

The CAP framework for business process modelling[☆]

Keith Thomas Phalp*

Empirical Software Engineering Research Group, Department of Computing, Bournemouth University, Talbot Campus, Poole House, Wallisdown, Poole, UK

Received 3 June 1996; accepted 5 May 1998

Abstract

Business process modelling is an area of work that is increasingly used in conjunction with software development. For example, many development methods note the importance of strategic or business modelling, typically as a prerequisite to analysis. In addition, Systems Engineering for Business Process Change suggests the need to model the business process in maintaining and evolving existing (legacy) systems.

In order to model business processes, one needs to consider what notations are most suitable, and what methods to adopt. However, the most appropriate notation typically depends on a number of contextual issues, the purpose of the modelling, the audience for the models and so on. Furthermore, this context changes with the progress of the modelling. Hence, the modeller needs guidance about appropriate approaches at different points in the modelling programme.

This paper introduces a framework for business process modelling that provides such guidance without prescribing particular notations. This is achieved by describing business process modelling in terms of three iterative and generic categories or phases: Capture, Analysis and Presentation. The paper shows how different kinds of notational approaches can be used within these categories, discussing the choices available to the modeller. The (CAP) framework is generally applicable, and is illustrated both by a simple theoretical example, and by examples from industrial business process modelling. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Business process; Framework; Methods; Enaction; Measurement

1. Business process modelling and software development

Process modelling is an activity that is familiar to many software engineers. Models of software development have been proposed in order to understand, guide and control the software process. However, this paper specifically concentrates on models of business processes, and on describing a framework for developing them. Hence, this first section argues why software developers should need to consider business processes and how this may be used to improve the software development process.

1.1. Software as a business process support system

Software developers are becoming aware of the need to understand and model the business processes of their clients, that is the organisations for whom they build software

systems. The business process is of relevance because software development may be viewed as producing support systems for business processes. Some of these business processes are easily discerned, for example, a Point of Sale system clearly supports a retail business process. However, many other systems also support processes, which may be viewed as carrying out some business purpose. For example, an Air Traffic Management system supports the work of Air Traffic Controllers (among others) who are in the business of ensuring the safe conduct of air traffic. Two important ways in which business process modelling has an impact on software development will now be briefly described. These are areas in which there are significant research efforts from both industrialists and academics, and which specifically address the modelling of client business processes and its impact upon software development.

Strategic modelling is a term used by a number of software development methods [2]. The term refers to those activities before analysis, which are used to understand, describe and validate business needs and the business context for the proposed system. The output from this strategic modelling may also be used within analysis and design. For example, role models, a common output of such strategic

* Tel.: 44 1202 595133; fax: 44 1202 595314; e-mail: kphalp@bournemouth.ac.uk

[☆]An overview of this framework was presented at the Fifth International Conference on Re-Technologies for Information Systems in December 97 [11], and the paper presented here shares the same industrial example.

modelling, are helpful in identifying candidate objects, and the likely interactions and dependencies among them. In addition, enactable versions of such role models may also be used to further experiment with the behaviour of the proposed business process, providing executable specifications of business processes [3]. Hence, business process modelling aids the software developer, by helping to reduce the problems associated with the elicitation of systems requirements.

Strategic modelling suggests the use of models to understand the context for the development of new systems. A further use of business modelling is to tackle the problem of changing or evolving existing (legacy) systems. Here, the existing system supports the current business process and the client intends to make changes to that business process. The hypothesis is that by understanding the relationship between the business process and the legacy system, such changes may be better compared, gauged and managed. For example, a change that affects only a small part of the business may require changes throughout the supporting system. Hence, a number of projects have suggested modelling the business process and the legacy system and then constructing a mapping between them [4]. This mapping is used to see how (and where) proposed business changes will have an impact, and ultimately to improve the evolution of the system.

In summary, business process modelling has an impact not only upon the client's business but also upon the processes and productivity of software developers. Hence, frameworks are needed which provide support for business process modelling to a variety of audiences who have different modelling needs.

2. Business process modelling methods

In considering 'how to' model business processes, two important considerations are notation and method. In other words, what representation scheme (or schemes) should be used, and how does this fit within a methodological framework. This can be seen as essentially the same problem that software engineers have in carrying out analysis or design. One might choose to use structured analysis notations, also choosing some associated method. Hence, this section will first discuss notations for process modelling, and then suggest some implications for methods.

An important reason for consideration of notation is that different notations have different capabilities. One cannot model the same things using a data flow diagram as a petri-net. Hence, in choosing different approaches, the modeller is (often implicitly) attempting to satisfy different goals. For example, pragmatic approaches [5] are mostly concerned with capturing and understanding processes. Conversely, rigorous paradigms are typically used for analysis of the process, [6,7], perhaps including some simulation or experimentation with process scenarios [8]. This suggests a need

for different notational approaches, for different modelling purposes. However, if multiple models are required, then a framework is needed which allows them to be used together in some coherent way.

Existing business process modelling methods appear to fall into two camps. High level methods concentrate on the general managerial activities that surround process modelling [9]. These activities, such as getting proper backing for the modelling, setting up workshops, and gathering evidence are clearly important but can be considered as separate from the modelling activity, and may often be carried out by a separate person or group. Consequently high-level methods give little consideration to the notations used for modelling, and little guidance about how the modelling activity is best carried out. In contrast, those methods that do consider notations appear to be much more prescriptive, suggesting that a particular notational approach is superior, without considering the environment for the modelling [10].

This paper provides a methodological framework (CAP), which supports modelling activity without prescribing a particular notational approach. To this end, a general framework is described, allowing the modeller to choose appropriate notations, at different points within the modelling exercise. Modelling is described in terms of three generic phases (or categories), and examples of notations appropriate for each phase are illustrated.

3. Evolution of the CAP framework

The (CAP) business modelling framework has evolved from experience of applying the kinds of techniques used within the software process to business process modelling [1,11,12]. For example, modelling business processes with formal notations such as Hoare's [13] CSP [14,15]. Observations from such undertakings can be distilled into a number of lessons learned.

- *Capturing the business process*: business users are not prepared to invest time in understanding complex or formal modelling approaches. In order to have meaningful dialogue with such users, a readily understandable, typically diagrammatic notation is required.
- *Analysing the business process*: formal or executable models do allow the modeller to gain a more thorough understanding of the process than can be achieved using simple diagrammatic techniques. Furthermore, in order to carry out process analysis it is necessary for the modeller to have more sophisticated mechanisms than qualitative analysis of static diagrammatic models. This paper will describe a number of alternative ways to achieve the added rigour required for process analysis.
- *Presenting the business process*: even simple diagrammatic notations lead to complex models which users find difficult to comprehend. The audience for models must

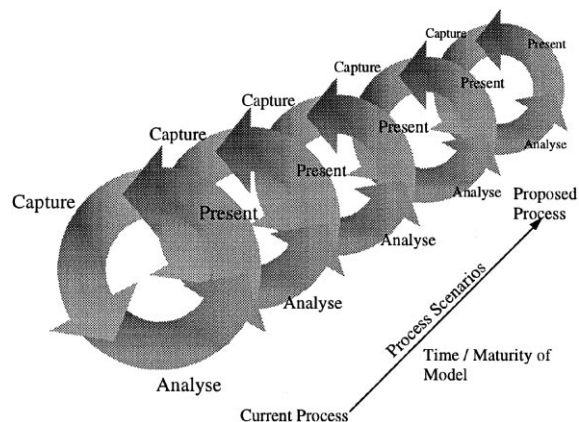


Fig. 1. CAP framework involves iteratively capturing, analysing and presenting process models.

be considered and approaches chosen that present relevant findings in a way that can be more easily understood.

The CAP framework mirrors these observations about business process modelling, with three distinct types of model being produced for process 'Capture', 'Analysis' and 'Presentation' (Fig. 1).

Initial process descriptions are produced (and validated) using Capture models. These Capture models are then used as the basis for analysis. Analysis models (and the results of analysis) are used to produce models for presentation. Presentation models are used to clarify issues, to highlight areas of further interest, to discuss findings and so on. Use of the CAP framework is therefore iterative. Early iterations focus on increasing understanding and on producing accurate descriptions of the existing process. Later iterations are used to suggest where improvements will be made, to experiment with different process scenarios and to choose among alternative process redesigns. Finalised models are then used to provide education about proposed change, and to provide guidance once the new process is implemented.

4. The CAP phases and framework

It is important to start the modelling programme by deciding what the models will be used for, and what issues they should highlight [16]. As with the selection of metrics, a goal-based approach seems appropriate. One could even construct a hierarchy of goals (modelling purposes) questions and notations; similar to the way GQM [17] has a hierarchy of Goals Questions and Metrics. This may be rather an elaborate solution to deciding notations, however, and others [5] suggest a simpler goal-based checklist approach (GUIDE)². Whatever selection mechanism is used, it seems reasonable that choice of notation should take into account the goals and context of the modelling.

This paper suggests that the goals, context and audience change for each 'phase' of the modelling. Consequently, different notations may be required for each phase, or at least that the choice of notation needs to be reconsidered. Rather than prescribe specific notations CAP is intended to be generally applicable, allowing modellers to choose various notations. However, the paper illustrates the kind of approaches that are appropriate for each of these modelling phases. An additional constraint is that the capture and analysis notations should allow for information from capture to be utilised within analysis, and subsequently within presentation. Although there is unlikely to be a complete mapping from one phase to the next notations should be selected that can be used in a coherent manner, so that meaning is preserved in each transformation. (This issue of mapping will be further discussed in presenting examples of capture, analysis and presentation models.)

The following sections describe each of the CAP phases. To illustrate these phases, a small artificial business process example has been chosen. The example, taken from Ould [18], describes the interaction between three roles: a Divisional Director; a Project Manager and a Designer; all involved in 'carrying out a design project'. For pedagogic reasons, only a single model of each type is given, though for a genuine business process the application of CAP would be far more iterative. Each category or phase would be visited many times, and many versions of each type of model would be produced.

4.1. A note about process support environments

A number of authors have extended the idea of enactable models in the attempt to build process support environments; and this quest has been an important motivation for much process modelling work [19,20]. The idea has been to model the way users work and then to instantiate a model description in the environment [21]. This running process description allows (and controls) user access to the tools they require to carry out their work.³ A number of impressive support environments currently exist. For example, both Process Weaver [23,24] and SPADE [25] allow the modeller to describe the process in a diagrammatic form (based upon Petri-nets), and to add detail to this description to support enaction of the process model within the environment.

² The GUIDE checklist is used in order to specify the Goal, Use, Investment, Deliverables, and Environment for the models. Ideally, goals should be quantified, and questions attached which help to clarify the thinking behind the goal. Other categories are intended to clarify thinking about what would be an appropriate modelling approach. The 'Use' category addresses the audience for the models, and how they are to be used. This also has a significant impact upon choice of notation, particularly for presentation. Further categories address the 'Investment' for the modelling programme (which clearly affect the feasibility of various modelling solutions), the 'Deliverables', and the 'Environment' (which includes the experiences, attitudes, and aptitudes of those involved or affected).

³ A number of authors now believe that more flexible support may be achieved by using agents [22].

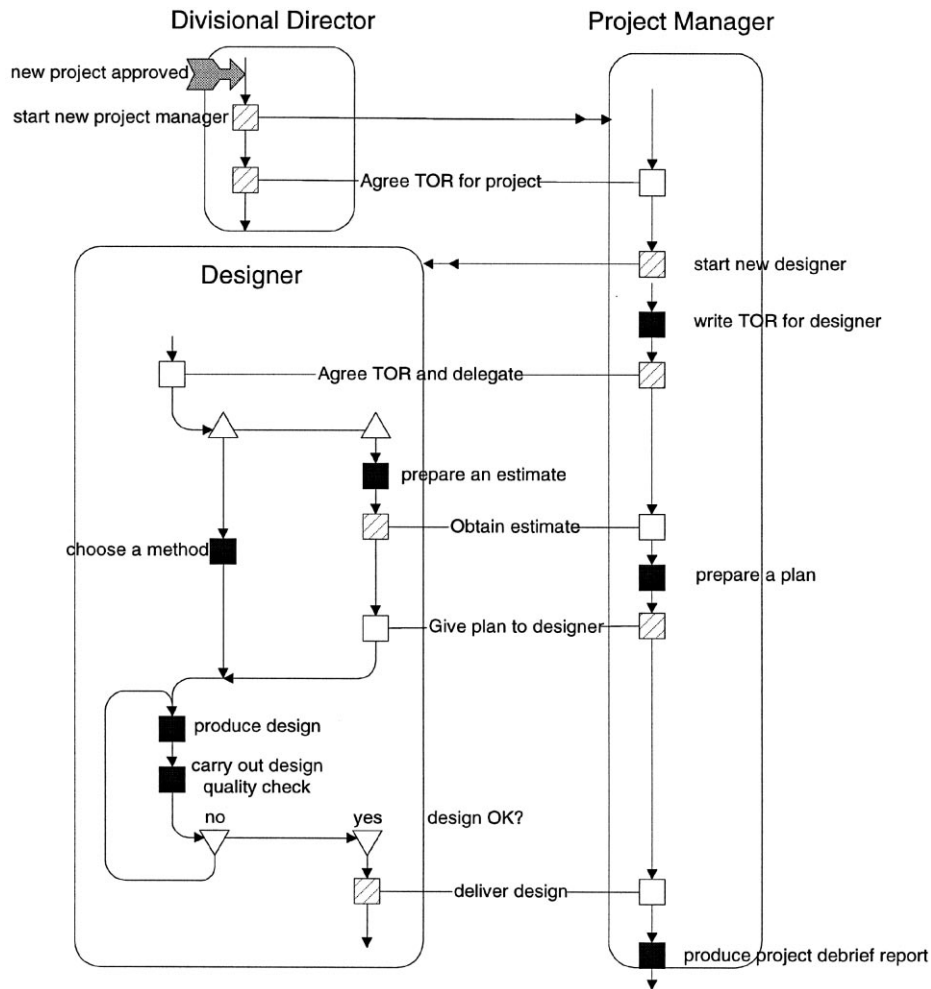


Fig. 2. Role activity diagram for 'designer'.

This paper considers the adoption (or otherwise) of such tools as a special case of the selection of notation, indeed, one may see how modelling context has an impact. For example, such tools are a significant investment for an organisation, and assume a level of control in the instantiated process that others argue is inappropriate [26]. In addition, for many the use of these tools would be beyond their budget, yet they may still improve their processes by using some simple (low cost) modelling technology.

5. Capture

As process evidence is gathered, it is necessary to produce models that validate understanding or interpretation of the process, and to use these models to focus discussion with process users. In theory these models could be of many different kinds, however, the major proviso is that they must be easy to understand for the uninitiated. Hence, diagrammatic representations are usually best suited for process capture. This paper illustrates process capture with role activity diagrams (RADs), a commonly used process modelling notation [27]. However, other diagrammatic

notations such as data flow diagrams [28], IDEF [29] or derivatives of petri-nets [30] are frequently used in this way. The RAD notation will now be briefly described. Those wishing a more thorough introduction are referred to Ould [18].

5.1. Role activity diagrams

Roles (shown as large rounded rectangles) group together a type of person, group or system. Roles are intended to be cohesive groupings, with the role representing the unit of responsibility for carrying out activities. A role is a type description; it describes a class of behaviour. Hence, a role may have a number of instances. For example, a cashier role would be enacted by a number of cashiers; therefore, a role can be assigned to a number of people or actors (resources). Similarly, a single resource may actually take on a number of roles; a supervisor may sometimes 'act' as a cashier.

Each role has a thread of control. A point on this thread represents the state of the role.⁴ All activities are linked by

⁴ Some authors explicitly label each state, using a circle to do so [27,31,32].

this thread of control, and all have a ‘before’ and ‘after’ state. Activities may be either actions (which are activities that an instance of the role carries out in isolation) or interactions (activities which are carried out co-operatively). An action is shown as a dark (shaded) square. An interaction is shown as a horizontal line connecting clear (unshaded) squares. This paper adopts the convention of showing the initiating (or driving) end of an interaction as crosshatched.

In carrying out an action, a role will move from its ‘before’ state to its ‘after’ state. The consequence of an interaction is that all roles involved move from the ‘before’ to the ‘after’ state. Roles also have constructs corresponding to alternate choices (case refinement) and concurrent or parallel activities (part refinement). Concurrent or parallel threads are preceded by a triangle, whereas alternate paths (threads) are denoted by an inverted triangle.

Starting at the top of the Divisional Director role there is an external event (shown as a horizontal arrow) which signifies the approval for a new project. As a result of this event the ‘Divisional Director’ role is able to start a new project.

This is followed by the creation of a new project manager role ‘start new project manager’. Creation is a commonly used RAD construct, normally denoted by an activity square. In this paper, a horizontal line with a double arrowhead emanates from the action square and points to the role being created.

The result of the creation is that an instance of a Project Manager role will be created. Now the Project Manager and Divisional Director roles are able to agree terms of reference (TOR) for the project. Thus, the Divisional Director takes part in an interaction with a Project Manager role; interactions are shown as squares joined by a horizontal line. The driving role’s interaction square is shaded; whereas in all other roles involved in the interaction the corresponding interaction square is unshaded. Hence, the Divisional Director side of the interaction has a shaded square showing that this role drives the interaction. The completion of the interaction ‘Agree TOR for project’, moves the two roles to their next state.

The Project Manager role is (as a result of the previous interaction) able to take part in the action ‘start new designer’. As with the creation of the Project Manager role, the ‘start new designer’ creation will create an instance of a Designer. In this case the Project Manager must write a TOR (the action ‘write TOR for designer’) before being able to initiate (drive) an interaction with the Designer to agree the TOR. Once this interaction has taken place the Project Manager is in a waiting state, waiting for an estimate from the Designer.

The Designer, having agreed the TOR, follows two parallel or concurrent paths. The paths are each denoted by an upward pointing triangle. The left-hand path involves a single action, ‘choose a method’. The right-hand path is more complex. The Designer must first ‘prepare an estimate’ (an action), and then take part in an interaction with the Project Manager, the result of which is that both roles have the estimate agreed. Since the interaction takes place only after the Designer has prepared the estimate, the Designer drives the interaction. Only when this interaction ‘obtain estimate’ has taken place, can the Project Manager go ahead and ‘prepare a plan’. Similarly, on completion of the action ‘prepare a plan’ the Project Manager drives an interaction that gives the plan to the Designer.

When both the interaction ‘give plan to designer’ and the action ‘choose a method’ have been completed the Designer is in a state of being ready to design. All concurrent paths must rejoin in a Role Activity Diagram. At this point, the Designer may take part in the action ‘produce design’. Having produced a design, the Designer must then ‘carry out design quality check’. Again, the role now has two paths. However, the downward pointing triangle symbol shows that there is a choice between these paths (only one may be taken). If the design is not OK, then the role moves back to a state of being ready to design, shown on the RAD as a loop back; from the bottom of the inverted (no) triangle back to the activity ‘produce design’. There may be any number⁵ of iterations of this loop, of design and then carry out a design check, before the design is acceptable. Once the design is acceptable, an interaction is initiated by the Designer to ‘deliver design’ to the Project Manager. Finally, for this depiction of the process, the Project Manger takes part in an action to ‘produce project debrief report’.

Fig. 3. Description of the role activity diagram (capture model) of the designer.

⁵ 0 to n where ‘n’ is a natural number.

5.2. The 'designer' as a role activity diagram

The role activity diagram in Fig. 2 shows three roles 'Divisional Director', 'Project manager', and 'Designer'. Fig. 2 has no explicit labelling of states, but the vertical lines between actions and interactions implicitly represent states.

The textual description of the designer (Fig. 3) makes reference to these (implicit) states. The reader may choose to ignore the details of this description, or the state based information it contains, since it is not necessary in order to understand how Role Activity Diagrams may be used as a capture model. It is included since both the analysis and presentation models described in following sections can be mapped directly from the RAD of designer. In particular the enactable presentation model makes explicit the states described, using them as pre and post conditions for actions and interactions.

6. Analysis

Analysis frequently requires a degree of rigour not found in models used to elicit or understand processes (as in capture). This rigour may be gained in a number of ways.

- *Expert judgement and heuristics*: the majority of business process analysis consists of inspection of diagrams. Inspection involves the use of expert judgement perhaps aided by the application of a number of heuristics. This is very similar to the application of design guidelines seen in software engineering. One might look to reduce certain features, for example, to minimise coupling or to remove redundancy.
- *Measurement*: as with software measurement one can view measures as ways to aid the analysis of the business or of the business model. One may have direct measures of the process (e.g. of effort and time) or of process products [33]. However, collection of such 'real world' data often requires substantial effort both on the part of the modeller and the organisation. In contrast, measurement may be used to aid the analysis of static business process models, adding a quantitative dimension to the application of heuristics [34].
- *Formality*: one of the arguments for formal modelling is that the modeller is forced to think more clearly to describe the process and thus achieves a deeper understanding. In addition, formal models do not suffer from the kind of multiple interpretation that afflicts many diagrammatic descriptions. For example, a CSP model has far clearer semantics than a DFD.
- *Enaction*: an enactable model is one that can be executed, thereby providing a dynamic view of the process. Hence, enaction allows for a further degree of scrutiny which is not possible using only a static process description. Typically, one might investigate temporal aspects of process behaviour, for example, cycle time using tools [35] based on Forrester's Systems Dynamics [36].⁶

⁶ Recent applications of Systems dynamics approaches are described in Ref. [37].

The following sections briefly describe how each of the above general strategies can be used to add rigour to the analysis of the process. The paper assumes that Role Activity Diagrams were used for process capture, though clearly the general strategy could be applied to other capture notations. The aim is to show how different approaches to analysis may be utilised. However, enaction of role models is described more fully, since this approach will be revisited for presentation.

6.1. Applying judgement and heuristics

Much of the useful analysis of a given process is dependent on an understanding of the domain [38]. For example, knowledge of financial systems is useful in the analysis of business models of financial institutions. Despite this caveat, it is also possible to discern some generally applicable guidelines for the analysis of business processes.

Ould [18] describes a number of desirable (and undesirable) business process features and shows how these manifest themselves in Role Activity Diagrams. As an example, two familiar concepts, coupling and cohesion are discussed. Of cohesion Ould states, "A role should have high cohesion, that is, the activities that form it should be closely related and collectively have a single purpose" and of coupling, "As a set, the roles should be loosely coupled, i.e. we should expect few interactions between them". Ould argues that restructuring the roles in the business process, to minimise coupling (and maximise cohesion) leads to greater efficiency, for example by reducing delays inherent in interaction.

6.2. Collecting and using counts

Phalp and Counsell [34] describe how heuristics can be quantified; producing 'counts' or measures to aid the analysis of static process models. They illustrate the general principle by describing measures, based upon Ould's heuristics; which help the identification of highly coupled (and low cohesion) roles. These counts may be easily derived from capture models such as the designer RAD described in Fig. 2. As an example the 'coupling factor' (CpF) is calculated by forming the following quotient:

$$\text{CpF} = \frac{(\text{Total interactions in role } X)}{(\text{Total actions} + \text{Total interactions in } X)}$$

Consider the Designer roles in turn. The Divisional Director role has a single interaction and a single action, hence the coupling factor is 1/2. The Project Manager role has five interactions and four actions, giving a coupling factor of 5/9, and the Designer role four actions and four interactions, giving a coupling factor (after reduction) of 1/2.

The modeller uses these measures to look for highly coupled roles, and aims to restructure the process to minimise coupling between roles. Low coupling indicates roles with a high degree of autonomy. Such roles have less

interaction (and synchronisation) with other roles and will typically have less delay in the completion of their tasks.

A discussion of meaningful levels of outliers for role coupling is beyond the scope of this paper (see Phalp and Counsell [39]). Indeed, it is unlikely that any definitive levels would be equally appropriate across all domains, or indeed all organisations within a domain. However, such measures help the modeller gain further insight, and aid identification of areas that need to be restructured. Phalp and Counsell report examination of a fragment of a business processes where of 10 roles depicted, eight had a coupling factor of one (i.e. there were only interactions — co-operative activity) and exceptionally of the 30 roles examined approximately 90% had a coupling factor greater than 4/5. This process exhibited many delays and the authors described it as “...highly bureaucratic...where few actions could be carried out by roles independently, and where the key actors in the process spent at least 50% of their time in gaining approval for documents”.

6.3. Formal models

This paper has noted the argument that being forced to describe something formally forces the modeller to think more clearly. The added rigour of the modelling exercise, coupled with the clear semantics of the formal notation leads to a more accurate and unambiguous description. However, the problems of validating such ‘difficult to understand’ models often preclude their use in business modelling.

Abeyasinghe and Phalp [15] show how role activity diagrams (as a user-facing notation), can be mapped to Hoare’s notation for communicating sequential processes (CSP) [13]. They argue that this provides the advantages of a simple model for process capture, with the addition of a formal notation for more rigorous process analysis. Their paper describes experience of using CSP to model a variety of processes in this way, revealing an iterative approach to process modelling.

‘We have found the RAD to CSP mapping itself to be an iterative process, with the mapping to CSP forcing us to re-think the original depiction of the process in RADS. Indeed, the mapping enables us to make changes in CSP, then go back and change the RAD and to go round such a cycle until we reach a stable and agreed process description. This kind of process is vital to the process modeller. It enhances the understanding of the process, and adds value to the process modelling exercise.’

Henderson [40] describes a simple ‘stepper’ which enables the modeller to ‘step through’ the states in a model written CSP. Abeyasinghe and Phalp show how use of this stepper provides some limited enaction of the formal model, but not in a form that is presentable to users. Hence, one strategy for analysis would be to map the capture model

(Role Activity Diagram) to CSP (providing a formal description of the process), with some enaction capability for use solely by the modeller. A further example of the use of enaction within process analysis will now be described.

6.4. Enactable models

Enaction increases the rigour of the modelling activity. The modeller must describe the process with sufficient clarity that it is unambiguous to a computer (e.g. for compilation); and the experimentation with process behaviour engenders greater understanding. However, the main advantage of running a dynamic model is that it allows both modellers and representatives of the client organisation to experiment with (or check) process understanding. The model can be tried (or run), problems encountered documented, and the model adjusted or revised until a more satisfactory process scenario is evolved. Hence, enaction is particularly useful in a presentation role. The main difference in the kind of models suitable for analysis and presentation being the extent to which the models can be made accessible to the end-user.

The enactable notation RolEnact will now be described. RolEnact may be used (with different interfaces) for both analysis and presentation of business processes, hence its syntax is discussed here, whilst a simple interface is described in the following section on presentation models.

6.4.1. RolEnact models and syntax

RolEnact syntax is based upon a condition–action paradigm. Its primitives match those of Role Activity Diagrams, allowing a simple mapping from process capture. In RolEnact, processes are described in terms of roles, the states of these roles, and the actions and interactions in which roles take part. As with a Role Activity Diagram a description in RolEnact is of a type of behaviour. However, when running an Enact scenario the modelling engine invokes instances of roles, each which will then behave according to that general (type) description. By observing how these role instances act when running in parallel and how they influence each other, the user of the model is able to gain a thorough insight into the process.

In brief, the syntax describes the creation of a new role, or the invocation of an activity (action or interaction). The behaviour of the process is dependent upon the rules (pre and post states) of these activities. The general form of syntax for an activity is as follows (keywords shown in bold):

```
Activity Role1.Activity
    Me(before1 → after1)
    Role2(before2 → after2)
    ...
    RoleN(beforeN → afterN)
End
```

The driving role (Role1) moves from a before state to an after state, as do any other roles involved in the activity. All

of the roles involved (Role2 to RoleN) must be in the required ‘before’ state for the activity to be able to take place, and all move to their described ‘after’ state as a result.

This general form can be seen in the syntax for a two-role interaction.

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

A simple interaction, taken from the example process, is as follows:

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

Actions are a special case where only a single role is involved. The syntax for an action is:

```
Action Role.Action
    Me(before → after)
End
```

A simple action, again from the example process, is:

```
Action Project_Manager.prepare_a_plan
    Me(estimate_received → plan_prepared)
End
```

A full description of RolEnact and its syntax is given in a related paper [3]. For the purposes of this paper, it is sufficient to see RolEnact as an example of a modelling notation, which may be used either for process analysis, or presentation or both. A further (industrial) example will show a larger scale version of a presentation model that utilised RolEnact.

7. Presentation models

Presentation is used to highlight pertinent findings from process analysis, and to suggest where changes should be made. Presentation models are used to present process scenarios to end-users, to validate new (or desired) behaviour and to choose among alternatives. Finally, presentation models are used to educate process users about change, before, and during implementation of the new process. As with capture, the presentation model is user facing, and understandability is of importance. However, the need to illustrate findings from (these often complex) analyses requires further capability.

The following section concentrates on the use of enactable models in presentation, and specifically RolEnact. It should be clear, however, that such models are often used in conjunction with other, typically static, models. For example, one might describe the overall process with a RAD, and use RolEnact to highlight particular issues or concerns. Indeed, this is a strategy familiar to the author,

and the industrial example that follows used both static and enactable models to describe different aspects of the process to the client organisation.

7.1. RolEnact models

A dynamic scenario in RolEnact takes the form of a set of windows, where each window represents an instance of a role in the process, and where each of these role windows acts as a separate Windows program. Each role instance runs in parallel and communicates with other roles via a Windows DLL (see Refs. [3,41]). The separate windows are each instances of a generic Windows program. Each time a role instance is created this program is parameterised by information from the RolEnact description. The role instance then acts according to the rules described for its particular role type.

It should be clear that the description of the process is separate from the interface. Each RolEnact description is kept in a text file, which may then be changed without altering the implementation of Role instances. Hence, it is very easy to make changes or to run a different scenario by loading a different file.

A further consequence of this approach is that it is also extremely easy to change the interface of the model. A number of different RolEnact interfaces have been used to present models to business users. The most common form of interface (of which again there are a number of slight variations) is now described. This ‘raw’ interface has been found to be acceptable to many (though not all) business users, and was the most commonly used by a number of process modelling projects based at the University of Southampton [4,42,43]. This ‘raw’ interface has also been used by the author in teaching modelling to students (at Bournemouth University) and process modelling to industrial audiences. Where RolEnact is used primarily for analysis, again this simple form of interface is prevalent.

7.2. The designer in RolEnact

Returning to the example process, the first role to be created would be the Divisional_Director. In the ‘raw’ RolEnact interface, an instance of the role appears as shown in Fig. 4.

Each role may have a number of instances at any one time. Each instance of the role is launched by the user and represented by a window. Hence, there may be any number of instances of the role Divisional Director, or any other role, in any given scenario.

From Fig. 4, it can be seen that each role instance (window) consists of:

- *The Role Instance name:* This is the name of the role type followed by an integer. The integer is the number of roles of that type already launched in the current scenario. Hence, the default⁷ numbering scheme is 0

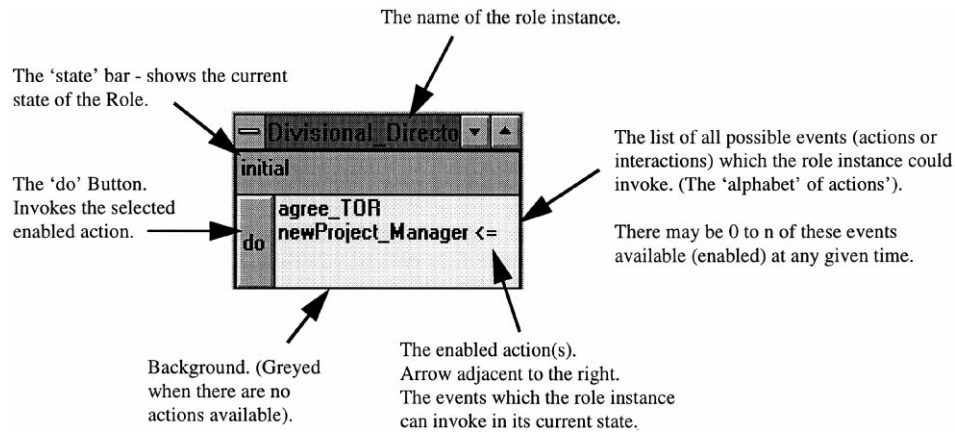


Fig. 4. A RolEnact: role instance window.

to n , where the first instance launched is roletype0, the next roletype1 and so on (see Fig. 4). This allows for multiple instances of the same role type in the scenario.

- A *state bar*: which displays the current state of the role instance. All roles being launched in an ‘initial’ default state.
- A *‘do’ button*. This invokes a chosen event. If the event is not enabled (available) then no action is taken. The ‘do’ button can be ignored completely, and the user may double-click on events to invoke them.
- A list of all of the possible events that the role may perform. Since events are governed by preconditions that take into account the state of the role, and other co-operating roles, some events may never become available within any given running of a scenario.
 - When no events are available within a role instance, then the role instance window is said to be inactive. The list of events is greyed out, to show that none of them can currently be invoked.
- *Enabled actions*: indicated by an arrow on the adjacent right of the entry in the list.

The list of events can be seen as similar to the idea of an alphabet in CSP. It comprises all of the possible events (actions or interactions) which the role may take part in. The events, which are available, depend on the state of the running scenario (or system). This allows users to experiment with a dynamic model showing how role instances co-operate with and affect each other.

Consider four⁸ role instances acting as described in the ‘Designer’ example (Fig. 5). The possible events at this point are ‘choose a method’, an action of the designer; and ‘prepare_estimate’, an action of the Designer Estimator. The Project Manager is in a waiting state; its state bar is

⁷ This has been found to be confusing for users, therefore, new implementations of RolEnact will number role instances from 1 to n .

⁸ Note that parallel threads in a role activity diagram are represented by a separate role. Hence, the parallel threads in the designer role activity diagram are shown as two separate role instances, Designer_Estimator and Designer. For a full discussion of this mechanism for describing parallel threads the reader is referred to Ref. [3].

white and its list of actions disabled. In addition, the Divisional Director may create a new project manager instance (currently highlighted by the user).

Suppose the user wishes to invoke an event, e.g., to prepare an estimate. A single click selects (and highlights) the action which may be invoked either with the ‘do’ button or with a double-click. The (enabled) action is then carried out, and RolEnact updates the states and list of enabled actions in all role-instances. For ‘prepare estimate’ (an action) only the state of the Designer Estimator would change. However, consequently, it would be possible to interact with the Project Manager. Hence, the state changes of both actions and interactions determine those available next. The user chooses available states and continues in this way, experimenting with the process behaviour.

8. CAP applied to an industrial process

The scenarios above have all been produced by iterating around a number of cycles of production of static models, experimenting with process scenarios, revisiting static models and so on. The scale of the models presented thus far is deliberately small, in order to present understandable examples. However, large complex and real business processes have been modelled using the method described.



Fig. 5. Co-operating role instance windows in a RolEnact scenario.

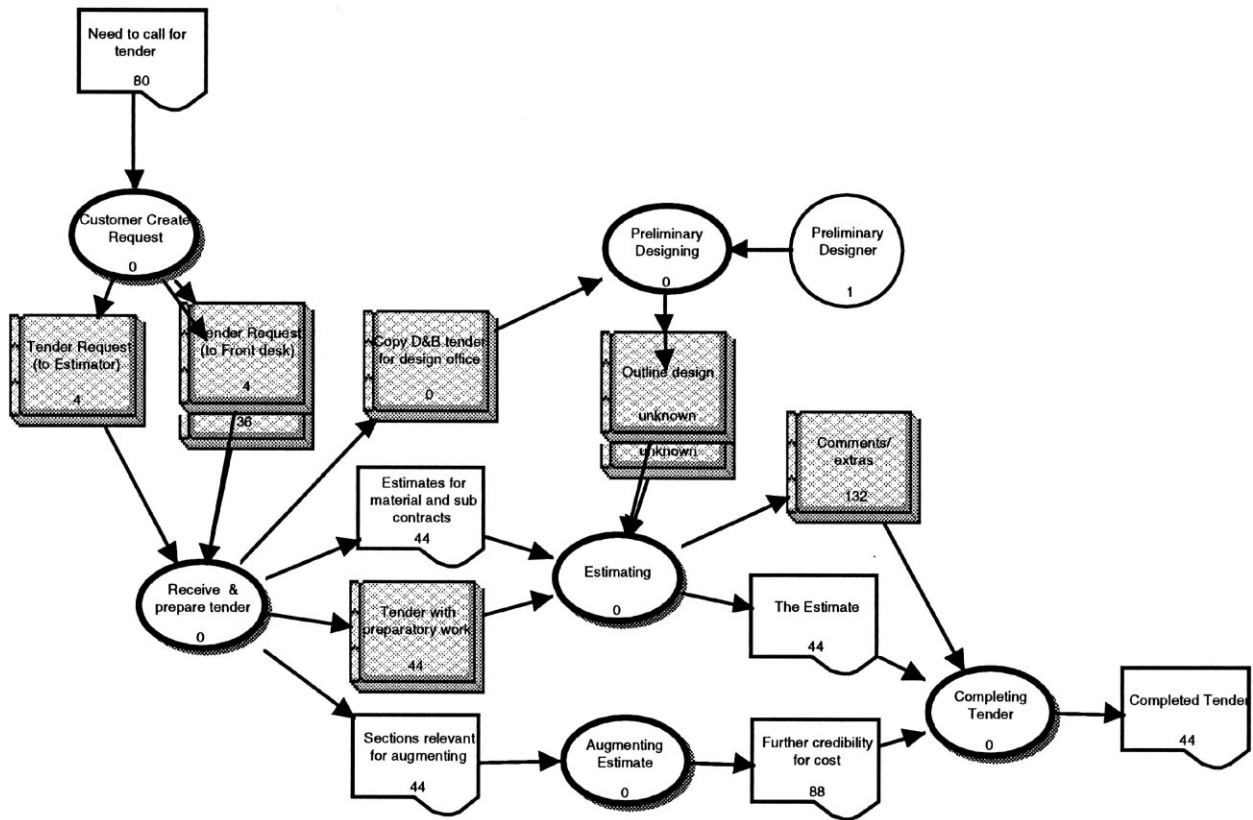


Fig. 6. Top level PWBS diagram in the tendering process.

Two large families of models have recently been produced to describe the tendering processes of two European Construction companies, as part of a collaborative European project [42]. The presentation models both used over a dozen separate multiple event role types, each of which have many instances, both in the process mock-up, and in reality. One of these processes along with associated models will now be briefly described.

8.1. Capturing an industrial process

The capture models for the tendering process use a different notation to those so far presented, adopting a largely procedural rather than a behavioural (role-based) modelling paradigm.⁹ This illustrates the point that the CAP framework does not prescribe notations for each category of model, allowing the modeller choice.

The industrial capture modelling used the ProcessWise WorkBench Standard (PWBS) tool [45] which creates data-flow-like diagrams. This satisfied one of the main aims of process capture; to have an understandable modelling notation. In addition, the tool provides some cumulative analysis. This allowed the same basic notation to be used both for capture and analysis; easing the transformation from one model type to the next.

Diagrams in PWBS consist of three types of object:

1. *Processes*: describe actions performed. As with data flow approaches, processes may be decomposed into further processes, and so on. Processes are depicted as ellipses.
2. *Business objects*: are passed between processes. The tool supports appropriate levelling of processes, such that business objects which are consumed or output by a process at one level of the diagram hierarchy must appear in the decomposition of that process, and vice-versa. Business objects have a number of subtypes. For the tendering process, two types of objects were used (to aid inspection of models by the uninitiated). Default objects (rectangles with an irregular lower edges) and book-like shapes (depicting actual documents in the process).
3. *Resources*: these are called roles within PWBS, but are mostly used as containers for units of resources. Roles are shown as circles. All processes must have one or more role that performs them; processes not roles being the grouping mechanism within PWBS. Roles are sometimes used to show responsibility for a process, where the role may not be the resource used to complete the process or task. For example, a manager might be responsible for the completion of a task, but would assign some other resource to carry it out.

Fig. 6 shows the top-level view of the tendering process. In all, there were 12 levelled views in the PWBS model.

⁹ See Ref. [44] for a categorisation of process modelling approaches.

Processes that are decomposed (expanded) in another view are shown with a heavy outline.

8.2. Analysing an industrial process

The PWBS tool used for process capture also has a number of analysis options. A description of these options is outside the scope of this paper. However, in brief four main types of analysis may be carried out.

1. *Volume analysis*: enables the user to see how input business move through the process.
2. *Resource analysis*: is used to calculate the amount of resource (staff) required for each role performing an activity. Resource analysis requires information about total volume for an activity, the time the activity takes to process a unit of volume (worktime) and the units of effort this will require from the supporting role.
3. *Cost analysis*: is used to calculate monetary costs for activities, based on volumes, worktimes, and costs per worktime.
4. *Elapsed time*: calculates cumulative duration at each point in the process.

For the tendering process, obtaining figures needed for these kinds of analyses required considerable effort on the part of the modellers, and extensive co-operation with the organisation concerned. The main analysis effort was driven by the needs of the organisation (resource levels were fairly fixed) and focused upon volume, costs and time. In addition, it was found that it was necessary to manipulate the model in

order to gain the analyses required by the organisation, since the tool was highly constrained in the kind of calculations it allowed. This lack of flexibility (which would not be found in say a spreadsheet) was to some extent compensated by the ease of mapping from capture to analysis. However, a major concern was that the detail shown in the model often hid rather than highlighted analysis results, particularly those related to costs (a theme that will be further discussed in describing presentation of findings).

8.3. Presenting an industrial process

8.3.1. Presentation of sequencing

During meetings with representatives of the organisation, it became clear that there was a need to clarify and examine sequencing issues. RolEnact models were produced specifically to address this need. These models were used primarily in a presentation mode; as a vehicle for discussion and argument about both the current and proposed process behaviour.

An example of one of the RolEnact models used is shown below (Fig. 7). Notice that this is essentially the same as the raw RolEnact model previously described (in Figs. 4 and 5). However, in this case graphics have been added to illustrate the states of each role. This extended RolEnact implementation allows users to define pictures for each state stored, for example as bitmaps, in an appropriate directory. On execution the Role instance program (a separate Windows executable) determines its own state and then fetches and displays the appropriate picture. Hence, by changing the appropriate picture file, it is possible to change how any

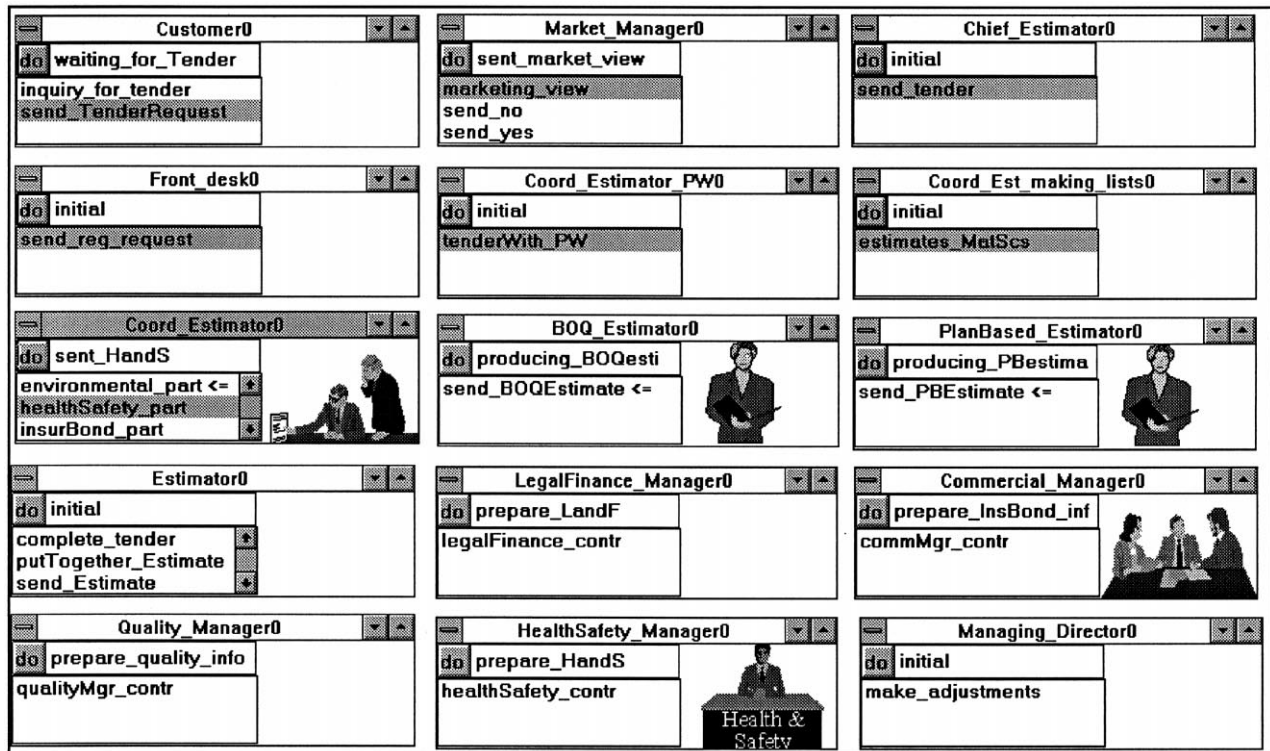


Fig. 7. RolEnact for the tendering process.

state is depicted, without needing to change the RolEnact description.

Representatives of the organisation found that this implementation change made it easier for them to see state changes, and to understand how for each different state there would be a number of possible activities, determined by the states of other roles. The most useful consequence of the RolEnact modelling was that it identified a number of areas of misunderstanding, which led to further process changes. This is because the RolEnact models explicitly and dynamically showed those sequencing issues that were only implicit in capture and analysis models. Although a number of personnel had agreed that PWBS models were an accurate depiction of their process, they had been interpreting the models differently.

This illustrates how different approaches highlight different aspects of the process. For the tendering process, RolEnact was used to highlight sequencing issues. However, a further notation (POSD) was necessary in order to illustrate findings related to cost.

8.3.1.1. A note about interfaces and presentation.

The use of RolEnact, with the addition of pictures to represent states, reveals how simple changes to an interface may have a significant impact upon users. Clearly, a discussion of human–computer interaction is beyond the scope of this paper. However, changes to the interface can be viewed as another example of the need to consider the audience for each model; an essential reason for the adoption of the framework described.

8.3.2. Presentation of structure and findings

The main reason for differentiating presentation models from those used in analysis and capture is that conventional models are hard to present to process users. Even with models like PWBS, which are relatively simple and which can be structured by hierarchical decomposition, users find that models quickly become too difficult to comprehend. Hence, the overall process architecture, or process findings cannot be easily discerned. In order to alleviate this comprehension problem, the PROCESS project (and other associated projects CORE and GISIP) have used a notation specifically designed for presentation; Process Oriented Ssystem Description (POSD) [46].

8.3.2.1. POSD notation.

A significant advantage of using POSD is that it helps to alleviate a phenomenon known as ‘wire syndrome’. This phenomena was first discussed in electronics where processing units could be decomposed into other processing units but where the number of connections would remain high. It may also be seen in other modelling approaches, which allow only for decomposition of one primitive, typically process. Hence, high level diagrams often consist of small numbers of processes with large numbers of flows obscuring the overall structure.

Having mechanisms to guide the reader through complex models is not unique to software engineering. For example, in object oriented analysis [47] ‘subjects’ are used to group together objects and classes into logical units. However, whereas subjects may either include or overlap with other subjects POSD has a simple mechanism (called a ‘promise’) to describe the interaction or inclusion of other behaviours. The POSD notation is described in Henderson and Pratten [46], and its use in process modelling in Abeysinghe et al. [48]. The reader is referred to these papers for a full explanation of POSD; however, a brief description now follows.

POSD utilises a single primitive; called a ‘behaviour’. As with subjects, behaviours (typically represented as rectangles) may sometimes completely include other behaviours. Where behaviours overlap they are shown as touching, and the shared details are not shown. Touching behaviours indicate a ‘promise’ that the shared behaviour will be described somewhere further down the model hierarchy. Often the shared behaviour includes simple interaction or information, but it may sometimes be a complex process in its own right. For example, if the interaction were via electronic mail, then it is not necessary to describe the workings of the mail system to understand the high-level process. However, if models were later to be used to move towards building a support system then it might be necessary to describe aspects of this mechanism. Hence, a number of views of the system are possible, again depending on the purpose of the model.

By moving unnecessary detail to lower levels the overall process structure may be more easily discerned (and the wire syndrome alleviated). In addition, behaviours may also be decomposed into other notations, as long as ‘promises are kept’. Hence, a high level POSD picture of the process may include lower level detail in other notations, for example, in the analysis notation from which it was derived.

8.3.2.2. Using POSD in the presentation of industrial processes

A mild example of wire syndrome can be seen in the PWBS model taken from the tendering process (Fig. 6).

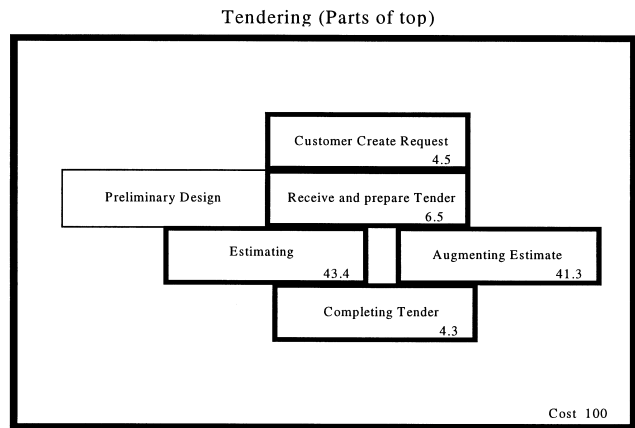


Fig. 8. Top level POSD model of the tendering process.

Note how the following POSD representation of the same top-level process is far easier to understand (Fig. 8). All of the details of the interactions are still described, but have been moved down the model hierarchy.

In addition, numbers representing costs have been added to the model. This allowed representatives of the organisation to quickly discern not only the overall structure of the process but also the relative costs of each major component.

This increased ‘ease of use’ of POSD models led to a far more productive relationship between the modeller and the end-users. Moreover as a general point it shows that what to a modeller may seem a trivial difference (both models offer essentially the same description) can have a significant influence. Hence, consideration must be given to the audience for the modelling.¹⁰ Such consideration is made explicit by the categorisation described in this paper.

9. Conclusions

This paper has described ways in which business process modelling has an impact on the efficiency of the software development process. The paper further argues that the choice of notation has an impact on the efficiency of this business process modelling. Hence, that the modeller needs guidance as to suitable notations or approaches to adopt.

However, current methods either do not consider notation, or they prescribe a particular approach, denying choice to the modeller. This paper argues that different process modelling notations are suitable, depending on the context of the business modelling, its purpose and its progress. Notably, that not only the goals of the modelling but also the audience for the models should influence the choice of notation. Further, that these goals and audiences vary at different stages (or phases) of the modelling programme. Three distinct modelling phases: Capture, Analysis and Presentation are described, and this forms the basis of a framework for business modelling; (CAP).

Different notations may be used within each CAP phase, but the paper suggests why those with certain characteristics may be most beneficial, and gives examples. A simple process example is used to illustrate the CAP framework, choices at each phase are illustrated and the problems of mapping between phases are discussed. An industrial business process example is also described, showing how the framework is applicable to the larger scale.

By using the CAP framework, the business modeller is guided without being forced into a particular notational approach. This allows the modeller to target the modelling, within each phase, such that it more closely matches the goals and experience of the client organisation. Hence, business process models will be produced which accurately

represent the client’s environment and needs, ultimately leading to greater efficiency in the production of the software that supports their business processes.

References

- [1] G. Abeysinghe, P. Henderson, K. Phalp, R. Walters, An audience centred approach to modelling for business processes, Fifth International Conference on Re-technologies for Information Systems: RETIS97 Klagenfurt, Austria, OCG Schriftenreihe, 1997.
- [2] E. Yourdon, Object-oriented systems design: an integrated approach, Prentice Hall, New York, 1994.
- [3] K.T. Phalp, RolEnact: role based enactable models of business processes, Information and Software Technology 40(3) (1998) 123–133.
- [4] PROCESS, modelling and mapping the business process, EPSRC project, home page at: <http://www.ecs.soton.ac.uk/~kp/process.html>.
- [5] K. Phalp, M. Shepperd, A pragmatic approach to process modelling, Proceedings of the Third European Workshop on Software Process Technology, 1994, Vilard de Lans, near Grenoble, France. Springer, Berlin.
- [6] J. Sa, B.C. Warboys, Modelling processes using a stepwise refinement technique, Proceedings of the Third European Workshop on Software Process Technology, 1994, Vilard de Lans, near Grenoble, France. Springer, Berlin.
- [7] J.Y. Chen, C.M. Tu, An Ada-like software process language, Journal of Systems and Software 27 (1) (1994) 17–25.
- [8] G. Hansen, Simulating software development processes, Computer Jan (1996).
- [9] C. Coulson-Thomas, Business process re-engineering: myth and reality. Kogan Page, London, 1994.
- [10] T. Huckvale, M. McHugh, C. Roberts, Process Modelling Workshop for the BCS Bristol Branch. Bristol University, Praxis Systems plc, 1993.
- [11] W. Scacchi, Business processes can be software too, Proceedings of the Third International Conference on the Software Process, 1994, Reston, Virginia. IEEE Computer Society Press, New York.
- [12] P. Henderson, P. Software processes are business processes too, Third International Conference on the software-process, 1994. Reston, Virginia. IEEE Computer Society Press, New York.
- [13] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, New York, 1985.
- [14] M. Greenwood, A declarative approach to process modelling, IOP-ENER 1 (5) (1992) 7–8.
- [15] G.A. Abeysinghe, K.T. Phalp, Combining process modelling methods, Information and Software Technology 39 (2) (1997) 107–124.
- [16] G. Tate, Software process modelling and metrics: a CASE study, Information and Software Technology, Software Process Modelling in Practice 35 (617) (1993) 323–330. Special Issue.
- [17] V.R. Basili, H.D. Rombach, Tailoring the software process to project goals and environments, Proceedings of the 9th International Conference on Software Engineering, Monterey, 1987.
- [18] M.A. Ould, Business Processes: Modelling and Analysis for Reengineering and Improvement. Wiley, New York, 1995.
- [19] R.A. Snowdon, A brief overview of the IPSE.2.5 project, Ada User 9 (4) (1988) 156–161.
- [20] B. Warboys, The IPSE 2.5 project process modelling as the basis for a support environment, in: N. Madhavji, W. Schafer, H. Weber (Eds), Proceedings of the First International Conference on System Development Environments and Factories. Pitman, London, 1989.
- [21] R.A. Snowdon, Tutorial: software process modelling, technical aspects. Software Process Modelling in Practice. Kensington Town Hall Conference Centre, London. Butterworth-Heinemann, Oxford, 1993.

¹⁰ The modeller may well decide to choose to present findings in a complex notation, perhaps the same as was used for analysis, as long as this is appropriate for use with the end-users or representatives of the organisation (the audience).

- [22] R.A. Snowdon, Active models and process support. Proceedings of the Fifth European Workshop on Software Process Technology, EWSPT96, 1996. Lecture Notes in Computer Science 1149, Springer, Berlin.
- [23] C. Fernstrom, Process centred environments, software process modelling in practice, Kensington Town Hall Conference Centre, London. Butterworth-Heinemann, Oxford, 1993.
- [24] C. Fernstrom, PROCESS WEAVER: adding process support to Unix. Proceedings of the Second International Conference on the Software Process, Berlin, 1993.
- [25] S. Bandinelli, M. Braga, A. Fieggetta, L. Lauerzza, The Architecture of the SPADE-1 Process Centred SEE, Proceedings of the Third European Workshop on Software Process Technology. Vilard de Lans, near Grenoble, France. Springer, Berlin, 1994.
- [26] M.M. Uhman, Some reservations on software process programming, Proceedings of the 4th International Software Process Workshop, 1988. Moretonhampstead, Devon. ACM Press.
- [27] D. Miers, Use of tools and technology within a BPR initiative, in: C. Coulson-Thomas (Ed.), Business Process Re-engineering: Myth and Reality, Kogan Page, London, 1994.
- [28] E. Yourdon, Modern Structured Analysis, Prentice-Hall, New York, 1989.
- [29] W.E. Hefley, The cobbler's children: applying total quality management to business process improvement, information engineering and software engineering, ACM SIGSOFT Software Engineering Notes 18 (4) (1993) 19–25.
- [30] S. Bandinelli, E. Di Nitto, A. Fuggetta, Supporting cooperation in the SPADE4 environment, IEEE Transactions on Software Engineering 22 (12) (1996) 841–865.
- [31] C. Roberts, Modelling and co-ordinating change in business processes. Process Modelling Workshop for the BCS Bristol Branch, 1993. Bristol University, Co-ordination Systems Limited.
- [32] Co-ordination, RADitor version 1.5: Users Manual. Co-ordination Systems, 1994.
- [33] M. Shepperd, Products, processes and metrics, *Information and Software Technology* 34 (10) (1992) 674–680.
- [34] K.T. Phalp, S.J. Counsell, Counts and heuristics for the analysis of static models, Proceedings of the ICSE'97 Workshop on Process Modelling and Empirical Studies of Software Engineering, Boston, 1997.
- [35] HPS, Stella, High Performance Systems, 1996.
- [36] J. Forrester, *Industrial Dynamics*, MIT Press, Massachusetts, 1961.
- [37] A.G. Rodrigues, T.M. Williams, System dynamics in software project management: towards the development of a formal integrated framework, *European Journal of Information Systems* 6 (1997) 51–66.
- [38] T. Rodden, V. King, J. Hughes, I. Sommerville, Process modelling and development practice, Proceedings of the Third European Workshop on Software Process Technology, 1994. Vilard de Lans, near Grenoble, France. Springer, Berlin.
- [39] K.T. Phalp, S.J. Counsell, Using counts and heuristics for the analysis of static models, 1998. Available at <http://www.xanadu.bournemouth.ac.uk/staff/kphalp/usechasm.html>.
- [40] P. Henderson, The CSP Stepper in Enact — an Executable Specification, 1992. Available as an ftp source from <ftp://ecs.soton.ac.uk/pub/peter/various/csp.ps>.
- [41] P. Henderson, Modelling Process Support in Enact, 1995. Available at <http://dsse.ecs.soton.ac.uk/~ph/cv.html>.
- [42] CORE, CORE: CONstruction Companies Process ReEngineering, 1997. Project home page at <http://iwi.uni-sb.de/forschungsprojekte/core/core-e.html>.
- [43] GISIP, Geographical Information Systems Integration Process, 1996. Project home page at <http://dsse.ecs.soton.ac.uk/~kp/gisip.html>.
- [44] B. Curtis, M.I. Kellner, J. Over, Process modelling, *Communications of the ACM* 35 (9) (1992) 75–90.
- [45] ICL, Process Wise Workbench: Version 5.4 Users Guide, International Computers Limited, 1995.
- [46] P. Henderson, G. Pratten, POSD: a notation for presenting complex systems of spare processes. First IEEE International Conference on Engineering Complex Systems, IEEE Computer Society Press, 1995.
- [47] P. Coad, E. Yourdon, *Object Oriented Analysis*. Prentice Hall, Hemel Hempstead, 1990.
- [48] G. Abeysinghe, et al., Presentation of Business Process Models using Process oriented Systems Description, 1997. Available at <http://www.xanadu.bournemouth.ac.uk/staff/kphalp/posd.html>.