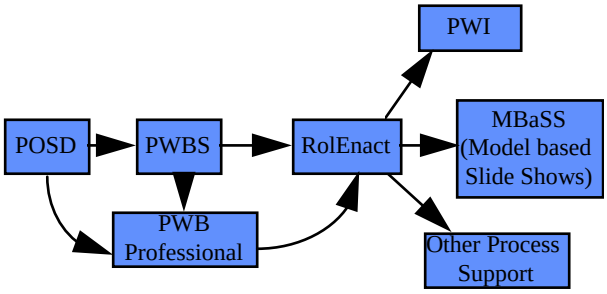
The aim of the work described in this report was to be able to have not only understandable notations for process elicitation and validation, but also to have executable process models which are suitable for presentation to users. That is, the work aims:

- To initially describe processes using a notation which users understand.
- To be able to experiment with processes using an executable notation.
- To be able to present static understandable models of the process solution to users.

- To use a structuring notation (Process Oriented Systems Design (POSD) - (Henderson and Pratten 95)) which simplifies the presentation of complex models into a more understandable hierarchical model.
- To be able to present executable models of processes to users, which they can understand, and with which they can experiment with the logic of the process.
- To be able to map between models.
- To be able to go through this process quickly enough to be able to provide feedback to users when they need it.
- To have a method which links these models into a unifying structure.
- To try out our methods on real business processes.

This report describes work which meets the above criteria, and presents examples of the application of these methods to Local Authority Business processes, in The UK and Denmark, as part of a European Commission funded collaborative project GISIP (Geographical Information Systems Integration Project - (GISIP, 95)). Four examples of the work, spanning three local authorities are presented. Between them these examples show how each modelling phase complements the other, and how this forms a coherent picture of a business process. However, the core aspect of the work, the production of a coherent set of graphical and executable process models, which are not only rigorous, but also presentable to non-technical users, is described by the first - somewhat longer - example process.

Modelling Methods in Place



Modelling Phases and their Relationships

This report briefly describes five modelling descriptions which we have used in order to meet the criteria expressed in the previous section. Of these descriptions, three notations (POSD), PWB Standard (ICL, 95), and the PWB Professional (ICL, 95) notation used are what might be commonly considered graphical modelling notations. RolEnact (Henderson, 95) is basically a process program, which can be executed on a computer in order to run process simulations. MBaSS, provides a more understandable interface to RolEnact models, by providing a slide show interface which is controlled by the underlying RolEnact process model.

3. Notations and Their Place in the Modelling Method

A brief description of each notation, and its use in the method now follows. This is intended only to give a flavour of the notation.

Although elements of the notation will be further described in presenting examples, it should be noted that it is not the purpose of this report to provide an introduction to these notations, and the reader will be referred to more exhaustive introductions where they are available.

POSD is a notation with single primitive (a behaviour) which can be decomposed into processes, objects, roles, and so on. Where behaviours touch this implies that they have some shared behaviour, typically an interaction, and this must be maintained consistently throughout the POSD model hierarchy. For a description of POSD see (Henderson and Pratten 95). The major advantage of POSD, is that it allows a model to be decomposed not only in terms of processes or activities, but also in terms of interaction. This alleviates the 'wire syndrome' so commonly found in process based notations, which often causes overly complex higher level diagrams.

A process modelling tool (ProcessWise WorkBench - PWB) (ICL, 95) has been used in order to produce the two types of graphical process model. The standard version of this tool allows one to produce standard (data flow like) models quickly, and to run some basic model checking for consistency and completeness.

The meta-model of the Professional tool used in this work is based upon a state-based condition-action paradigm developed within the Process Group, which is akin to a simplified Role Activity Diagram. This particular meta-model of the professional tool also enables one to automatically generate RolEnact code from these models. (Note that this meta-model and conversion was developed by a member of the ProcessWise Group at ICL). However, the modelling method presented does not rely upon this automatic conversion, and it is possible to write RolEnact relatively quickly without utilising these tools. Indeed, it is sometimes more appropriate to do so, particularly if one wishes to describe aspects of processes which require elements of other paradigms.

RolEnact is again a particular instance of a modelling language called Enact. Enact is used (within this context) as a process modelling engine, with which to run modelling paradigms. A stepper for CSP is a previous incarnation of a paradigm which also used Enact as the process modelling engine. RolEnact code has rigorous semantics - being based upon a state model of processes (for which a further diagrammatic representation is given) and can thus be executed. These executable models run under Windows, launching roles as independent Windows programs which communicate through the Enact DLL. This allows for rigorous experimentation with the process.

4. Example Processes and Modelling Solutions

Some examples of models showing how the various modelling phases or paradigms fit together are now described. The following models are presented:

- RolEnact Process Simulation for Example South Jutland Council (SJC).

  This represents the core of the modelling method presented in this report. The example shows modelling phases from the production of initial data-flow like diagrams, through the production of RolEnact process simulations, to executable Model Based Slide Show (MBaSS) models which can be used to present process solution to non-technical users. This will be described in section five.

- PWBS - POSD for Roskilde
- POSD to PWBS for Cannock Gateway

These two examples of portions of business processes show how two user-facing modelling descriptions can be combined, using data-flow like techniques (augmented with supporting roles) to describe process detail, and using POSD (briefly outlined above) to structure processes into a more understandable and hierarchic model.

In the first case a POSD model was produced after the production of PWB models in order to structure them more neatly. In the second case models produced in POSD were represented in PWB in order to check the POSD models for consistency and completeness. These two process examples will be described in section six.

- PWB Model of Generic Case Handling

  This model (described in section seven) shows an improved (re-engineered process) in one of the user facing paradigms (PWB Standard). Some business process metrics taken from one of the municipalities are shown. These metrics, and the possible effect of differing scenarios will be used as part of the argument for implementation of this proposed business process change.

5. Moving from Process Description to Process Enaction: Process Simulation for Example South Jutland Council (SJC) Process

In moving from process elicitation to process simulation the following modelling phases are undertaken:

Create PWB Model - Identify Roles
Create Role Model
Create RolEnact
Experiment with Process Logic in RolEnact

Create Visual interpretation of process
Present / Repeat

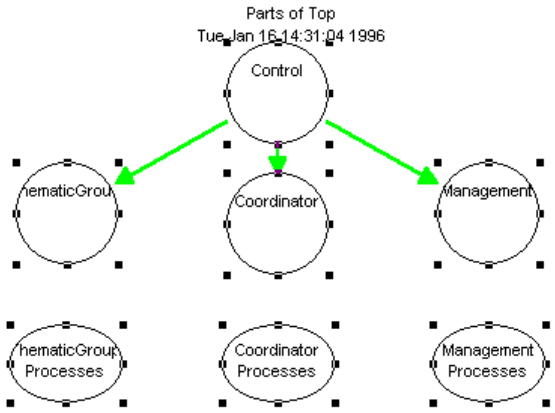
These phases will now be briefly described for the example SJC process.

- Create PWB Model - Identify Roles

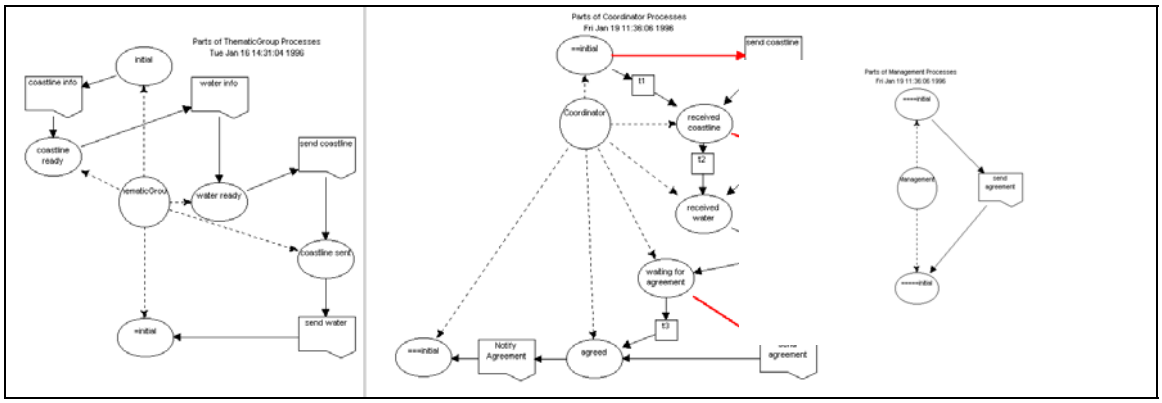
In this case the data flow models were produced by Danish consultants working closely with the municipality. However, the Process Group did run some basic consistency checking exercises on these models, by utilising one of the simulation features of the standard version of ProcessWise WorkBench (notably static volume analysis).

- Create Role Model

In order to create an example process simulation an initial simplified role model was produced. This hierarchical model will now be briefly explained.



The top level, shows four roles (represented by circles). There are three roles which come from the SJC processes. A Thematic Group role (who produce documents on various environmental themes for a plan), a Coordinator role (responsible for co-ordinating inputs form the group and producing a report), and a Management role (who assess the political implications in order to decide whether to agree to the plan). In addition, there is a fourth role which corresponds to the control role of the simulation. Each SJC role has an associated expandable behaviour. These behaviours are then expanded in order to show the details of how the roles behave (the activities which they take part in, and the states in which they can be) and how they interact with each other.



The above figure shows the expansions of the behaviours of three roles. Each ellipse is a state in which the role may exist. These states are connected to the role via dotted arrowed lines. The role moves from state to state via an event, or activity (shown as a box). The movement from one state to another, via an event, is shown by an arrowed lined. A further bolder arrow shows the mechanism for shared interactions. This is based upon the idea of a channel model. For example, the co-ordinator sends out a channel 'send coastline'. This channel waits for a send coastline event from the Thematic Group (formally one would say that an interaction occurs between these two roles, in that they have a shared event; using a channel is a mechanism which allows for this to be simulated dynamically). On receipt of information (in our model as a result of the event) both the Thematic Group and the Coordinator move to the next state. That is, that the result of an interaction (or shared behaviour) is that both roles move to their next state. (This is consistent with commonly used Role based process modelling diagramming techniques such as Role Activity Diagrams (Ould 95)).

The exact behaviour of the process can be more easily seen by examination of instances (screenshots) of the model based slide show (MBaSS) simulation, and thus the thread of control through this process model will not be further examined at this stage. However, it is worth noting that this process model has been written in a particularly restrictive way (taking the process description literally), in order to illustrate the fact that the co-ordinator is a potential bottleneck (see the later discussions of the scenario). In addition, the process simulation can also be used in order to show that a choice needs to be made about whether the thematic group can produce and send different kinds of information in parallel or (as in this case) only in a particular sequence. Again these issues will be made clearer by examination of the more visual representations of this model. This is, after all, the reason for having such later models, to present back to non-technical users who are less familiar with the formal descriptions utilised by the modellers.

- Create RolEnact

As noted in the introduction, RolEnact is only one instance of a family of models which can be created (using a paradigm) file to run dynamically using Enact as their process enactment engine. It should also be noted that even RolEnact has a number of constructs in addition to those which are represented in the Role Model notation described above.

For example, it is relatively easy to add ‘and’ and ‘or’ constructs to RolEnact code. Thus, the models produced, and the simulations produced from them are really a subset of RolEnact, utilising only a core set of constructs. However, by using Role Model diagrams RolEnact code can be generated automatically from the diagrams (literally at a click). This has the following advantages:

1) It provides a fast track route from diagrams of process to executable models which may be experimented with. This is akin to very rapid prototyping of process solutions.

2) It provides a direct mapping from one representation of the model to the next, thus ensuring consistency.

3) It provides a reliable framework (or code stub) which can then be used as a base model to add further RolEnact features.

Again the exact nature of RolEnact code is not within the scope of this report. However, for completeness, a description of the simple process model in RolEnact is included below.

Declaration of Roles:

```
class ThematicGroup < Role.
class Coordinator < Role.
class Management < Role.

management():=role(new Management).

class Control < Role.
```

Events which roles carry out, have a ready condition, to check whether they may take place and a ‘doit’ which shows the effect of the event.

```
Events: Ready and Doit
send_water:=event(ThematicGroup).
send_water.ready(aThematicGroup):=
    Coordinator.exists('received_coastline) and
    aThematicGroup.inState('coastline_sent).

send_water.doit(aThematicGroup):=
    (
    aCoordinator:=Coordinator.choose('received_coastline);
    aThematicGroup.vCoordinator:=aCoordinator;
    aCoordinator.vThematicGroup:=aThematicGroup;
    aThematicGroup.vCoordinator.setState('received_water);
    aThematicGroup.setState('initial)
     ).

send_coastline:=event(ThematicGroup).
send_coastline.ready(aThematicGroup):=
    Coordinator.exists('initial) and
    aThematicGroup.inState('water_ready).


send_coastline.doit(aThematicGroup):=
```

```
    (
aCoordinator:=Coordinator.choose('initial);
aThematicGroup.vCoordinator:=aCoordinator;
aCoordinator.vThematicGroup:=
                  aThematicGroup;
aThematicGroup.setState('coastline_sent);
aThematicGroup.vCoordinator.setState
                  ('received_coastline) ).

notify_Agreement:=event(Coordinator).
notify_Agreement.ready(aCoordinator):=
    aCoordinator.inState('agreed).
notify_Agreement.doit(aCoordinator):=
    (
    aCoordinator.setState('initial)
    ).

produce_report:=event(Coordinator).
produce_report.ready(aCoordinator):=
    aCoordinator.inState('received_water).
produce_report.doit(aCoordinator):=
    (
aCoordinator.setState('waiting_for_agreement)
    ).
coordinator():=role(new Coordinator).
```

For the purposes of simulation, link roles to an inital control role. The conrol being the fist role to be launched, so that others may be launched from it.

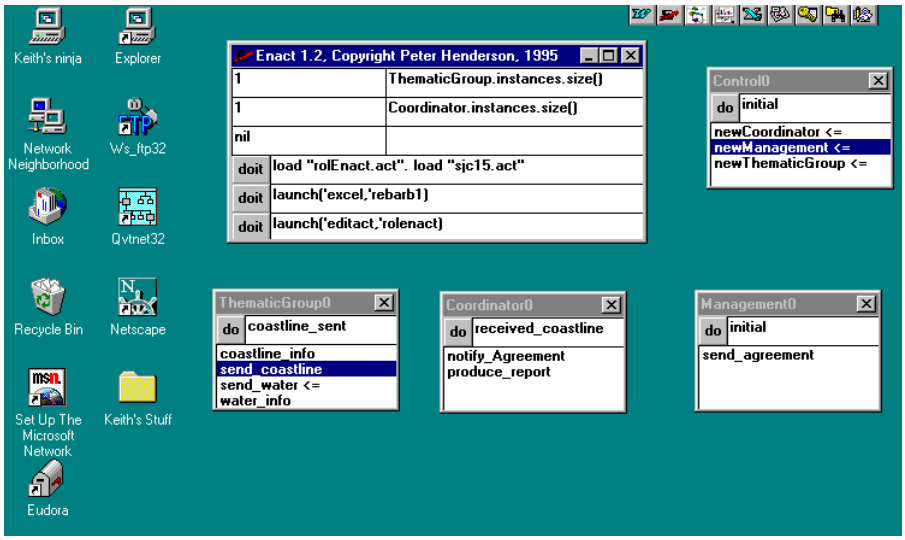
```
newManagement:=event(Control).
newManagement.ready(aControl):=true.
newManagement.doit(aControl):=
    launch('role, Management).

newCoordinator:=event(Control).
newCoordinator.ready(aControl):=true.
newCoordinator.doit(aControl):=
    launch('role, Coordinator).

launch('role,Control).
```

- Experiment with Process Logic in RolEnact

The point of creating RolEnact is not to examine code, but to play with a simulation, and thus experiment with the process, before committing to expense process redesign. This can be done either with the core representation of RolEnact, or with the later more visual representation. It is expected that these two representations may be used by different classes of users. The experience (and expectation) of the Process group, is that bare RolEnact may be used (often within the group or by more technical users) to experiment with process change incrementally and very rapidly. For example, one might go through a number of phases of drawing models, creating RolEnact, experimenting with process, finding problems, going back and changing the Role Model, running a further simulation, and so on, within a couple of hours. Having reached a more stable state MBaSS models would then be used to create a model which would be presented to users, for their

experimentation and approval. This might then necessitate another cycle of prototyping the process model until agreement was reached.

Below is a single screenshot of how RolEnact appears in its core form.



The large rectangle centre screen is the Enact Control, which starts up the simulation, and is not itself part of it. The smaller rectangles are the roles which have been launched.

These are actually separate windows programs which each check their own state when any changes are made to the state of the Enact DLL. When any role takes part in an event this changes the overall process, and may change the state of other roles as well. Hence each role checks their own state when any other changes state.

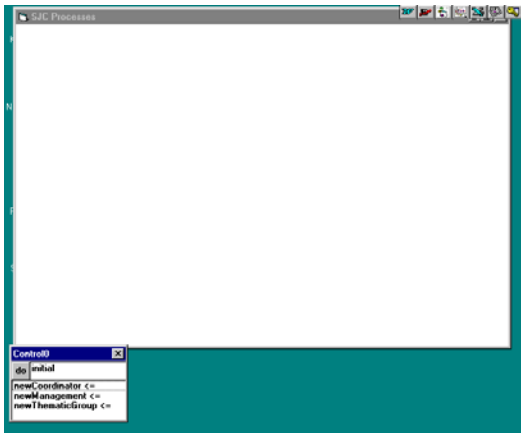
The first role launched is the control, which has subsequently launched three other roles, ThematicGroup, Coordinator and Management. Each role is divided by a thick line. Above the line is the do button, and a field showing the current state of the role. Below the line is a list of events in which the role may participate. However, those events which are allowable at the current time, are shown with an arrow to their right. For example, the diagram above shows that Thematic Group can only participate in the event 'send_water'. Furthermore, it can be seen that both Coordinator and Management Roles cannot participate in any of their events. This is because Coordinator is awaiting further input from the Thematic Group (note that the Coordinator state is 'received_coastline', and thus the Coordinator also wishes to receive water), and the Management role is awaiting the report which will be produced by the Coordinator. It should be apparent that it possible to run through a number of process scenarios using this executable model, and thus have a greater process understanding, and a clearer idea of how the process might be supported by Information Technology.

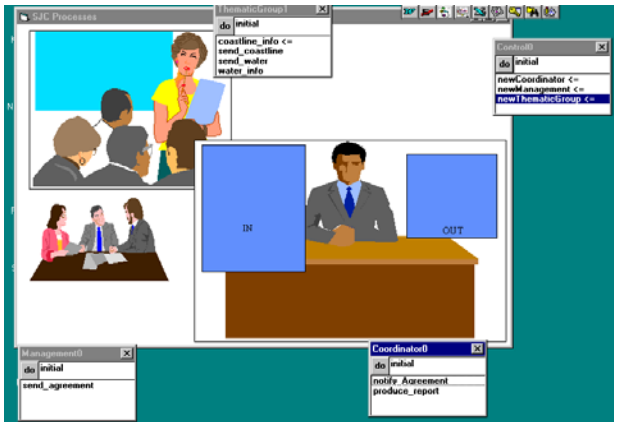
- Create Visual interpretation of process

Just as Enact itself has a number of incarnations, there are also a number of visual interpretations which communicate with RolEnact. The MBaSS model is intended to be akin to a slide show of a process thread (based on an idea presented by Skankort) but with the added advantage of choosing what to do next, and being able to experiment with the process. This is possible because the appearance of the images on screen is controlled by the RolEnact process model.

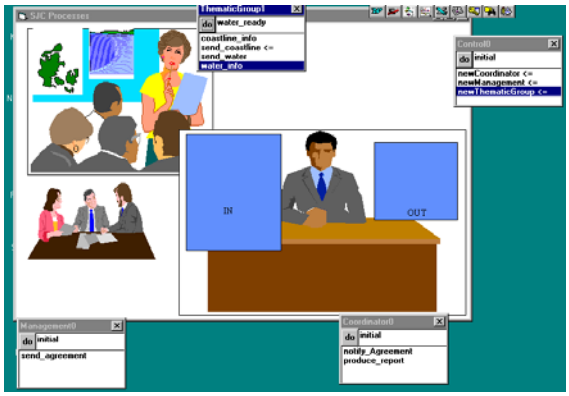
A simple scenario will now be outlined, showing some of the screenshots of the simulation. Note how this may be related back to the Role Model diagrams described previously.

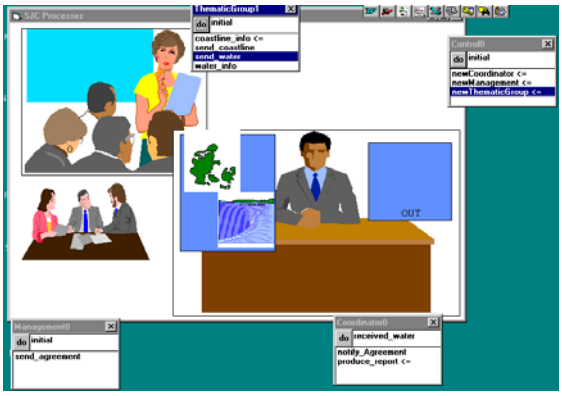
Initially the control role is launched. The associated scenario screen shows no roles. The three roles are then launched. ThematicGroup (in mid-presentation) appears top left, Coordinator (sitting at his desk) bottom right, and Management (in a meeting) bottom left. The associated RolEnact Windows show the states of each role. All roles are launched in an initial state.



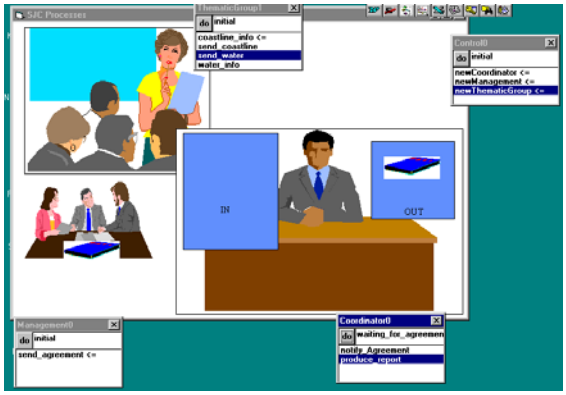


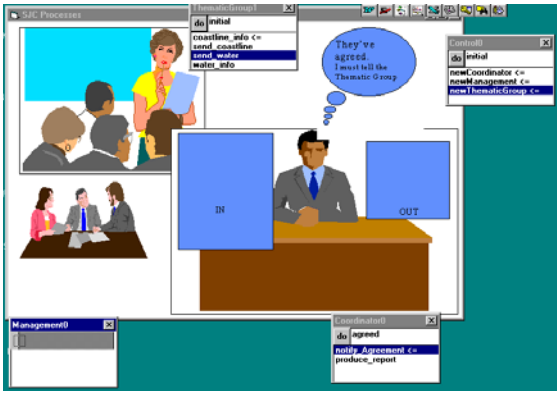
The only allowable action (other than the control role creating new instances of roles) is for the Thematic Group to produce ‘coastline_info’ - shown by the arrow adjacent to this event. Having done this, the Thematic Group will move to their next state ‘coastline_ready’. This will result in allowing their next event ‘water_info’. Having ‘done’ this event the Thematic Group role moves to the state ‘water ready’, and the event ‘send coastline’ becomes allowed. This is the state of the process as represented by the third (following) scenario screenshot.





Note that the production of water and coastline information can be seen by pictures of a coastline (in this case Denmark) and a water feature appearing on the board upon which the ThematicGroup members are focusing their attention. However, once the ThematicGroup role takes part in its next two events, namely 'send_coastline' and 'send_water', these pictures move across to the in-tray of the Coordinator (scenario screenshot four). The states of the RolEnact windows also reflect these changes, since Thematic Group now returns to its initial state and the Coordinator (having been furnished with the requisite information, being in the state 'received water') is now allowed to 'do' the event 'produce_report'. This event results in the disappearance of the pictures of coastline and water, and the appearance of a report in the Coordinator's out tray, (now in the state 'waiting for agreement') and upon the desk of the role Management (scenario screenshot five - below).

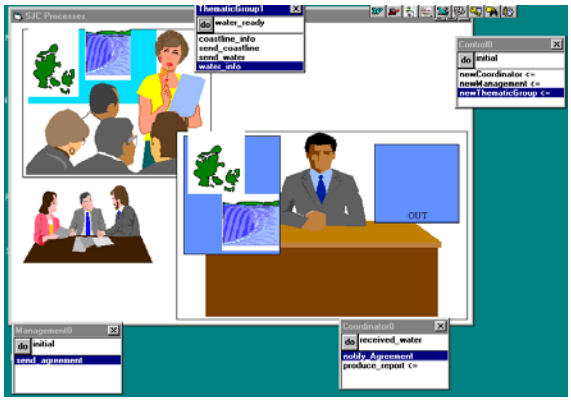


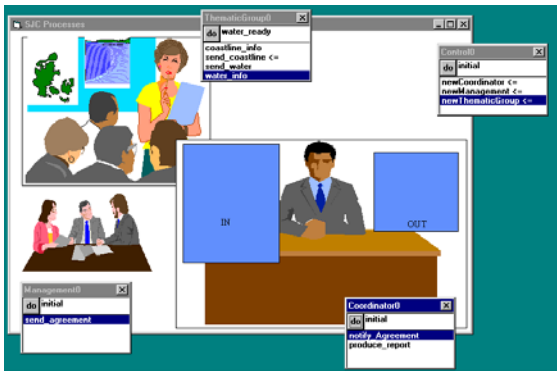


Management are now able to 'send_agreement' to the Coordinator which results in the Coordinator being in the state 'agreed', and being allowed to do the event 'notify' agreement. (Scenario screenshot six shows the Coordinator considering his next event).. After this 'notify_agreement' event all roles would be back in their initial state.

Bottlenecks

The scenario above shows a single process thread, and thus does not show the need for driving the screen with a process model. However, one of the benefits of running a scenario on a computer is that it can illustrate that bottlenecks occur when certain choices are made. The following screenshot (scenario screenshot seven) shows the effect of such choices, and illustrates that the Coordinator is a potential bottleneck. Having produced 'coastline_info', and 'water_info', and having sent these to the Coordinator the Thematic Group have immediately produced another plan with its associated 'coastline_info' and 'water_info'. The screen shows pictures of coastline and water features, both on the Thematic Group's wall-chart, and in the Coordinator's in-tray. Furthermore, it can be seen from the RolEnact windows that the ThematicGroup are unable to do any further events, since they cannot send on water or coastline to the Coordinator.





The only way out of this impasse is for the Coordinator to 'produce_report', and for Management to 'send_agreement' (to the report), and for the Coordinator to 'notify_agreement' and hence return to the state 'initial'. Only then will the ThematicGroup have an allowable event 'send_coastline' (scenario screenshot eight).

This is perhaps an exaggerated process example. However, without running through such process simulations it is quite possible to make similar errors in process redesigns. When such errors are compounded by supporting process support tools (enabling Information Technology) only allowing process users to take part in certain activities, the consequences can be significant. Hence, the need for presenting executable process scenarios to users, in ways in which they themselves can experiment with models and understand more fully the consequences of possible process redesign.

An Extension

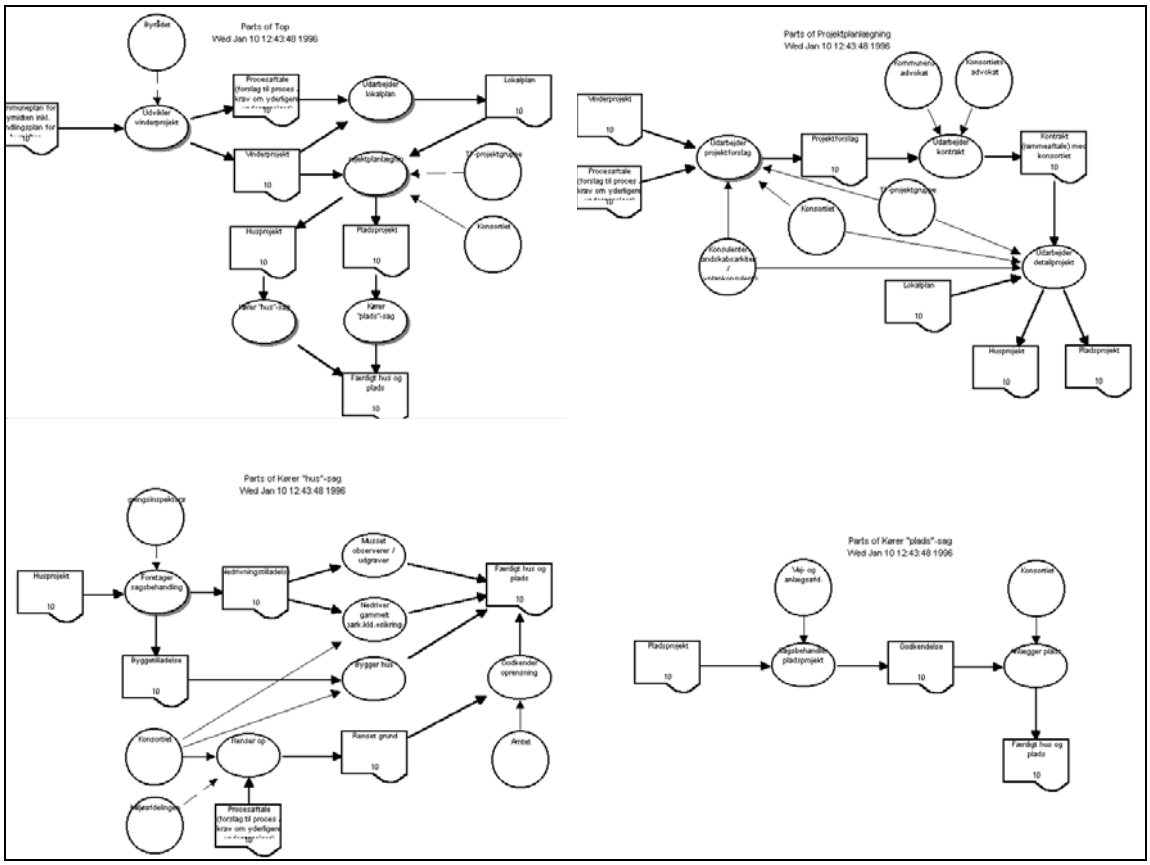
It is possible to adapt the visual interpretation shown so that once roles are launched it is this which calls the underlying RolEnact model, without using the Core RolEnact as detailed above. For example, we may have a pop-up menu which appears when the mouse is dragged over a particular role, and thus remove the need to show RolEnact boxes corresponding to each role. However, this to some extent hampers the speed with

which modellers may iterate round the cycle of process experimentation, process change, and further experimentation. Nevertheless, once the process has reached a relatively stable state, this is a viable option for presentation back to process users.

6. Linking Notations (PWB to POSD)

As noted earlier, this report is not intended to be an introduction either to modelling in the kind of notations used by PWB, nor an introduction to POSD. The following sub-sections will thus very briefly present portions of the models of respective business processes, and then attempt to draw some general conclusions about the need for a link between these two notations. Further work will concentrate on the details of mapping between these notations.

6.1 PWBS Roskilde - POSD Roskilde



Initial diagrams, in Danish were partially translated, and then represented in POSD. This representation used POSD to structure the model into five top level behaviours, arranged to show interactions between touching behaviours. These five top-level behaviours were then expanded, and a data-flow notation shows the details of each behaviour in turn, and its interaction with adjacent touching behaviours. The PWB diagrams above are those which were sent back to the Danish contributors. These diagrams are complete, in that

they balance properly across all levels, but the named detail is not important in order to understand how POSD works. It is interesting that the omission of a link was found by modelling with POSD (see the POSD diagram 'Project Planning'). A link was then added to the PWB diagrams in order to make them complete. The successful running of a simple (static volume) analysis, confirmed this completeness, and these are the diagrams shown above.

POSD for Roskilde

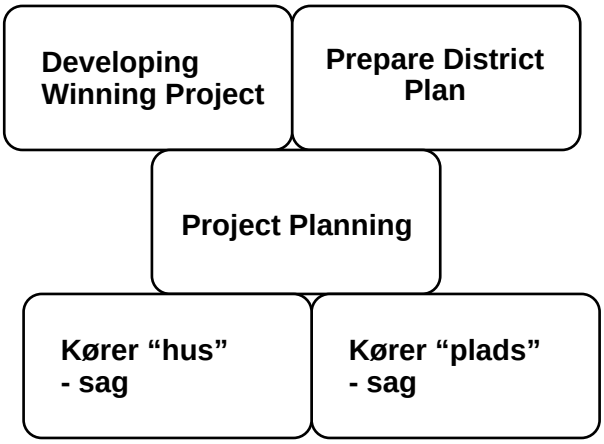
The following set of POSD diagrams relate to the associated PWBS diagrams which are part of the business process of the Municipality of Roskilde.

This example was meant to be mainly illustrative, and was presented back to the Danish partners, in order to show how POSD could be used to structure the diagrams produced in PWBS. Translation was incomplete and was by use of a dictionary only, and it was felt that some confusion over terminology might arise. However, in presenting these models back, not only to the consultants who had produced the PWBS diagrams, but also to representatives of the municipality, it was felt that the POSD depiction did accurately reflect the process under scrutiny in a simple and understandable way.

It is also worth noting that the production of POSD diagrams led to the discovery of some inconsistencies in the PWBS diagrams upon which they were to be based. That is, that in ensuring that touching behaviours have interaction or shared behaviour some gaps became apparent. For example, the link between project planning and Kører "hus" sag and Kører "plads" - sag seen at the top level of the PWB hierarchy was missing lower down.

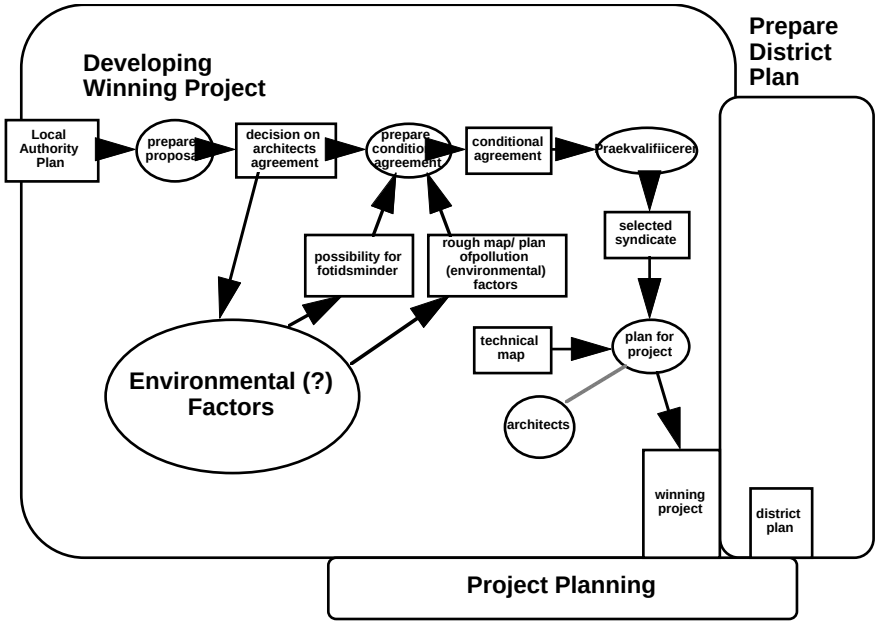
Note also the way that interactions between behaviours are hidden until encountering lower levels of process detail, where is it is more appropriate to show the interaction in similar detail. This is not possible using traditional notations (like PWB) which only allow one to decompose processes. Even for this very simple example process the top level PWB diagram is more complex, and thus more difficult to comprehend, because of the flow of business objects from process to process. Compare this with the simplicity and clarity of the equivalent top-level POSD, which shows just five touching behaviours.

Parts of Top for Roskilde

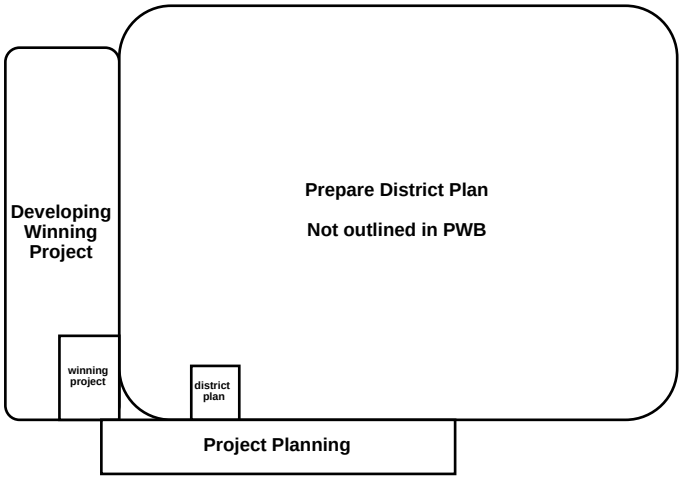


Each of the top level behaviours is exploded in turn. The detail of the behaviour is then modelled, as are the interactions with its touching behaviours. In this case the detailed modelling used a data flow notation augmented with roles, in order to be mapped to the PWBS diagrams on which the model is based. However, it is possible to explode POSD behaviours into a number of different modelling notations, and this is one of the strengths of POSD.

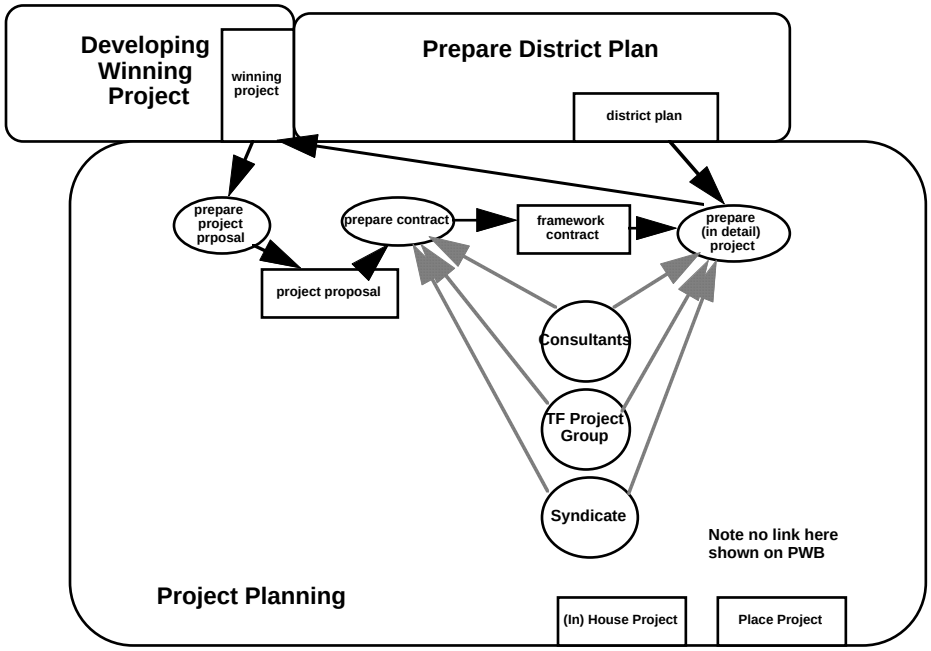
The explosion of 'Developing Winning Project' shows not only the behaviour in more detail, but also clearly maintains the interaction between 'Developing Winning Project' and its adjacent touching behaviours.



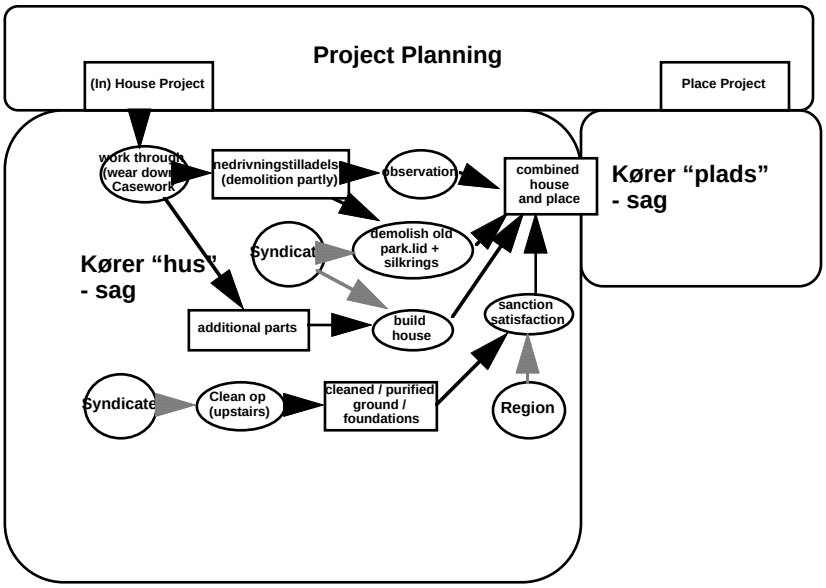
This is continued for the other behaviours, so that a single behaviour is always expanded, and the interaction with its touching behaviours is also shown.



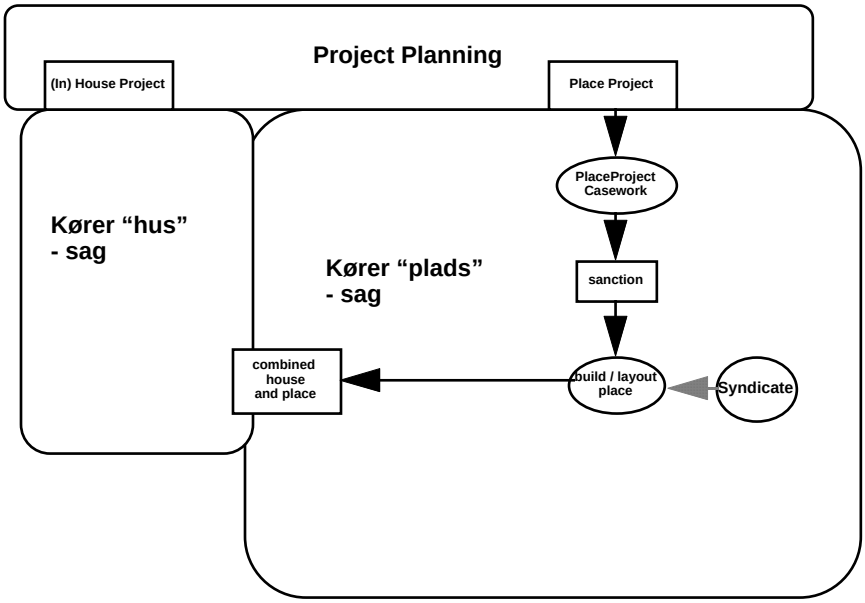
The expansion of 'Project Planning' shows how there has been some interaction overlooked in the original PWBS model, since there is an output to 'Korer "hus" sag' and one to 'Korer "plads' sag", but no detail of where these come form within the project planning behaviour.



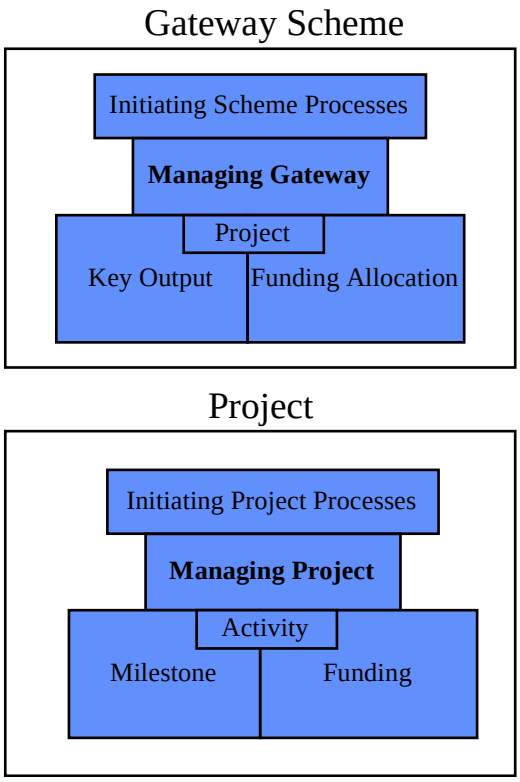
The need for interaction between these behaviours is again made clear by the examination of the expansion of both 'Korer "hus" sag' and 'Korer "plads" sag', since they both show connections back to 'Project Planning'.



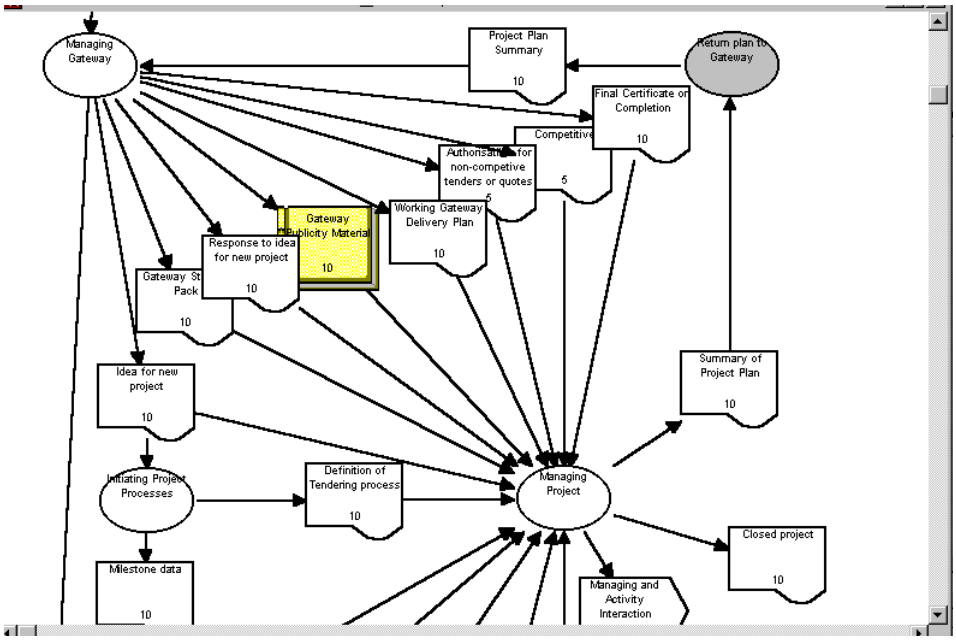
Note that by maintaining the topology of adjacent behaviours it is easier to discover inconsistencies, where behaviours do not share interaction.



6.2 Cannock POSD to Cannock PWBS



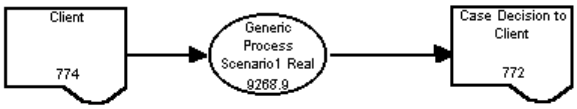
This process example, was worked in the reverse way to Roskilde, in that a portion of an original huge POSD model was expressed in PWB. The reason for doing to have a further check of the completeness and consistency of the model by running some analyses. The key point to note is that the POSD representation of this very complex business process allows the interaction to be decomposed and shown at lower - appropriate - levels of detail. Hence, the POSD diagram is easily understood and digested, even by non-technical users. This problem of wire syndrome is apparent even by looking only at a portion of the (simplified) top level of the PWB diagram. Despite the fact that there are only three processes shown on this diagram, the picture is very busy. This busyness is caused by the fact that there are so many interactions between the two core behaviours of ‘Managing Gateway’ and ‘Managing Project’. In the POSD diagram, this interaction can be decomposed, and shown where appropriate, since all the modeller needs to do is to place the two behaviours adjacent and touching to represent that they have some shared behaviour (often lower level interactions).



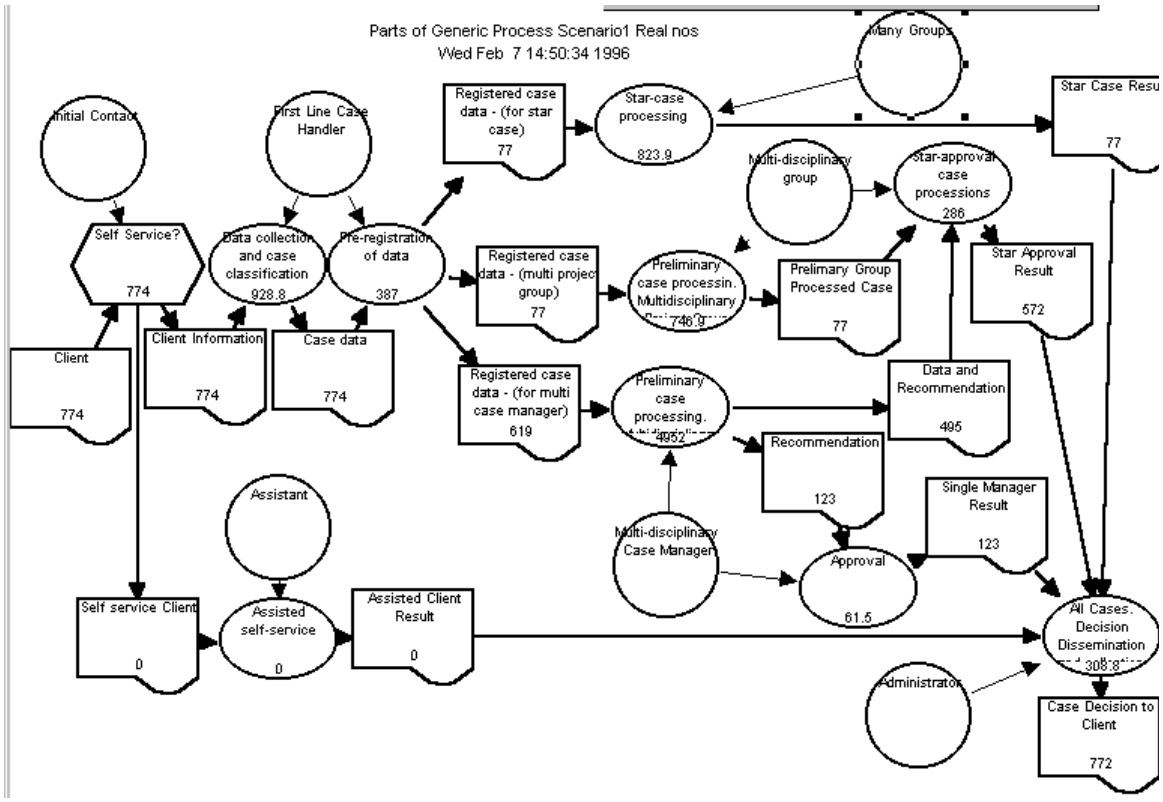
7. Generic Models of Case-Handling

The business process modelling carried out by the Danish consultants (Skankort - JBP) as part of the GISIP project has identified areas in the case-handling procedures of local authorities, where some improvements may be made. A new case -handling process has been proposed by the Danish consultants, and agreed across all participating partners. This process is possible because the use of GIS means that it is possible for upstream process contacts to have information earlier, and that information can be more easily accessed by single individuals. In the extreme this means that some cases may be handled by a single multi-disciplinary case manager, whereas previously they would have involved a number of specialists groups.

The PWB models below show the costing of data which will be used to argue the efficacy of implementing the new redesigned generic case handling process. The first model shows a PWB equivalent of the generic case handling process with numbers taken from the study of current authority processes. This model has been split into two hierarchical layers.

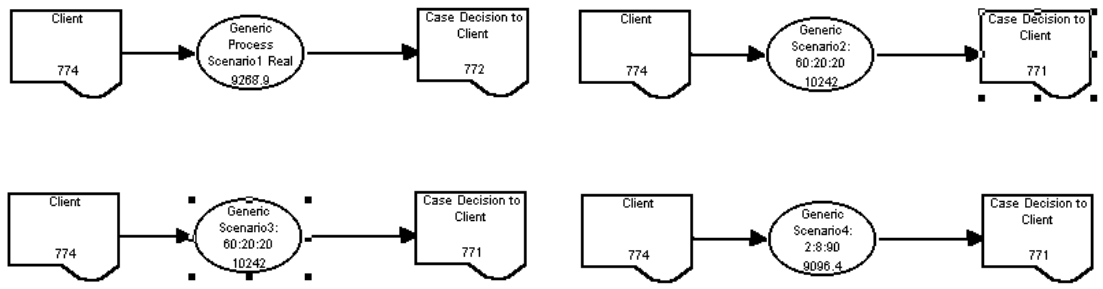


The upper layer shows a single process bubble for the case handling process with a figure representing the total time that the process uses. The lower layer shows the process in greater detail. Each lower level process (represented by an ellipse) has a worktime and a volume. Each business object (represented by roughly rectangular document shape) has a volume. The worktime is the time to process a single volume (a case). Hence, the product of worktime and volume will give an overall processing time used by that process. The business object volumes can be seen entering and leaving processes, and being distributed in different ratios. The sum of all of these overall processing time figures, gives a total processing time for the case handling process (as shown in the high level process bubble).



The amount of clients or customers going down a particular process stream is represented by the ratios of business objects. It is these ratios which may change, and which account for the different scenarios which are presented. For example, the scenarios give a feeling for the impact of greater numbers of clients using a single multi-disciplinary case handler as opposed the traditional multi-group (star case) processing or vice-versa. In essence the scenarios change the percentages going in one of three streams, and the name of each top level process includes this ratio. Rather than present the detail of each scenario one may just show the top levels to users, with the total figure, and explain how the ratios relate to the volume streams. For example one might present the following scenarios as:

Scenario 1: Real numbers: volumes split in ratio 10:10:80 (also shown above).
Scenario 2: volumes split in ratio 60:20:20
Scenario 3: volumes split in ratio 80:10:10
Scenario 4: volumes split in ratio 2:8:90



Hence, not only have modelling methods been used which allow for the elicitation of process, experimentation with process redesign, and presentation to users of process scenarios, but also to gauge the quantitative impact of process change. Furthermore, by integrating the business process metrics with the process model, it is far easier for users to understand where, and why to collect these measures, and to feed the results back for future process change.

8. Conclusions

Rather than develop new paradigms throughout, this method has chosen to use some notations which are commonly used for process elicitation and which users can easily understand. The value of the work is in mapping these kinds of notations to more rigorous representations of process (developed within the process group) which allow the creation of executable models. This provides a coherent route from user-facing models through to process simulations. The development of (MBaSS) models which provide accessible process simulations to non-technical users extends the mapping to provide an executable user-facing modelling solution.

In addition, the use of POSD in order to provide a framework for other notations has also been described. This allows complex models to be decomposed into understandable structures. Finally, an example of process redesign has been presented, which again used an appropriate part of the method to present scenarios to business users.

Further work within the Process Group is divided into parallel strands. The group will endeavour to develop better modelling paradigms to run using Enact as a process engine. Different interfaces will be developed both for existing and new modelling paradigms. Finally, as with the work described above these process technologies will be applied to real business processes, in order to assess their efficacy, and to further refine and develop these tools.

References

Abeysinghe, G.A. and Phalp, K.T. Comparing Process Modelling Methods, submitted to Information and Software Technology, currently available at: http://dsse.ecs.soton.ac.uk/~ga/process.html

GISIP: Geographical Information Systems Integration Process: Home page at: http://dsse.ecs.soton.ac.uk/~kp/gisip.html

Henderson, P. The CSP Stepper in Enact - an executable specification, July 1992, available by ftp at: http://dsse.ecs.soton.ac.uk/~ph/cv.html

Henderson, P. Object-Oriented Specification and design with C++, McGraw-Hill, 1993.

Henderson, P. Modelling Process Support in Enact, November 1995, available as a Word document or in postscript at http://dsse.ecs.soton.ac.uk/~ph/cv.html

Henderson, P. Enact User Manual, April, 1995, available as a Word document or in postscript at http://dsse.ecs.soton.ac.uk/~ph/cv.html

Henderson, P, and Pratten, G. POSD - Process Oriented System Design, Jan. 1995, available at http://dsse.ecs.soton.ac.uk/~ph/cv.html

Hoare, C.A.R. Communicating Sequential Processes, Prentice-Hall, 1985.

ICL ProcessWise WorkBench User Guide, PWB/usrguide/S5.4, International Computers Limited, August 1995.

ICL ProcessWise WorkBench User Guide, PWB/Guide/5.3, International Computers Limited, May 1995.

Osterweil, L.J. Software Processes are Software Too, in Proceedings of the Third International Software Process Workshop, Breckendridge, Colorado, IEEE Computer Society Press, 1986.

Ould, M.A. Business Processes: Modelling and Analysis for Reengineering and Improvement, Wiley, 1995.

Phalp, K. and M. Shepperd, A Pragmatic Approach to Process Modelling, Proceedings of the Third European Workshop on Software Process Technology, Vilard de Lans, Feb. 94, edited by B.C. Warboys, in Lecture Notes in Computer Science, vol. 772, Springer-Verlag 1994.

Phalp, K.T. An Investigation of Process Modelling in Practice, PhD Thesis, Bournemouth University, 1995.