Are There Requirements for BPS?

Ian Alexander

Scenario Plus, London, England
ian@scenarioplus.org.uk

Abstract. A huge amount of effort is consumed in modelling business processes, and then in developing and maintaining tools, documentation, and training courses to support the modelled processes. This discussion paper looks informally at the activities, costs, and benefits involved, and asks some simple and down-to-earth questions about the entire BPS enterprise. Why does it cost so much? Is it worth its salt? What does it deliver? Why do people believe in it? Is it relevant to business? Would other approaches be more effective?

1 Introduction

Why do people bother to model and try to support business processes? It clearly consumes a large amount of effort – indeed, a whole industry of process modellers, process consultants, process modelling tool makers, gurus and trainers subsists on the presumed value of the modelling enterprise. Is the effort justified?

The purpose of this paper is not to present a position, far less to rehearse erudite methodological arguments. Its purpose is to stimulate debate and reflection, initially in the framework of this year's workshop [1], and ultimately perhaps to encourage the more efficient use of available resources of time, money, and skill in Business Process Support. It is therefore written simply as an essay, without the apparatus of statistics and references, taking a fresh and direct look at what businesses get for their BPS money.

In this context, questions may be more valuable than answers; untested hypotheses may sometimes be more useful than carefully-justified evidence; and impertinent the-emperor-has-no-clothes observations than polite dissertations.

2 BPS Activities

BPS depends on a chain of activities, in which each link is vital. The metaphor says that if any link in the chain is weak, the result must be weak also. What is that chain? Is there anything special about it? Is it indeed different in any way from 'system', 'software' or even 'product' development?

The basic chain of activities starts (for the purposes of simplicity) with a request for a business process to be analysed and modelled so as to understand it better, and hence to support it better. Interviews, observations and workshops are conducted, and

on the basis of the collected data, a model is constructed. Let us call this business process modelling or BPM, even though in fact the step involves much more than just model-making.

The next link in the chain is to optimise or modify the model so as to improve it with respect to some measure of quality, such as time or cost to complete an operation. This step, business process re-engineering or BPR, is not mandatory, though we can observe in passing that it was for a while hugely fashionable, creating another industry, an army of consultants led by industry gurus, and consuming a prodigal amount of money for what were often paltry returns. Does this constitute a lesson for BPS as a whole?

The next link in the chain is to identify from the model what needs to be supported. Traditionally this was done on an individual activity or department basis, e.g. the finance department identified that the billing process was too slow and decided to improve it, generally by automating it with a computer-based system; we will return to this issue below. Let us call this step business process improvement (BPI) project initiation; actually it contains a considerable range of activities including identification of candidates for improvement, cost-benefit and other trade-off analyses, budget allocation, and project planning.

Next, the selected BPI development projects are run; this step involves specifying, designing, building and testing the software, computer hardware, and any other equipment needed to provide the 'support' part of the new system. Let us call this step BPI development.

Finally, the business process support comes into service, perhaps with a period of trial running or parallel operations; usually with training; and itself supported by a helpdesk and a maintenance programming team. Let's call this step BP operations and maintenance.

3 Costs & Benefits

3.1 Silo-ism and Process Propaganda

The traditional chain of BPS activities described above was clearly vulnerable to several diseases, notably silo-ism or lack of joined-up process thinking. Indeed, if a process model was commissioned by a department, the interfaces to other departments were probably sketchy if not downright inaccurate, being on the periphery of vision and interest.

If the much-overused term 'process' has any meaning (beyond creating a vague feeling of being up-to-date), it must be that the defensive walls of the silos are breached, so that the business activities are considered end-to-end and from the point of view of the customer, i.e. of the quality of the services or products delivered.

This is more easily said than done: there are powerful forces maintaining the silo walls – careers and jobs, the limits of individual knowledge and competence, fear of the unknown, power politics, personal rivalries, and empire-building, and so on. Is it possible to sweep away the walls without another, less visible set of barriers springing

into place? If these homeostatic forces [2] – the departments' immune systems, we might say – are destroyed, can the organisation survive, or will it be so disrupted and demoralised that it ceases to function effectively? Conversely, if the silos are allowed to remain in place, can 'process' thinking realistically be applied, or is it doomed to remain no more than propaganda and pious window-dressing? To put this another way, installing a faster network and a messaging 'system' to speed communications between departments is a waste of time if the slowness comes from quite other causes, such as overwork, poor training, or interdepartmental distrust. A real system, in other words, never consists only of equipment, but includes the people, procedures, politics, and social infrastructure of a business as well. I once saw a bottle of engine oil advertised as an 'engine maintenance system': are you sure you are not using terms like system and process in a similarly vacuous sense?

3.2 Process-Speak, Development-Speak

If a vast industry is based on a concept, it is worth asking whether that concept is actually distinct and well-formed. The chain of BPS activities outlined above was:

- BPM
- BPR (optional)
- BPI project initiation
- BPI development
- BP Operations & maintenance

A software developer used to UML and Use Cases, a user interface designer used to task modelling, or a contextual design engineer used to scenarios – note the diversity of job descriptions, communities, and languages! – might choose to describe the same chain as a story, i.e. in functional terms:

- Model the business
- Re-engineer the business (optional)
- Evaluate & select projects
- Develop software / systems
- Operate & maintain

A requirements engineer used to thinking in terms of stakeholder needs and system specifications (I confess to being in this camp) might be more inclined to focus on a slightly different story:

- Identify business goals
- Identify stakeholders & viewpoints
- Define stakeholder needs (as scenarios)
- Specify required systems & software
- Develop them
- Operate & maintain

What has happened in these transformations? Why? It isn't just a matter of grammar. The focus of attention has shifted from modelling the business to identifying problems and solving them. If you are not going to bother with BPR, and you are going to conduct a thorough trade-off analysis of the possible improvement

projects that you might choose to spend your money on, then why do you want to model the existing business?

The obvious answer is that it might help you identify candidate areas for improvement. But is this anything like sufficient justification for such a costly activity? After all, anyone who has ever gone into an organisation as a consultant knows that you can often tell within an hour what the organisation is like, and what is wrong with it. Further, you only have to ask anyone in the business over a cup of coffee what they'd most like to see improved, and they will instantly tell you a dozen grievances. For instance:

- IT has been outsourced, so it is slow, difficult, and costly to move a PC or get ink for the printer. You aren't allowed to put any software package on your own PC; you have to fill in a form, get it authorised, get the software tested by the IT company at your own department's expense, and then (a few weeks later if you're lucky) you can have a look at the software to see if it is worth using.

Does that not sound familiar? Do you need 6 months of process modelling to find it out? The problem, perhaps, is that it is a message that managers don't want to hear. It's easy to see why they might prefer to lose the message in a mass of diagrams.

So, are the developers (and others) not right when they gloss over process modelling, and focus on tangible things like software development and staff training? Tools certainly aren't the answer in themselves: tools are only fragments of systems, and 'a fool with a tool is still a fool' is as true a saying today as ever it was.

But aren't end-to-end approaches like goal models, task models, scenarios and Use Cases just another way of saying 'process' and 'systems thinking' without the p-word? Perhaps; but they can lead swiftly to implementation decisions, whereas the path from BP models is long and indirect. Indeed, John Carroll [3], a user interface and scenario pioneers, says that 'When we design systems and applications, we are, most essentially, designing scenarios of interaction.' and we might add a corollary, that thinking out scenarios of interaction is inherently and directly a system design activity.

The fragmentation of languages and schools of thought among engineers is at least as severe a problem as the silo-ism within organisations. Wheels are reinvented; knowledge is laboriously re-created; duplication of effort is the norm.

4 Discussion

So, are there requirements for BPS? To put it another way, is there anything distinctive and valuable about BPS that is not provided by other approaches such as systems engineering, object-oriented software development, or for that matter extreme programming and so-called agile methods? If the life-cycle is 'find out the problem - solve it, and repeat', then what value is added by talking about and modelling processes? Does BPS-thinking, for example, in some way cause development projects to be selected more rationally? Does it somehow address the social, political and environmental issues better? Does it provide a basis for choosing between development approaches, offering a ready-made set of criteria for deciding whether

this project would benefit from formal methods, while that one could be done by rapid prototyping? It isn't obvious that it can deliver benefits like these.

Why, then, should "a detailed process model .. be built in addition" to "a data-model" [1]? A data definition is indeed plainly not enough on its own.

It is interesting that traditional systems analysis focused mainly on a functional model, namely dataflow, which was essentially a process description with low emphasis on stakeholders; it was supplemented with a suite of other models, including state transition, entity-relationship (data), and if need be (for real-time systems) timing diagrams.

In reaction against this, object-oriented analysis (e.g. with UML) focuses mainly on data, giving rather little effort to functions and some guidance on software structure; it models processes initially with use cases to give an end-to-end view, and then proceeds to make both flowcharts (activity diagrams) and message sequence diagrams; the flowcharts can be annotated with roles (actors) to create swimlanes diagrams which overlap oddly with sequence models in their appearance and semantics.

Component-based approaches are in their infancy, but they are likely to combine object-orientation with an emphasis on finding components that can be bought off-the-shelf, on interfaces (so that components fit together first time) and on reuse. They would bring a strong emphasis on software structure to the party.

Agile or extreme methods tend not to make models at all; they briefly document 'user stories' and then create test cases and code. This approach can run into blind alleys, in which case the code has to be 'refactored' to get it out of trouble. Success is claimed in some business information systems for this surprising approach.

All of this suggests that while everyone agrees you need to spend at least a little time (two minutes writing down a 'user story') thinking about what your software and the system that contains it has to do, you do not necessarily have to spend time on BP modelling what the previous system (including people, etc) used to do.

The other substantial claim in [1] is that "a traditional business information system is .. designed to support a business as is (but more effectively), while a BPS system should be designed to support a new way of running the business, the one which is not possible without the system. The latter leads to substantial requirement changes even long after the system has been deployed." But if development is always iterative – you develop a system, its users complain or ask for extra features, you develop some more – then doesn't the difference between BPS and other approaches disappear? Requirement change is nothing new. What is more, any successful intervention, even if it is designed to support a business "as is", is guaranteed to change that business in some way. So what, if anything, does BPS have to offer?

References

1. Ian Alexander, Ilia Bider, Gil Regev: REBPS'03: Motivation, Objectives and Overview, CAISE'03 Workshops Proceedings, Klagenfurt, 2003.
2. Gil Regev, Ian Alexander, Alain Wegmann: Use Cases and Misuse Cases Elegantly Model the Regulatory Roles of Business Processes, BPMJ 2003 (to appear)
3. John M. Carroll: Scenario-Based Design, John Wiley, 1995