13 Processes and process systems

PROCESS MOBILITY – THE THEATRE OF THE THIRD WAVE

One very noticeable thing about the e-mail example in Chapter 7 is that, as the world 'runs', so the entire 'e-mail process' evolves into an increasingly complex and intertwined beast. There is a cascade, a flux, a continuous succession of changing process instances that are created and that peter out. Every e-mail corresponds to a process instance. Every thread corresponds to a process instance. Every conversation corresponds to a process instance. And every process instance – whether it is for an e-mail or a thread or a conversation – is associated with an individual with an e-mail address. I can turn that round and say that every e-mail and every thread and every conversation is a process (instance). As a conversation develops and as more conversations are started up, there are an increasing number of process instances running in parallel – and we have increasing concurrency.

E-mail addresses are the lubricant: the *mobility* of e-mail addresses helps create and cement the link between the process instances (and their associated individuals). By knowing e-mail addresses and passing them around within emails we can expand the web of threads and conversations: the running process evolves. Linkages – in particular between role instances – are created and changed. If I send an email to a group of people and I make their email addresses visible in the message header, I am effectively introducing them all to each other – they can now communicate amongst themselves and the conversation can blossom. In *Riva* terms, process instances know about each other and can communicate thanks to that knowledge. The process evolves as a constantly mobile network of parallel, interacting and communicating threads of activity.

So, the active process evolves, but the definition remains constant. Put crudely, the same RAD will be 'operating' tomorrow as is 'operating' today. Or will it? We need one more level of dynamism: on-the-fly change of the definition. Let's listen in on our Tutor and Pupil.

Overheard at the water-cooler

Pupil: A while back we looked at the organisation as a theatre – a rather chaotic theatre, which proved to have lots of stages with plays being performed, actors running from role instance to role instance, and a mass of interaction between role instances and process instances (performances) going on constantly. But I did notice one thing: the plays were fixed. Is that the only constant in this madness?

Tutor: I'm afraid not. It's about to get much worse. The scripts of plays can be rewritten. We might have liked *Hamlet* the way Shakespeare wrote it, but we might

also consider changing it for modern times – strengthening the substance abuse angle at the end for instance.

Pupil: Eeeek. So who is it that is changing the script?

Tutor: That depends. In some situations the original author might come along and produce new versions, perhaps rewriting the final act to give it a different outcome, or refining some of the smaller parts for better characterisation, or removing unnecessary material. When a new performance of the play is about to start, the actors can use the new version.

Pupil: That's understandable in the real world: we have all sorts of reasons for wanting to change the way we do things – our processes. But presumably the whole business of changing a script happens outside the theatre?

Tutor: NO! The scripts are in the theatre – that's the only place you can change them. Not only are they in the theatre, they are part of the subject matter of the theatre: in other words, you can get hold of them. Now, you can only change a script by using the *process* for changing scripts. **Handle a script** – let us say – would be just another process, indeed a case process. Scripts are units of work!

The whole point of this theatre is that it is where *everything* happens – there is no 'outside' – and one of the things that can happen there is that you work with your processes. Putting it another way: the theatre supports you *in* your processes, by managing all the instantiation and concurrency; but it also supports you *with* your processes, by giving you all the means you need to write new ones or change existing ones.

Pupil: I can accept that a script can get changed and new performances use the new script. But presumably any performance in progress is unaffected? Please?

Tutor: Why so? Why should a performance not switch to the new version as soon as it's available? Why should it be forced to carry on with the old one? In some situations that might make sense – for consistency reasons perhaps – but in principle we don't need to make performances stick with old scripts.

Pupil: I'm struggling … I have a picture of a 'master' script which could be changed. Any performance in progress might continue with the old version, or switch to the new version. But at least they are using a master.

Tutor: Who said anything about 'masters'? Why shouldn't a performance use its own variation of a script?

Pupil: … because … they … Ok, why not? So, they might start with the 'standard' script for Hamlet but decide to change it in some way for just this performance?

Tutor: Of course! They're doing a lunch-time slot, they're ten minutes in and realise that people don't have time for a full *Hamlet*, so they quickly do a rewrite and present a reduced version. There is nothing fixed or sacred about a process.

Pupil: Now I'm getting concerned about the sort of chaos that will ensue if we let everyone tinker with processes as they please.

Tutor: Then don't let everyone tinker with processes as they please: you must script the **Handle a script** process to control what can and cannot happen to a script. Everything is in the theatre, including control over use of the theatre.

Pupil: My head is hurting.

In Appendix C of his book on the Third Wave, Howard Smith says 'Any theory of process management must … break down distinctions between the process of change, the process under change and change in both.' If this makes our brains hurt, we must work at it in the same way that we made the shift in mind-set from data-orientation to object-orientation. Now we must shift our mind-set to process-orientation.

THE NEW ORDER

In this new environment the business process emerges from the shadows and takes centre stage not just in our thinking but in our computer systems. In the past, the idea of 'process' was a useful back-drop for thinking about an information system, but the system and its behaviour were expressed in terms of data. With the Third Wave, processes become the *subject matter* of the system, not just the background or underlying framework. BPMSs allow their human users to work *in* their processes and to work *with* their processes.

In this chapter I want to examine the requirements that this first-class citizenship of the process places on the technology required to build BPMSs and how *Riva* provides us with the intellectual tools we need in order to use it. Let's reflect for a moment on the history.

I'm old enough to remember when, in the 1970s, a new generation of data-processing systems was touted that end-users would build for themselves using a 'design by example' paradigm. Off-hand I can't remember what happened to it, but I suspect the technology of the time wasn't fully up to the job, and end-users couldn't be trusted of course.

Anyway, the relational database folks took over and ensured that control stayed with the IT Department, by making the levers so many and fangled that the end-users had to be kept well away. But at least the organisation could have a system that allowed it to differentiate itself, even if it meant depending on the IT Department to deliver it.

There is a reliance in the relational model on the constancy of the business's entity-relationship diagram. Our RDBMS can therefore concentrate on adding varying rows to fixed tables; or one could say that it now ossifies our information access system around a fixed set of tables. Of course, if the organisation changes how it works, the diagram is invalidated, as is our information access system. Suppose we draw a picture of the organisation's processes, and suppose we base our process enactment system – our BPMS – on it. What happens when the organisation or its business changes? Will our BPMS be invalidated? Like our pedestrian information access systems, will it be a drag on the business?

You don't have to be very old to remember how the ERP package folks then commoditised information and process. And how the big consultancy firms were ready with teams of lever pullers and knob twiddlers who would 'customise' the standard 'solution' for you. Now you could be like everyone else, more or less, and your staff could move to other employers and

immediately feel at home. Process was no longer a business differentiator. But the process was still lodged on the other side of the door to the IT Department.

When BPR was in full swing, by its very nature it called for major step changes in the organisation and the way it did business, reinforcing and facilitating those changes by supporting them with ERP packages which brought organisation-wide processes that were precoded. We had one set of processes one day and a completely different set the next – with all the disruption that went with it. Given that change is the only constant in business, how many organisations are prepared to undergo that sort of upheaval again, and again, and again, as their business environment changes? We can start to see how important it becomes for the process itself to be there in front of us, the thing that drives the computerised support and – above all – there for us to change as our business environment changes.

Instead of radical change to the entire organisation, suppose we want to start small and support a bit of one process in a part of a department; and that we want to have the end-users – the *process actors* as we could call them – adjust that process until it works the way they want; and suppose that, once they're comfortable, they want to extend its use to the whole department, perhaps incorporating more of that process, or even new, connected processes, and then extend out to another department, or a supplier, or a customer. Shouldn't we expect the process actors to be able to do all that? Adapt the process, extend it, spread it, add another, etc? Surely we don't want them to have to wait for the Process Technology Department to get round to it. Under the Old Order we had a business–IT divide. Under the New Order process owners design and deploy their own processes, obliterating, not bridging, the business-IT divide. To quote an example given by Howard Smith and Peter Fingar 'The ideas of the reengineers had created a chasm between thought and action, between process design and process deployment. TI began to realize that their problem had never been the capacity to dream up new processes. Instead, the real problem was how to make change happen, and do so incrementally and on a continuous basis. They needed to make the newly designed processes operational, and to do so step-by-step, process-by-process. Instead of radical reengineering grand designs, what they needed was an actionable and sustainable approach to operational innovation.' (Howard Smith and Peter Fingar, *Outoperate your competition using the BPMS*, BPTrends.com, May 2004)

Let's imagine another scenario in the New Order. In the world of the fixed and invisible process, buried in the RDBMS or ERP, each patient in our hospital must be processed in the same way if IT have anything to do with it. But surely, when the patient enters the hospital shouldn't the process that they will follow be customised for them, and won't the health care professionals then want to adjust that process in the light of the outcomes of treatment? And if new therapies become available shouldn't it be possible for the patient to be switched to the new process? Not only do we want to change a case process over time, but we may want to customise it for an individual case.

Let's see what all this tells us about the technology.

What is the process equivalent of the entity-relationship model? What is the 'process-relationship model' for the organisation? Well, we have seen how *Riva* provides a fast and reliable way of preparing such a 'process architecture' for an organisation, in particular an architecture that stays the same as long as the business stays in the same business. This sort of 'invariant' architecture is exactly what we need: when we choose a process to enact on our

BPMS we need to choose it from a constant architecture; when we add a process to our BPMS we must add one from the constant architecture. Our end-users must have a shared view of what the processes are and where their boundaries and relationships are. Such a process architecture provides the bed-rock on which we shall 'grow as we go': it chunks the whole process – *the* process (because, in truth, there is only one) – in a way that we can bite off sensible pieces. That's the first feature: our BPMS must be built around a constant process architecture of the sort that *Riva* gives us.

At any one moment our BPMS will hold – I hesitate to use the term – a *process model*. I hesitate because of course it's not a model in the sense of a model ship, something different from the ship. So instead I'll call it the *process potential*. It's the thing that defines all the potential future behaviour of the process.

Of course the BPMS will also hold all the past actual behaviour in the form of all past instances: the *process past*. (Indeed, the process past is really part of the process potential: where we have been helps determine where we shall go. Time future is contained in time past.) The BPMS will hold what actually happened, in principle forever: nothing need be forgotten – why should it be? Total persistence. In particular, total *context* persistence. That's the second feature: our BPMS must provide *total persistence*. When I sign on to my PC it just sits there and doesn't do anything. Doesn't it know what I am doing? Clearly not. One of the things I want to do is work on this book. In fact I am working on several right now, in different ways. I have to remember where in my folder structure I have put files about *this* book, and I have to remember what their names are: what *did* I call that file of typesetting instructions from the publisher for this book and where did I put it? Heavens, why am I worrying about the names of things? I just want things to do with this book on my desktop when I decide to work on this book. We only give things names because we need to be able to find them again! I want the BPMS to *know where I am with what I am doing*, and when I ask to pick up the threads (of that role instance) I want to be presented with everything appropriate, everything in context. No more names! … except for cases: I must be able to tell the system which book I want to work on, but I don't expect to have to know where everything about that book is – it should just come to my desktop. When I choose to work on a particular chapter I must expect to identify it, perhaps by pointing to it in the contents list, but I don't expect to have to know where the files containing it and all its diagrams are: I just want them brought to my desktop again when I point to it. Context is all.

A process is just the way a bunch of people agree to get together – collaborate – to make something happen. The process potential says what, at this moment at least, they've agreed. (The process past is what they have actually done.) If collaboration is what a process is about, then collaboration must be a primitive of the process potential and of the BPMS, the engine that turns potential into past. A *Riva*-based BPMS would support *roles* and mediate their *interactions* and hence make the intended collaboration happen. That's the third feature: our BPMS must be *collaboration-centric*.

The process architecture will remain constant (if we stay in the same business), but of course we shall want to allow our end-users to adjust their processes … on-the-fly. I'll put that another way: at any moment an end-user may change the process potential in the BPMS, i.e. change the possible futures. While the process is going on. There's an immediate implication: nothing must exist until it must. The BPMS must not anticipate the future; it must only be in

the present – and then only at the last moment. That's the fourth feature: our BPMS must use *lazy instantiation* as its prime mechanism. It must instantiate at the last possible moment. Roles, for instance, unwind action by action, part-interaction by part-interaction: the BPMS does not instantiate all the actions and part-interactions in the role when the role comes into being – it waits until the activating conditions become true and then instantiates.

But can we trust those end-users, the process actors, to change the process? Perhaps, perhaps not. Perhaps the CEO would like to ensure that any change is properly controlled (not the CIO, note, nor even the CPO – the Chief Process Officer). So presumably we shall have a process change control mechanism front-end to the BPMS? Of course not, changing a process is a process and is therefore in the BPMS, part of the process potential. In fact – to be perfectly general – everything's in the process potential. If you can do it, it's in the potential. If it's not in the potential, you can't do it. (And perhaps only the CEO can use the process change process – but that will be in the potential too.) Perhaps different processes have different change processes – these will all be in the potential … and open to change themselves (I feel a recursion coming on). That's the fifth feature: our BPMS *contains all there is or has been*. Or, rather, all there is today – tomorrow we shall change it.

So now we could define a simple process potential for a bit of our department and a bit of our process. We'll use that for a while and tune it, on-the-fly. Then when we're happy, we'll grow the process potential to cover more of our activity or more of our department, or another department, or one of our suppliers or whatever. We'll grow as we go, the BPMS lazily instantiating and hence forever in step with us, our past always accessible, our future always ours to control.

Processes exist in their own right. There is a process life cycle: discovery, design, deployment, execution, monitoring, control, change. The Third Wave of BPM is more continuous, more incremental than BPR–ERP and closer to real TQM in that process-centric change is now in the hands of the actors. The process enactment engine is process-neutral; in other words, the process becomes the business of its actors, not the IT department.

In early deployments of the technology we might expect to have to connect legacy systems, packaging them up as web services perhaps. But, in truth, that's a dreary prospect. The BPMS would become just glue. A real grow-as-you-go deployment would use the BPMS and the potential *as its entire world* – after all, everything's in the potential and the past: the process and all, everything – who could ask for anything more? (If you could, then put it in the potential!) Because everything is persistent, including relationships, nothing need be named – everything is in context, the end-user's context. We shall have no need of any RDBMS or document management system or ERP to remember stuff. Indeed, they all become redundant – they have always been small, closed worlds and we must do away with them. And we shall have no need of diary systems, workgroup computing (whatever that was), intranets, or any other of the half-baked mechanisms we have invented to coordinate the workings of the organisation, to support collaboration. All collaboration is in the potential. And the potential is in the BPMS. And the potential is the BPMS.

Later, next to the water-cooler again

Tutor: To calm you down when you started getting worried about uncontrolled changes to scripts, I suggested that you simply needed to get a grip on

things using the **Handle a script** process. I invite you to think through the implications of the existence of this process.

Pupil: Well, I guess that for each script there is an instance of **Handle a script** running. Which means there is a stage where that performance is going on, and that performance has the actual ... the paper script on it. Presumably if someone wants to use the script they get it from that process, from that stage. They can take it away in some form and use it for a new performance on its own stage. So far so good?

Tutor: Yes. But I want to put what you said in a different way: *scripts can be handed around*.

Pupil: Oh dear. One performance can give a script to another performance?

Tutor: Of course. Processes are truly mobile. When an interaction occurs between two process instances, the gram can be a process. In traditional computer systems, data was passed around, or messages were passed between objects. Now you can see why I suggested that the object-oriented paradigm was only halfway to full process thinking: the unit of currency is the process.

Any thoughts on what would make up the **Handle a script** process?

Pupil: Well, there'll be an instance of the process for each script (... and one for itself .. oh dear, now I feel a recursion coming on!) ... and that instance will have as one of its props the process model – or some representation of the process ... as it stands ... the current potential. There will be an editing role of some sort and within that role there will be threads that allow the different actions one might want when editing a process model: adding new roles to the process, adding actions and interactions to a role, and so on. It would be nice if all this was done diagrammatically – we would like to edit the RAD itself ideally.

We've talked about versions of things, but since *Riva*'s theoretical underpinnings are object-oriented I assume that what really happens is that we refine object classes (that define roles, actions, etc) rather than replacing them?

Tutor: Yes, that's right.

KEY POINTS

A BPMS holds both the process potential and the process past.

The process potential can be structured using the *Riva* process architecture.

Individual processes can be represented as RADs.

RADs can be actionable diagrams.

Everything is in the BPMS and in its process context.

If everything is in context, we no longer need to name things except the units of work.

The BPMS is collaboration-centric.

Change on the fly is the norm.

Processes replace data as the new currency.

Revolution is overthrown by evolution: we grow as we go.

Power to the people: a BPMS potentially puts the process back in the hands of the organisation.

A final thought. We have talked glibly about 'instances' of processes as a convenient modelling metaphor. Remember that strictly speaking there are no process instances. The clue was in the way that we 'activated' a process: the word 'activated' was carefully chosen. All that actually happened, of course, was that we instantiated the lead role of the process concerned. Just that and only that. We did not instantiate the process. We had no need to. We associate that instance of the lead role with the case it is handling: the lead role has the responsibility for that case and can be associated with it precisely, one to one. That case (and its lead role instance) are the closest we get to an instance of the case process. We have to ask what would be meant anyway by '*p* is an instance of process *P*'. At the moment of instantiation of *P*, what would happen? What properties would *p* have which were different from those of the case? We clearly could not instantiate anything in *P* ahead of time – any actions, interactions, other roles – because we do not know now what *P* will look like when we get to them. Indeed they might not exist if *P* is changed on the fly. That's what I meant by lazy instantiation.

PROCESS CALCULI

'In many industries, business and systems architects strive to create software applications that accurately reflect their business. Sometimes they do not realize that a perfect simulation is their ultimate aim. Architects in other industries know precisely that this is their task. In the logistics industry, companies often model their IT architecture closely around the behavior of the physical logistics networks they monitor and control. … Gradually [computer system] architects are finding ways to represent the behavior of complex systems – interconnected and inter-related mobile processes – within the business applications they develop. Soon they will realize that mapping business concepts into artificial IT artifacts such as objects, interfaces and procedure calls, should be replaced, or at least complemented, by the process calculus models of the Third Wave. These artificial constructs arose to support the composition of software, not the representation of business. [They] are now looking for methods, tools and systems that are purpose built for business. Increasingly they are looking to business process modeling languages for solutions.'

Thus wrote Howard Smith and Peter Fingar in their book *Business Process Management: The Third Wave*. They hit the nail on the head. Because computers started out as … computers, things that computed, early computer language development was centred around ways of describing computations on numbers (see FORTRAN). When useful amounts of storage became a realistic matter and symbolic data could be kept *about* things, languages shifted slightly to the side and added ways of describing symbolic as well as numeric data and of 'computing' with data – typically moving it around and rearranging it (see COBOL). Even

though our focus of interest has now moved to the *process* about which data must be kept, we still see desperate attempts to use languages for data and data computation to describe mobile processes. It won't work. IDEF0 is one notorious example, and UML another. (There is an old joke in software engineering: you can write FORTRAN in any language.) The process-oriented world needs process-oriented languages. The business process world needs business process oriented languages. Data-oriented and even object-oriented languages can only be tortured into supporting business process thinking. Process-oriented languages need process-oriented methods – like *Riva* – to make them work for us.

Calculi for reasoning about systems of processes have been around for decades. Petri Nets have been used for representing systems but suffer two shortcomings: firstly they lack any business-oriented semantics, and secondly they have limited capacity for the sort of cascading and evolution that we know underpins real-life processes. The original RAD notation was based on Petri Nets. It didn't allow processes to dynamically change: a process had a fixed structure. The *Riva* adaptation added replicated part refinement and role instantiation as ways of generating new process at run-time. *Riva* was based on a derivative of Greenspan's RML that described the dynamics of a process in terms of an object-oriented meta-model with formal semantics. IPSE 2.5 added an operational 'process engine' supporting process mobility.

A crucial question when working with a mass of concurrent activity is how one role instance knows which other role instances it is to interact with. When we walk into a room full of people whom we do not know, we can walk up to someone and introduce ourselves:

> Hi, I'm Martyn.
>
> Hello, I'm Angela.

Once this introduction has taken place, we're subsequently able to interact. But how did we know that the thing we walked up to and said 'Hi, I'm Martyn' to was a person? This might sound a stupid question, but suppose you have just joined GlobCorp and your role is to collect project status information from all the project managers – how do you know which of the 1,200 people in the building counts as a project manager? You need to be introduced in some way. When we walk into that room, how helpful it is for the host to take us by the arm and say 'Martyn, I'd like to introduce you to Brian.' Now we know Brian. We might have an interaction with Brian and then Brian might introduce us to someone else with whom we might then interact. Later we might start another interaction with Brian. Sounds like a civilised party.

When we draw a *Riva* RAD for pure modelling purposes – perhaps as part of a process discovery activity – we don't worry too much about such things. If a Project Manager has to provide a report at the end of the month to their Divisional Director, we simply show an appropriate interaction taking place between *Manager* and *Divisional Director*. We don't worry about how, in particular, the instance of *Project Manager* knows which instance of *Divisional Director* to interact with – how they know which is *their* Divisional Director. But if we are going to get a BPMS to run this process, nothing will happen unless (a) a Project Manager can be introduced to the appropriate Divisional Director, and (b) subsequently the two can be correlated when the reporting interaction has to take place. There will be many instances of *Divisional Director* and our instance of *Project Manager* has to be able to say which one it wants to interact with. These notions of introduction and correlation are vital for enactment.

Introduction typically takes one of two forms: *one-to-one* and *directory*. When we were introduced to Brian at the party, it was a one-to-one introduction. We made a mental note of Brian – his face and his name – and Brian made a mental note of us, so that either of us could start a new interaction later. On the other hand, if we are running a small van hire firm we shall probably have an entry in a trade directory giving our phone number. Someone wanting to interact with us to hire a van can look at the directory listing, choose a hire firm, pick up their phone number and start the interaction with a phone call. At GlobCorp, we might hope to be given a list of all the project managers so that we can interact with them to ask for their reports, without having to be personally introduced to all 42 of them.

Let's look at the *Riva* equivalents. Suppose I am a role instance, *A*. When I instantiate another role, I clearly know the role instance I have created, *B* – I don't need to be introduced. But *B* might need to know who I am so that we can interact later. So when *B* is created we must tell it who its 'parent' was. Then we can have an interaction where I pass over some terms of reference or whatever. I might also want *B* to interact with a third role instance *C*, so I shall have to introduce the two of them – on a one-to-one basis. I shall have to say to *B* 'You need to know about *C* for this matter,' and I shall need to say to *C* 'You need to know about *B* for this matter.' *B* and *C* will then be able to correlate and interact. This introduction takes place over interactions of course … provided the two ends have already been introduced.

Now, in real life there are situations where we don't have to be introduced for an interaction to occur – what I called *service interactions* in Chapter 2. I don't have to be introduced to the teller at my bank in order to deposit some money. I simply walk up to a teller and they interact with me and I with them. No need for the manager (to whom I have already been introduced) to pop out and say 'Mr Ould, this is Ms Farrier. Ms Farrier, this is Mr Ould.' In *Riva* terms, some role instances are ready and able to interact anonymously.

Let's stand back for a moment. We know that processes evolve through the instantiation of roles, as new responsibilities are created. In order to collaborate – interact – those newly created role instances need to 'be aware of' each other. Their identities must be available for handing around, for introducing. Their identities must be 'mobile'. It is the mobility of identities that allows the process to change its structure and evolution to occur. As role instances are created and introduced, so the network of introductions develops and the corresponding network of potential interactions can evolve.

The π-calculus of Milner, Parrow and Walker (Robin Milner, *Communicating and Mobile Systems : The Pi-Calculus*, Cambridge University Press, 1999) generalised earlier process calculi by allowing 'channel names' to be dynamically created and communicated, thereby allowing process mobility, and a new level of dynamism. Taking its cue from π-calculus, BPMI.org has published a representation for business processes, the Business Process Modelling Language (BPML) and a Business Process Modeling Notation (BPMN) – see www.bpmi.org. At the time of writing, the situation with BPM standards is still evolving, with a raft of acronyms (BPSS, BPEL, BPELJ, BPEL4WS, ebXML, ebBP…) washing around.

Wherever the ball comes to rest, as long as the metaphor is one of collaborative, concurrent and mobile processes, *Riva* will help in their design.

SIX VISIONS

Because this process-oriented world is so different from the information-soaked world we currently inhabit, I shall end the chapter and the book with six visions of the New Order. They were originally prepared with David Perrin and Clive Roberts as part of a visioning exercise for a new BPMS. Each is designed to capture one facet of the sort of process-driven world that becomes possible when we base our business systems on our business processes.

Managing globally audited and distributed processes

A Clinical Trials Manager in the pharmaceutical industry can spend a lot of time on planes. Hating to waste time, Francesca uses her four-hour flight back to the UK to work on the three clinical studies she's managing. Via the browser on her PDA, operating over the airborne satellite phone system, she connects to the BPMS back at HQ outside London. After completing security checks, she resumes her role as Manager of the Phase III trial for the new drug Dimoxinol and finds herself taking part in interactions with investigators in France and Belgium. Decisions made and communicated, she moves on to reviewing a proposed change to the protocol – it has been through earlier stages in the process but the BPMS knows that it now needs her approval. As soon as she decides to work on it, the BPMS assembles the necessary documents and places them on her PDA: no need to remember where things are, everything is presented to her in context. A quick scan of the change reveals a serious flaw, so she adds her reasoning and rejects it. The BPMS pushes that part of the study process on automatically by sending the rejection immediately by e-mail to the change proposer.

The Terathroxine trial in Japan has been suffering from poor patient recruitment rates – as she finds out when she switches to acting the role of its manager. So what has happened recently? The BPMS memorises everything that happens, so it's possible to look back at the conduct of any process and see who did what and when. As Trial Manager, Francesca has the necessary rights to act another role in the process – Auditor – and to browse the history so far. It's not long before she discovers that one of her investigators has been taking an inordinately long time taking part in a joint decision that sits on the critical path. Stepping out of that role and back into the Trial Manager role, she checks the background of the investigator to find that this is his first experience of this sort of trial. Time to act . . .

Spreading best practice

The Arts & Media Faculty at Wellow University has recently taken the opportunity presented by some internal reorganisation to redefine its processes. Using *Riva*, they prepare a Role Activity Diagram for each of their processes.

The Faculty team quickly put the processes into the BPMS and start to get immediate benefits – they know that all the work items covered in their process architecture will be managed and tracked through to completion. Not surprisingly, news of the improvements they've made soon gets around, and before long they find themselves leading a 'best practice team' tasked with rolling out their processes to other faculties.

The related group in the Fine Arts Faculty, not known for their love of new technology, is the first to raise its performance. There are slight differences between the two faculties, but it's such a simple matter to add Fine Arts administrative staff as role actors in the BPMS that

they simply pick up the best practice processes, with a view to making on-the-fly modifications not long after roll-out.

The Fine Arts group are most impressed by the fact that they need only register in the BPMS to start adopting the processes by simply playing their roles in them. No software had to be written for them, they simply started using the system with their normal browser at their PCs. After two months using the Arts & Media processes, they are soon adapting them for their own environment.

Building one-off collaborative processes on the fly

Dealing with an emergency is not just about each individual agency doing its job. Time and again rehearsals have shown that close collaboration between agencies is essential in mounting a rapid response.

An urgent message arrives at the Emergency Response Centre: Barrack Hill caves and the houses built over them have collapsed. The Response Centre Manager, Mark, turns straight to the BPMS. The bones of the collaborative process have been drawn up as a *Riva* model over the last two years and are now waiting in the BPMS ready to be brought into play to co-ordinate the emergency services.

No two emergencies are alike – the process has to be constructed in broad outline and details added as the emergency unfolds. Mark has used the process authoring part of the BPMS many times before in rehearsals, taking just minutes to build an appropriate collaborative process, and it's not long before the BPMS is running the collaboration of teams from the fire services, cave rescue, the major utilities, and the hospital crisis centre. Even automatic information feeds to the media are already built into the process and can start as soon as the process is kicked into action. Lives and property are saved.

Customer-focused processes from functional silos

People in need of medical care are frequently treated by staff from many different disciplines, and all too frequently these work almost in isolation from each other even though they share a common objective in treating the patient. This isolation is reflected in separate processes, separate roles and separate information resources.

Each of the resulting 'silos' of activity may well optimise the performance of the individual service, but that doesn't necessarily mean that the total service delivered to the patient is the best it can be.

As a pilot study in more effective focusing of all services on the patient, the authorities in Barsetshire form a cross-functional team from the different groups that need to co-ordinate their work to deliver a service package to victims of strokes. Speech therapy, occupational therapy, physiotherapy, social services, the medical team, wheel-chair mechanics, and many more, first use *Riva* to develop a process architecture for their individual areas, and then work together to place these in a larger architecture for integrating stroke management services. As a result, they are able to construct a single process, focused on the patient, which integrates the processes of each individual group with very few changes.

In stage 2, Barsetshire puts a BPMS in place to run the overall stroke management process, principally by co-ordinating information flows between the groups and notifying each group when its contribution is required. In stage 3, processes from the individual groups are added to the BPMS, and gradually an integrated process develops and spreads.

The stroke patient now feels they are the focus of the work of all the teams that collaborate to improve their quality of life.

Built-in measurement leads to process improvement

The administration team for Rode University's Engineering Faculty has adopted the processes developed at Wellow University and is successfully running the administration of students, courses, awards, and modules using those processes. When the Vice-Chancellor announces that 5% of funds will shift from administration to teaching, yet more savings have to be found, and the Faculty administration team turns to the BPMS in which they play their processes.

As part of the initial *Riva*-based process definition work they have done, each process for dealing with a different type of work item has been defined in terms of the actions of a set of roles and the collaboration between those roles. Those processes are now playing in the BPMS. Thanks to the way *Riva* has grouped their work into case processes and case management processes, it's easy to identify where to insert 'measurement probes'. The collection of data about the performance of the processes immediately becomes automatic.

Some extra analysis activities are added to two management roles and within days performance hot-spots become visible and point at opportunities for process improvement. Almost invariably these occur at boundaries with other, non-BPMS processes owned by other groups, where collaboration can break down all too easily and impact performance. With diagnostic data to hand, the Faculty's processes are extended into the other groups and more reliable collaboration is soon in place, with cost savings following.

Keeping processes under audited control

In his role as Manager of Standard Operating Procedures (SOPs) at pharmaceutical company BenePharm, Sanjay is a man with processes for looking after processes. The industry regulator takes a very keen interest in how things are done in their labs, requiring documented processes and moreover – this is where Mark comes in – processes for controlling those definitions themselves, the SOPs.

His group is small and their processes are relatively few, but managing SOPs involves a great many people around the business: front-line staff must draft them, business managers must approve drafts and changes, changes must be evaluated by an SOP committee, and finally of course they have to be published on the intranet for people to be able to use them. And Sanjay has to be able to demonstrate to the regulatory inspector that all of this is being done in accordance with their SOP management processes – the SOPs for managing SOPs.

This is one load that the BPMS has taken off Sanjay's mind: with his processes loaded into the BPMS they run just as required – no chasing or checking by him or his staff. They can concentrate on the content and usage of the SOPs. And because the BPMS keeps a full record of what happens, that mandatory audit trail he needs comes for free.

Tutor: Wouldn't that all be nice?

Pupil: It would indeed.