# Designing a Re-engineering-proof Process Architecture

MARTYN A OULD
VENICE CONSULTING LTD
The Old School, Hinton Charterhouse,
Bath BA2 7TJ, UK

*phone*: +44 (0)1225 723 822
*e-mail*: mao@veniceconsulting.co.uk

## ABSTRACT

Suppose we have come to an organisation and have the job (for whatever reason) of designing or modelling some or all of its processes. It can be very hard to know how to cut the mass of organisational activity up into component processes and to see how they are related, in other words to determine the *process architecture* of the organisation. We want a partitioning into processes that is, as far as we can make it, completely aligned to the business the organisation is in and as independent as possible of how it chooses to organise itself. This paper describes the approach to solving this problem offered by the ***Riva*** business process management method.

## KEYWORDS

*process architecture, process modelling, process design*

## INTRODUCTION

Business Process Management (BPM) situations come in many different guises:

- In a BPR context we will want to define a whole new process architecture for the organisation as a first step to defining an organisational structure aligned to the processes.
- In a TQM or incremental improvement context our need is to prepare a model of an existing process as a diagnostic tool supporting its improvement. But where are the boundaries of this process with others?
- When we design IT systems designed to support specific business processes, especially those involving workflow management, we need to understand those processes in order to see how the proposed systems will enable – or even disable – them.
- For many organisations coming to grips with the way they operate, a shared understanding of the way things work – the processes and models of them – can be immensely valuable and revealing.

Copyright © Martyn Ould 2002

– An organisation wanting to lay down defined procedures in, say, a Quality Management System, especially in a regulated environment, needs a clear and unambiguous way of defining processes and hence needs models that are good enough to act as work instructions and that clearly delineate processes from each other.

The common concerns here are twofold.

Firstly, the process designer or modeller[1] needs a method for working with processes, one that yields clear, unambiguous and revealing models usable by ordinary people.

Secondly, and almost as importantly, the designer needs a way of deciding what are the processes in the first place and how they fit together when the world is 'running'.

Both of these concerns are answered by ***Riva***, an approach to process design, modelling and analysis that has been developed and used in a great many situations and industries. ***Riva***, as described in *Business Processes*,[2] concentrates on the first, process-level question: how can we most revealingly model a process? This paper describes how ***Riva*** has now been extended to cover the second question: what are the *real* processes and what are their dynamic relationships?

I shall use the term *process architecture* to refer to a definition of the *processes* within an organisation and the *dynamic relationships* between them. By the end of the paper I shall have given a very precise meaning to the term.

## Does it matter?

Why is getting the process architecture right so important? There are several urgent reasons.

An inappropriate division of organisational activity into processes can easily lead to unnecessarily complex designs or models. There is a lesson from software development: when we design software modules we look for high 'cohesion' and low 'coupling'; when modelling the real world, if we break a process into two along a natural 'fault line' we will get fewer broken connections. We want to exploit any cohesion present in the real world. Our aim must be to find those natural fault lines.

If re-engineering is being considered, a bad initial division can at best obscure the possibility of a radical change to a process and at worst lead to the local optimisation of part-processes but overall pessimisation of the total process. This is especially true if we arbitrarily chop things up

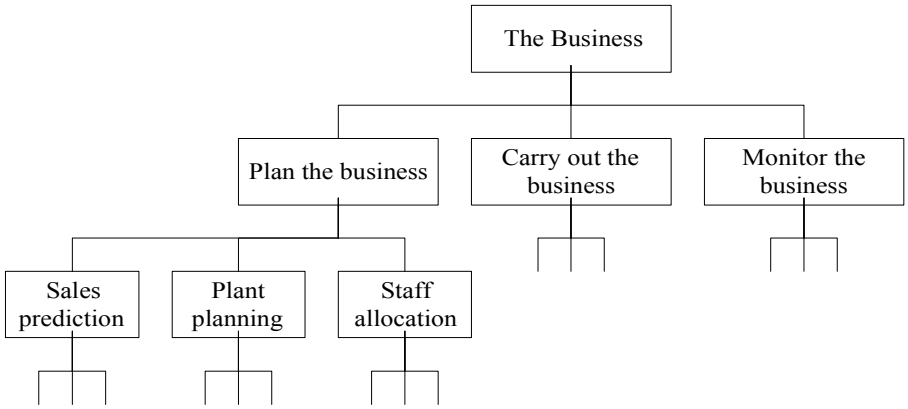
[1] For brevity I shall refer to this person as a *process designer*, but the argument runs exactly the same for the person solely concerned with mapping the status quo.

[2] *Business Processes – Modelling and Analysis for Re-engineering and Improvement*, M A Ould, Wiley, 1995.

into lumps in a sequence and call each lump a 'process', something that is all too often done.

People (especially those of us with software engineering backgrounds) have a natural desire to decompose a process into successively smaller sub-processes, in other words to draw up a *hierarchical* structure. The observation followed by ***Riva*** is that the world is rarely constructed in such a neat way and that the processes in an organisation are invariably connected in a *network*, rather than being contained in one another. The danger of thinking hierarchically cannot be over-emphasised. To draw a diagram such as the one in figure 1 is not to prepare a process architecture. What does it mean to say, as the diagram implies, that one process is the 'sum' of four others? Where in the diagram are the *dynamic* relationships between processes represented? It is, after all, the dynamic relationships that are crucial to an understanding of how an organisation *works*. This emphasis on network dynamics rather than hierarchical composition will be clearly made in this paper.

*Figure 1 – Not a process architecture*




It is equally all too easy to draw up a list of 'major' processes in an organisation by looking for functional units in the organisation and imagining that they in some sense represent 'top level processes', eg the 'Finance process', the 'Analytical Chemistry process', or the 'HR process'. These are all meaningless and probably misleading titles that should be resisted at all costs. At best such a hierarchy will allow us to group processes within 'areas' . . . but no more. And, if we change the organisation's structure, its list of processes would change too, which would be absurd if it was still in the same business.

There is of course no simple mechanical way of determining the process architecture of an organisation. But ***Riva*** does offer some powerful heuristics designed to lead towards an architecture that satisfies our re-

quirements and avoids the dangers. The method described below is designed to help us *hypothesise* a likely architecture of processes and their relationships, as a *starting point* for our final decision on how to carve up the organisational activity.

We will see that central to ***Riva*** is the notion that organisational activity forms a *network* of related processes, rather than a hierarchy. The network and its relationships capture the real dynamic structure, whereas a hierarchy is rarely present in the real world.

## FUNDAMENTALS

We start by recapping some of the basic concepts of ***Riva***. (All the machinery in ***Riva*** necessary to understand this paper is outlined here.)

### Basic concepts

The main structuring concept in a ***Riva*** process model is the *role*. A role is an area of responsibility. The notion ranges from concrete things such as organisational posts and functional units (eg *Finance Director* and *Finance Department*) through job titles (eg *Senior Engineer*) to abstractions such as *Responsible Chemist*, *Customer*, *Expense Claimant*, and *Approval of Large Claims*. Roles can also include computer systems and applications, meetings, committees and teams, and rooms.

Everything that happens in a process happens in a role. Typically we will find activities and decisions. We will find concurrent threads of activity, maybe replicated many times. Generally roles operate independently. But in ***Riva*** we stress the collaborative content of business processes and that collaboration takes the form of *interactions* between roles. An interaction is any collaborative act involving two or more roles. Typically we find interactions to do with approval, delegation, reporting, agreement, authorisation, negotiation, questioning and informing.

A ***Riva*** process model takes the form of a *Role Activity Diagram* (RAD) showing the roles and their activities, decisions, threads, and interactions. In addition a RAD will show *events* (triggers) and *goals* or *outcomes*.

A sample RAD is shown in figure 2. Almost as important as the activities and decisions are the innocent-looking vertical lines in the diagram that apparently only connect blobs: in fact these represent *states*. Some of the states are interesting or desirable: they are milestones, outcomes and goals.

### Process relationships

The RAD notation allows us to capture the flow of an individual process. We now want to look at how different processes, and indeed models, can be related. ***Riva*** recognises three forms of relationship:

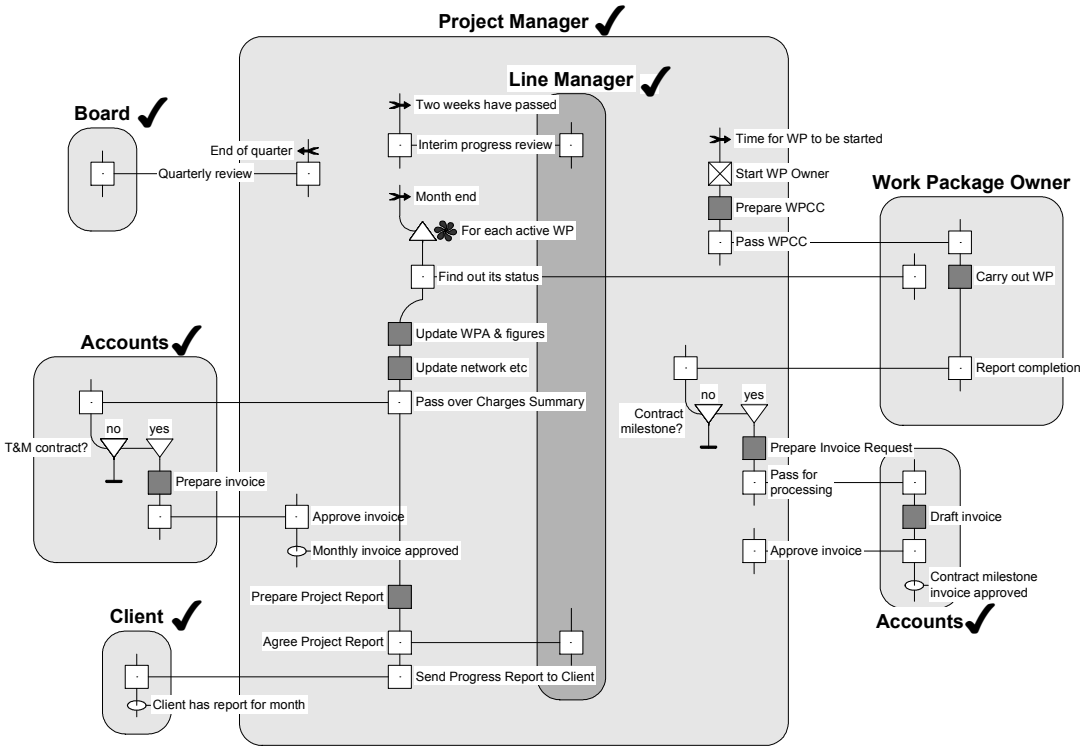
– composition

– activation

– encapsulation.

*Composition* can be thought of as the interaction of *processes*. For instance, during the annual budget setting process, the Board will set financial targets for each product and communicate them to the Product Managers who take their targets back into their respective product development processes. A number of independently operating processes – the budget setting process and the several product development processes – come together and interact, in this case through a shared role: the Product Manager. (Strictly, the single *instance* of the budget setting process interacts with the many *instances* of the product development process. This notion of *instance* is central to a proper understanding of ***Riva***. It is also central to adequate modelling of business processes and yet is missing from most other modelling approaches which thereby provide no way to capture one of the most important sorts of concurrency in organisations – the simultaneous execution of the same process for many cases.[3])

*Figure 2 – A simple sample RAD*




[3] Note that ***Riva*** actually offers three dimensions of concurrency: concurrent threads within a role, concurrent instances of a role, concurrent instances of a process. All of these have very real existence in the real world.

*Activation* occurs when one process starts up another (again, strictly, when one process instance *instantiates* another process). For instance, if the Board decides to take on a new product into the company's product portfolio it will start up the product development process to do the work. When a software house wins a contract to deliver a system to a client it starts the project process to do that work. Activation is typically related to task forces and project teams.

*Encapsulation* is the closest that ***Riva*** gets to hierarchical decomposition! When we draw a RAD we recognise that it might be appropriate to 'bottle up' a mass of action into a single 'black box' activity on the RAD. But we might then want to draw a further RAD that treats that single activity as a process in its own right. The second RAD is then said to be encapsulated in the activity of the first RAD. (Even then ***Riva*** encapsulation is not like the unhelpful strict decomposition of Data Flow Diagrams and IDEF0 diagrams: a black box activity behaves more like a window when you open it than a box, but this is outside the scope of this paper.)

## Types of processes: case processes and case management processes

When we observe an organisation at work we see *units of work*. In an insurance company we see insurance claims being processed, and we expect to see a process for dealing with a single case of an insurance claim. This is a typical example of a *case process*: the process for dealing with a single case, from the time it comes into being (eg when the claim arrives at the post room of the insurance company) to the time when it has been dealt with (eg when a reply to the claimant leaves the building). Every time a unit of work arrives it starts its way through its case process. During the process many roles might play a part in handling the case. And at any one moment there might be many cases in progress, each at a different stage in the process: cases just arriving, cases currently with the loss adjuster, cases with the claims officer; and very often different parts of the processing of the case going on in different roles at the same time. We can model (or design) such a case process as a RAD.

But, the moment we find more than one case potentially in progress, we can expect to find another, connected process that is concerned with managing the flow of cases through the case process. This *case management process* is responsible for scheduling, allocating work to people, monitoring flows, prioritising, accepting new cases, negotiating in the event of priority problems, in short: management activities. Again, we can model (or design) such a case management process as a RAD, and we can show how it is connected to its case process.

So, if there is a unit of work, we will probably find some sort of case process and a corresponding case management process to deal with it. This is a key principle in the ***Riva*** method for constructing the process architecture.

Case processes can vary in 'size'. In a pharmaceutical development company, one of the core business processes (these are almost invariably case processes) is what is sometimes called the 'Molecule to Market' process, which starts with the discovery of a promising molecule and ends (it is hoped) with a safe and efficacious drug on the market. This is a process that can take a dozen years and involves many roles. We might say that the unit of work is 'very large'. Equally units of work can be very small in the overall scale of things. In a pharmaceutical company a common unit of work is an *assay* where a sample of chemical is tested against a specification for purity. This might take a few hours and a few roles.

Within the company you will also find a case management process corresponding to the Molecule to Market case process: this process is responsible for managing the company's compound development pipeline, choosing the therapeutic areas, putting promising compounds into development, stopping the development of those that fail to live up to their initial promise, deciding priorities between compounds for development funds, and so on. This is a constant and crucial process with many parts to it, and one that is equally important to the success of the company.

We can see quickly that case processes will be triggered by the birth or arrival of the unit of work and finish at its death or completion. Case management processes, on the other hand, tend to be triggered by the clock or the calendar, or exception conditions occurring somewhere in the set of case processes.

## Relationships between units of work

When we examine the *dynamic* side of an organisation from the point of view of the units of work we frequently see one unit of work generating other units of work. For instance, during the development of a drug compound in a pharmaceutical company, a number of clinical trials will be carried out to determine the compound's efficacy and safety. In ***Riva*** terms, the unit of work corresponding to a single compound generates many units of work that are clinical trials. To be even more technical, 'one instance of the compound case process generates many instances of the clinical trial case process'. A software house will undertake projects for clients. Each project is a unit of work that goes through a defined project case process. During the project there will be a number of work packages, each of which goes through our work package case process: the project unit of work generates many work package units of work.

This recognition that the principal dynamics of an organisation are captured through the 'generates' relationship is central to ***Riva***. In fact we can perhaps already glimpse the idea that the relationships between the units of work in the organisation tell us about the relationships between their corresponding case and case management processes. This is the clue we need for constructing the organisation's process architecture.

## ESSENTIAL BUSINESS ENTITIES : WHAT BUSINESS IS THIS ORGANISATION IN?

In any business there are 'things', *entities*, that one cannot get away from. They are there simply because of the business the organisation is in. For instance, in a pharmaceutical company there are drug compounds, clinical trials, assays, and batches of raw compound. As long as the pharmaceutical company is in the business of developing compounds that are subject to regulatory control, these things will exist: they are the essence of the business. They are its *essential business entities* (EBEs). We can also regard something as an EBE if it is a 'given', ie we cannot re-engineer it away.[4]

Sometimes we find an EBE that we suspect is there only because of the way the organisation has decided to do its business (eg 'Purchase order'). We immediately ask whether it is 'essential'. Of course, our task might actually be to look at the process for dealing with a purchase order ignoring whether they are a good thing or not, or it might be that purchase orders are here to stay in this organisation. But in a re-engineering environment we might consider whether the more useful EBE is the 'purchase' and that the 'purchase order' in simply an uninteresting mechanism.

An EBE can be something physical and concrete such as a batch of drug compound (you can stick your finger in it in the barrel once it exists), or something rather abstract such as a clinical trial (you could see a clinical trial going on but you couldn't touch it), or something entirely abstract such as a request to change a clinical trial (eg 'I have decided to double the strength of the tablets'). The entirely abstract EBEs often prove to be very important.

Some EBEs will correspond in the organisation to *units of work* (UOWs). For instance, a clinical trial is a unit of work: it starts, proceeds and stops. A drug compound is a unit of work: it is invented, tested and developed, taken to market, and finally withdrawn. Some EBEs will not correspond to units of work: a 'Company standard' will have a lifetime but we might choose not to think of that lifetime as a unit of work (but, on the other hand, we might).

At this point we might also look around for units of work that we might have missed. Individual departments are often in place to deal with one sort of unit of work (eg the Analytical Chemistry Department deals with assays), so a scan of departments might yield units of work. Some of the most important are also not obvious or even unrecognised. At one

[4] The older among us will identify this with Michael Jackson's JSD (*System Development*, M Jackson, Prentice-Hall, 1983). A JSD entity must "perform or suffer actions, in a significant time-ordering, exist in the real world ...; and be capable of being regarded as an individual, and, if there is more than one entity of a type, of being uniquely labelled." Much of what follows has strong parallels with JSD's concepts.

pharmaceutical company we found people who thought their unit of work was preparing the patient packs for clinical trials. When we looked they did indeed do that, but additionally – and very importantly – they spent a great deal of time dealing with changes of mind from clinicians, so they had a further unit of work which was the 'Change to clinical trial': it arrived, it had to be dealt with, and finally it was incorporated into readjusted schedules.

Our first step is therefore to list the EBEs and our second will be to filter those down to UOWs.

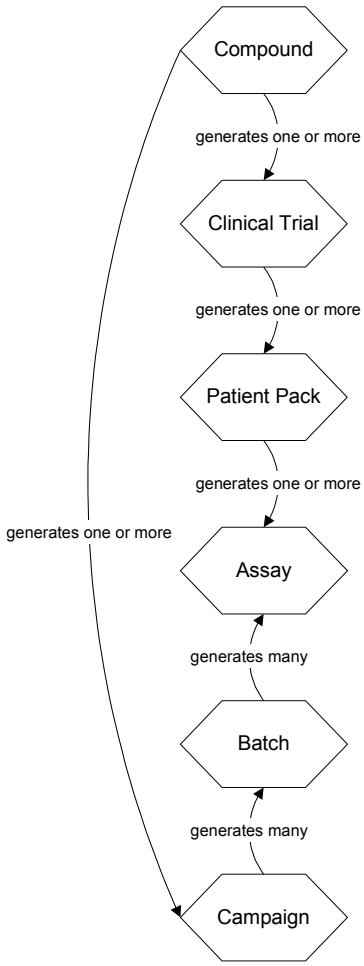
## THE UOW MODEL

Now that we have a list of the units of work of the organisation we map their relationships.

How do we decide which relationships are interesting? Because we are interested in the *dynamic* relationships between the processes we will want to concentrate on the *dynamic* relationships between the UOWs. The fact that 'a Standard Procedure is related to a Task' is not an interesting relationship, since there is no dynamic content to the relationship. However, 'a Clinical Trial requires many Patient Packs' is an interesting relationship because it tells us that we can expect the process for a clinical trial to 'generate' or 'activate' or call upon in some way the process to do with a patient pack (the box containing the drugs for one patient in a clinical trial, and, moreover, we can expect that relationship to be repeated ('requires many'). So the relationships that we will draw up will especially be those that could involve words such as 'require', 'call for', 'generate', and 'activate'.

We can now construct a diagram in which we show each UOW and the dynamic relationships between them; we name each relationship and identify its cardinality (one-one or one-many). For example, 'A Compound requires many Clinical Trials'. Figure 3 gives an example of the sort of UOW model we might produce.

*Figure 3 – A sample UOW model*




## THE HYPOTHESES

We now hypothesise that *the process architecture of the business is aligned to these units of work*, on the grounds that each has a life history and that corresponding to that life history there will be a process – a case process (for a discussion of such, see chapter 7 of *Business Processes*).

In particular, we hypothesise that *for each unit of work there will be a case process*. For instance, the case process for the UOW 'Drug compound' might be called 'Handle this unit of work'. We might also have 'Handle a drug compound', 'Handle a clinical trial', or 'Handle an insurance claim'. Such a case process might be expected to start somewhere around the point where the entity comes into being or enters the realm of the model (eg a promising molecule leaves Discovery and enters Development) –

this corresponds to the trigger of the process – and to finish where the entity ceases to be, or leaves the realms of the model (eg a compound is withdrawn from the market), which probably corresponds to some goal of the process, eg 'Successful closure of the insurance claim'.

Moreover, at any one time, for any one unit of work, we can expect the organisation to have a number of cases at various points in the process, ie for there to be a number of instances of the case process. For instance, a drug company has a number of compounds in different stages of development at any one moment – its 'portfolio'. We then hypothesise that *for each unit of work there will also be a case management process* which is there to manage the set of case process instances. For example, in a pharmaceutical company, the process that manages the portfolio – deciding which compounds should enter Development (new case process instances), which should be abandoned (termination of process instances), which should be given priority, etc – is the case management process for the 'drug compound' unit of work.

Consider the general situation. Suppose that 'UOW *X* requires one or more instances of UOW *Y*. We first posit the existence of the following processes:

– *X* case process
– *X* case management process
– *Y* case process
– *Y* case management process.[5]

We reinterpret the relationship between *X* and *Y* by saying that the *X* case process 'calls up' potentially many instances of the *Y* case process, and we posit that this is done via the case *Y* management process, in particular that the *X* case process communicates with the *Y* case management process which in turn instantiates a *Y* case process. This reflects what happens in the real world. During a project, I will wish to raise a number of invoices for payment from the client. To do this I go to the person who manages invoicing and ask them to start the process to prepare an invoice for me. *X* is the Project UOW and *Y* is the Invoice UOW. The Project case process asks the Invoice case management process to start an Invoice case process.

The *X*-*Y* relationship can take one of two forms: the *task force* relationship and the *service request* relationship. I shall illustrate the difference with the example in figure 3.

At various points in the life history of a pharmaceutical drug compound (ie during the Compound case process), decisions are made to start, change or terminate clinical trials. The management of clinical trials

[5] In the full ***Riva*** approach, outside the scope of this paper, we also hypothesise the existence of a *case strategy process* which has as its subject matter the case process and the case management process.

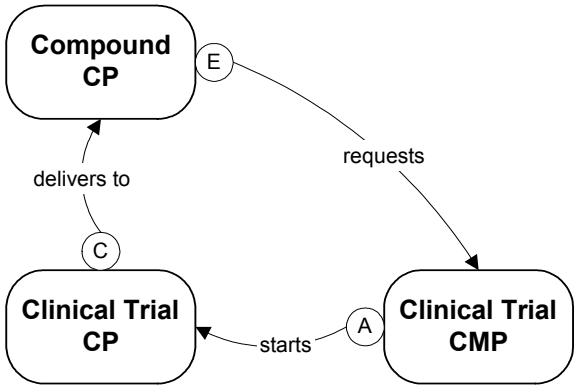
is probably entirely in the purview of the compound team and what happens is that the compound team itself starts off the process for carrying out the clinical trial. So the Compound case process in fact *contains* the Clinical Trial case management process; in RAD terms we would expect to find Clinical Trial case management for a compound *within* the RAD for the Compound case process: *encapsulation*.

The Clinical Trial case process however will have a separate existence; in RAD terms it will have its own RAD and the relationship with that of Clinical Trial case management will be one of *activation*: the Clinical Trial case management process activates (instantiates) the Clinical Trial case process.

When a clinical trial finishes we might expect it to report back with results to its 'customer process', ie the Compound case process, rather than going back through the case management process. The relationship in this case is one of *composition*, where two independently running processes interact.

We can represent this as in figure 4.

*Figure 4 – A hypothesised process architecture for a 'task force' style request*




In the case of a clinical trial, a wholly new group (a sort of task force) is probably created to carry out the process of running the clinical trial: the clinical trial team. But in other situations, the relationship is indirect and the execution of the new case is requested *from a group who provide that as a service*; an example in figure 3 is the Clinical Trial process requesting Patient Packs (of drugs for a certain patient for a certain trial of a certain drug) from the Pharmacy, a permanent group that specialises in preparing patient packs. At any one time the Pharmacy is preparing patient packs for any number of clinical trials, so when our clinical trial team wants patient packs to be prepared they send a request to the Pharmacy who then schedule the request into their other work. In other words our Clinical

Trial case process instance communicates directly with the Patient Pack *case management* process instance which in turn creates a new Patient Pack case process instance for each patient pack required. A further example is to be found where 'a Batch of compound involves the testing of many samples'; such samples go to a dedicated group – Analytical Chemistry – for analysis. This is typical of how a service function operates.

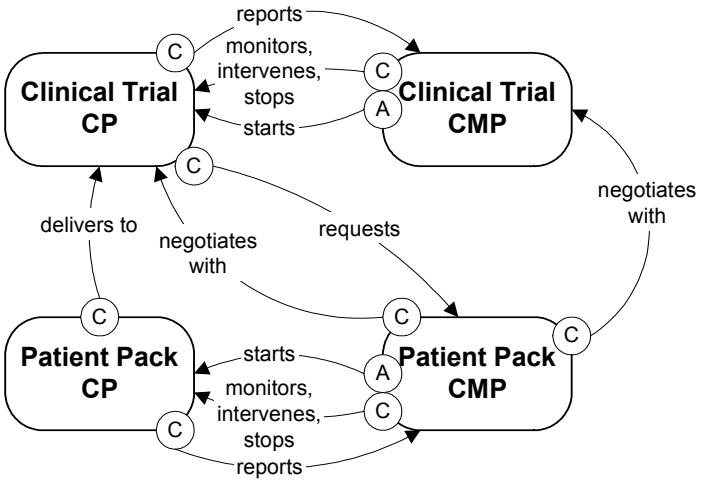
The relationship between the Clinical Trial case process and the Patient Pack case process is therefore indirect and is mediated by the Patient Pack case management process. It is made up of three relationships:

– the Clinical Trial case process *sends a request to* the Patient Pack case management process (composition);
– the Patient Pack case management process activates the Patient Pack case process (activation) to prepare a patient pack;
– the Patient Pack case process delivers its output (a patient pack) to the Clinical Trial case process (composition).

However, we have one more refinement to make. We can expect that because a service function case management process receives service requests from a variety of sources (typically different case processes) there will be times when priorities and schedules cause potential conflict between contesting groups for the attention of this shared resource. When this happens we observe the case management process negotiating priorities with the requesting parties, potentially just the individual requesting case processes, but possibly with their case management process(es) who perhaps have the overview necessary to talk priorities and negotiate. This is shown in figure 5.

In all cases negotiation is a *composition* between the processes concerned.

*Figure 5 – A hypothesised process architecture for a 'service function' style request including negotiation*




The difference between a task force and a service function deserves a little more investigation. One way of distinguishing between them is to observe that, in the task force situation, the management of the set of task forces (eg *Clinical Trial CMP*) is under the control of the requesting case process (*Compound CP*) which will allocate *its own* resources and funds to the task force; this is reflected in the fact that the *Clinical Trial CMP* process is encapsulated in the *Compound CP* process. On the other hand, in the service function situation, the management of the service is under the management of another body (eg the Pharmacy that prepares patient packs) which has its own resources and funds and manages them to service requests from many sources. In truth, this alternative structuring does represent a re-engineering opportunity within the 're-engineering-proof' process architecture we are creating: an inflexible centralised service could be disbanded in favour of giving individual groups the freedom to set up their own task forces as and when they require them; or inefficiently replicated DIY processes could be replaced by a central service achieving economies of scale.

## THE FIRST-CUT ARCHITECTURE

Knowing how to deal with the two relationships of service function and task force we can, essentially mechanically, transform our UOW model of figure 3 into a first-cut process architecture as in figure 6.

In some cases a case management process can simply consist of a few activities inside another process. In this case it will not benefit the modelling much to draw separate RADs for the case and the case management process: we can combine the two in one RAD. In our example, the Clini-

cal Trial CMP and the Campaign CMP can be folded into the Compound case process. The result is as shown in figure 7.

*Figure 6 – The full hypothesised process architecture for the UOW model in figure 3*

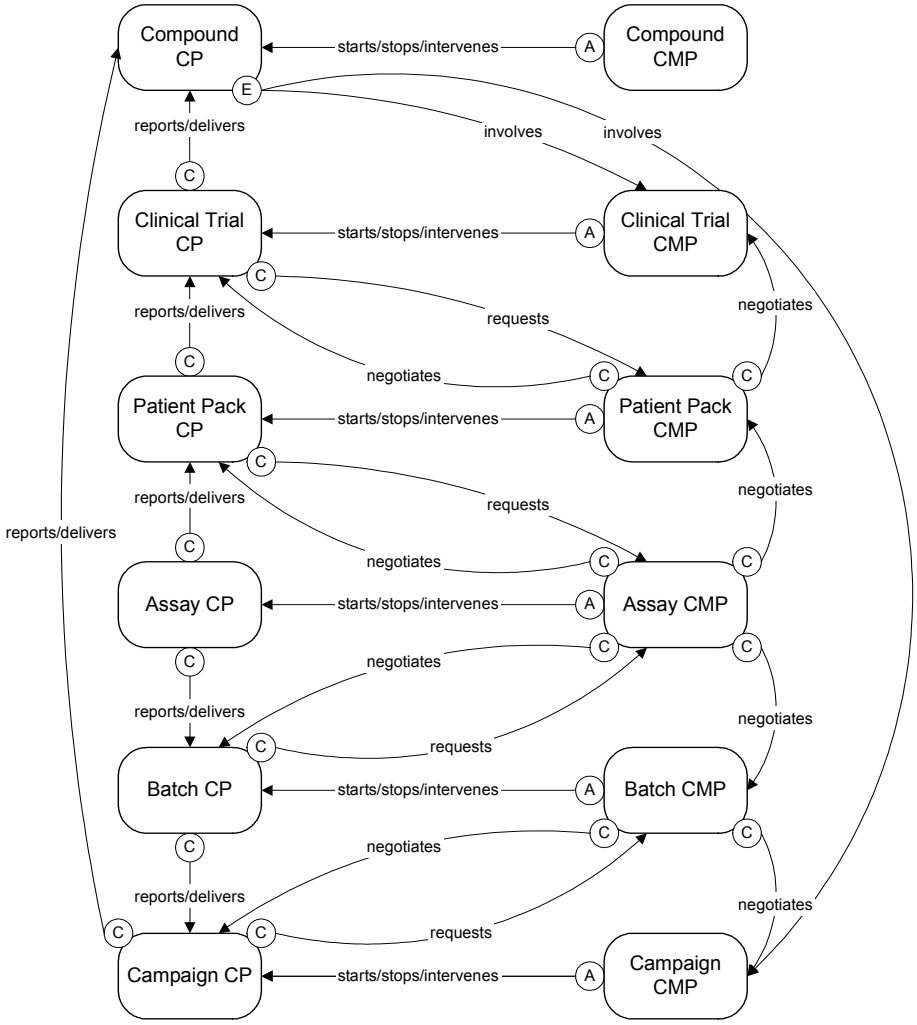
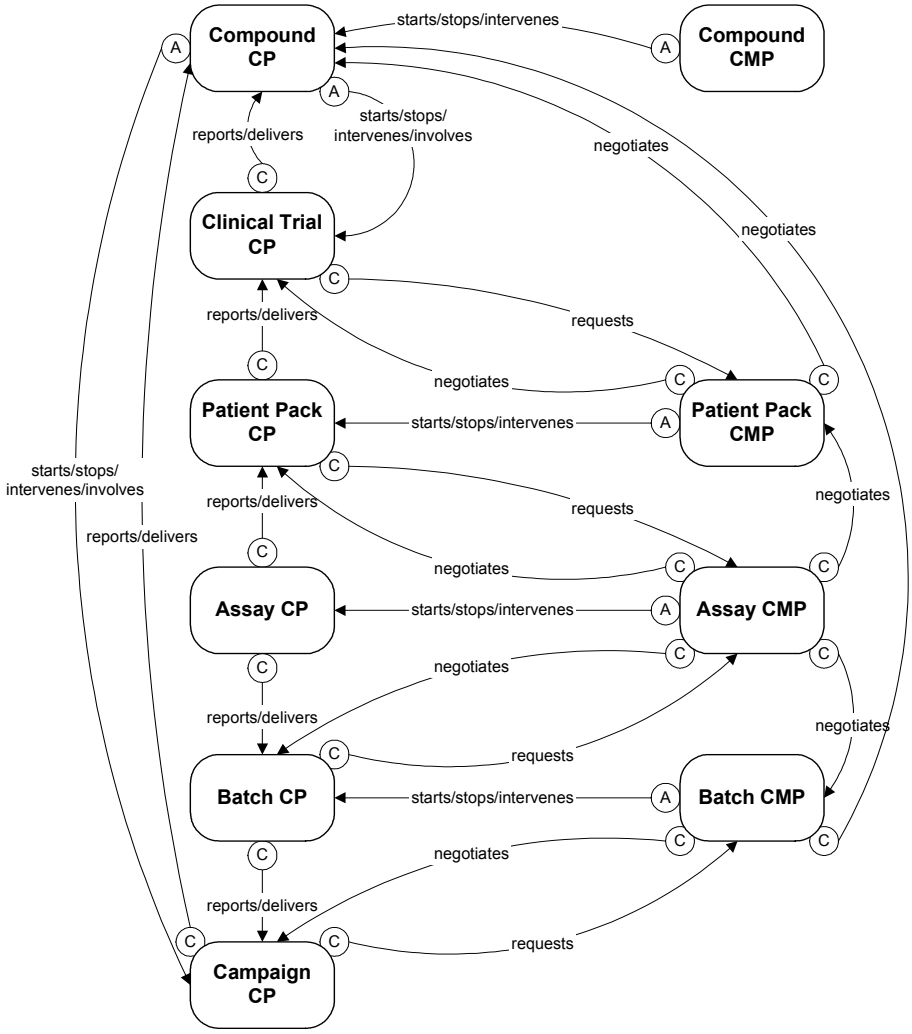


*Figure 7 – A reduced process architecture for the UOW model in figure 3*




## THE PROCESS ARCHITECTURE AS SEARCHLIGHT

The point was made earlier that units of work come in a variety of sizes: a Compound represents a much larger unit of work than an Assay. We can also see a partial ordering ('generates') amongst units of work. It is likely, though not certain, that as we follow the 'generates' relationships we will find our way to 'smaller' and 'smaller' units of work. This is exemplified by the (real-life) example in figure 3.

This immediately suggests that, in ***Riva***, to add more 'detail' to our process architecture means adding more UOWs to our UOW diagram, with the new UOWs being generated by (probably) 'larger' UOWs already on the diagram. Note that we do not add more detail by decompos-

ing into lower levels of a hierarchy. We add more detail by *adding more nodes to our network*. And indeed this reflects the real world. For all processes have separate, *peer-level* existences in the real world, and interact as and when necessary. The question is only whether we choose to include them in the model or exclude them from it. By choosing to include them we are turning the searchlight on that part of the organisation's activity.

As an example, take the UOW model in figure 3. Currently we only have the spotlight turned on patient packs as UOWs generated by a clinical trial, as indeed we might if we were interested in the world of the pharmacy in our pharmaceutical company. But suppose that we wanted to bring in more detail about the planning and recording of a clinical trial. We might recognise the *Clinical Record Form* (CRF) as an essential business entity, essential to the business of conducting and recording a trial and we would add the UOW to the UOW Diagram. For simplicity we have also supposed in the figure that we are not interested in the Patient Pack UOW. We have moved the searchlight and illuminated this area of the organisation's activity further. We could equally well illuminate several areas at the same time depending, as ever, on our purpose.

## USING THE PROCESS ARCHITECTURE

At the beginning of the paper I identified a number of reasons why a sound architecture was essential in several situations where the business process is under examination.

The architecture generated by the approach described here is designed to be sound in the following senses.

Firstly, by deriving the architecture from the entities that are at the heart of the organisation's existence we ensure that the architecture we have designed is entirely deduced from *the business the organisation is in*. We have not considered at all how the organisation chooses to do business: its organisational structure, whether it is a command-and-control or empowered or consensus organisation, whether it operates strict hierarchical procedures or allows cross-functional communications, whether information is treated as shared or on a need-to-know basis, and so on. The process architecture says 'if you are in this business you will have these processes in these dynamic relationships'. It is only when we start to look inside the individual processes and at how their relationships are implemented that we introduce culture and style by our choice of roles and the interactions we require between them, in particular in the use of approval, delegation, reporting, agreement, authorisation, negotiation, questioning and informing, and the mechanisms for these interactions.

Secondly, by mapping the relationships between those entities into relationships between their respective case and case management processes we ensure that the dynamic content of the architecture matches what is happening in the real world. Our architecture does not impose structure

on our processes in the form of some arbitrary decomposition with no analogue in the real world.

Thirdly, by recognising three well defined ways in which processes can be related we again capture reality, and moreover in a way that can be directly modelled when we look at the detail within processes in a Role Activity Diagram.

The benefits are that we have a process architecture that is likely to 'survive' re-engineering, in other words it is in some sense an *invariant* of the organisation; we have divided the organisational activity along the natural cleavage lines that we were looking for and as a result our processes will have greater cohesion and less coupling; and we have an approach that allows us to turn spotlights wherever appropriate to our concern.

## CONCLUSIONS AND SUMMARY

As a BPM method, ***Riva*** is designed to provide the process designer with a rich set of real-world analogues through the concepts it supports and through its notation. In particular, the notions of role, activity, decision, thread and interaction all reflect real-world phenomena; the notion of instance allows a true representation of concurrency; and the case and case management processes correspond strongly with reality.

***Riva*** also provides the process designer with a simple heuristic approach to hypothesising the process architecture for an organisation, allowing an appropriate degree of detail to be gone to, and allowing context to be retained. The architecture produced is based solely on the business the organisation is in. It can be expected to be an invariant of the organisation and hence as sound a basis as one could find for process design, process discovery and process diagnosis.

(This paper has concentrated on some of the key concepts underlying ***Riva***. For details of the method based on these concepts please contact the author.)

## ACKNOWLEDGEMENTS

My thanks go to colleagues who have contributed to the ideas in this paper, in particular Tim Huckvale. The work has its roots in research undertaken by Clive Roberts (now of Co-ordination Systems Limited) and me, which itself was based on work by Anatol Holt to whom I extend my thanks.