

Business Process Architecture and Business Transformation

Chris Lawrence, Old Mutual South Africa

INTRODUCTION

Logical process architecture and change methodology are two sides of the same coin. Organizations who know this and know what to do with it will beat the competition. This is why BPM is so crucial. But architecture and methodology sound academic and expensive, so people in organizations are often reluctant to think about them long enough to see the coin.

This paper takes, as its point of departure, a proven business process metamodel which has been elaborated from a number of perspectives elsewhere.^{1, 2, 3, 4, 5} The metamodel sees process as an architectural entity derived from data and rules rather than an empirical fact or democratic artifact. It can support selection, design and configuration of IT systems, and therefore also the evaluation of IT systems in the service of business processes.

Systems shape work and how people see and think about work. Internal and external support teams employ cognitive models shaped by the systems their careers and incomes depend on. Systems and system ownership have so shaped organizational design and politics that even 'logical' architecture and delivery methodologies often only have meaning in terms of systems which happen to be in place.

From a process perspective business IT history was back to front. Recordkeeping first, with process (workflow) a later add-on. This was understandable as history but less forgivable in its unexamined, protectionist result: back-to-front legacy thinking, translating the profound tautology 'process-centric = customer-centric' into a complex overhead of duplication and falsehood. Meanwhile Lean and Six Sigma stay aloof and go for soft targets. But this won't do for 'BPM', and it won't beat the competition. A new generation of business analysts must not take everything vendors and consultants throw at them, but must understand a business as a set of interacting processes; understand a process as an architectural entity derived from data and rules; and only then evaluate the solution space.

¹ Lawrence, C. P. (2005a). *Make work make sense: An introduction to business process architecture*. Cape Town, South Africa: Future Managers (Pty) Ltd. (<http://www.makeworkmakesense.com>)

² Lawrence, C. P. (2005b). *Integrated function and workflow*. In Layna Fischer (Ed.), *Workflow handbook 2005*. Lighthouse Point, Florida: Future Strategies Inc, in association with the Workflow Management Coalition.

³ Lawrence, C. P. (2007a). *Business process architecture and the Workflow Reference Model*. In Layna Fischer (Ed.), *BPM & Workflow handbook 2007*. Lighthouse Point, Florida: Future Strategies Inc, in association with the Workflow Management Coalition.

⁴ Lawrence, C. P. (2007b). *Architecture-driven business transformation*. In Pallab Saha (Ed.), *Handbook of enterprise systems architecture in practice*. Hershey, Pennsylvania: Idea Group Inc.

⁵ Lawrence, C. P. (2007c). *Business process integration in a knowledge-intensive service industry*. In Wing Lam & Venky Shankararaman (Ed.), *Enterprise architecture and integration: Methods, implementation, and technologies*. Hershey, Pennsylvania: Idea Group Inc.

AN ELEPHANT IN CONTEXT

Business process architecture, or rather the need for it, is like the elephant in the room. We know it's there but we don't know what to say about it. Why is this? One reason is the IT investment mantra: *reuse, then buy, then build*. From a corporate IT customer's perspective anything that smacks of *build*—design included—is immediately on the back foot. Product selection meanwhile enjoys the happy end of the continuum, particularly if product promises to make legacy reusable.

There are good reasons for *reuse, buy, build*. But it can discourage analytical thinking about business architecture and critical thinking about the alignment between technology and business. If *reuse* is top prize, then arguments that what you have is OK will be rewarded. If *buy* gets silver, then discovering that what is for sale is OK will also be popular.

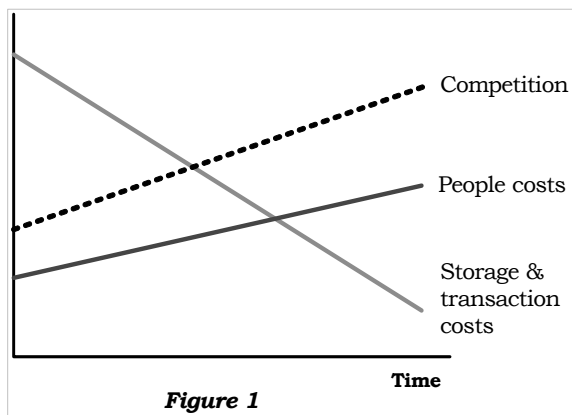
This is not to imply that legacy and proprietary technologies are always bad. That would be absurd. The intention is in any case not to critique legacy or proprietary technologies *per se*, but expose assumptions they might encourage or reinforce.

A word on context. Much of the discussion will involve generalizations, but a generalization is not a universal truth. The context is not necessarily all business processes. It is that large subset describable as rule-based administration: financial services, local and central government, purchase ordering, contracting, HR management etc etc. It is a huge domain—the potential market for BPM systems in fact.

A BRIEF HISTORY

Economic trends affecting business IT over time can be summed up as in *Figure 1*.

It is no surprise that systems first focused on record-keeping: products, orders, payments, contracts, applications, agents, customers; and then transactions—event records changing the status of these 'master' records. A business process was something people did around (and in interaction with) systems, occasionally fast-tracked by bursts of automation inside or in between the systems themselves.



Then workflow made at least some parts of some offices paperless. A 'workflow' is kind of the same thing as a business process and kind of not. Workflow starts when it starts, which might be after the 'end-to-end process' begins. It controls sequences of interaction and dis-

tributes work between and within teams. But important subsets of the business rules coded as either data or program logic inside administration systems have process implications. These rules must be duplicated in or somehow accessible to workflow if workflow is to be coherent and comprehensive. Hence the debate about what is a product rule, what is a process rule, what is a compliance rule;

and where does process logic belong—as if this was a question about the world like ‘where do fish belong?’.

IT has archaeological strata, but IT systems are not natural kinds. Administration systems are rarely process-architected to any extent—the concept of ‘business process’ is rarely implemented inside them. Even their ‘long-running transactions’ reflect few of the ifs, buts and maybes of a real end-to-end process. Workflow systems on the other hand typically come from vendors who need their products to be generic like email and spreadsheets. Workflow fits outside or alongside administration systems because it primarily supports, controls, organizes and empowers the human users of those systems. Ergo the business process is something people do around administration systems.

But this is a false paradigm, created by IT history. Before exposing it though we must bring our history more up to date.

PROCESSES AND SERVICES

Service-oriented architecture (SOA) is the 21st-Century panacea, the redemption of legacy and proprietary technology. What we have is OK or can be made OK by componentizing it. What we can buy is OK as long as its services are exposed. ‘Services’ is business-speak. We have all we need—or do we?

A business needs services, needs to offer services to customers, suppliers and partners. How does a business process relate to a service? Is a process a service or a type of service? Is a process a set of services? Is a service a set of processes? Do we need processes if we have services? We need workflow, because we need people’s work to be organized, distributed and controlled: is workflow a service, or a set of services?

We have a metamuddle. We want a number of things to be true, and true together.

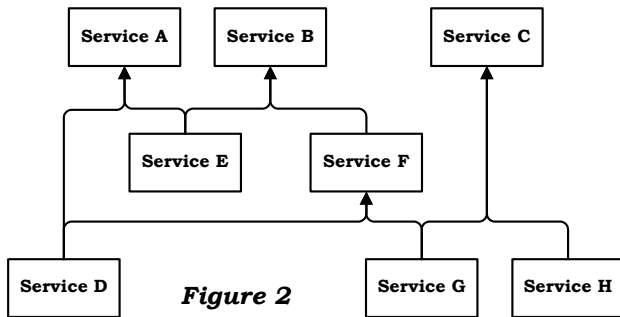


Figure 2

We want componentized services, and we want them modular and hierarchical. In *Figure 2* service B is made from services E and F, while F is made from D and G.

We want end-to-end business processes; and relationships between processes and services. In *Figure 3* process BP1 uses services A, F and H, and BP2 uses B and H.

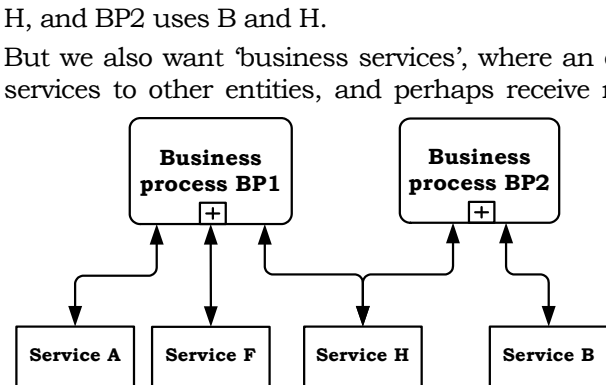


Figure 3

Figure 4.

How do business services BS1-BS5 in *Figure 4* relate to processes BP1 and BP2 in *Figure 3*? Are they the same thing? Or are BS1-BS5 in *Figure 4* the same thing as services A, B, C etc in *Figure 2*?

The vision seems to be of organizational entities controlling portfolios of services and offering them to internal and external customers. They are modular and composable—sets of services assembled into composite services as in *Figure 2*. But does this work all the way down, so the service portfolio of division D1 in *Figure 4* could include (say) services B, E and H in *Figure 2*, and division D2's could include services A, C, D, F and G?

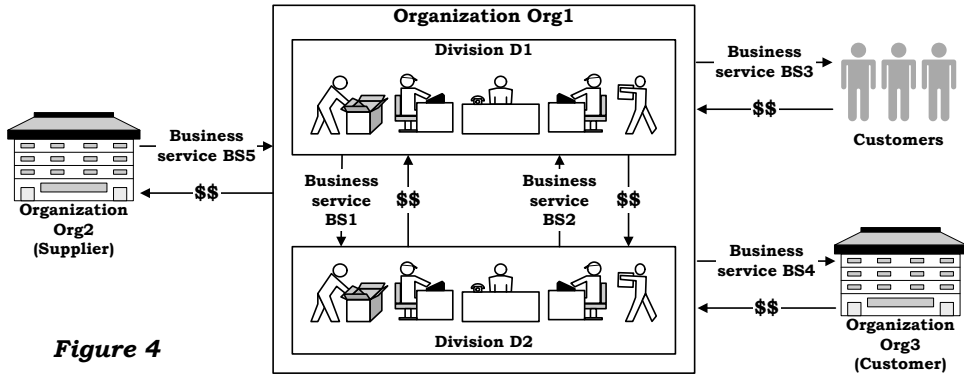


Figure 4

Questions like these need choices rather than answers. At a business-architectural level an organization must choose its 'primitives', its 'axioms'. Otherwise those accountable for technology architecture will not have clear guidelines to follow or share.

In particular, if the definitions of, and relationships between, *service* and *business process* are left undefined the result could be a conceptual free-for-all eating up

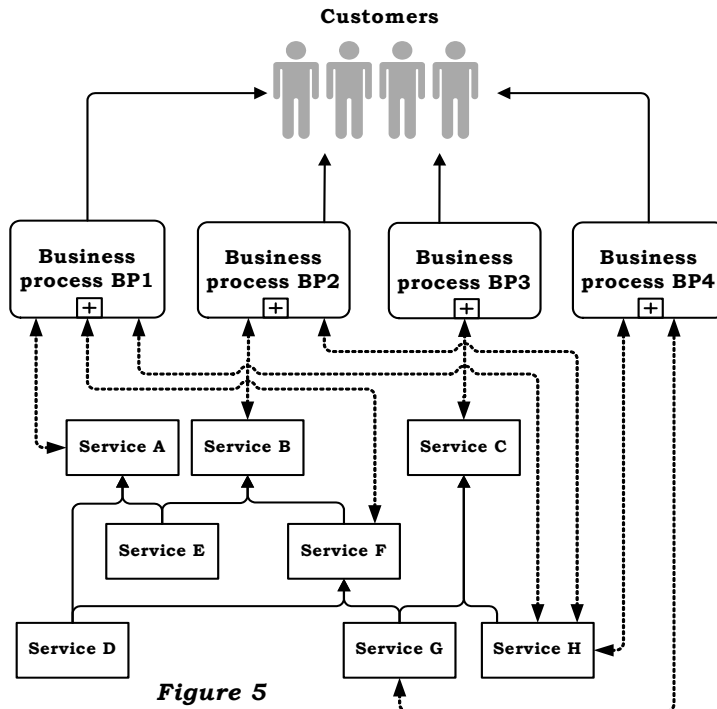


Figure 5

both real cost and opportunity cost. This kind of thinking is not trivial. It is after

all what underpins debate about how BPM and SOA can or should work together to deliver business value.

Three models will now be contrasted. It is not that one is right and the others wrong. Nor are they the only models. But they show the kind of choice organizations need to make at logical architectural level. They may not all hold true for the same place at the same time. Remember also the context: rule-based administration in financial services, government, ordering, contracting etc.

Figure 5 is a model where the reference point is the business process. Processes BP1-BP4 call on one or more componentized services A-H, some of which will be composite. Processes BP1-BP4 and services A-H are physically and explicitly implemented: physical solution constructs control and coordinate services A-H within processes BP1-BP4. Business services do not feature, unless they are the same as business processes.

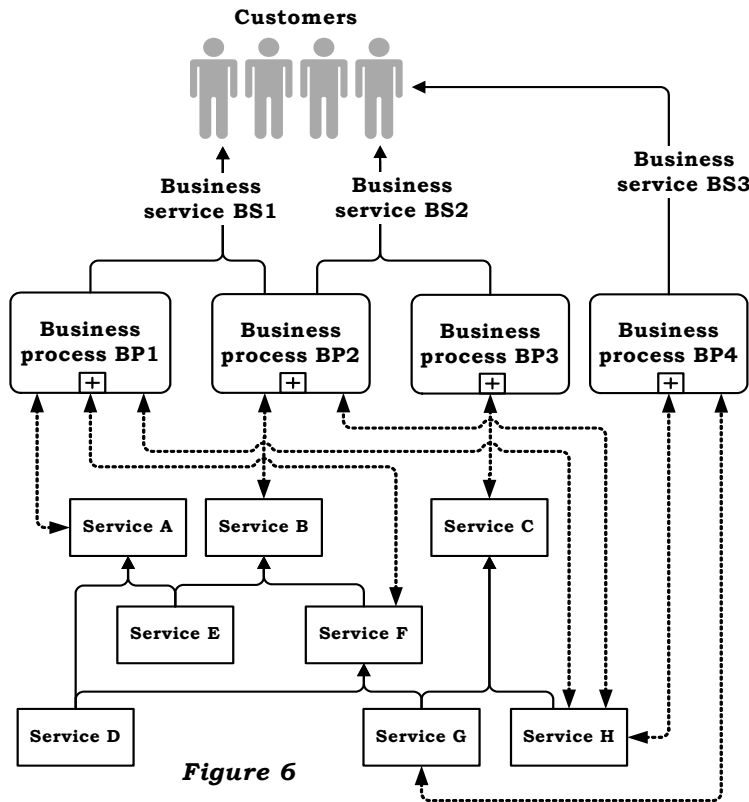


Figure 6

Figure 6 is a variant of Figure 5 where business services do feature, but as 'bundles' of one or more business processes. Business services BS1-BS3 may or may not be physically and explicitly implemented.

Figure 7 is a service-based model, where componentized services A-H are assembled into business services BS1-BS3. Business processes do not feature unless they are the same thing as business services. Componentized services are physically and explicitly implemented, while business services may or may not be.

If business service BS1 is physically and explicitly implemented, solution constructs control and coordinate services A, B, F and H as BS1. If BS1 is not physically and explicitly implemented, then operational procedures may instead ensure

that when business service BS1 is provided, services A, B, F and H are called upon.

A FALSE PARADIGM

Business systems deliver functionality, which is specified by users and business analysts and designed and built by developers. Users interact with systems as

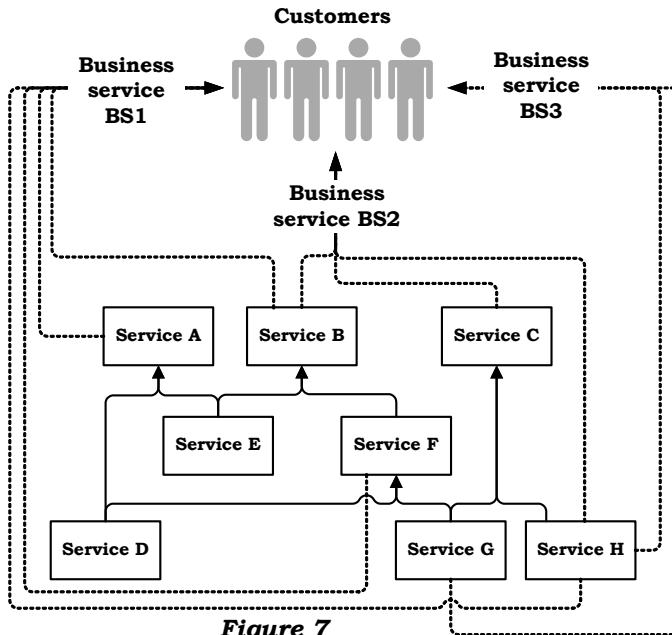


Figure 7

they do their work. A series of interactions with one or more systems (plus related off-line activities and/or relevant automation) is a business process. If workflow organizes, distributes, delivers or automates the work, then it is a workflow-supported business process.

The paragraph above may be true as history but a business process management (BPM) initiative based on it may not succeed as well as it could.

It assumes a business process is nothing but

an empirical fact: person X does Activity1; person Y does Activity2 using system S1; person Z does Activity3; and so on. The same paradigm generates optimized processes by getting X, Y and Z in a room with other stakeholders and experts to map and debate until leaner, tighter, ‘to-be’ models emerge, complete with lists of system changes to make everyone’s lives easier.

The problem is the ‘nothing but’. It ignores the elephant, perhaps because it is a supplier-centric paradigm. An organizational entity, say division D1 in Figure 4, is responsible for a set of functions, employs a number of people and uses and invests in one or more systems. It will be rewarded on how efficiently and effectively it carries out its functions. Its IT investment will be geared to optimizing its functions. It may have close relationships with internal or external IT suppliers, who will be rewarded on how efficiently and effectively they maintain, develop and support the systems D1 depends on. Anything likely to extend the functionality and deployment of those systems will be in the IT suppliers’ interest, as careers and incomes depend on it. This is a world where ‘business analysts’ specialize in system S1 or system S2 or ‘workflow’, where it would be incomprehensible to see a business process other than in terms of the systems which happen to be implemented; a world as in Figure 8.

It is a world where system boundaries influence organogram. But local ownership of systems and devolved IT funding also entrench system boundaries. Key to extending the functionality and deployment of systems is the successful promotion of the design patterns the systems are based on. The more they are promoted, the better the fit, as if system and business context were made for each other. Even-

tually the patterns become the cognitive models the support teams think and communicate with. Workflow team D3.5 sees the business context in terms of workflow system S5; team D3.2 supporting product P1 sees the business context in terms of system S2. All well and good unless S2 axioms and S5 axioms conflict with each other, and until S2 and S5 need to integrate. Then if they do clash, shared self-interest will bring a negotiated truce. So what if this means duplication? Both teams get work.

A good example is the business process itself. For workflow team D3.5 the business process is a set of queues plus allowable pathways. For support team D3.2 the business process is a series of screens in system S2, along with automated processing when a long-running transaction reaches specific statuses. As psychologist Abraham Maslow would say: *It is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.*⁶

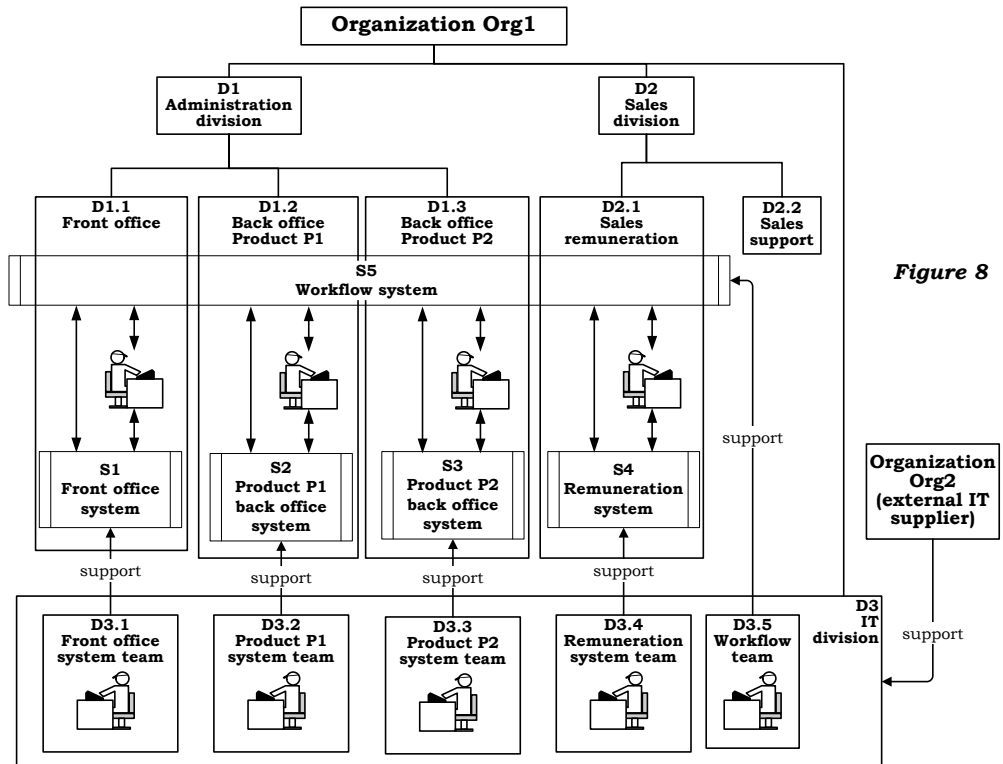


Figure 8

It is in the interest of both teams to see two separate things which need to be kept in step, as the alternative is for the business process to be only in S2 or only in S5. Either decision throws a different baby out with the bathwater, as the real issue is that neither system treats process very well because the original designers had a flawed process paradigm or no process paradigm at all. Which in turn would mean the teams' cognitive models are flawed—an inconvenient truth. In the (rather sexist) words of novelist Upton Sinclair: *It is difficult to get a man to understand something if his salary depends on his not understanding it.*⁷

⁶ Maslow, A. (1966). *The psychology of science: A reconnaissance*. New York: Harper, 1966.

⁷ Sinclair, U.B. (1935). *I, candidate for Governor: And how I got licked*. New York: Farrar & Rinehart.

BUSINESS PROCESS ARCHITECTURE AS A METHODOLOGY

The intention is not to demolish all other models to leave just one standing. Instead the benefits of a particular model will be offered as a way of navigating through choices and providing clear methodology and architecture guidelines.

The model is based on the premise that the business process comes first. The business process is not what people do as they interact with systems; nor what systems do between user interactions; nor a combination of the two—except in the case of a process-architected solution where the physical model corresponds exactly to the logical process model.

The model has been described in detail elsewhere: see INTRODUCTION for references, and in particular www.makeworkmakesense.com. So the focus here will be on the implications of employing it within a development and transformation initiative.

We start with organization Org1 in *Figure 8* and a project scope covering one or more end-to-end processes spanning front office D1.1 and the two back offices D1.2 and D1.3. The sales remuneration department D2.1 may also be affected. An early question will be on project resourcing. Should it be the front-office team D3.1 as processes start in the front office? Or workflow team D3.5 as all processes involve workflow? Or product support teams D3.2 and D3.3 as systems S2 and S3 handle most of the process detail, so those teams hold and control most of the relevant knowledge?

We need only ask questions like these to see what possible answers imply. Let's assume Org1 is bold enough to set up a new team D3.6 including members from at least teams D3.1-3 and D3.5. An immediate risk is that a large and powerful group of people left behind in teams D3.1-3 and D3.5, along with client colleagues in departments D1.1-3 and supplier colleagues in Org2, may want to see team D3.6's process project fail. They will not say that of course. More likely they will rubbish the process team's methodology.

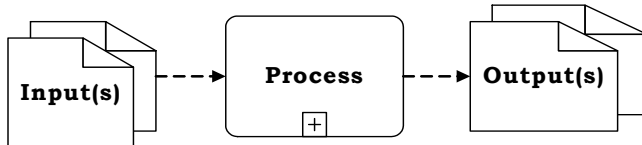


Figure 9

The methodology starts by qualifying a familiar generic model: *Figure 9*. In the case of a business process there is advantage in identifying a specific input (the **request**) and a specific output

(the **outcome**). The model in *Figure 10* has three immediate benefits:

- (i) It is customer-centric: the request is typically from a customer; the outcome typically for the customer.
- (ii) The request and outcome are linked: the request is for the outcome; the outcome is typically the thing requested.
- (iii) The request can be understood as a data entity belonging to the organization's logical data model, therefore with foreign keys to other business entities.



Figure 10

The link (ii) between request and outcome borders on identity—certainly close enough to draw a key architectural implication: the re-

quest entity initiates the process and changes status as it passes through the process, until the last status change of all, representing the outcome.

The next steps bring in business rules. A **process** is a sequence of status changes from request to outcome, and the status changes are governed by rules. The rules applicable to all request entities of the relevant type (eg all orders, for a purchase order process) split the process into **subprocesses**, as in *Figure 11*.

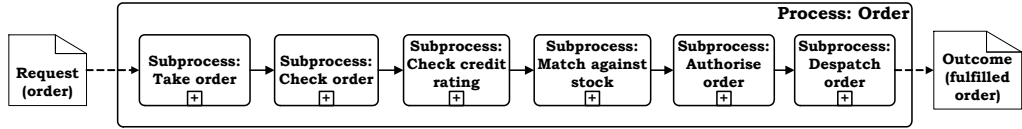


Figure 11

Subprocesses are typically sequential, but could be in parallel if that is what the rules specify. For example **Subprocess: Check credit rating** could run parallel to **Subprocess: Match against stock**. The BPMN notation in *Figure 11* is ultimately just another way of saying:

- First take the order.
- Then check the order.
- Then check the customer’s credit rating.
- ...etc.

The third and final level is **task**. A subprocess consists of one or more tasks plus the routing between them. Where subprocess-level routing applies to all request entities for the process, task-level routing differs for different request entities—because the attributes of the request instances will have different values. The task structure of the process must accommodate every possible request instance. A request instance will typically pass through every subprocess, but only through the tasks it needs to pass through.

A task is either automatic (all data available; the next status change achieved by applying rules mechanically); or manual (human interaction needed, eg because data is missing, authorization is required, or a decision must be made).

Figure 12 shows a simple task structure for the first two subprocesses—allowing manual data capture, automated validation, and correction of validation errors. *Figure 13* applies the same design principles to show a rather more complex structure for the third subprocess.

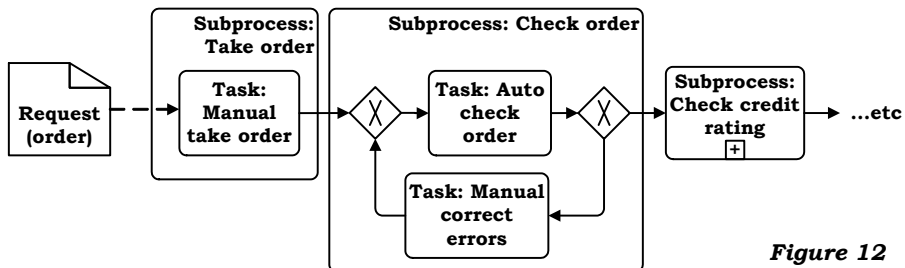


Figure 12

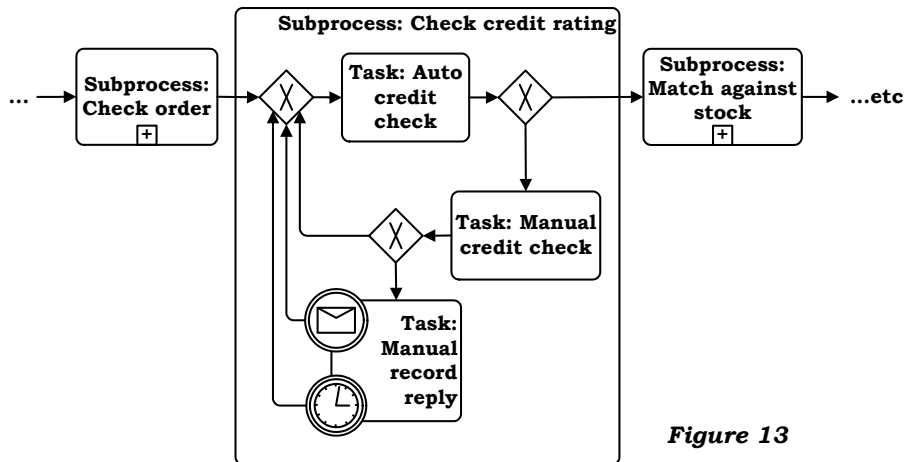


Figure 13

It is important to see the model free from existing systems. This is not to throw everything out and start from scratch, but to have a basis for evaluation and a target to aim at.

The model is at request entity instance level—the individual order. If implemented it would provide complete control over each instance, regardless of attribute values. This is true by definition, as the task structure is designed to handle every possible combination of values. The task structure of **Subprocess: Check order** in *Figure 12* only allows well-formed orders to progress further. The task structure of **Subprocess: Check credit rating** in *Figure 13* automatically passes orders already satisfying the organization’s credit rules to **Subprocess: Match against stock**, and routes all others to a manual task where an authorized user applies discretion and/or communicates with the customer (for eg advance payment or a bank reference justifying increase in credit limit).

Each subprocess (except perhaps the first if the request entity can only be captured manually) will typically have a controlling automatic task representing the ‘straight-through processing’ (STP) route. STP is at subprocess level: an error-free order would pass straight through **Task: Auto check order** in *Figure 12* but might need manual credit authorization in **Task: Manual credit check** in *Figure 13*. It might then pass straight through **Task: Auto match against stock** (not illustrated) because the ordered goods are in stock, and so on.

Relating this now to existing systems we could envisage a ‘purchase order system’ providing, say, a screen to capture orders (the main part of **Task: Manual take order** in *Figure 12*), but where the only data validation (**Task: Auto check order** and **Task: Manual correct errors** in *Figure 12*) is within the capture screen itself. This might mean an incomplete order cannot be captured—following the garbage-in-garbage-out (GIGO) design principle. There might also be a workflow system presenting scanned orders to users for data capture, but the order which is the controlling entity in workflow is a physically different data record from the order captured in the order system. It must be different in the example just given because it exists in workflow but has not yet been accepted by the order system. It may also be a different entity logically—in workflow it may be a document (image) whereas in the order system it is a data set including foreign keys to customer, product etc.

More generally, a workflow system would focus on user interactions: in our model, **Task: Manual take order**; possibly **Task: Manual correct errors**; **Task:**

Manual credit check; Task: Manual record reply; etc. Figure 14 shows an example queue structure.

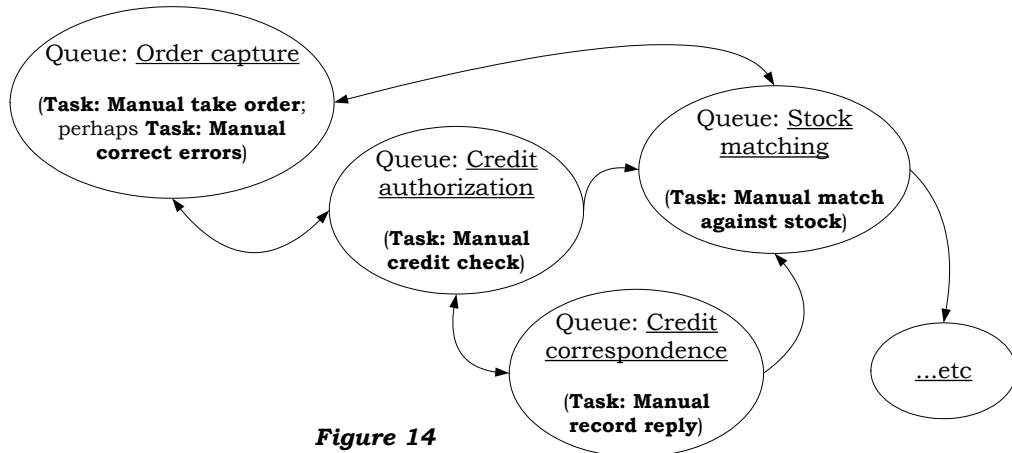


Figure 14

Meanwhile the likely home for the data and rules featuring in **Task: Auto check order, Task: Auto credit check, Task: Auto match against stock** etc would be the purchase order system itself. So must we have two processes implemented or reflected, one in each system? How do they keep in step? Is one always in control, and if so, which one?

These issues arise not from ‘poor design’, but from IT history following IT economics: record-keeping first, then generic workflow. Now that storage and transaction costs have dropped to make integrated, instance-level process support economically viable (and competitively vital) we have two obstacles. One is ‘how to get there from here’: the recalcitrant legacy architectures we must navigate. The more insidious obstacle is legacy thinking—assuming the reasons system boundaries and system design are as they are must hold true for all time.

The model depicted in Figure 10 to Figure 13 may also not hold true for all time, but it is a far more effective response to current IT economics and competitive pressures. It can be implemented given appropriate BPM and SOA architectures. The cost and difficulty of ‘getting there from here’ are constraints on logistics and phased delivery, not reasons for rejecting it.

Relating this now to the componentized services of Figure 5 brings us to the overall schematic of Figure 15. The final step should now be obvious: task is the fundamental unit of a business process, so componentized services link to tasks. We thus have a framework for logical and physical architecture and delivery methodology, which suits process-based transformation projects and avoids both supplier-centric legacy thinking and the kind of empirical, anti-architectural pragmatism which too often drives Lean-type process re-engineering.

These last two are related. It is tempting to assume that, because business processes are generally poorly implemented in IT systems, business processes must be ‘something else’, something people do with systems, or the bits in between system functionality. That ‘something else’ is a popular target of Lean-type transformations which pride themselves on not getting bogged down in IT development backlogs.

There is nothing wrong with Lean *engineering*. Process architecture is Lean by design. The problem is empirically based *re-engineering* initiatives which see a

process as a *de facto* sequence of improvable activities, but which lack the analytical tools and/or appetite to cut through layers of technology.

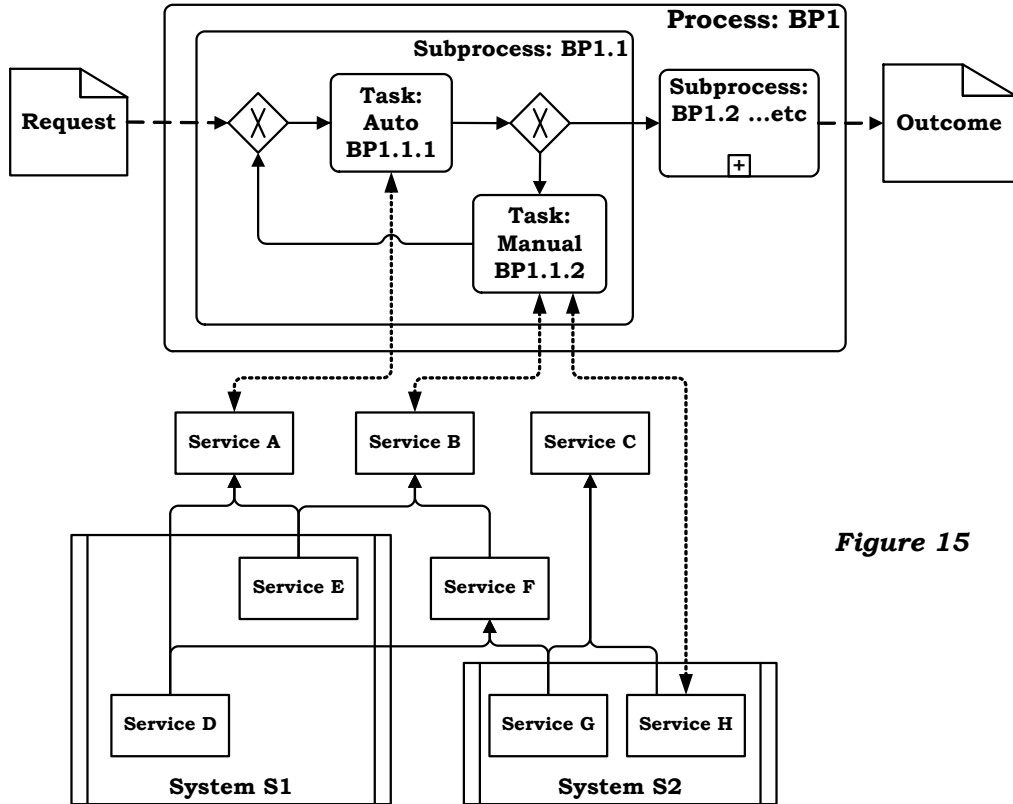


Figure 15

Seeing or not seeing a logical process model is like seeing or not seeing a logical data model. In a forest it is breathtaking how close you can be to an elephant before you realize it's there.

EXAMPLES

We shall now apply the metamodel to real-life examples to draw out a few methodology implications.

Request

If we focus on the customer request as the initiating entity then the format the request is in should be secondary. An order could be captured by the customer on a web page, posted or faxed as a form or a letter, telephoned to a call centre, or received by email. The process should cater for all of these. There will be format-specific rules, but also generic rules regardless of format. There should not be one project to develop a 'web-based order process' resourced by a 'web team' and another 'back-office order process' project resourced by a different team.

Figure 16 shows an example process design for a context where orders and other incoming requests can be received by a variety of channels.

Organization Org1 in Figure 8 could be a financial services company, where (say) P1 is a loan product and P2 is an investment product. Org1 may not have a purchase order process as such but a range of other processes initiated by different request types: loan application, loan redemption, investment application, invest-

ment payout, change of address etc. Again each process would be the same regardless of format or channel, but no doubt with format- or channel-specific rules. So a truly process-based Org1 should not have a ‘web team’ developing a ‘web-based loan system’.

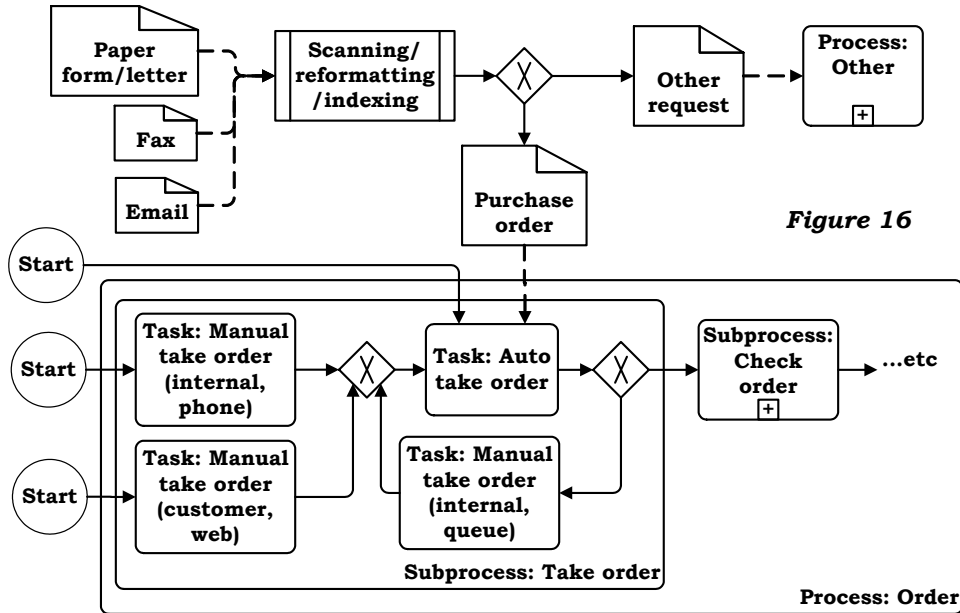


Figure 16

Data and Rules

Empirical observation is often key to re-engineering initiatives, whether to understand and measure a process or to identify improvement opportunities. The metamodel provides a framework for structuring the results and, more fundamentally, a program for what to observe—so as to understand the data and identify the rules.

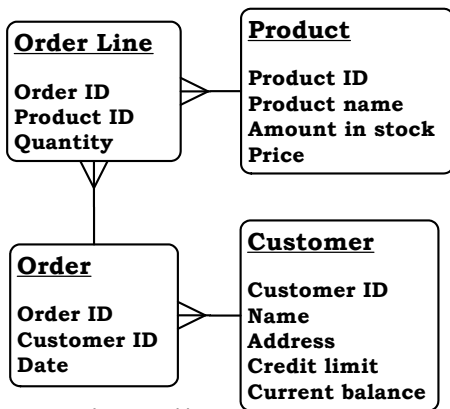


Figure 17

In the order process the request is the purchase order. Figure 17 shows a straightforward logical data model.

An order could be for many products. In the draft process design of Figure 11, when the order gets to **Subprocess: Match against stock**, some items could be in stock but others not. We need a rule: can we despatch part orders? We will assume yes, but only if the part order value exceeds a certain parameter.

Our draft design must change to something more like Figure 18. Here one instance of **Process: Order** creates an

instance of **Process: Fulfillment** for each part order which can be authorized and despatched. **Subprocess: Complete order** terminates **Process: Order** when all part orders have completed.

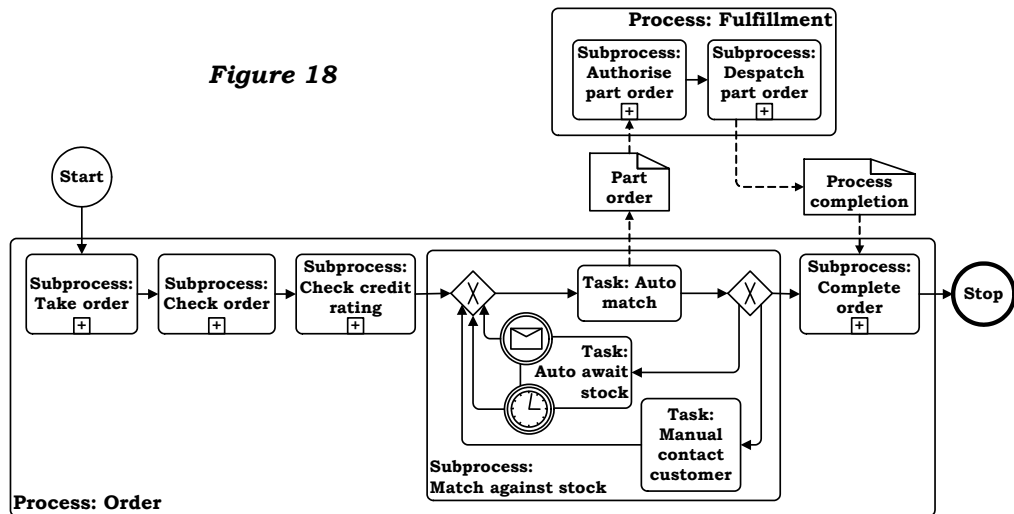
The process implications of part orders may call into question the logic and position of **Subprocess: Check credit rating**. In both Figure 11 and Figure 18 the

assumption is that the value of the whole order is checked against the customer’s credit limit and current balance. But the value of the first part order could be within the credit limit, so maybe the rules should apply at part order level—which would mean a more sophisticated task structure factoring the credit checking rules into **Subprocess: Match against stock?**

CONCLUSION: DESIGN FIRST

Examples like these show a draft logical process design and draft logical data design being used to tease out and clarify business rules. It is hard to overstate the significance of this for a process project. This is the elephant in the room. The process metamodel both allows and requires logical design to start early, in order to identify requirements. We do not ‘define process requirements’ and then pass them to a development team for ‘process design’. This is fatal, as business users and business analysts typically define requirements in terms of systems they are familiar with. Logical design needs to be done by someone who understands data analysis and process architecture. That someone could be called a process analyst or process designer or process architect or process modeler or business analyst or business architect or business modeler. What is crucial is that he or she must understand how process components (request entity, rule, process, subprocess, task) fit together at a logical level in order to engineer a customer-centric process solution which works within the logical data model. A business transformation initiative which has a significant process component will be less successful than it could be if it does not acknowledge this and does not factor it into its project and engagement design.

Figure 18



This approach is very different from familiar systems projects, and both ‘business’ and ‘technology’ people can find it a struggle—particularly if it questions structures of power and influence. (It can be difficult to get a man or woman with a hammer to understand that everything isn’t a nail.) It calls for a holistic engagement and delivery model and holistic data and process design skills. But it works. It avoids massive duplication, and it delivers: because it opens people’s eyes to the elephant.

administration systems, 3
business process architecture, 1, 2
business rules, 9
componentized services, 3
customer-centric, 1, 8, 14
economic trends, 2

metamuddle, 3
organogram, 6
re-engineering initiatives, 13
Service-oriented architecture (SOA),
3