12 Processes and information systems

INTRODUCTION

We know that, for every unit of work, we are likely to find, in some measure, three processes: the case, case management, and case strategy processes. Each of these can be expected to be both a provider and a user of information, and hence a potential user of computer-based information systems (IS).

The case process

IS support for the case process will be about the capture, storage, presentation and processing of information about individual cases. When I call my water company, my name and account number (and perhaps the number of the telephone I am calling from) will allow the information systems to bring up my details and a summary of recent contacts on a screen in front of the person taking my call. The time frame of IS support for the 'lifetime' of a customer contact will be short- to medium-term: some contacts will be cleared during the call, some will need action that will continue for perhaps weeks as visits to the customer's premises are arranged, take place and get closed off. The time frame of IS support for the 'lifetime' of a customer will be medium-term to long-term, since an instance of the case process for an individual customer will last as long as the customer is a customer.

The case management process

IS support for the case management process is about the flow of cases at any one time – typically rates and levels – and about exceptions that are raised by the case process instances. As Call Centre Manager, I expect to see on a screen in front of me who is doing what, what the waiting queue of callers looks like, and how it is building or falling. The actual allocation of calls to call centre staff is being dealt with automatically by my smart telephony system, leaving me to manage the staffing levels and deal with exception conditions. The time frame for such IS support will be about minute-to-minute management of the centre. In other cases the time-frame could be years.

The case strategy process

IS support for the case strategy process is about the longer-term capture, storage, analysis and presentation of past cases. We will think in terms of data warehousing and data marts, the ability to slice and dice the case data, computer simulation of the business and its flows, exploration of trends, experimentation with different future scenarios, and modelling of new structures and processes. The time frame for such IS support will be medium- and possibly long-term depending on the time horizon of the business.

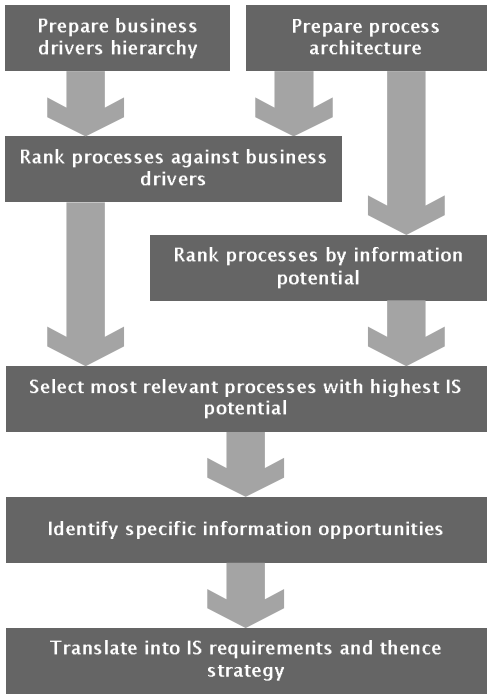
RIVA AND TRADITIONAL INFORMATION SYSTEM DEVELOPMENT

From *Riva* process architecture to IS strategy

Utility company *R* sought a business-driven IS strategy, one driven by the needs of the business rather than the capabilities of technology. For any business, the issues that drive management action vary in time as priorities change. Those business drivers will have different time horizons and the role of IS is to support them accordingly. But the translation from business drivers to information technology strategy is a big one: we needed to break it into a number of smaller steps that could be taken individually with senior managers.

The *Riva* approach in this area is summarised in Figure 12-1. The first step was to brainstorm those business drivers: ignoring information systems, what's driving the business and changes in the business today and in the foreseeable future? Following Gilb's *System Attribute Specification* (SAS) method (*Design by Objectives*, Tom Gilb, North-Holland, 1987), we developed a hierarchy of drivers with lower levels making the 'woolly' upper levels more specific and finally, where possible, measurable. At the highest level in our case study the drivers included safety, expanding the customer base, and satisfying the regulator. Each of these was then decomposed in turn to bring the area into finer focus. A driver such as 'We must expand the customer base' might be decomposed into drivers such as 'We must have more flexible pricing,' 'We must package our offerings better,' 'We must improve the quality of our *A* service,' and 'We must understand better who is buying what.' The driver 'We must have more flexible pricing' became 'We must know which of our assets are being used and how much' and 'We must be able to construct more flexible contracts.' This sort of analysis might already have been done by the organisation and we can pick it up directly; otherwise a traditional facilitated workshop will yield the information.

Figure 12-1 – Steps from the business drivers to an IS strategy



While this work was progressing, a simple *Riva* process architecture was constructed for the organisation.

We worked with senior managers responsible for large chunks of the company. They examined the list of processes in the process architecture to see which potentially had the most to gain from having accessible information. Simple round-the-table scoring allowed us to rank the processes according to their 'information potential' – clearly, processes with the most to gain from strong IS provision would be of greatest interest to us in developing the IS strategy. They examined the processes to assess how far each could help the organisation meet the demands of each of the 'leaf-node' drivers developed with SAS. A traditional matrix of drivers against processes was used to score each process according to its contribution to each driver and thence to derive a further ranking. By simply multiplying the process–driver matrix by the information potential rankings vector, we were able to quickly prepare a shortlist of the processes that were most important to the business *and* had most to gain from strong IS provision. In a final bout of workshops we drew out improvements in information quality or accessibility for each of those processes, knowing now that we were addressing the big opportunities. These requirements could then be translated into requirements on the IS provision.

At this stage we were of course simply characterising the IS provision in terms of what it would *achieve* as opposed to what it would look like: 'Timely analysis of the usage that each asset component had and of its costs, available at the desk of group *D*,' 'Timely costing of

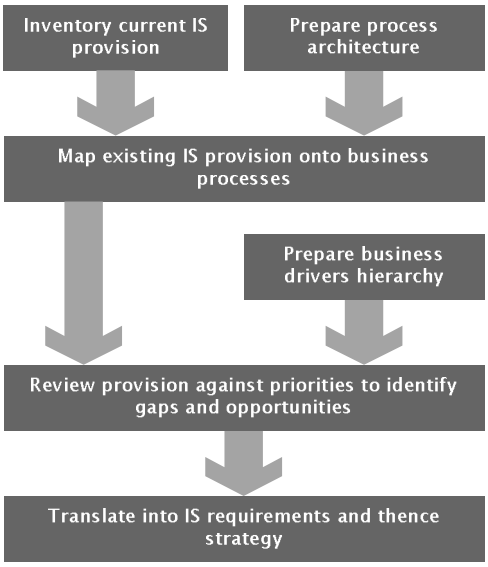
alternatives packages, available at the desk of group *E*,' and the like. The remaining step was to decide the technologies that could be used: data warehousing, client-server architectures, web-based information gathering etc. This was a purely technical judgement to be made by those familiar with the technologies concerned, but the senior managers now had a clear vision of why those technologies were relevant to their business – one of the hardest steps to make – and we made the bridge for them using the process architecture. If, say, a particular IS architecture was proposed to satisfy the information needs of the asset case strategy process, managers could trace the decision back to the business drivers concerned with their need for pricing flexibility, and the proposal became meaningful.

From *Riva* process architecture to IS gap analysis

Coming at the question from the other direction, the IS group at utility company *W* wanted to develop an IS strategy that was aligned to the business in a way that was accessible by senior management. The strategy would need to map a path from the existing provision to the new. Again the process architecture provided the bridge.

Figure 12-2 summarises the approach.

Figure 12-2 – Steps from the current IS provision to an IS strategy

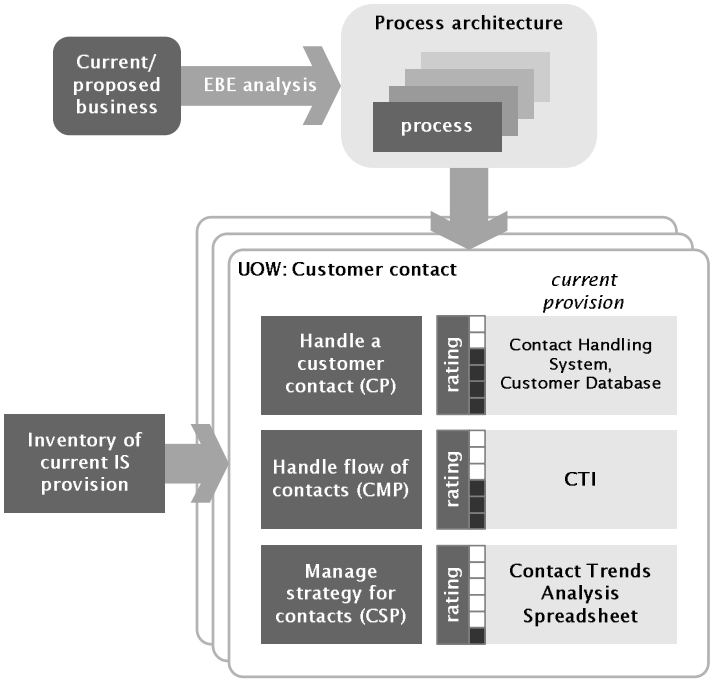


For an IS group the preparation of a process architecture can be kick-started by using the corporate data model as a checklist of UOWs. (There will be many things that an organisation collects data about that do not represent the 'work items' that we look for in UOWs, but it would be strange for there to be a UOW about which no data is kept.) While this work was going on at *W*, a parallel team was preparing an inventory of existing systems and their users. Those systems were then mapped onto the process architecture – knowing the users of the system allowed us to determine what processes they were involved in when they made use of the system.

The result was a gap analysis. Certain processes were well supported by the current IS provision. But an analysis of the business drivers of *W* pointed to process areas of the business that would become important before long and which would need increased support from IS. It was now possible to prepare a 'shopping list' of systems and architectural changes necessary to make them possible. The business case made to senior management could now be aligned to the business's processes and drivers, offering the opportunity of a much richer form of justification than a cost-benefit comparison – IS could be painted much more easily as an *enabler* for the business, a case that has traditionally been difficult for any IS group.

Figure 12-3 summarises the approach to using a *Riva* process architecture in preparing an IS gap analysis.

Figure 12-3 – Preparing a gap analysis



Preparing an appropriate process architecture

Using a process architecture in these two ways places few new requirements on the way we build the process architecture in the first place. The approach in Chapter 6 requires little adjustment.

The one decision we must make is whether to include in the process architecture the processes for designed as well as essential UOWs. An IS strategy that is aligned to the business strategy will, minimally, provide an adequate level of support to each of the processes for each of the essential UOWs. But an adequate IS strategy will also support the way the organisation

has chosen to do its business – in other words the designed UOWs, so we can expect that these will appear in the analysis.

Using a *Riva* process model in traditional IS development

Because *Riva* gets to the heart of what is going on in an organisation, the process models it produces are ideal starting points for the development of traditional information systems. Each such development should start with some sort of analysis of the business and, thereby, where and how an information system can help. But traditional approaches have not looked seriously at the *process*, something that we now have the machinery to do.

Traditional methods (such as SSADM) concentrate on the information needs of the individual and the way that information finds its way between individuals and groups through 'plumbing' between them and some database. To do that they use techniques such as *entity relationship modelling*, *data flow modelling*, and *entity life history modelling*. The information collected during a *Riva* analysis connects tidily into these:

- When we look at the units of work in order to get at the case structure of the organisation, and model their relationships as a step to understanding the likely relationships between processes, we have a sound starting point for preparing the traditional entity relationship, or data, model. The latter will add more entities and more (static) relationships.
- By examining the actions, interactions and decisions within roles, we identify the information needs of those operating the process.
- When we examine the information inputs and outputs of actions and the flow of grams in interactions, we build the basis for a data flow model of the process.
- When we follow the history of a UOW entity (typically through the case process that deals with it), we can map out its state history and hence have the basis for its entity life history.
    1. The flow of an entity through the actions and interactions will be shown on the RAD; the actions and interactions through which it flows will change its state.
    2. The state changes caused by the process will be shown on the ELH. We can check where in the process those state changes occur, and ensure that the ELH and the RAD tell a consistent story.
    3. The attributes of the entity, as listed in the ER model, should cover its state. We can check that the attributes are sufficient to describe the entity's different states.

In short, having done its job in helping us understand, analyse, or re-design a process, a RAD also provides us with a sound starting point for the data-oriented development of an information system to support that process.

Information systems, and ERP systems perhaps even more so, are notorious for being wet concrete before they are built and set concrete once they are built. Worse, they set in concrete the very processes they support, making them hard if not impossible to change once the concrete of software has set. All too easily, crazy processes become ossified, a phenomenon known as 'paving cow paths': things have always wandered this way in the past and now,

thanks to the information system, they are doomed to wander this way in future. In the very worst case, it is also a matter of 'paving cow pats'.

The message here is of course that we should take an abstract perspective in our RAD in order to ensure that our new process support system does not simply mimic, say, the existing paper-based system, but instead takes advantage of the potential of a fully electronic environment: document management, document imaging, smart telephony, web services, PDAs, network communications, group support products, etc. Our process models will therefore be strongly abstract.

RIVA AND OBJECT-ORIENTED SYSTEM DEVELOPMENT

One of the central notions in *Riva*, whose roots go back to the early 1980s to the formally-defined object-oriented language SPML, is that of the *type* (or *class*) and the *instance*. We saw how everything on a RAD is actually a type, and that we can 'run' a RAD by looking at how, when and what instances of types of roles, actions and interactions are created as the process runs. *Riva*'s object-orientation makes it a natural business analysis precursor to the use of object-oriented software development approaches, and in particular the object-oriented notations in UML (Unified Modeling Language). *Riva*'s two notations are not a replacement for UML any more than UML's fifteen or so are for *Riva*'s. Their shared object-oriented roots make them good bed-fellows but some apparent similarities must be handled with care. Unfortunately, the attempt to crowbar UML into the world of business process modelling has been made and we must hand over to the Tutor to rebuff it …

During a break, next to the water-cooler

Pupil: I've had a lot of dealings with software developers recently and they have really taken to UML for software specification and design. I know that a number of people also use UML in business process modelling. It sounds like a sensible idea – so why try and persuade them to use a different approach for the business process side from the software design side?

Tutor: Well, you've given me the reason without realising it: UML and all its diagrams come from the world of software engineering. Let me tell you a story. You might have come across the IDEF0 notation. Its precursor (SADT) was invented in the 1970s at TRW for the analysis and design of computer systems using 70s technology and for which it was well suited. It has been terribly abused by being transplanted from the world of system design to the world of business process modelling, a purpose it was never intended for when it was invented. Like UML, IDEF0 has no business- or organisation-oriented semantics. Worse, it was based on hierarchical decomposition, which made great sense in the 1970s when software development had that paradigm, but it makes no sense in the modelling of organic business processes.

We shouldn't be surprised that using software-oriented methods to model anything other than software systems proves hard work. It's possible of course, but then we could model business processes with Turing machines … and much good may it do us. My message is that *Riva*'s diagrams are there for a

purpose, and UML's for other purposes. Attempts have been made – in particular by tools vendors eyeing up the business process management marketplace – to adapt UML for business modelling despite its lack of business-oriented semantics.

Pupil: You're showing your age ... But let's take UML's use case diagrams – they're very popular and they've been proposed as a starting point for building a model of a business – I understand Jacobson *et al*'s *The Object Advantage* (Addison-Wesley, 1994) was the influential first attempt. What's wrong with them?

Tutor: Well, let's check on definitions first. A use case was originally defined in the world of computer systems as a sequence of transactions designed to produce value to a user of the system. If we translate that out of the world of computer systems to the world of business we get something like 'a sequence of transactions designed to produce value to someone outside the business.' Fine, except that there's little agreement after more than a decade on what use cases are and how to work with them – what does that tell us? The real meat in use cases comes with the *use case description* which is typically a page or two of text that describes the use case. So what's happened? We've ended up writing serial text to describe a definitely non-serial thing. All sorts of contrivances are then necessary to handle exceptions, alternative routes, etc. These are just attempts to render a complex network structure into serial form: text! The question of use case 'extension' for example confuses the issue, and, it appears, practitioners. At best, use case practitioners apologetically suggest that use cases are just an exploratory tool. Not much good for real understanding then? As a way of finding out what processes an organisation has, what its dynamic structure is, use cases prove very weak.

Pupil: OK, accepting that a set of use cases would only 'explore' the business activity as you put it, what about the other notations that are then used to support the use cases: the state diagrams, activity diagrams, interaction diagrams, and so on?

Tutor: I think you've answered your own question again: we are now going to collect a mass of detail into a number of quite separate and very *technical* viewpoints: it's down to the reader to try and integrate all these different viewpoints into a cogent picture of what actually happens in business terms in the organisation. So the structure of the resulting 'model' isn't business-orientated: it's concept-orientated. And the concepts – *interface objects, control objects, entity objects* – are not business-oriented concepts, they are technical concepts.

People who have used UML diagram notations for business process modelling often remark on the fact that several diagrams with different notations are needed to capture a single business process. As a result, (a) it's hard for an analyst – let alone someone in the business – to easily see from the multi-concept, multi-viewpoint set of diagrams what's really going on out there in

the organisation, and (b) the many and detailed cross-relationships that arise between the diagrams have to be maintained, and this makes for problems with change as well as understanding.

Pupil: So you maintain that *Riva*'s concepts such as the unit of work, the **Handle a** process, the **Manage the flow of** process, creating a responsibility, responsibility-roles, organisation-roles, actions, interactions, decisions … that these are the proper metaphor for business process modelling?

Tutor: Of course. They have analogues in the real world. Real-world people recognise them. Moreover, all of them are underpinned with an object-oriented theory with formal semantics, but we don't expose the poor business folks to that. In this book – for analysts – we have used the object-oriented concepts and even the word 'instantiation' but, as I stressed at the outset, we don't need to trouble the business folks with them.

The problem is that with a UML toolset on our PC, it's impossible not to slide into ways of thinking that are perfectly valid when thinking about software design, but that make no sense when thinking about the real world. A UML business modeller would tell us that in a restaurant we have an *Order* object which must have an associated operation *Which dish* which is stimulated by, for instance, the *Order Handler* interface object – I have read this somewhere. This would feel perfectly natural if we were discussing a software design, but 'invoking the operation *Which dish* from an *Order* object' has absolutely no analogue in the real world. It means nothing in the real-world. To claim that it models something that happens in the real world is nonsense. It might be a sensible description of some class interactions in a software system, but it has no meaning for someone in the restaurant business.

Pupil: Isn't this just a question of terminology?

Tutor: NO! It's a question of having the right metaphor, the right idiom. The metaphor of 'objects responding to received messages' is fine where we are building a software system using object-oriented tools that share that metaphor. But that metaphor has no analogue in the real world: chefs do not send messages to orders to find out what dishes are required.

Pupil: Point taken.

Tutor: Don't despair. By all means use UML where it was intended to be used and is entirely appropriate: in the analysis and design of software. So, let's move on and look at how we can transition from a sound *business* model in the form of a *Riva* process architecture and a set of RADs to a *technical* description of a system in the various notations in UML.

From Riva *Process Architecture Diagram to UML use case diagram*

We know that a *Riva* process architecture is precisely based on a clear characterisation of the organisation's business in terms of its EBEs. A clearly defined analysis path leads us to identify all the organisation's processes and produce a second-cut process architecture. Case

processes in particular can be identified with business use cases: after all, each case process delivers some value to a 'user'. A case management process can also be identified with a 'large' business use case, one that we will be able to decompose when we examine the different threads within it at RAD level.

A traditional problem with use cases is knowing at what 'level' we should be thinking. Are all the following system use cases: 'I go to hospital,' 'I visit the Haematology Department,' 'I have my blood pressure taken,' 'The nurse records my blood pressure'? This problem might not have been solved for UML system use cases, but by using the notion of 'fainter and fainter' UOWs that we introduced in Chapter 6 we are now able to get our heads round the problem, rather than simply moving it from the system arena to the business arena by inventing 'business use cases'. Our *Riva* analysis will readily identify, in particular, *Patient Hospital Visit*, *Haematology Visit*, *Blood Pressure Reading* as units of work, and clearly identify their dynamic ('generates') relationships. Because the process architecture was produced without consideration of technology or systems, we can be confident that the corresponding business use cases are devoid of system design – a good thing.

From Riva *UOW Diagram to UML object model*

UML class diagrams must not be confused with *Riva* UOW diagrams. The former show all relationships and are avowedly static views of the world. The latter show only dynamic relationships, as they are a step towards the all-important *process* architecture. We can certainly expect that essential and designed units of work will appear on a UML class diagram for a system supporting the organisation characterised by those units of work. But neither is a replacement for the other. That said, a *Riva* UOW diagram is a perfect starting point for a UML class diagram, being based on a strong analysis as described in Chapter 6. To the dynamic relationships in the UOW diagram we must add the static relationships of interest in system development.

From Riva *Role Activity Diagram to UML activity diagrams*

At first glance, UML activity diagrams with swim-lanes suggest *Riva* Role Activity Diagrams. But they lack a number of important concepts. Firstly, a swim-lane is a static thing, the responsibilities of a certain class, rather than a responsibility that can itself be instantiated or even passed around – a swimlane cannot generate the network of collaboration that characterises organisational activity. Secondly, where activity flow crosses from one swim-lane to another, it is just that: a movement of the locus of activity; the implied interaction is not recognised, the contribution made by each end is not recognised, and the object-oriented integrity of the interaction is lost. Roles don't just partition activity, the way that swim-lanes do: they also express the interactions – the strands of mozzarella – that are generated by the partitioning. The activity diagram ignores interactions as joint activities that synchronise state, which is what happens in reality. A UML activity diagram taken together with an interaction diagram, and sequence diagram could cover the ground of a RAD, but without the clarity or precision. For instance, UML interaction diagrams capture some of the dynamics of a 'system' in terms of messages passed between objects, though typically they are potentially incomplete since 'returns' are not always drawn, for instance, being regarded as 'clutter'.

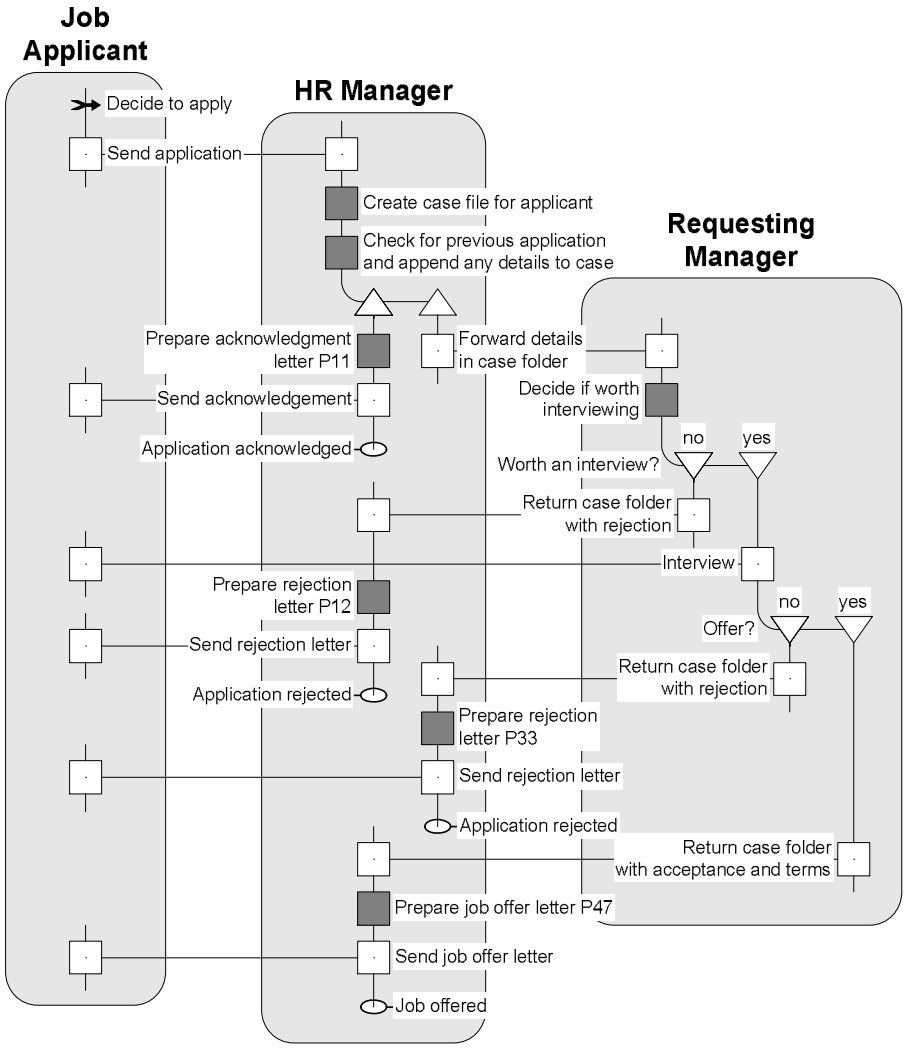
Preparing the process model for object-oriented system development

If we are starting with a clean sheet of paper, our process model should be of the abstract variety so that subsequent work is not tainted with design decisions that should be left open at this early stage.

RIVA AND WORKFLOW SYSTEM DEVELOPMENT

A *Riva* process architecture, complete with processes for designed units of work, exposes the workflows to be supported by a workflow management system. Such systems are typically designed to support case processes in which a thread of activity leads from case inception to one of a number of possible outcomes. A degree of branching of the flow is generally supported and some parallel working, though this is typically less general than we would like. Take the RAD in Figure 12-4 as an example. It shows a simple, single-threaded case process for handling a job application. It is the sort of RAD that might result from work to analyse and design a process that we want to support through workflow technology. We have taken it more or less to the point where we can start to decide which areas of the process can be supported with which sort of technology.

Figure 12-4 A simple workflow process



Every applicant gets an acknowledgement in the form of a standard letter (a P11), some are turned down on the basis of the application form and receive a P12 letter; others get an interview but are turned down with a P33 letter; and successful applicants get a job offer in a P47 letter. The *HR Manager* handles all interactions with the *Job Applicant* except for the interview which is done by the *Requesting Manager*. Let's examine how workflow technology would naturally support this sort of process.

When the *HR Manager* receives an application from the *Job Applicant*, they put together a 'case file' for this application. In a manual system, that file could be just a labelled manila envelope which is then passed by hand from person to person. In a 'pure' IT system it could be a new record on a central database which can be accessed by anyone involved in handling

the case. In a workflow management system it would be an electronic case folder which the system conveys with the case on its travels round the workflow.

Before processing the application further, the *HR Manager* checks whether the *Job Applicant* has ever applied for a job with the company before, and, if they have, attaches the details of the previous application to the case folder. It might well be that there is an existing database which holds such details and which therefore the *HR Manager* needs to interrogate through an existing legacy system. We can therefore expect them to use some standard forms to get at that database, and to transfer the results into the e-folder for the case.

The e-folder can now be forwarded to the *Requesting Manager* for further work. That interaction is a 'hard' interaction and could be simply supported within the workflow management part of the system.

Subsequently the *HR Manager* will interact with the *Job Applicant* in various ways: acknowledging the application, turning down the application, getting in for interview, turning down after interview, and making a job offer. Each of these involves preparing a standard letter. Here is an opportunity for traditional personal productivity tools: in particular a word processor, perhaps used in conjunction with the e-folder to get applicant-related details automatically into the letter. In fact we can imagine an implementation of this system which does not require the intervention of the actual *HR Manager* at all: the system could prepare and despatch the letters automatically on their behalf.

What have we seen with this simple example?

- Some actions require the individual to have access to the right information at the right time and perhaps to update that information. Such information needs are traditionally supported by plumbing people into corporate or function-related databases – the domain of traditional database products.
- Other actions require the individual to prepare material, perhaps drawing information from a variety of sources. Here we might integrate databases with personal productivity tools such as word processors and spreadsheets.
- Some interactions and workflows are 'hard', that is they are pre-defined and straightforward in nature, involving the transfer of materials. They are precisely the sort of thing that workflow management products are designed to support.
- Other interactions are 'soft'; that is, they cannot be pre-defined in detail but can still be mediated by workgroup computing products.
- Other interactions will still remain the province of the face-to-face meeting and the phone call.

Preparing a RAD for workflow

Since a RAD is a natural way to capture a workflow, little extra needs to be done when preparing a RAD with a workflow implementation in mind. The only significant issue is the need to be aware of any limitations the workflow management system might impose on the complexity possible in the workflow itself.

SUMMARY

KEY POINTS

A process architecture provides a sound basis for ensuring that an IS strategy is aligned to the business, and for gap analysis.

An abstract RAD provides a solid starting point for traditional data-oriented system specification and design.

The object-oriented nature of a RAD makes it a suitable precursor to a detailed implementation analysis using notations within UML.

Provided we take care with process complexity, a RAD is an ideal starting point for direct implementation of a workflow on a workflow management system.