7 Dynamism in the world

INTRODUCTION

In Chapter 1, our Tutor stressed that things happen in organisations because a mass of concurrent activity takes place, and that, thanks to instantiation, there is a flux of instances that make that concurrency happen. In Chapter 3 we looked at how a RAD captures the concurrency possible in a single process. Now that we also have an understanding of the process architecture of the organisation – in particular the way that processes activate each other and interact – we can extend our horizons and look at how concurrency occurs across the world that we are concerned with: the 'organisation'. This is the right moment to summarise where we capture concurrency in *Riva* models:

- An organisation has a number of process types, each of which can be 'instantiated'.
- Each process 'instance' operates independently. Process 'instances' can interact.
- Within a process 'instance' each role type in the process can be instantiated.
- Each role instance operates independently. Role instances can interact.
- Within a role instance separate threads can be 'instantiated'.
- Each thread 'instance' operates independently. Thread 'instances' can combine and divide.

I have been strict here: a process instance only exists inasmuch as we have instantiated the lead role of the process, and that role instance has in turn instantiated other roles. Similarly, we don't instantiate threads as such, rather the actions and interactions on those threads. Put another way, if we look inside a 'running' *Riva* model of an organisation, we shall see a flux of role instances, each role instance having a number of action and interaction instances in progress. This reflects precisely what is happening inside the building when we walk in: the world is a mass of acting and interacting responsibilities.

In this chapter, we shall explore the notion of concurrency through two rather different case studies. In the first – a fairly conventional situation – we examine how we can represent concurrency in a single RAD of the organisational activity, or more usefully as a set of processes deduced from the process architecture. In the second we look at a completely different sort of process with a rather different dynamic: email.

CASE STUDY

In writing a book such as this, it is hard to decide at what level to pitch worked examples. A real example, from the real world, will be too big. A sufficiently reduced example from the real world will probably be too simple to be convincing. And the real world rarely offers tidy case studies for textbooks. I have chosen a real project that starts from exactly the sort of ill-formed input that typifies real projects, and it allows me to show the revelatory power of the *Riva* method.

The problem – as originally stated – was to prepare a description of 'the process' (singular) outlined in the following piece of descriptive text. I shall do what I always do: start with a process architecture, in order to chunk the organisation's activity into processes. Once we have the process architecture in place we can start to look at the individual processes in it. Yes, there is more than one.

When we have finished dissecting the processes, we shall model the problem as if it were indeed one process, conflating all the processes onto a single RAD.

The Sentinel Case Study

This case study concerns the work of a European project in the control of antibiotic-resistant infections in children. Often, children are asymptomatic carriers of bacteria that they can pass on to other children by playing with the same toys, touching each other, etc.

The medical infrastructure in place to monitor, contain and destroy the infectious organism typically involves a medical practitioner who collects samples from human individuals and sends them for further analysis. Consequently, three levels of operation can be recognized:

1 The field level – where the medical practitioner interacts directly with the potential human hosts, collects samples and basic characterization of the individual and sends the sample for further analysis at the next level, the microbiological/biochemical laboratory.

2 The laboratory level – which corresponds to the facilities that process the samples sent in from the field level.

3 A central location – where data integration and decision making takes place.

Typically, the human host is mostly characterized at the field level and the infectious organism is characterized mostly at the laboratory level. In our case study, Care Centres, Microbiology Centre and the Epidemiological Information Centre represent the three levels.

At the Care Centre (for example, a nursery school), within a Care Centre Period, swabs are taken from children's noses for analysis. The Care Centre Operative collects these swabs, referred to as C Samples together into a C Sample Batch to send to the Microbiology Centre. This is usually done one class or group of children at a time.

These are called Care Centre Units and the period when their swabs are taken is called the Care Centre Unit Period – this must fall within the Care Centre Period. The Care Centre Operative records all the necessary details (id numbers, etc) in order to ensure that the link between Attendee (i.e. a child) and his or her C Sample is maintained.

At the Microbiology Centre (usually a laboratory), the Microbiology Centre Operative receives a batch of samples, i.e. a C Sample Batch. She records them as M Sample Batches in the local system and then reorganizes them into one or more Sample Analysis Groups to suit the process at the Microbiology Centre. (To make this point clearer, imagine that one Care Centre Unit sends 19 swabs to the Microbiology Centre, another sends 22 and a third 26. The Microbiology Centre can process, say, 35 swabs at a time. The three distinct M Sample Batches – of 19, 22 and 26 – are reorganized into two Sample Analysis Groups of 32 and 35 respectively.) Each swab is cultured on a Petri dish and an attempt is made to isolate different strains of bacteria and then to identify them. This is done using various tests, the main ones being by serotype, by antibiogram, and by pulse field gel electrophoresis (PFGE). Information about strains identified in each swab is sent back to the Care Centre.

Information about strains is also sent to the *Epidemiological Information Centre*. The role of this centre is partly to organize and manage the overall campaigns (e.g. to define the Care Centre Periods) but, more importantly, it determines whether alarms must be raised and to what level. For example, if any antibiotic resistant strains are present, an alarm is raised. However, actions at the Care Centre following an alarm vary from country to country. In Sweden the children must be sent home and the parents are paid by the state to stay at home with them. In Portugal, they are also likely to be sent home, but they return earlier because there is no social provision. If a situation is deemed serious enough, a Care Centre may be closed for a time.

Let's start by listing candidate EBEs using only the text provided. When faced with this sort of situation, spotting nouns is a productive approach. Clearly, in a normal process study we would be running brainstorming workshops.

Child

Medical Practitioner

Sample

Laboratory

Care Centre

Microbiology Centre

Epidemiological Information Centre

Care Centre

Care Centre Period

Care Centre Operative

C Sample

C Sample Batch

M Sample Batch

Class (of children)

Care Centre Unit

Care Centre Unit Period

Attendee

Sample Analysis Group

Swab

Bacterium

Strain of bacteria

Petri dish

Test (e.g. PFGE)

C Sample Result

Campaign

Alarm

Resistant strain

Country.

You can see I have already been careful to ensure they are all entities: we can meaningfully put the word 'a' or 'the' in front of each.

The next step is to filter them using the guidelines listed in Chapter 6 to get to a list of units of work for this 'organisation'. Because we want to model the organisation as-is, we include designed as well as essential UOWs. Arrowed items are those that pass through the filters. Bracketed items didn't and the reasons appear against them. (Aside: this list looks very much the way it did on the white-board during the EBE brainstorming session.)

(Child) = C Sample: we are interested in the life history of the sample the child gives but not of the child. If anything, the child is only a role that plays a part.

(Medical practitioner) a role that plays a part, lifetime not of interest to us.

(Sample) see C Sample.

(Laboratory) lifetime not of interest to us.

(Microbiology Centre) lifetime not of interest to us.

(Epidemiological Information Centre) lifetime not of interest to us.

(Care Centre) lifetime not of interest to us.

(Care Centre Period) a period of time.

(Care Centre Operative) a role that plays a part, lifetime not of interest to us.

☞ **C Sample**

☞ **C Sample Batch**

(M Sample Batch) = C Sample Batch.

(Class (of children)) lifetime not of interest to us.

☞ **Care Centre Unit**

(Care Centre Unit Period) a period of time.

(Attendee) see Child.

☞ **Sample Analysis Group**

(Swab) = C Sample.

(Bacterium) lifetime not of interest to us.

(Strain of Bacteria) lifetime not of interest to us.

(Petri Dish) lifetime not of interest to us.

(Test) lifetime not of interest to us.

☞ **C Sample Result**

☞ **Campaign**

(Alarm) lifetime not of interest to us.

(Resistant Strain) lifetime not of interest to us.

(Country) lifetime not of interest to us.

During the break, next to the water-cooler

Pupil: I was concerned when you so easily knocked *Child* out of the list of UOWs – surely this whole thing is about children?

Tutor: Well, is it? Remember that when we decide on the essential business entities and filter out the units of work, we choose those that characterise the organisation we are interested in – we turn the searchlight on some areas and not on others. We also adjust the intensity of the searchlight, perhaps differently in different areas, to illuminate just those things that are 'large enough' for our purposes. You might feel that my choice in the filtering would not be your choice: that simply means we are thinking about different 'organisations'. We could widen our interest a bit and bring in how we deal with children who are found to be carriers for instance.

Pupil: OK, so you're saying that that's a scoping decision. If we decide we want to include that part of the business, we shall indeed have *Child* in the list and ... there will be a process called **Handle a child** and another called **Manage the flow of children** (though I'm not too sure what that would be about in this situation).

Tutor: Exactly. We could then cover what happens to each child, in particular if they are found to be a carrier, or an alarm is raised. But you might agree that the description we have been given doesn't get into any of that, so we should scope it out by leaving *Child* out of the list of units of work.

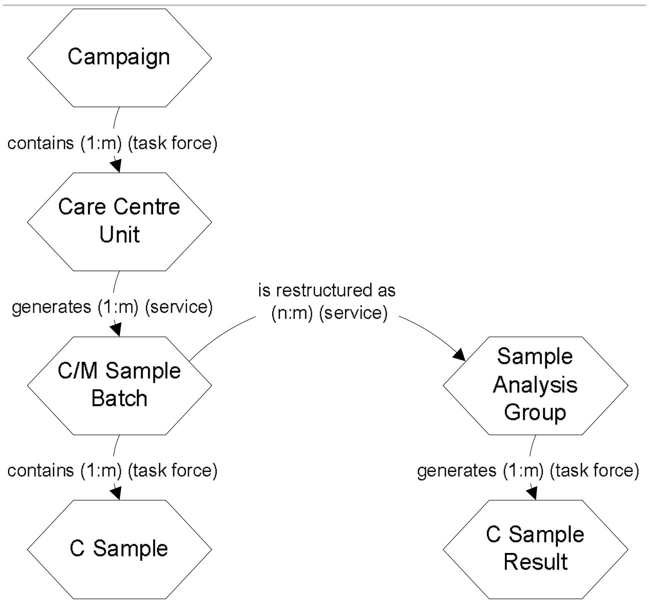
Pupil: OK. Then following the same reasoning, *Alarm* was an interesting candidate. Alarms have a lifetime of interest to someone ... OK, but not us, we're saying. They are the business of national bodies. I guess I've just made a decision about where the searchlight is pointing and how wide I want its beam to be.

I also thought *Sample Analysis Group* was interesting. Sample Analysis Groups are – as the text makes clear – only there to suit the way the Microbiology Centre works: they are designed EBEs.

Tutor: Right. We're faced with the typical decision: are we concerned with developing a pure process architecture which is independent of how we choose to do things, or do we want to include designed mechanisms? In this instance, our concern is with the way things are. If we were looking for re-engineering opportunities we might take the opposite view. In later sessions we shall look at how our purpose will determine which way decisions like this should go.

We can now transfer the six surviving UOWs to a UOW diagram, adding the 'generates' relationships that hold between them. The result is Figure 7-1.

Figure 7-1 – UOW diagram for the Sentinel case study


Campaign
contains (1:m) (task force)
Care Centre Unit
generates (1:m) (service)
is restructured as (n:m) (service)
C/M Sample Batch
Sample Analysis Group
contains (1:m) (task force)
generates (1:m) (task force)
C Sample
C Sample Result

Let's unpick what this is saying:

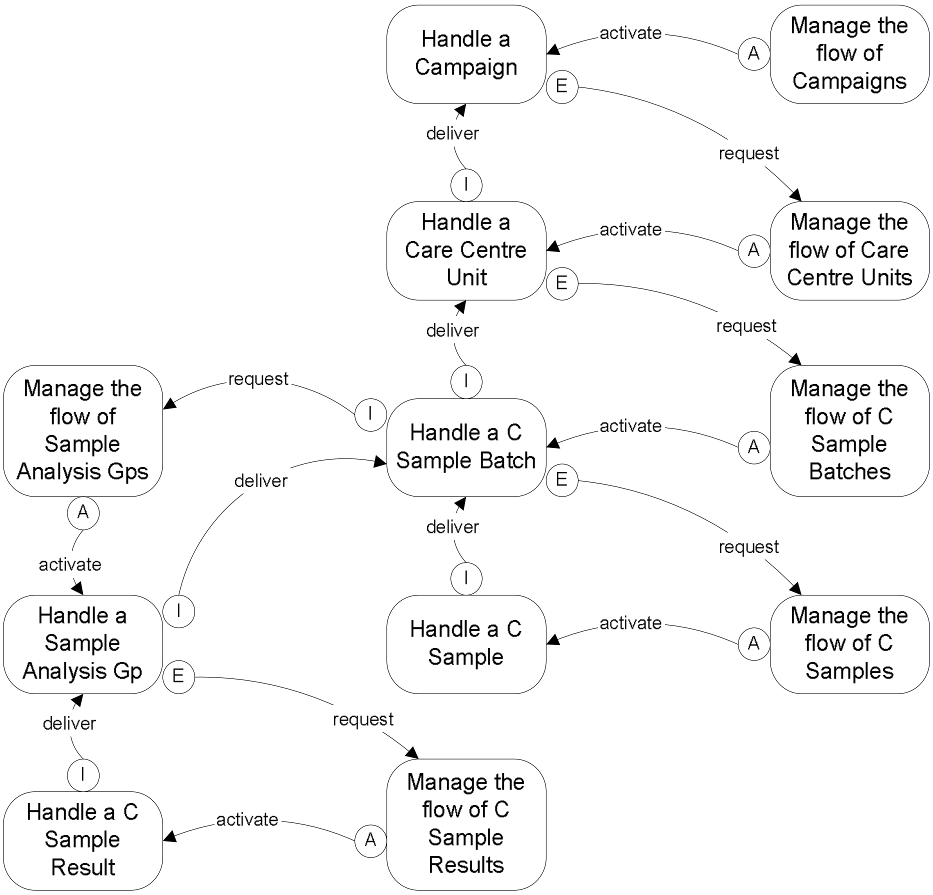
- The 'big' unit of work is the Campaign. Campaigns are defined at the Epidemiological Information Centre.
- During each Campaign, a number of Care Centre Units are generated. This is done within the handling of the Campaign – no outside service is involved, so the relationship is a 'task force' relationship.
- A Care Centre Unit generates C Sample Batches. The latter are the units in which C Samples are sent to the Microbiology Centre which deals with these as a service. The Microbiology Centre has them coming in from many Care Centres.
- C/M Sample Batches are restructured as Sample Analysis Groups for the 'convenience' of the Microbiology Centre. Note the *n:m* cardinality. Sample Analysis Groups are dealt with as a service.
- C Samples arise from the Care Centre's own handling of C Sample Batches. There is no separate service dealing with C Samples for them and others. This is a 'task force' relationship.
- During the lifetime of a Sample Analysis Group, the results for the constituent C Samples are generated. C Sample Results are managed within the context of each Sample Analysis group, so this is also a task force relationship.

We might first think that C Sample Results arise from (are generated by) C Samples. We could certainly say that C Sample Results are 'related to' their corresponding C Samples, in the obvious way. But that is not the dynamic 'generates' relationship that we are looking for, and which does exist between Sample Analysis Group and C Sample Result: it is during the lifetime of the Sample Analysis Group, which is processed in Microbiology, that the C Sample Results are generated.

Note how in this UOW diagram things get smaller as we follow the arrows.

The next step is to mechanically transform the UOW diagram into the first-cut process architecture in Figure 7-2.

Figure 7-2 – First-cut process architecture for the Sentinel study


Handle a Campaign
activate
A
Manage the flow of Campaigns
E
deliver
request
I
Handle a Care Centre Unit
activate
A
Manage the flow of Care Centre Units
E
deliver
request
Manage the flow of Sample Analysis Gps
request
I
I
Handle a C Sample Batch
activate
A
Manage the flow of C Sample Batches
deliver
E
A
deliver
request
activate
I
Handle a Sample Analysis Gp
I
Handle a C Sample
activate
A
Manage the flow of C Samples
E
request
deliver
I
Manage the flow of C Sample Results
Handle a C Sample Result
activate
A

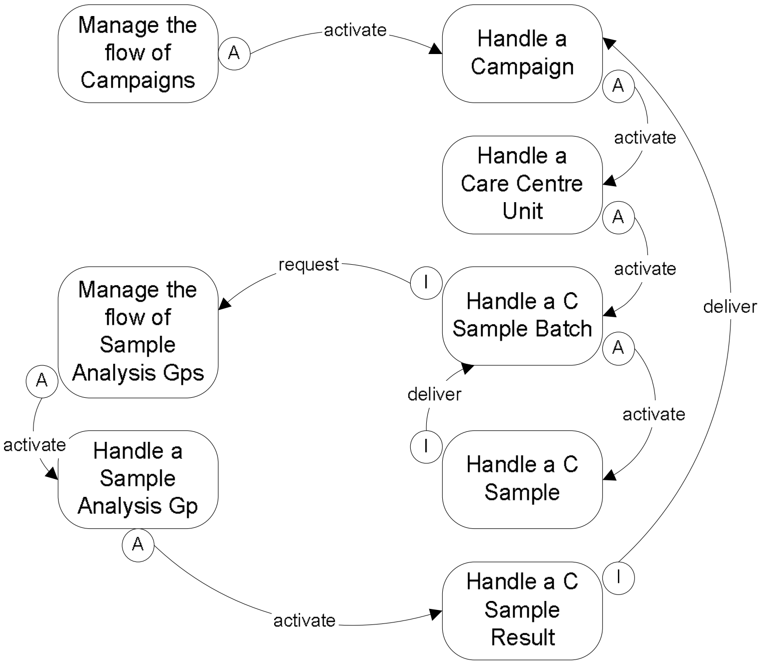
Let's walk through this. (Note that we have not added the case strategy processes. This is common but not inevitable. The case strategy processes sit 'around' their case and case management processes, and this study was concerned with operational activity, rather than strategic planning.)

There is a 'root process' – Manage the flow of Campaigns – which, at appropriate moments, activates Handle a Campaign in order to get a campaign done. A Campaign is made up of a number of Care Centre Units, which are managed by the campaign. The diagram therefore shows Manage the flow of Care Centre Units encapsulated in Handle a Campaign. Manage the flow of Care Centre Units has its own existence and needs to be thought about in its own right, but we are recognising that when we come to model it, it will be 'found in' or best modelled in Handle a Campaign since that is where the flow management actually takes place.

During Handle a Care Centre Unit, C Sample Batches are generated in task force mode. C/M Sample Batches are then converted at the Microbiology Centre into Sample Analysis Groups for which there is separate flow management. Handle a C Sample Batch makes its request to Manage the flow of Sample Analysis Groups which activates Handle a Sample Analysis Group when ready, which in turn goes to Manage the flow of C Sample Results for each C Sample Result it requires. There is a chain of deliveries back through the sequence.

We can now fold processes together where a task force relationship is involved. This yields the second-cut process architecture in Figure 7-3.

Figure 7-3 – A second-cut process architecture for the Sentinel study



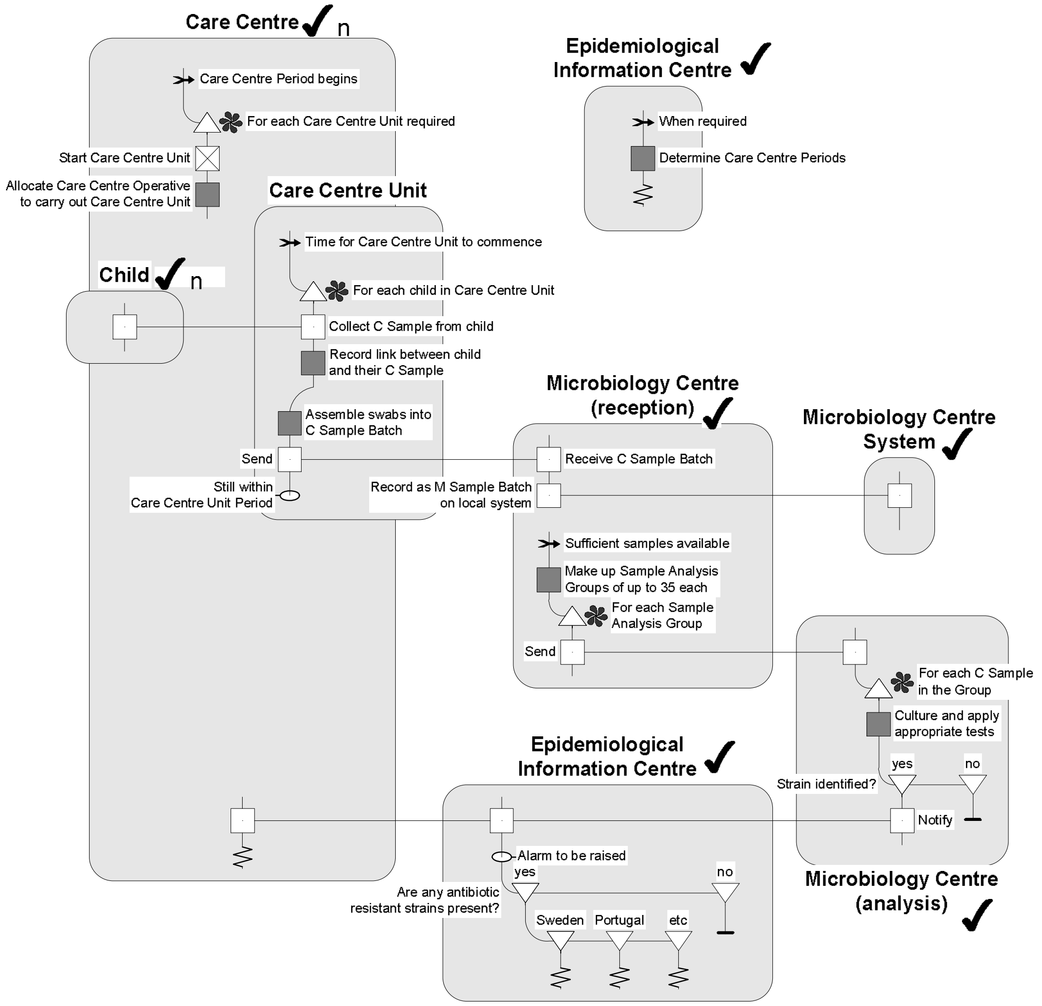
What has happened?

- ☞ We have folded each encapsulated case management process into the single case process that uses it.
- ☞ We have recognised that the chain of delivery from Handle a C Sample Result to Handle a C Sample Batch to Handle a Care Centre Unit to Handle a Campaign can be short-circuited leaving one 'delivers' relationship from Handle a C Sample Result to Handle a Campaign. Remember that the first-cut architecture is derived

mechanically from the UOW diagram and needs to be put against the world. It is Handle a Campaign that wants the results in order to determine whether an alarm needs to be raised.

The net result is that we can recognise eight discrete processes that are covered in the one description. It is certainly possible to draw a RAD as if this piece of the world really were a single process – Figure 7-4 does precisely this, in one way. Spend a few moments chunking the RAD along the lines of Figure 7-3. You will find all eight processes in there – some of them consisting of a single interaction. You will also be able to see the activation and interaction of the processes: sometimes as a role instantiation and sometimes as an interaction between roles. The concurrency that builds up in the process when things get going is created from the instantiation of Care Centre Units and the multiple instantiation of new threads in various roles thanks to replicated part refinements.

Figure 7-4 – The Sentinel case study drawn on one RAD


Care Centre ✓ n
Care Centre Period begins
For each Care Centre Unit required
Start Care Centre Unit
Allocate Care Centre Operative to carry out Care Centre Unit
Care Centre Unit
Time for Care Centre Unit to commence
For each child in Care Centre Unit
Child ✓ n
Collect C Sample from child
Record link between child and their C Sample
Assemble swabs into C Sample Batch
Send
Still within Care Centre Unit Period
Epidemiological Information Centre ✓
When required
Determine Care Centre Periods
Microbiology Centre (reception) ✓
Receive C Sample Batch
Record as M Sample Batch on local system
Sufficient samples available
Make up Sample Analysis Groups of up to 35 each
For each Sample Analysis Group
Send
Microbiology Centre System ✓
For each C Sample in the Group
Culture and apply appropriate tests
Strain identified?
yes
no
Notify
Microbiology Centre (analysis) ✓
Epidemiological Information Centre ✓
Alarm to be raised
yes
no
Are any antibiotic resistant strains present?
Sweden
Portugal
etc

Later, next to the water-cooler

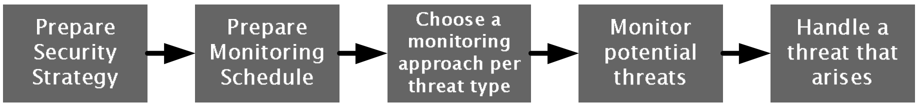
Tutor: I hope you're beginning to see how, by taking the process architecture view of things, we get a much more realistic understanding of the mass of concurrent activity that the organisation indulges in.

Pupil: Yes, I can. It was interesting to see how the RAD you drew looks rather sequential at first glance, but does in fact capture how – when this part of the world 'runs' – a mass of concurrent activity spreads out and then dies away – I can see that it's the replicated part refinements that do that: they create lots of concurrent threads of activity. But the process architecture view and the eight separate processes really make that so much clearer. I can visualise how a single Campaign starts, kicks off several Care Centre Units, each of which ... etc etc, and before we know it there are scores of process instances running independently. In time they die off and the world is quiet again.

Tutor: Yes, people all too easily look at an area of organisational activity and draw it as a sequential thread as if that is what happens. If they think a little harder they might perhaps use a single RAD to capture some or all of the concurrency that is actually going on. This is better – we are getting closer to reality. But if they start with a process architecture then they are thinking 'concurrency' right from the beginning: the separate processes, the dynamic relationships between them that cause new instances to be started ... we're capturing concurrency before getting anywhere near the workings of roles and so on. When we start to open up the individual processes we can explore the concurrency between roles, and by opening up roles we can explore the concurrency within a role. So by taking concurrency in three steps – architecture first, then processes, then roles – we get a firmer and more accurate handle on it.

I once saw a flowchart someone had drawn of their 'process' – in essence it was the sequence of boxes shown in Figure 7-5. Would you care to criticise it with your new understanding?

Figure 7-5 – Not what happens in the real world


Prepare Security Strategy
Prepare Monitoring Schedule
Choose a monitoring approach per threat type
Monitor potential threats
Handle a threat that arises

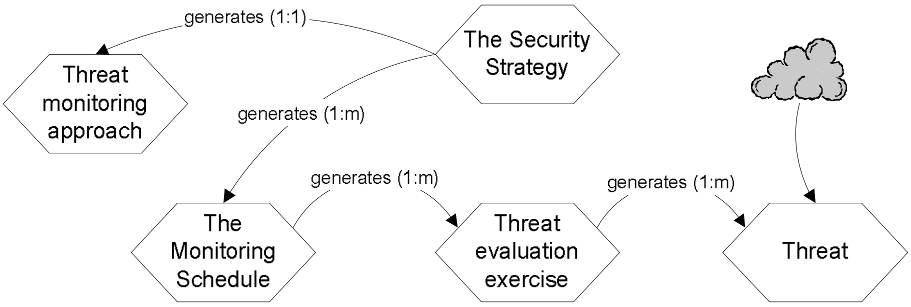
Pupil: Well, the first thing is that I'm sure that they didn't do things in the strict order shown: devise a Security Strategy, then devise a Monitoring Schedule, then choose a way of monitoring one sort of threat, then spot a threat, then do something about it. Does the one security Strategy last forever? Does the one Monitoring Schedule last forever? Does monitoring for threats cease once the first threat is spotted? What happens when the second threat comes along? The more I think about it the more unrealistic it becomes.

Tutor: You're right, it's not what they did, of course. Moreover, in the small print it was clear that the Monitoring Schedule was maintained and adjusted over time. It had its own independent life. What about Units of work?

Pupil: Just what I was thinking. A Threat is a unit of work, to name one. So there will be a case process for that – Handle a threat – and a case management process – Manage the flow of threats. The Monitoring Schedule sounds like something with its own life, and since there is only one there won't be any case management. Ditto for the Security Strategy. So we probably have four interacting processes here instead of one strange sequential one.

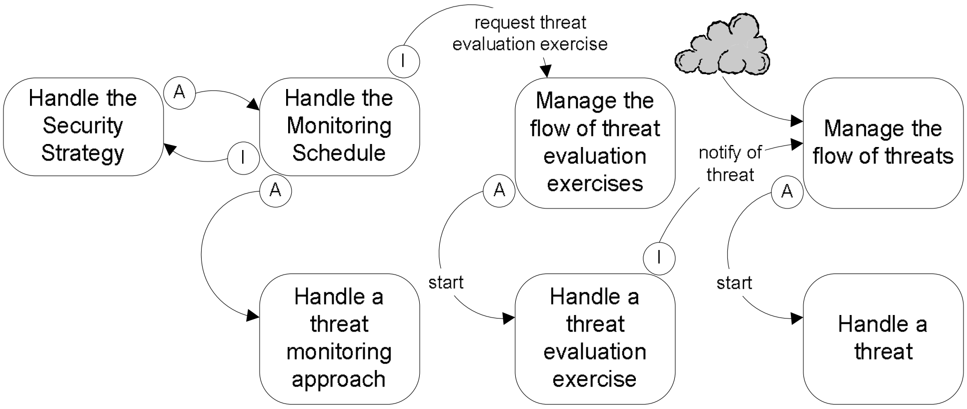
OK, I really ought to start with a UOW diagram. Let me sketch one in Figure 7-6 just from the bare facts that we have.

Figure 7-6 – A UOW diagram for the 'process'



I've guessed that the Monitoring Schedule lists Threat evaluation exercises which take place and reveal threats. I guess threats will also appear out of the blue. So, after a bit of work the process architecture will probably look like Figure 7-7.

Figure 7-7 – A process architecture for the 'process'


request threat evaluation exercise
Handle the Security Strategy
A
I
Handle the Monitoring Schedule
I
Manage the flow of threat evaluation exercises
Manage the flow of threats
notify of threat
A
A
A
I
start
start
Handle a threat monitoring approach
Handle a threat evaluation exercise
Handle a threat

I haven't added subsidiary interactions but I guess it's enough to demonstrate that the flowchart process really didn't capture reality at all.

Tutor: Exactly. Let's pull this together.

If we string things together that do occasionally happen in a particular order, we shall not capture reality, in particular we shall probably miss the potential concurrency.

If we mix up units of work, we shall not capture reality. It will be plain wrong.

If we conflate case processes and case management processes, we shall not capture reality. We shall miss concurrency and most probably get it just plain wrong!

If we don't start with a unit of work analysis and derive the process architecture from that, we shall end up ignoring the potentially concurrent processes and describing a world that does not exist. The resulting process model will not reflect reality and it will be impossible to use. It will not be a basis for process improvement and would be a disastrous starting point for developing systems to support the activity.

An organisation really is a portfolio of processes. We can be more specific: an organisation is a portfolio of case (end-to-end) processes together with the necessary case management and case strategy processes. That's why a *Riva* process architecture fully captures the organisation in process terms.

DYNAMISM AT WORK

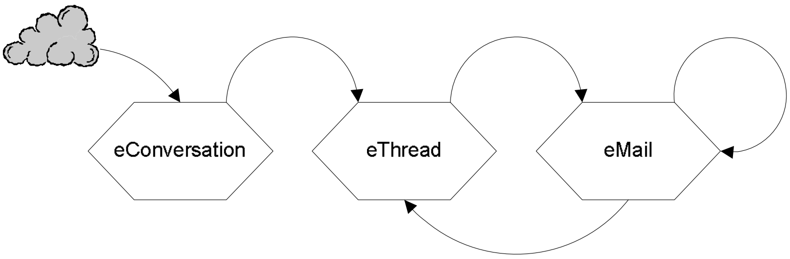
Pupil: That last case study was about ... shall I say ... a conventional process. But my thoughts are straying to something a little less tangible: the world of email conversations. The 'organisation' here is certainly not some functional group. It's the rather abstract world of structured communication by email. I have to ask you whether you think *Riva*'s concepts work here.

Tutor: And I have to say 'Of course!' In fact email is a very nice example of the sort of very fluid and very dynamic process that goes on in organisations and which traditional flowcharts and other notations simply can't deal with. Let's give it a go. Informally, I could describe the world of email as one in which people can start conversations. A 'conversation' is just an area that someone wants to start some 'threads' going on.

Suppose I'm interested in having a conversation about William Morris and the Arts and Crafts Movement. I might start three threads of conversation: one on his Red House pattern designs, one on his Kelmscott books, and another on his Socialist writings. I start each thread simply by sending out an email to some people I think might be interested in it. Each recipient picks up the thread either by responding to it – perhaps to the same group of people or a different group – by ignoring it, or perhaps by sending out an email to another group of recipients on a new thread of their own within the conversation. That sounds like a nice dynamic, fluid sort of process. Where shall we start in our analysis?

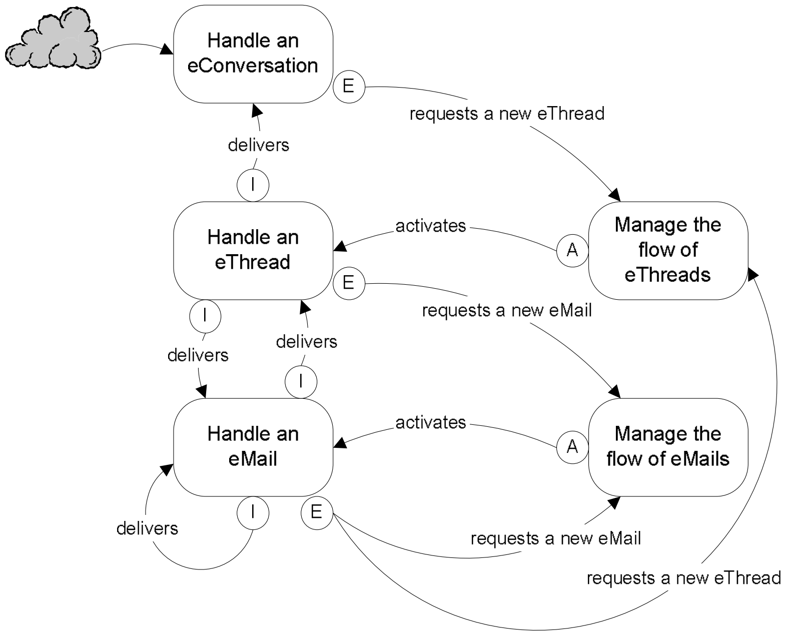
Pupil: Well, as good *Riva* analysts, our first instinct will be to draw a Unit of Work Diagram (Figure 7-8). I'll assume that an eConversation can start spontaneously, in someone's mind. And that an eConversation starts a new eThread with a particular message title. Let's equate each eThread with a message title. Then each eConversation evolves in the form of one or more eThreads where an eThread takes the form of one or more eMails. Obviously, eMails can generate new eMails and/or eThreads.

Figure 7-8 – UOW diagram for an email system



Our second instinct will be to transform this into a first-cut architecture (Figure 7-9).

Figure 7-9 – First-cut process architecture for an email system

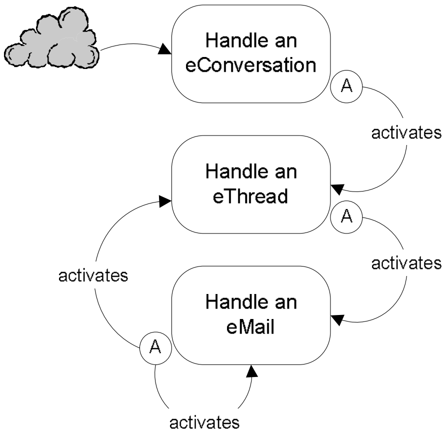


Tutor: Good. Now reduce it to the second-cut architecture.

Pupil: I can see some obvious reductions: 'delivery' resulting from 'generates' means nothing in this context, so we can remove those process interactions. All case management is clearly done in task force mode and so a CP does the CMP for generated UOWs. I guess that all case management is null anyway in this simple case ... but I can imagine useful facilities to do with logging and measurement and so on. Anyway, we seem to be just left with a set of 'activates' relationships between the three case processes reflecting the UOW structure (Figure 7-10).

Should we be surprised by the final, second-cut architecture?

Figure 7-10 – Second-cut process architecture for an email system

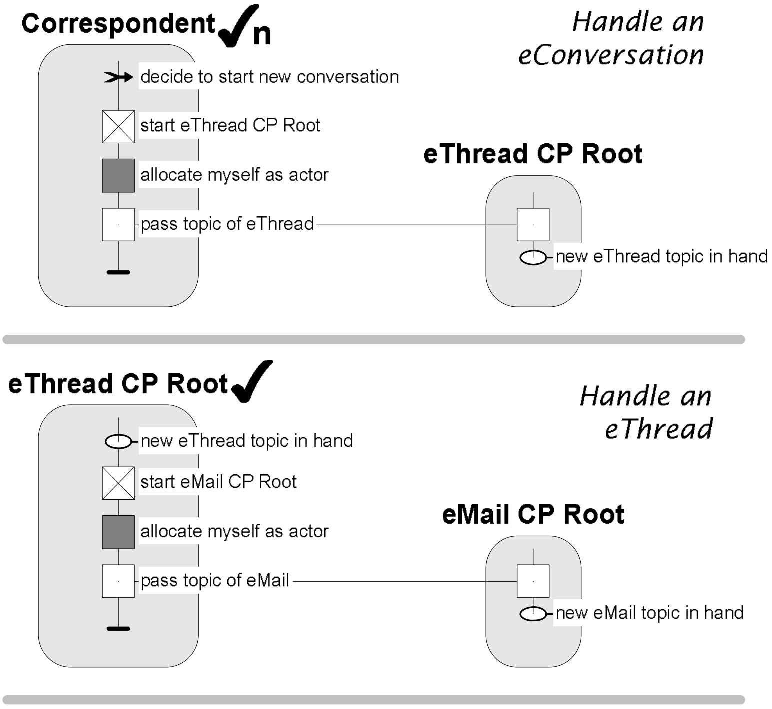


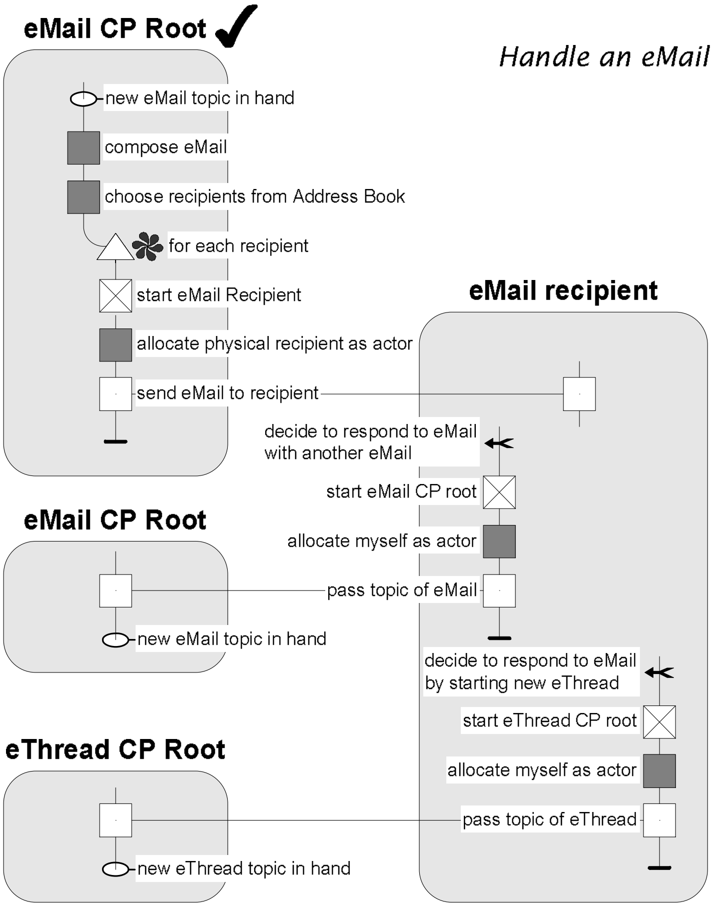
Tutor: Not in the slightest. Because there is no case management, the process architecture will have exactly the same structure as the UOW diagram. The real point that we can draw from this example is that this simple picture represents completely the way that things can blossom – even explode – on email. A conversation can develop in an infinity of ways. One conversation might consist of a single thread of a single message that causes no further email. Others can cause torrents of new threads that all have their origin in that first spontaneous email. One day perhaps all the threads peter out and the conversation as a whole 'ends' at that moment. This picture captures the cascade of process instances that really does happen in the real world. Email is not a serial affair, and it's not workflow.

But why don't you take it one step further and draw some RADs for the three case processes that we're left with?

Pupil: OK. Three case processes. (Figure 7-11).

Figure 7-11 The three email case processes





Tutor: Now we can start to see how a RAD captures all the possible future behaviours of the 'organisation'. Talk me through what you've drawn.

Pupil: Well, a conversation starts spontaneously with a Correspondent who creates a new 'responsibility' for a thread and takes on that responsibility (role instance) themselves. That role instance then creates yet another new responsibility in the form of an instance of a role for an email, and again acts that instance.

Tutor: OK, that's the first two case processes. Then what?

Pupil: In that responsibility (*eMail CP Root*) the correspondent then actually writes the email, chooses a set of recipients from an Address Book, and sends each of them a copy. The interesting thing is that we have identified being the recipient of an email with the responsibility for dealing with it, though ignoring it is one way they can deal with it, as ever! And these (very tiny) responsibilities are created dynamically.

Tutor: Exactly. Then what?

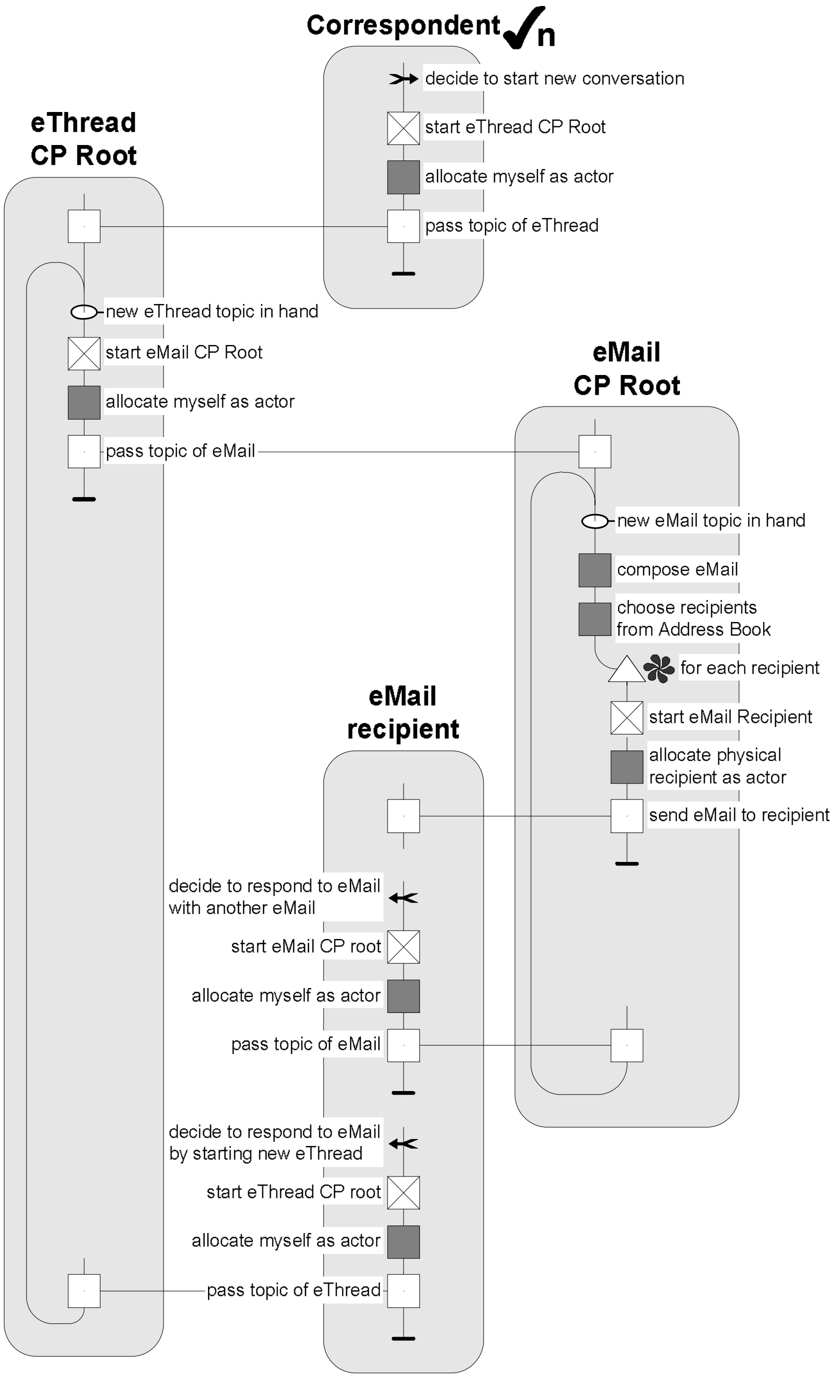
Pupil: The recipient doesn't actually have to do anything – they can ignore the email. But they can also decide to contribute to the thread by sending out a response email – again creating new responsibilities (role instances) for each recipient – or they can decide to start a whole new thread in the conversation.

tutor: Right. In fact of course, the RAD says that they can contribute to the existing thread *and* they can start a new thread ... and indeed they can do those two things as many times as they like – just as in real life: we leave a message in our in-box and two days later decide to send out a reply, and then next day decide to forward it with a different reply to a different group of people, and so on. The conversation can 'blossom' in the hands of recipients in an infinite number of ways. How does the concurrency show itself?

Pupil: I guess it's reflected in the multiplication of instances of the roles for conversations, threads and emails. Over time, these are generated by people sending emails and creating them. Once a conversation starts flowing you could have lots and lots of these at any one moment, all operating independently and deciding how they want to push things on, if at all.

Tutor: Yes. Now, even though our process architecture is 'small' it does highlight concurrency in terms of the separate *processes* that can be instantiated. There is of course nothing to stop me then drawing a single RAD combining all three processes – this is a modelling choice, remember, and not a statement about the world. If we do that, we'll get something like Figure 7-12. It describes that same behaviour as Figure 7-11 but we lose the visibility of the separate processes, and we might regard that as a bad modelling choice as a result. One thing you'll notice is that there is only one role with pre-existing instances: *Correspondent*.

Figure 7-12 – The three email case processes drawn on one RAD



DYNAMISM IN THE WORLD

Tutor: One thing that is currently fixed in the process we have drawn is the number of pre-existing instances of *Correspondent* – in other words we start with – and remain with – a fixed number of correspondents. This means we're actually missing a degree of variability that is present in the world. Now something that we haven't looked at is the Address Book.

Pupil: Well, we've – I've – assumed that it's just there and available to all. Isn't that fair?

Tutor: Yes, we could certainly regard it as some sort of shared resource. But – in the world of email – we can treat it as a designed unit of work: it has a lifetime, it undergoes change during that lifetime, and it needs handling. And if it's a UOW then it has a case process. Indeed, we can associate the Address Book with its case process. Then if someone wants email addresses they can interact with the **Handle the Address Book** process.

Pupil: I like it. Presumably if we wanted to go further we could identify a 'smaller' unit of work called *Account Holder* or some such which is generated by the *Address Book* unit of work.

Tutor: Keep going ...

Pupil: Well, in our first-cut process architecture we know that there will be a **Handle an Account Holder** process and a **Manage the flow of Account Holders** process. We can associate an instance of **Handle an Account Holder** with an account holder. I'm really getting the feeling that the world is full of processes rather than data or objects. Any significant object – in particular any with a lifetime in which things happen – can be associated with its case process instance.

Tutor: I think this demonstrates the power of the *Riva* process architecture. It moves us away from thinking about data and objects and towards processes.

In true BPM style, we start to identify things with their processes. An Address Book, or an Account Holder, becomes a process rather than an object about which we keep data, or an object which processes messages. The object-oriented paradigm converted us all from thinking about data on which computation is performed, to thinking about objects which have associated computation (to process messages) and state. The process-oriented paradigm finally completes the inversion: everything is a process and instances have state which we can treat as data. *Riva* moreover adds business-related concepts and dynamics to basic object-orientation – role and interaction and prop/resource in particular.

ALL THE WORLD'S A THEATRE

From the outset, there has been a theatrical theme running through the book: in particular the notions of roles and their actors. Let's round off the metaphor and see what it tells us about the world and how *Riva* helps us understand it and get our arms round it. We'll leave it to our Tutor and Pupil to explore the idea.

Later, next to the water-cooler

Tutor: We've been talking about activity in a part of the world, the part inhabited by our 'organisation'. We can think of that part of the world as the 'theatre'. This theatre will be a rather twenty-first-century one with some unusual characteristics. Let's start with processes.

Pupil: Well, we've got processes – plays, I guess. Plays are written down in scripts … process models, and …

Tutor: Stop there for a moment. Think about instances …

Pupil: Plays are performed … they have performances … process instances. So when we walk into the theatre we'll find performances of plays going on. Ah – the theatre seems to be a multiplex, because I have lots of plays being performed at the same time! Worse, some performances start new performances! And presumably they have to find a stage to operate on.

Tutor: Concentrate on one performance for a moment.

Pupil: Well, a play consists of a number of roles. Each role … ah, each role instance … is acted by an actor – who was cast in that part somewhere along the line. Things start getting a bit weird around here, I suspect, because during the performance some actors might act several roles, rushing around the stage from one to another. Worse, I suppose they might be acting roles in more than one performance, so they'll have to run from stage to stage, changing costume as they go. In the worst case they'll be acting several roles in several performances.

Tutor: When we're talking about performances remember we need to talk about role instances.

Pupil: Hmm. That's a bit weird too: how many Hamlets can you have in one performance of *Hamlet*? Only one, I guess; but we have seen plenty of processes in which some roles are instantiated many times – mercifully Shakespeare was content with a single instance of the Prince of Denmark on stage at any one moment. But it appears that in some plays new role instances are created while the play is in progress. And an actor has to be found to play the role instance. In extreme cases, the actors might be writing bits of play, inventing new roles, instantiating them, and then casting actors as they go along. This is truly contemporary!

Tutor: Absolutely. You mention casting …

Pupil: Yeeees. We said, didn't we, that the allocation of actors to role instances is just more process … so casting happens on stage, and possibly during the performance!

Tutor: Any thoughts about props?

Pupil: Well … the props are the resources actors need to play the role instance. It might be a book or a newspaper in a real play, and an information system or a software application in a real business process. The costume sounds a bit like the mind-set they need for the role.

Tutor: So we have a number of stages, each with a performance of a play going on. Performances are starting up and stopping all over the place. On each stage, role instances are being played by actors, who are possibly rushing from play to play and from role instance to role instance, putting costumes on and taking costumes off, and picking up and putting down props as appropriate. So far so good. What about role actions and interactions?

Pupil: I'm not sure I want to watch one of these plays. Sometimes a role instance is doing an action: so the actor is giving a soliloquy; sometimes it's interacting with another role instance, or even with several others at the same time. There may be several soliloquies and several conversations all going on at the same time. And occasionally proceedings will get held up for want of an actor … who might be acting another role instance on another stage. It's madness.

Tutor: Hold that thought. You've only dealt with role interactions in a single performance, but we have seen how process instances interact. And in the theatre?

Pupil: Oh dear. Some plays are connected. A performance of one play has to interact with a performance of another. That means that role instances in the two performances have to interact – there must be communications systems between the stages – telephones or email or something. Or perhaps the actors rush to and from each other's stage, or perhaps they meet in the corridor? My head is starting to spin.

Tutor: We've just described a chaotic-sounding theatre – but it's no more complex than the average organisation. The problem is of course that when most people look at an organisation and are faced with the task of capturing what goes on, they find it so overwhelming they resort to the simplest thing they can think of: the sequential process, with perhaps some branching and some swim-lanes. But these come nowhere near giving us insight into what is really happening in the Theatre of the Third Wave. That's why I made such a fuss about instantiation and concurrency being the key to really understanding organisations at the start of the book. That's why they are so central to *Riva*.

KEY POINTS

When a process architecture ‘runs’, processes are instantiated.

Process instances operate concurrently.

The dynamism in the world *is* the flux of concurrent process instances.

The process architecture expresses the potential concurrent behaviour of that part of the world, in the same way that a process model expresses the potential concurrent behaviour within one process.

We can identify business entities (essential and designed) with their case processes.

The world becomes a constantly changing network of process instances and a cascade of responsibilities.