How to simplify the evolution of business process lifecycles

Dr Alexander Samarin

Independent consultant, Switzerland
www.improving-bpm-systems.com
samarin@bluemail.ch

Abstract. My experience shows that business wants that separate requests for change be implemented quickly in existing systems. These changes are typically small (from the point of view of the business) and unpredictable (from the point of view of the IT). Obviously, all phases of the business process lifecycle should be coherently organised to be able to handle such a flow of requests for change. The aim of this article is to share my experience with a practical architectural framework for the improvement of complex business systems [1-4]. This framework provides recommendations on how to implement BPM systems which are easy to evolve.

Keywords: BPM, SOA, business process, artefacts, flexibility

1 Introduction

Improving the organisational business performance is a permanent imperative and a daunting task. Enterprises improve their performance by changing their business processes to perform more effectively and more efficiently. The main tool used for such improvements is a Business Process Management (BPM[1]) system. Obviously, all enterprises have their own BPM system (a collection of all business processes), but this system is often a "problem" of its history, and suffers from problems of complexity, inefficiency, etc. So, how do we improve a BPM system?

The business world understood a long time ago[2] that services and processes are the backbones of most businesses. The IT world recently "re-discovered" and accepted the notion of services, and so emerged Service Oriented Architecture (SOA). But IT is still not very comfortable with processes.

In reality, both the business world and the IT world work with different views of the same business. In a simplified way we can say that the business world sees the business as a coherent set of processes which are under the control of the business

1 "BPM allows you to model, automate, control, measure and optimize the flow of business process steps that span your organisation's systems, people, customers and partners within and beyond your corporate boundaries."

2 See, for example, the numerous articles on Business Process Re-engineering, and Quality Management Systems.

people. Typically these processes constitute the management model. On the other hand, the IT world sees the business as a set of IT services which are under the control of the IT people. Typically, these services constitute the implementation. From both perspectives, processes and services are the principal artefacts of the system, and these are complemented by business events, rules, data, roles, etc.

In current systems, processes, services and other artefacts are typically "diluted" in existing monolithic applications. This makes the business difficult to evolve since to do so, changes to program code are often necessary. To achieve high flexibility[3] in the business we want processes, services and other artefacts to be distinct and versionable, whereby each artefact can be evolved easily. We need to consider a business as a complex dynamic mixture of processes, services and other artefacts. The composition and the structure of this mixture are unique for each organisation, but the structures share hierarchical, multi-layer and fractal characteristics (or patterns).

Our framework approach provides guidelines and patterns for structuring processes, services and other artefacts to simplify their evolution.

2 Your BPM system must be architected

As we want that the BPM system will be easy to evolve, we have to anticipate potential changes in practically all aspects of the organisation such as the policies and priorities, organisational structures, business processes, external obligations (compliance), technology, level of computerisation within the company, stakeholders, culture of the users, and the size and complexity of problems to be addressed.

Most of these potential changes are changes of the artefacts which constitute the business processes. Any BPM system works with several types of artefact. We can quickly identify most of the artefacts simply by analysing a popular definition of a process: *who* (roles) is doing *what* (business objects and activities), *when* (coordination of activities), *why* (business rules), *how* (business activities) and with *which* results (performance indicators). So, the business process is a complex and dynamic set of many artefacts. (In this paper I will concentrate primarily on artefacts directly related to business processes, i.e. I do not consider typical IT artefacts such as servers, databases, operating systems, networks, etc.)

From the systemic point of view, easy evolution of the business process means that it should be easy to modify each and every artefact without causing any negative effects elsewhere in the system. As the artefacts are **interconnected** and **interdependent**, we need a comprehensive plan (i.e. the architecture) of how to build, to use and to evolve all artefacts and the relationships between them. So, the evolution of the BPM system is the management of the evolution of *all* its artefacts and the relationships between them simultaneously, as a system.

Formally, we define all these **versionable** artefacts throughout their life-cycle. We model **explicitly** all relationships between these artefacts. And, very importantly, all models are made to be **executable**. This means that in an implementation, a model

[3] *Flexibility* is defined as "the ability to change without losing identity" [5].

acts as a skeleton or foundation to which we attach services (stand-alone pieces of functionality which are available only via a formally defined interface).

Of course, to guarantee a high level of flexibility it is necessary to expend additional effort to build flexible systems, but this extra effort more than pays for itself in the long run as shown in figure 1. The data in figure 1 are derived from practical experience with the architectural framework. [One application was a production system comprising about 3 000 complex products per year, 50 persons, about 50 different tasks, 3 production chains, 6 repositories and 40 IT services. The system was in place for several years. The maintenance and evolution of this production system required in several times less resources. Several successful (and easy to do) migrations were undertaken.]

As figure 1 clearly demonstrates, the result of the use of the architectural framework is that the Total Cost of Ownership (TCO) is much less than that for traditional IT. In fact, the difference between development and maintenance virtually vanishes because in many cases the system can be evolved by changing only a small part of it (e.g. a particular service, process or other artefact) rather than having to review a larger part of it.

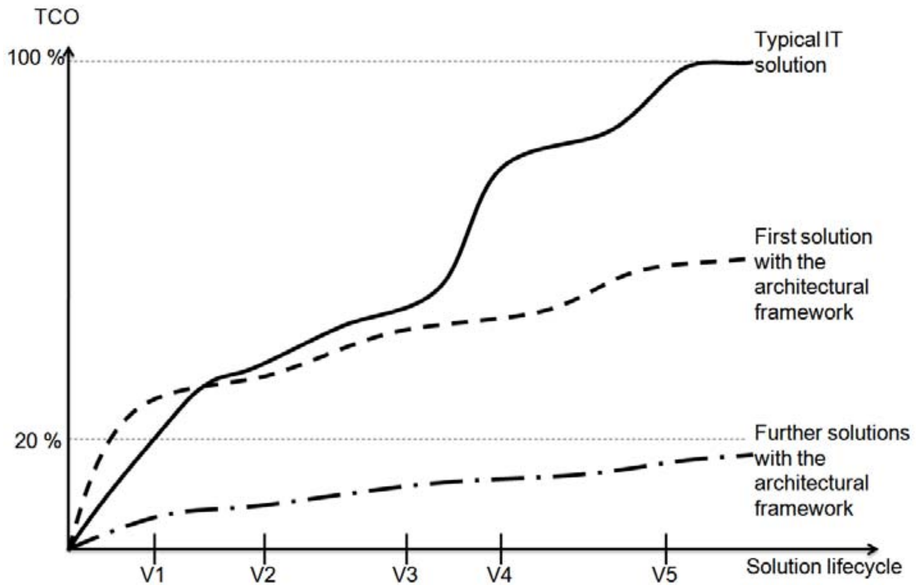

Fig. 1. Each subsequent solution is cheaper because it reuses the same tools, the same services and the same architecture

3 Common understanding of artefacts

Any BPM system has many stakeholders, each with their own concerns and with their own artefacts. The internal stakeholders are from three groups: strategy (top managers), business (line managers, super-users, users, modellers of business processes), and IT (managers, architects, developers, and operators). Each type of stakeholder sees and uses the BPM system in a different way and many system characteristics and features may be different for different stakeholders. For an implementation to be successful, it is necessary for a BPM system to address the concerns of all its stakeholders and to explain in advance how this system will change their work. Also, a common understanding of all BPM artefacts amongst all primary stakeholders (those directly involved in the design and implementation) is crucial.

In some cases I have observed a real confusion between the IT and the business in their discussions of BPM artefacts and artefact-related issues because of inconsistent terminology and unspoken assumptions. The IT and the business may use the same term, but it may mean different things, not only to these two business partners but also in different software products.

I recommend that you make any potential assumptions/differences explicit by talking about both "technical" and "business" artefacts. In addition to natural differences in personal understanding, the technical meaning may depend on the software product in use and the business meaning may depend on the business domain.

The following list details the types of artefact important for all BPM systems:

- added-value chain,
- macro-processes,
- events,
- processes,
- rules
- activities,
- roles,
- objects – data structures,
- objects – documents,
- audit trails,
- indicators, and
- services.

4 Improving artefacts

To work efficiently with artefacts, usually, we have to improve them. Firstly, all artefacts have to be **digitalised**, i.e. exist in electronic form. Typically, business objects (especially data structures) are digitalised, whilst other artefacts (e.g. processes and rules) are often defined on paper and implemented implicitly (in such a way that they are not easy to validate) in applications.

Secondly, artefacts have to be **externalised**, i.e. be available as separate and explicit entities. For each artefact we would like to know its naming convention, ownership, versioning, complexity, number, modification frequency (some estimation of the speed of evolution), security needs, traceability and, in general, lifecycle. Each artefact is versionable and several versions of the same artefact may co-exist. One may have a business rule expressed in a digital form, e.g. coded as a macro in an

Excel spreadsheet, but not yet properly externalised because of the absence of a well-defined lifecycle for this rule.

Thirdly, artefacts have to be **virtualised**, i.e. be available independently from particular IT resources such as servers, databases, media, formats, browsers, etc. For example, we should be able to use in a Java program a business rule defined in an Excel spreadsheet. With virtualisation we address how to transport some artefacts and to provide them to other artefacts.

Obviously, the best virtualisation is provided by the Service Oriented Architecture (SOA[4]). We can say that BPM, by revealing its artefacts and the relationships between them, provides the necessary context (e.g. granularity) for the definition of services and SOA provides their implementation, execution and governance.

It is important that some types of artefact are aligned across the whole enterprise to avoid integration problems and to improve reusability. For example, all roles, business objects, and services should be registered and maintained at the enterprise level. Other types of artefact have a lower enterprise visibility, but may have some enterprise-wide importance. For example, a change in the postal address of an insurance client may lead to the revision of his/her contract. This is certainly an event that has an impact elsewhere in the enterprise. Meanwhile a change in a client's e-mail address may have a limited impact.

5 Structuring relationships between artefacts

There are many relationships between BPM artefacts. For example, an informal relationship between artefacts can be described as follows.

- The business is driven by *business events*.
- For each business event there is an associated *business process* to be executed.
- A business process coordinates the execution of *business activities* (human or automated) , in accordance with *business rules*.
- A group of staff members (*business roles*) is responsible for the execution of each human activity.
- Each business activity operates with some *business objects* (data structures and documents).
- The execution of business processes produces *audit trails*, which are used for the calculation of *key performance indicators*.

Another informal relationship between artefacts is their **visibility**. One of the main difficulties in the implementation a BPM project is that the project team often underestimates how many artefacts are necessary to run a BPM system and the related efforts to improve these artefacts. Typically, the improvement of obvious artefacts will reveal their hidden supporting "structure" (see figure 2).

[4] SOA is an architectural approach for constructing complex software-intensive systems from a set of universally interconnected and interdependent building blocks, called services.

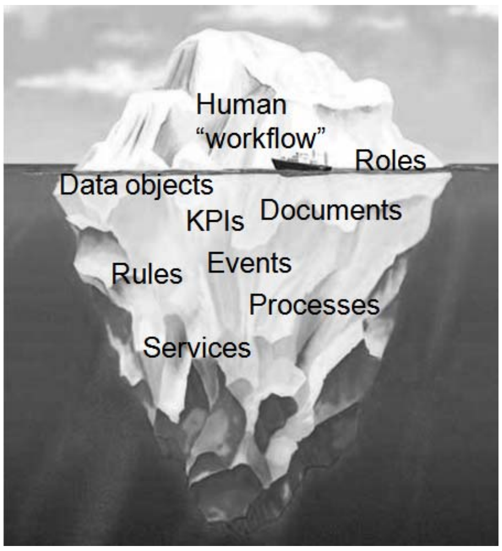

Fig. 2. Visibility of artefacts

Our goal is to reveal all hidden relationships and to structure them. Examples of such relationships are as follows:

- static (in design phase);
- dynamic (in execution phase);
- composition (from atomic artefacts to a composite artefact);
- instantiation (from a template to instances);
- compatibility (between different versions of different artefacts).

If possible, we model relationships as formal, explicit, traceable, testable, secure, Service Level Agreement (SLA) aware, and, finally, executable. This means that a formal model of relationships acts as a skeleton or foundation of an implementation.

Within the scope of this paper, I cannot cover all relationships used in the architectural framework, so I will discuss the most important relationship – the composition of business processes.

The two most important artefacts, services and processes, can be considered to be intimately related since in real terms

- all processes are services,
- some operation(s) of a service can be implemented as a process, and
- a process may include services in its implementation.

Often, it may be useful to consider a service as a black box where no information about its implementation is available – see the top part of figure 3. Alternatively, it may be useful to consider a service as a white box where implementation details of all its operations are available (for example, as a set of other building blocks where the execution of those blocks is coordinated in some way) – see the bottom part of figure 3, where one of its operations is a process.

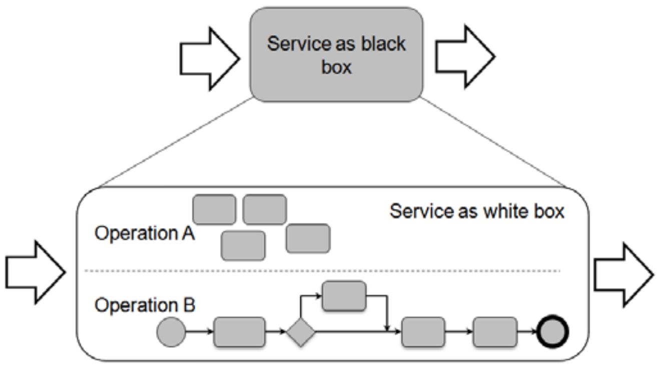

Fig. 3. A service as a black box and as a white box

6 To model executable business processes is to develop implementable business processes

Historically, the modelling of business processes is disconnected from their implementation. As a result, models are often un-implementable and implementations don't correspond to their models. I believe that business process modelling should deliver an executable process implementation that is **simple** (i.e. it is comprehensible by all stakeholders involved) and **complete** (i.e. it does something very similar to the final result). This may sound a daunting mission, because the modelling alone of real business processes is very difficult, and to combine this with implementation is contrary to many IT project management practices. But from my experience, the synergy of the agile combination of two traditionally "difficult" phases of any business process initiative is the way to change radically the amount of effort for the implementation of BPM solutions. Small cycles "model-implement-test-refactor" (similar to the famous Deming PDCA wheel [6]) considerably simplify both modelling and implementation: any wrong decisions are easily corrected; services are quickly adapted to the required granularity and so on. As a by-product of this approach, the evolution of business processes becomes much easier – most improvements are quickly implemented because modifications are catered for by design and are welcome.

This modelling procedure is designed for joint work between people from business and IT – modern BPM suites such as Intalio have made this possible by hiding the Business Process Management Language (BPEL) [7] complexity behind the business-friendly Business Process Modeling Notation (BPMN) [8].

The purpose of the modelling procedure is to **analyse** a building block (*what* it is supposed to do) and to **synthesise** its implementation (*how* it does this) as the explicit coordination of other building blocks (processes or activities). It is an iterative procedure – we can apply it until we have left only indivisible building blocks (i.e. activities). During modelling, we collect and refine different artefacts. We consider that building blocks are constructed recursively, like Russian dolls, to avoid getting bogged down in detail.

In some senses, modelling is similar to solving a puzzle – everyone has his/her own way, but there are a few practical tips, e.g. make the edges first, group together pieces with a similar colour or pattern, collect them into clusters, use the latter as "centres of crystallisation" and then fill in the rest. But, there are a few real-life difficulties: you have to do many puzzles at the same time, to use pieces from other puzzles, to cut new pieces, to optimise the number of pieces, to transform some puzzles, etc. It should be a lot of fun!

There are four main phases in the modelling procedure as shown in figure 4 (I use my own diagramming style in BPMN – see [9] for detail).

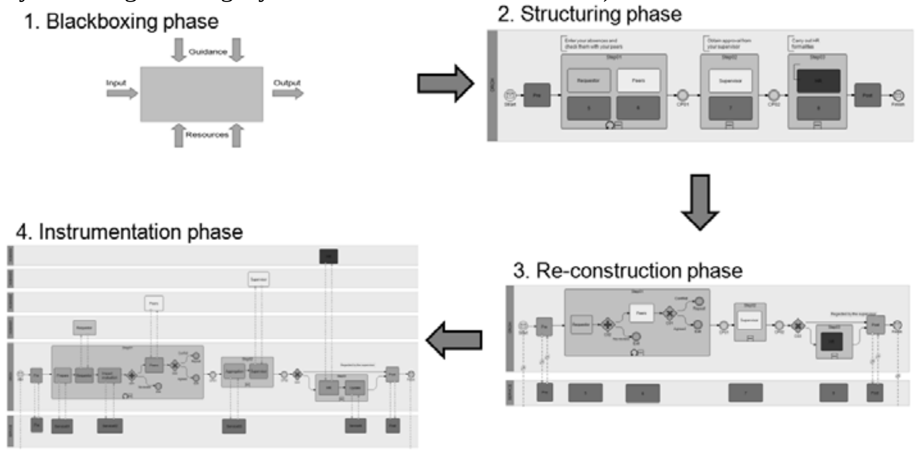

Fig. 4. Four main phases in the modelling procedure

The modelling procedure delivers an executable, and thus implementable, business process. But this business proecess is just the first version, and often requires further improvements. We have to balance the depth and breadth of these improvements as a function of the particular situation. It is important that the business people who participated in the modelling of a business process can recognise it when it comes to execution time – "what you model is what you run" is the motto of this procedure.

We may apply the modelling procedure to all newly found building blocks. The first phase of the modelling procedure (i.e. *blackboxing*) is a decision point: should this building block be further subdivided, or should we stop the modelling procedure and consider this building block as a human or automated activity? If the decision is to continue the subdivision, we continue the modelling procedure (i.e. *structuring* phase) and, as a result, we will identify several smaller building blocks (i.e. *re-construction* phase). Recursively, we will apply this modelling procedure for each of these building blocks. Lastly comes the *instrumentation* phase. The purpose of this phase is to enrich the process skeleton by adding more automated activities to make it fully executable.

Also, it is necessary to consider that the business process model may itself be improved, e.g. in the first version we implemented only the most frequently used sequence of activities and later we decide to add more variations. In any case, keep your modifications small and implement them step-by-step.

7 Conclusion

From both the business and the IT perspective, processes and services (complemented by business events, rules, data, roles, etc.) are the principal artefacts of systems. The evolution of a BPM system is the management of the evolution of *all* its artefacts and the relationships between them simultaneously, as a system. The use of an architectural framework (including a particular modelling approach) whereby BPM artefacts and the relationships between them are made to be digital, explicit and virtual, simplifies the evolution of business process lifecycles:

- the formal expression of relationships enables their automatic validation;
- the aggregation or assembly of services becomes the main implementation activity, which enables the delegation of some traditional IT activities to the business;
- small cycles "model-implement-test-refactor" considerably simplify both modelling and implementation;
- there is a good match between BPM (provision of the context for services) and SOA;
- the increase in the granularity of artefacts opens more opportunities for the use of open source products;
- there is a built-in possibility for the versioning of artefacts.

References

[1] Samarin, A.: ISO: integrating the WEB and document management, presentation at Documation conference, Paris, France, 2001.www.samarin.biz.
[2] Samarin, A.: Agile SOA for Process Automation and Integration, www.ebizq.net, 2004.www.samarin.biz.
[3] Samarin, A.: From agile development to agile evolution of enterprise systems, presentation at EuroPython Conference, Geneva, Switzerland, 2006. www.samarin.biz.
[4] Samarin, A.: Three pillars of a practical architectural framework: BPM, SOA and ECM, presentation at the Open Group's enterprise architecture practitioners conference, Lisbon, Portugal, 2006. www.samarin.biz.
[5] Regev, G., Wegmann, A.: A Regulation-Based View on Business Process and Supporting System Flexibility, Proceedings of the CAiSE'05 Workshop, p. 91-98.
[6] http://www.deming.org/
[7] OASIS (www.oasis-open.org) standard: Web Services Business Process Execution Language, 2007.
[8] OMG (www.omg.org) specification: Business Process Modeling Notation, 2008
[9] Samarin, A.: Practical Industrialisation of Information Technology (for alignment of business and IT), course lecture EPFL, Lausanne, 2007. www.samarin.biz.