A BPTrends Column January 2008

Human Processes

Keith Harrison-Broninski
CTO Role Modellers (www.rolemodellers.com)

harrison_broninski@acm.org

Is Your SOA a Disaster Waiting to Happen?

Last month I gave the keynote at SOA India 2007 – and the conference was a fascinating experience. In particular, it was interesting to talk to delegates from many different sizes and types of organization who are actually doing SOA! Not just talking about it ☺

India really is where it is all happening, IT-wise, and there are a great number of people there with a deep knowledge of SOA. Yet the discussions I had while in Bangalore, both with delegates and with other speakers, bear witness to an urgent need for better understanding of SOA governance. Many people charged with SOA implementation are technically savvy and have a good understanding both of technical innovations and of the corresponding software tools, but still possess little or no appreciation of the key organizational issue underpinning a move to SOA – namely that **SOA governance is a highly complex, constantly demanding, and deeply risky activity carried out not by machines but by humans**.

Hence, any organization doing or considering SOA must make it their first priority to work out how this human work will be managed. Common sense, surely? Yet typical SOA discussions in enterprise IT magazines, Web sites, forums and (particularly) vendor podcasts suggest no way in which to implement such management. The guidance provided to the SOA adopter with regard to governance is a varied assortment of vague recommendations, thrown out in scattergun fashion, and different in almost every source you consult. Listen to almost any presentation on this topic and you will come away more confused, not less!

What is worse, the SOA governance tools currently on offer from software vendors don't help at all with *managing* the work. A moment's thought should be enough to see that effective management of human activity involves a lot more than buying a repository for artifacts (in this case, SOA policies, contracts, etc) and maintaining a to-do list for each item. Rather, managing SOA governance properly involves a deep rethink of how your organization goes about IT.

In fact, SOA technology itself is nothing very new. It is essentially the latest evolution of principles such as encapsulation and abstraction that have been known since the 1960s. **The real innovation of SOA is not technical, but administrative.**

So let's see if we can bring a little order to the current administrative chaos surrounding SOA. In this article, I will show in a way that every business person will understand **what SOA governance is**, and **how to get started**.

The Purpose of SOA

We need to start at the beginning. Why do SOA at all?

For many people, the answer may be this: "Our current IT infrastructure is very expensive to maintain, and the IT department tells us SOA will reduce costs."

Actually, SOA may or may not reduce costs. Depending on your business circumstances, it is quite possible that SOA will *increase* costs. The determining factors are the level of change you make to business operations, and the degree to which you expect your IT infrastructure to reflect such change.

In the end, change is the sole justification for SOA. If you don't need to reflect frequent business changes in corresponding system changes, there is little or no point doing SOA. SOA adds various new forms of overhead – not just by introducing more complex and demanding development techniques, but because for many organizations SOA means new IT infrastructure (new IT infrastructure is not strictly necessary in order to do SOA, but few SOA projects make do without it). These new overheads cost money, and can only be justified by increasing your ability to support business change with IT.

However, change in an SOA environment is deeply risky. Here is your IT infrastructure before SOA:

Like a sailing ship, enterprise IT pre-SOA makes slow progress, is not easy to maneuver, and is vulnerable to external forces. By contrast, here is your IT infrastructure after SOA:

Like a fly-by-wire aircraft, enterprise IT post-SOA is efficient, highly responsive, and carefully secured against external threats.

All very encouraging. However, before getting too optimistic, let's pursue the analogy a bit further.

Suppose that you are the lead engineer for a new fly-by-wire fighter jet. After years of careful engineering, the testing both in simulation and in practice has been successful, and the jet has just taken off on its maiden flight. So far so good. Then you are approached by the generals who commissioned the project, who tell you that they want to add some new functions, remove some existing functions, and change some of the other functions.

You try to explain that the aircraft is an incredibly complex network of interacting systems, and that such changes should have been considered a long time ago, back at the drawing board stage. Nevertheless, the generals are insistent. It's your worst nightmare.

Or is it? Actually, this is a walk in the park compared to what you can expect from SOA. A closer analogy with SOA is this: the generals approach you, asking not only for a complex raft of changes to the aircraft's functions, but also that you make the changes **to the jet currently flying overhead, whilst the jet is in mid-air**. At this point, your safest option is probably to feign a heart attack.

Yet even this apocalyptic scenario is not a realistic comparison with the true complexity of SOA. Engineers in well-established fields such as aerospace can draw on a range of standard techniques for ensuring the safety of the systems they construct, techniques that are not available to service-oriented architects.

For example, a classic technique is known as **dissimilar redundancy**. Dissimilar redundancy involves commissioning a subsystem from a number of different manufacturers, who are not permitted to interact at all. The idea is that any basic flaw in one subsystem implementation does not make it into the other implementations. In operation, all the subsystems are used in parallel – both as checks on each others' operation and to support automated failover.

SOA, at least in its current form, does not possess the luxury of such standard techniques. In particular, a technique such as dissimilar redundancy is not applicable to an SOA infrastructure, in which a key intention is to remove redundancy entirely. More generally, changing to an SOA approach means altering the basic means by which enterprise software applications are constructed.

SOA encourages the system developer not to write (or buy) new programs, but rather to assemble them from pre-existing components. A core mantra of SOA advocates is **re-use**. Post-SOA, enterprise systems (Service-Oriented Business Applications, if you like) are not *programmed* but *configured*, and standard software quality assurance techniques have yet to mature for such systems.

For example, I recently heard a BPM Suite vendor speak proudly of a BPMN process they had implemented that contained 250,000 steps. This represents a vastly complicated piece of business software, so I asked how they had tested it. He replied that testing was unnecessary since their tools did not require the user to write lines of code (just to draw diagrams), and anyway their products contained simulation features if you really wanted to prove that an application worked as expected. He then went on to say how this approach must be OK, since other organizations are using their suite to build safety-critical applications.

This response made me blanch, and still sends shivers down my spine. What sort of world is being constructed using such tools? Whether created via diagrams or as text, software needs testing - and simulation is no replacement for structured, automated, exhaustive, regressive verification. Further, anyone who knows anything about the theory and practice of simulation will recognize immediately that simulation of a flowchart of 250,000 steps is itself a hugely complex and risky exercise.

In general, many key engineering techniques that are mainstream in enterprise IT pre-SOA are no longer applicable in enterprise IT post-SOA: static analysis, automated testing, continuous integration, and so on. Even the most basic quality controls, standard for decades, seem to have been thrown out – for example, a leading SOA consultancy firm claims "metadata configuration differences are too prevalent and too dynamic" to allow creation of a realistic test environment, so new services should be implemented and tested directly in the production environment. Does this scare you as much as it scares me?

By now, any reader that has already embarked on SOA (or committed to doing so) may be wondering whether there is some light at the end of the tunnel. We'll get there, but unfortunately we still haven't plumbed the depths of risk associated with SOA, since there are subtle business issues associated with a move to SOA. In particular, the notion of "a single source of truth" – much touted by SOA adherents as a benefit of moving to SOA - is not really applicable at all in an enterprise environment, at least when it comes to transactional data.

To see this, consider the periodic progress reports made by middle management to senior management. Anyone familiar with the operation of large corporations knows that such reports are not so much *generated* as *created*. The inputs include figures both from a range of operational systems (CRM, order entry, logistics, …) and from a range of back-end systems (ledgers, data marts, data warehouses, …). In any large-scale organization, such systems will all possess overlapping but different versions of transactional data, depending on each system's domain, financial period, validation controls, and so on. The figures are then assembled by the middle manager responsible for the report into a summary that in their view reflects the true reality of business operations – or at least, the reality they are prepared to present to senior management.

This process of report creation, common to most large organizations worldwide, is challenged by SOA and the associated new wave of technologies such as mashups. In the brave new world of SOA, reports presented to senior management can be generated automatically, which means defining a set of business rules by which the information is assembled – rules that do not vary in each period, but apply equally to each report. Is this really what an organization wants? Are management at all levels aware of the deep impact on their working practices?

SOA brings a host of new and varied challenges. It is very easy to get things wrong, and this risk is compounded by the necessity for constant system change – since constant system change is the only thing that justifies SOA in the first place. If you're going to do SOA, you can expect to be doing on-the-fly transformation to a highly complex network of interacting systems, in the absence of standard, systematic control techniques. Controlling such changes is the province of SOA governance. How can governance be done so as to reduce the risk to an acceptable level?

Reducing the Risk of SOA

The key insight necessary to cut the Gordian knot is that **SOA governance is done by humans collaborating**. Services may be run by computer software, but their contracts, service-level agreements, policies, and so on are all run by people – and these people do the work by interacting with each other.

Hence there are two main steps to SOA governance:

1. Work out what is involved – the **issues** associated with SOA governance

2. Work out how to deal with these issues – i.e., design and implement the corresponding **collaborative human work processes**.

With regard to step 1, a simple way to understand the issues of SOA governance is to recognize that they correspond directly to the high-level issues of general organizational governance. In other words, one can use standard executive management techniques such as Balanced Scorecard to understand and represent them, as below:

Balanced Scorecard for SOA Governance

For more details, see http://human-interaction-management.info

Financial	Customer

• Service lifecycle funding
  • Creation
  • Maintenance
  • Retirement
• Shared service cost allocation
  • Fixed price
  • According to usage
  • According to benefit
• Correlation with incentive schemes
  • For teams
  • For organizations
  • For individuals
	• Regulatory conformance
	  • Statutory
	  • Industry
	  • Organizational
	• Implementation and monitoring of Service Level Agreements
	  • For individual services
	  • For composite services
	  • For composite applications
	• Service consistency
	  • Interoperability
	  • Interface standards
	  • Redundancy

Internal Business Processes	**Learning & Growth**

• Data Management
  • Consolidation, harmonization, and centralization of reference data (Master Data Management)
  • Lifecycle management of transactional data (Create, Read, Update, Delete)
  • Reconciliation and usage of transactional data in management and financial Reporting
• Development
  • Architectural principles
  • Development practices
  • Technologies and tools
• Engineering
  • Safety analysis, testing and review
  • Administration and security instrumentation
  • Fault and policy exception management
	• Knowledge capture
	  • SOA in general
	  • Domain specific
	  • Organization specific
	• Knowledge dissemination
	  • Internal
	  • To partners
	  • To market
	• Skill maturity
	  • Technical
	  • Managerial
	  • Tools

As with the Balanced Scorecard approach generally, this picture does not give us a checklist of actions to carry out. Rather, it tells us what we need to be thinking about with respect to SOA governance. The next step is to decide how to deal with the issues.

Since you are reading this article, which is published in a Business Process Trends column, I expect that you accept the following assertion: better management of work means better *process* management of that work. In this case, however, we cannot apply mainstream workflow/BPM techniques and tools. Mainstream workflow/BPM can deal with interactions between computer systems, and even interactions between humans and computer systems – it cannot deal with interactions between humans and humans.

Trying to describe SOA governance processes using BPMN, for example, would not only be a frustrating exercise but ultimately a pointless one. The people involved are unlikely to stick to the steps prescribed in a pre-defined flowchart – and were they to try, it would be to the detriment of the process outcome. When collaborating, humans need to use their judgment on how best to proceed, since each case is different. Such flexible, innovative, adaptive behavior is what humans are extremely good at (and machines are extremely bad at).

Further, SOA governance brings complex boundary issues. Typically, a SOA initiative spans teams, departments, divisions – it may even span organizations. Hence, no-one "owns" the processes involved in its governance. A truly decentralized approach is required in order to design and implement such processes, and this is not to be found in mainstream workflow/BPM.

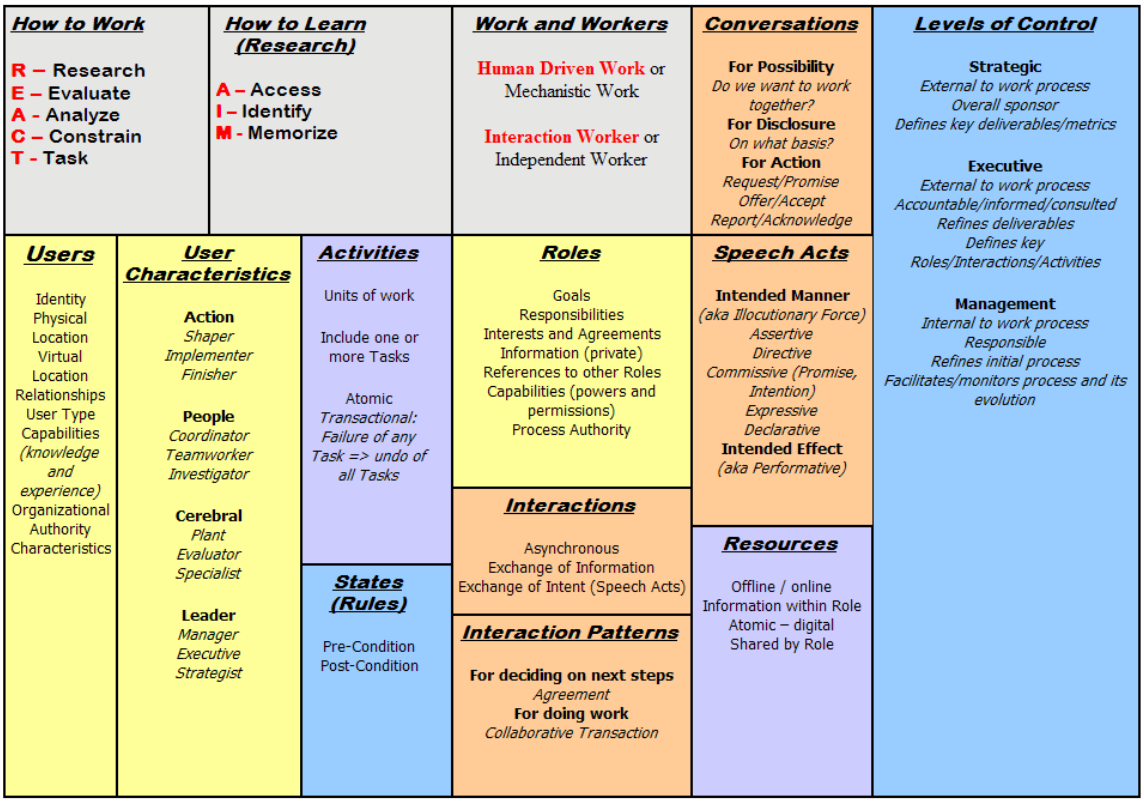
The best that BPMN, XPDL, et al can offer for SOA governance is a generalized step such as "at this point, people interact to do XYZ". This kind of hand-waving is not going to help anyone design, implement and improve collaborative human work processes.

Governing Governance

What is needed to manage collaborative, decentralized human work is an approach based on deep understanding of human behavior, in which "just enough" control is applied – enough control to standardize the processes, and help them run efficiently, but not so much control as to prevent the people involved from working to their maximum capability. The definition of such processes is the province of Human Interaction Management (HIM), and their implementation with software tools is the province of the Human Interaction Management System (HIMS).

A full description of HIM is far beyond the scope of this article. In brief, processes defined via HIM are intended for human, rather than machine, actors. Hence, they are based on human **goals, responsibilities and commitments**. Here are the elements of such processes:

Human Interaction Management (HIM) – Quick Reference Card



Not every aspect of HIM is required in order to model each process. However, taken as a whole, HIM provides a framework that can be used by organizations adopting SOA to describe and implement SOA governance processes.

Use of HIM is facilitated by a HIMS: a new type of Business Process Management System that supports collaborative human work rather than routine, mechanistic activities. A HIMS is not based on a flowchart-like notation such as BPMN, in which work is defined like a computer program: i.e., organized into (possibly parallel) task sequences. Rather, a HIMS supports the 5 basic principles of HIM:

1. Connection visibility - to work with people, you need to know who they are, what they can do, and what their responsibilities are as opposed to yours.

2. Structured messaging - if people are to manage their interactions with others better, their communications must be structured and goal-directed.

3. Support for mental work - organizations must learn to manage the time and mental effort their staff invest in researching, comparing, considering, deciding, and generally turning information into knowledge and ideas.

4. Supportive rather than prescriptive activity management - humans may not sequence their activities in the manner of a software program, but there is always structure to human work, which must be understood and institutionalized so that it can be managed and improved.

5. Processes change processes - human activities are concerned often with solving problems, or making something happen. Such activities routinely start in the same fashion - by establishing a way of proceeding. Before you can design your new widget, or develop your marketing plan, you need to work out how you are going to do so - which methodology to use, which tools are required, which people should be consulted, and so on. In other words, process definition is an intrinsic part of the process itself. Further, this is not a one-time thing - it happens continually throughout the life of the process.

In practice, since a HIMS must be able to support collaboration that spans various forms of organizational boundary, the software cannot depend on a server component. Though a HIMS may be able to run on a server if necessary (for instance, to support audit trail recording), it is typically intended for desktop use. A HIMS communicates **peer-to-peer** both with other instances of itself and with standard communication programs such as email clients in order to synchronize, monitor and support the work of the people involved.

To the average end user, a HIMS appears as a lightweight layer on top of their existing computing resources – other desktop programs, network files, enterprise applications, and Web 2.0 tools of various kinds. The HIMS is effectively a **more intelligent desktop interface**, enhancing the benefit of every kind of computing resource from emails to databases, by providing a rich process context via which such resources are accessed.

Selling SOA to the Business

HIM does more than reduce the risk of SOA. HIM also gives SOA evangelists a way to sell SOA to the business.

I have described how governance needs HIM. Well, so does the organization generally. In today's wired, globalized world, most routine work is being standardized, outsourced and automated. In other words, all that organizations have left to compete on is the interactive knowledge work (what I call "interaction work") that is left over. HIM provides a means of doing such work more efficiently, and managing it better.

HIM also offers the opportunity to overcome the inherently fickle nature of Internet trading by **binding your customers into shared, long-lived, collaborative processes**. This is the route towards what Pine and Gilmore (authors of "The Experience Economy") call the "fifth economic offering": **transformations**.

However, large-scale HIM adoption, like large-scale adoption of any other new operational approach to business, can be hard to fund and hard to implement. SOA evangelists can use this apparent obstacle to get funding for their own initiatives. Don't sell the directors SOA at all! Sell them HIM, and pitch SOA governance as the perfect pilot project.

Further, return on investment for SOA can be very hard to assess. If you use HIM techniques for governance, and support the human-driven processes thus defined via a HIMS, you can increase the value gained from SOA - since a HIMS can call services to automate key parts of a collaborative human process. Use of a HIMS in itself offers the chance to gain increased benefit from a service-oriented infrastructure.

TAKE AWAY: Moving to a Service Culture

Even among SOA experts with deep knowledge of SOA techniques, there seems currently to be very limited understanding of the need for **process control in SOA governance**. This is probably because such processes cannot be handled via familiar mainstream workflow/BPM. Rather, such processes require leading-edge HIM techniques for their design, and a new form of enterprise software for their implementation – the HIMS.

Certainly none of the vendors in the governance space have yet incorporated process support into their tools. The next step for SOA governance is to recognize the need for HIM – and the next step for SOA governance software is to integrate the functions of a HIMS such as HumanEdj into an existing, repository-based offering. The first consultants and the first software vendors in the SOA space to take these ideas on board will have the market to themselves.

However, I suspect that understanding of the need for process control in SOA governance will only truly become widespread once there have been a few public disasters - and these cannot be far off. Soon, early adopters of SOA will attempt to use their brand new system environment in the way promised by its original advocates - namely, to make rapid changes to align IT with new business requirements. Without effective process controls, reconfiguration of a hugely complex network of interacting services is bound to bring problems.

Further, I see no reason why the impact of such problems should be limited. Rather, they are quite likely to break some organizations entirely. It may be that for now, SOA change problems have been covered up - but large-scale operational or financial disaster is not so easy to hide.

So why wait until it happens to you? Would it not be better to bring some order to the chaos? In other words, **take responsibility for your SOA**. Do not treat SOA governance as if you were going out to dinner with colleagues - each choosing a few items from the menu on a whim. Rather, treat SOA governance as if you were all working together in the kitchen - collaborating with just enough structure to ensure efficiency, and adjusting your work patterns from day to day as necessary to meet requirements.

SOA evangelists claim that SOA will finally integrate IT with the business. If so, business people will no longer just be *consumers* of IT, like guests in a restaurant. They will need to get into the kitchen with the chefs, put on a toque and gain hands-on familiarity with the true complexities of service *provision*. SOA requires both business and IT to share responsibility for enterprise infrastructure.

This doesn't mean that every business person needs to learn how to write WSDL, or even what such acronyms mean. However, business people in a post-SOA environment do need to understand the human-driven processes involved in SOA governance, accept responsibility for certain aspects of such processes, and commit to playing their part in maintaining the infrastructure.

A key corollary of SOA is that the business-IT relationship changes at the most fundamental level. No longer is IT simply a supplier to the business consumer. The need for constant governance activity in an SOA environment forces the relationship to become two-way. Moving to SOA means that business people, as well as IT people, must start offering services.

Author

Keith Harrison-Broninski is a consultant, writer, researcher, and software developer working at the forefront of the IT and business worlds. He is author of the landmark book **"Human Interactions: The Heart And Soul Of Business Process Management"** (Meghan-Kiffer Press, 2005, www.mkpress.com/hi), described by a BP Trends review as *"the overarching framework for 21st century business technology,"* and by the BPM Group as *"a must read for Process Professionals and Systems Analysts alike."* Keith is also a contributing "thought leader" to the BPM Group book "In Search Of BPM Excellence" (Meghan-Kiffer Press, 2005, www.mkpress.com/bpmg.html).

Along with his research and consulting work, Keith is the CTO of Role Modellers Ltd, whose company mission is to develop understanding and support of collaborative human work processes across industry, a field that Keith has pioneered with his work on Human Interaction Management (see http://human-interaction-management.info). Role Modellers' free software, HumanEdj, is the reference implementation of a Human Interaction Management System.

Find out more about Keith and his work at http://keith.harrison-broninski.info.