Towards BPM 2.0

Hans Wierenga

Few concepts have had more impact on today's organizations than BPM: the idea that each business process should be perceived, designed, implemented, supported by IT and managed as a whole, in which all the activities of which it is comprised work together to achieve a result that is of value to the customer. The power of this idea has enabled many organizations to vastly improve their performance by eliminating unproductive activities, reducing delays and allocating resources more efficiently.

Throughout its history, BPM has been practiced in an expert-driven, projectwise manner, in which the people who execute the current process are regarded as part of the problem rather than as part of the solution. By analogy with the one-way communication patterns that are now collectively labelled Web 1.0, this could be termed BPM 1.0. In this paper we shall examine what is necessary in order to move on from established ways of performing BPM towards BPM approaches that deserve the 2.0 predicate.

What does 2.0 stand for?

The predicate 2.0 has been applied in very many terms in addition to the term it originated in, Web 2.0. Examples include Library 2.0, Social Work 2.0, Enterprise 2.0, PR 2.0, Classroom 2.0, Publishing 2.0, Medicine 2.0, Telco 2.0, Travel 2.0 and Government 2.0. In all of these usages, the intention is to indicate a change from top-down, one-way communication between source and consumer to multidirectional communication between all participants, from projectwise development to continuous development, and from individual users to user communities. 2.0 as a predicate refers to new patterns of information exchange.

Unfortunately, 2.0 is such an attractive predicate that it is exceedingly tempting to apply it to any development that is headed in the right general direction, even though it doesn't go far enough. It is our considered opinion that the term 'BPM 2.0' has been applied in this fashion by Silver (Silver, 2006), Ghalimi (Ghalimi, 2008) and others. True, they describe developments that move away from one way communication from process designers to the rest of the world and towards a world in which less well trained process experts can take over part of the process design effort, but that is still a long way away from a pattern in which ordinary process users substantially affect process designs. Further, in this use of the predicate 2.0, there is no movement away from projectwise development, and only the faint beginnings of a concept of community.

Note that the predicate 2.0 is not a statement about the BPM maturity level of the organization to which it is applied. Just as there are many websites that are fairly chaotic and ad hoc, but nevertheless demonstrate the Web 2.0 characteristics of multidirectional communication, continuous development and user communities, so an organization can have BPM with these characteristics and yet be fairly inconsistent in the way it operates. 2.0 differs from 1.0 primarily in the source of change; change is organic rather than externally induced. One could visualize the difference by imagining that 2.0 is a snowball that gets larger and larger by absorbing more of its environment as it rolls downhill, whereas 1.0 is a balloon that inflates because an external agent pumps more air into it.

Is BPM 2.0 possible?

Are Silver, Ghalimi and consorts correct in their apparent assessment that process design and management are by their very nature the domain of experts, thereby reducing BPM 2.0 to a pipe dream? In this article, we will articulate and defend the position that they are not, and that process design and management can be made into a community affair, carried out on a continuous basis by many participants including those who are not schooled in it, thereby making BPM 2.0 viable. We shall argue that the perceived need for process design experts is rooted in BPM 1.0's conception of what a process is, and present a new concept of process - one that

makes process designs simpler and more effective, easier to change and implement, easier to measure and benchmark, more robust, more customer friendly and more governable.

In passing, we shall discuss the consequences of the loss in process improvement repertoire which the move from BPM 1.0 to 2.0 entails. In particular, we shall examine how the process improvement measures on which have made BPM 1.0 a successful approach can be generated when BPM 2.0 is applied.

When reading this article, the reader should bear in mind that none of the ingredients of BPM 2.0 are in any way new. Each of the ingredients as presented in this article is tried and true. Indeed, they delineate the way in which market economies have driven process improvement for centuries. They are also applied in current BPM practice, although they are mostly not mainstream. It is the recipe that distinguishes BPM 2.0 from earlier approaches, not the ingredients.

The Concept of Process

Processes are customer processes

For change from within to flourish, it must be possible for all process participants to contribute. That requires that two conditions be fulfilled. Firstly, there must be a means by which each process participant can identify, assess and implement improvements to the process, whilst remaining within their own sphere of responsibility and competence. Secondly, there must be a way of allocating responsibility in such a manner that for each and every process improvement in our repertoire there is some agent with both the motivation and the means to apply it. Neither of these conditions is on offer with BPM 1.0. Only participants who can oversee the entire process are equipped to identify process improvements and assess their usefulness. Only those rare agents who happen to be responsible for the entire process and every little part of it have both the motivation and the means to apply the full process improvement repertoire.

In this paper we shall argue that a paradigm in which all processes are customer processes does satisfy the above conditions and must therefore be regarded as an essential element of BPM 2.0. A customer process starts with a request to do something and ending with the delivery of the product to the requestor. In BPM 1.0 that is not necessarily so, the request is not an essential part of the process. The process concept on which the BPM 1.0 paradigm arguably is rooted, as worded by Michael Hammer (Hammer, 1993) , contains no reference to it. In his definition – “a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer” – even just the final step of a customer process is of itself also a process. In this respect more recent BPM 1.0 process conceptions, such as the IGOE (Input, Guides, Outputs and Enablers) framework, as described, for example, by Kathy Long (Long, 2009), are no better. And you still see people taking just part of a customer process, optimizing it, and calling what they do BPM. For example, the optimizations starts from the point that an order has been accepted, rather than from the point that the customer issues it. More generally, process improvement projects are wont to consider only those parts of a process that happen to fall under the political influence of the project sponsor.

The restriction to customer processes is more than just an issue of scope. Those who optimize just part of a customer process are not – except perhaps by accident – busy optimizing the process from the point of view of the customer. To optimize the customer experience, you have to take the process from request to delivery. If you shut your eyes to the customer, you may improve your own operation, but you ignore the fact that the customer is the ultimate judge of your performance and that improving the customer experience is therefore the ultimate point of the exercise. This ignorance may very well result in such things as increasing the efficiency at the cost of reducing the effectiveness of the customer process.

Supply chains are hierarchies of customer processes

One consequence of this restriction is that BPM 2.0 does not recognize supply chains as processes, except in the somewhat extreme case in which absolutely everything is done in direct

response to a customer request, and therefore nothing at all is kept in stock in order to be able to cope with future customer requests. That does not mean that BPM 2.0 cannot represent and support supply chains. Any supply chain can be decomposed into a hierarchy of customer processes; in other words, it can be represented in such a way that every activity is carried out because some agent requests a provider to carry it out. At the top of the hierarchy is the customer, and one level below is the provider that accepts the customer's order and ultimately delivers it. Directly below this provider are all the subproviders which directly serve the provider with intermediate products and services. Each of these subproviders in turn may have providers who are subproviders to them, and so on until we come to the providers of raw materials. This is visualised in Figure 1: a supply chain as customer process hierarchy. If you turn the hierarchy on its side, you get a fan of customer processes which converge onto the customer process in which the final product is ordered and delivered. In this fan, the supply chain consists flow of the dominant raw material from provider to provider, with a progressive addition of value culminating in the delivery of the end product to the customer.

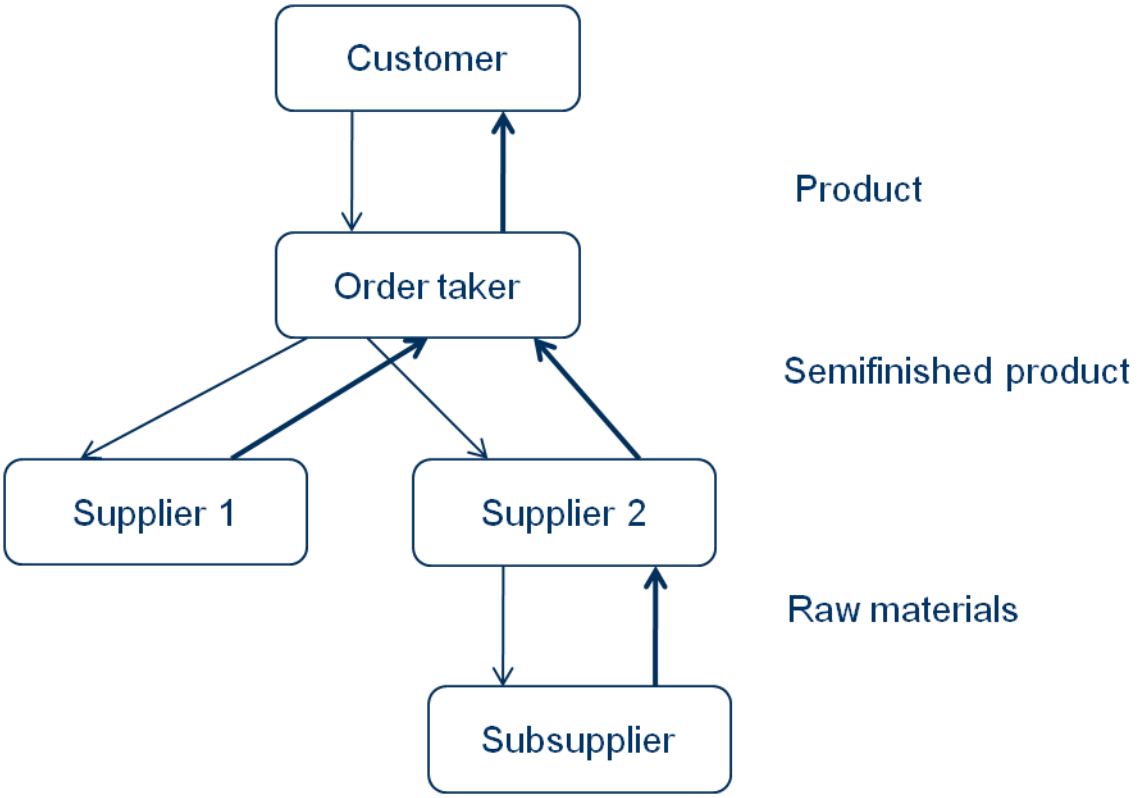

Figure 1: a supply chain as customer process hierarchy

In order to optimize a supply chain, it is imperative that the ultimate customer process is recognized as such, and it is useful that each component customer process is optimized using appropriate BPM techniques. A supply chain which includes a suboptimal customer process is itself suboptimal, because it is possible to improve that customer process without impinging on the other customer processes within the chain, thereby improving the chain. Note that it doesn't work the other way around: a supply chain consisting of only optimal customer processes is not necessarily optimal. Only when supply chains are recognized as being different from processes and supply chain optimization is recognized as a science in itself is it possible to optimize them. However, this optimization is facilitated by applying a BPM 2.0 perspective. In particular, it allows the "make or buy?" decision to be made at any level in the customer process hierarchy; in principle any intermediate product can be outsourced.

Each and Every Interaction Adds Value to the Requestor

The added value reflex

In the BPM 2.0 paradigm, because each and every interaction originating from the customer is a customer process, it is almost unthinkable that a requestor asks for something that is of no value to him. Of course, it doesn't make much sense in BPM 1.0 either, but in BPM 1.0 it is not a reflex to ask "What is this information exchange doing for the requestor?" as it is with BPM 2.0. Consider, for example, a customer process in which a customer reserves a hotel room long in advance of his holiday. Somebody designing the process from a BPM 2.0 perspective must ask what value it adds for a customer and under what circumstances it does so. As a result, the designer should at least not require reservations in the low season, or alternatively make it worthwhile for the customer by giving him a lower room rate.

Unfortunately, many BPM 1.0 logistical processes are designed with steps in which agents reserve resources for subsequent use within their customer process, but without rewarding them for doing so. As long as the customers cannot switch to other suppliers, that is not necessarily a problem for the provider. But otherwise it is essential to ask the critical question: "Suppose the customer had a choice between our process and an otherwise equivalent one in which placing a reservation was unnecessary, which would he choose?". Applying the BPM 2.0 paradigm tends to make your processes more customer-friendly.

BPM 2.0 goes further than Six Sigma in this respect. True, Six Sigma is also very much concerned that each and every activity adds value for the customer. However, it fixes the scope within which that value can be added at the outset and therefore tends to miss new sources of added value.

Added value drives process improvement

It may be objected that the focus on added value results in the intermediate products being fixed early on in process improvement trajectories, thereby limiting the process improvement options. Aren't the intermediate products artifacts of process design, which could be rendered unnecessary by an improved design? For example, for hundreds of years the business world thought that invoices were necessary in order for suppliers to get paid for their products, but we now know how to make purchasing processes work more simply and effectively without them. Would we have identified this process improvement using the BPM 2.0 paradigm? The answer to that question is: "Yes". By applying the BPM 2.0 paradigm, the process designer is encouraged to ask what value the invoice adds and to consider alternative means of achieving that value.

In general, for every possible process improvement, there is always a process in the BPM 2.0 hierarchy for which asking the right questions leads to identification of the improvement possibility. At that level, the lower intermediate products are not fixed, or, to put it more precisely, even in the picture. BPM 2.0 doesn't mean less oversight of the process, but rather oversight at every level.

The added value perspective changes our notion of process

Once the perspective has shifted from the BPM 1.0 question "How can we transform these inputs into the desired output?" to the BPM 2.0 question "How do we add value for the customer?" the whole concept of process undergoes a radical revision.

The revision affects our notion as to when the process starts. We don't need to wait for the customer to place an order, because it is possible to start adding value before that happens. For example, the customer may be helped by information about our products. But then we must remember what we told him, so that when he places the order he is not surprised with the goods or services we deliver or the prices we charge. In other words, there must be continuity across the entire customer process, in which each and every utterance, regardless of time, medium or agent must be remembered and honored. That does not sit well with BPM 1.0, because it has no place for any concept of continuity apart from the continuity that arises when activities are carried out in sequence.

The added value perspective also causes us to revise our conception as to what constitutes an instance of a process. BPM 1.0 is strongly biased towards processes in which each activity is carried out once for a process instance, but we must be prepared to cope with processes in which the various activities are carried out at differing levels of aggregation. Even within what is normally thought to be a single customer process, differences in intention and the degree in which the customer request has been specified often mean that different subproducts are aggregated differently. For example, a customer order process may go through distinct stages, as follows:

- Sales push: the provider makes known that he is capable of providing particular goods or services.
- Sales pull: the customer identifies the provider as a potential supplier of the goods or services he wants
- Order negotiation: the customer and the provider reach agreement on what is to be delivered and which terms and conditions apply
- Order placement: the customer orders specific goods or services to be delivered
- Order fulfillment: the provider delivers the goods or services to the customer
- Order payment: the customer remunerates the provider for the order fulfillment.

It is very restrictive to insist that each and every customer order process invokes a process sequence consisting of just one instance of each of these stages. Often there is one negotiation stage instance, resulting in an agreement which is applied in many order placements. It is quite usual to perform just one payment for many fulfilled orders, arising from many negotiations. The interconnection of stage instances is just as likely to be a mesh as a hierarchy. Nevertheless, the organization must remember the results of one stage for use in subsequent stages. If the interconnection is a mesh, it just won't work to make the process the carrier of that memory. If this is the case even for a straightforward process, we must conclude that a specification of a process purely in process terms misses essential aspects, for which some notion of shared memory is required in order to do them justice. The shared memory takes the form of a register of all relevant instances, around which the individual stages are placed, as pictured for the customer order stage model in Figure 2: Customer order stage model.

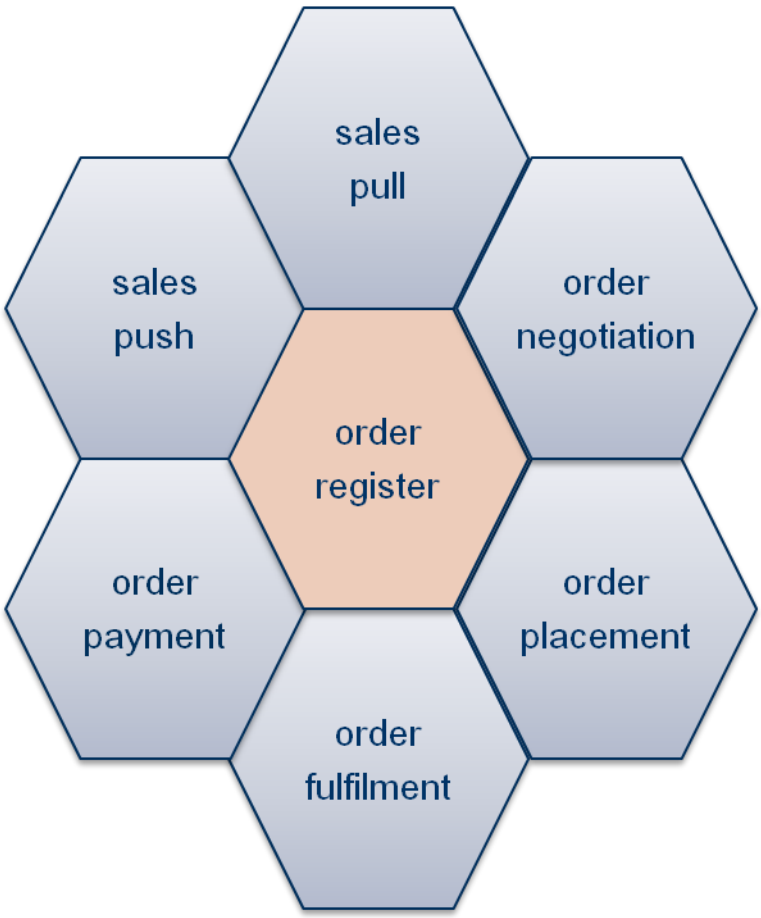

Figure 2: Customer order stage model

The above example concerned the continuity which is required for the execution of customer orders. Similar considerations apply to maintaining the continuity of the organisation's memory with respect to its products, its customers and each of its resources.

Life cycles

The stipulation that each interaction adds value to the requestor rules out life cycles as candidate customer processes. A life cycle may start with a customer order, but it normally continues after the order has been delivered. The part after order delivery does not belong to the same customer process as the rest, because it has another requestor. Of course, it is closely related to the customer order part of the life cycle, but the motivation to optimise a lifecycle as a whole is generally primarily concerned with internal issues.

IT Aligns with the Customer Process Hierarchy

New demands on IT

BPM 2.0 requires new forms of IT support in order to thrive, because it is viable only if participants have genuine freedom to change their processes. In practice this freedom is typically limited by both the flexibility of the IT as such and the way in which it is bundled.

The flexibility of the IT as such is often seriously deficient. A process design can be changed in days, people can be trained to apply the design within a month, but it often takes a year or more to change the IT. Of course, not all process changes require substantial changes to the IT, but

when the process logic has been incorporated into the IT the room to change the process without changing the IT is severely constrained. In such cases there is the additional problem of what to do with process instances that have been started with old process designs once the IT has been changed to support a new process design.

The way in which IT is bundled causes problems too. Shifting alliances at a business level do not sit well with alliances which have been baked in concrete within the IT. BPM 2.0 cannot live up to its promises if the ability to change and mix providers is built into the process design but the IT can only support a connection which is specific to the current provider. The ability to shift from an internal to an external provider is often hampered by internal IT which implements the connection by means of an integrated system to which no external provider has access.

Service oriented architecture meets the demands

Fortunately, IT solutions that meet BPM 2.0's demands are becoming widely available in the form of services which have been designed to interoperate within a Service Oriented Architecture (SOA). SOA is an approach to organizing information processing which consists of describing all interactions between information systems in terms of services, in which a requestor asks an agent – the provider – for something to be done, and the provider ensures that it gets done and delivers a response to the requestor. BPM 2.0 applies this way of thinking is applied at a business level, in order to describe interactions between organizations, and at a functional level, to describe how the activities of which business processes are comprised interact, whereas SOA applies it at the level of information systems, in order to describe how systems and parts of systems interact. In other words, BPM 2.0 and SOA employ the same way of thinking, but at different levels of abstraction. At each of these levels the principle is the same: how the provider does what needs to be done is no concern of the requestor, not even whether it is done completely automatically, completely manually or something in between. Nor is the requestor concerned whether the provider delegates part or even all of the work to others. All the requestor needs to agree on with this agent is how the request and the response should be formulated, and what the effect of the service is. Each service can be regarded as a customer process which starts with the request, includes as activities all actions which are carried out as part of the service, and ends with the delivery of the response.

BPM and SOA are widely touted as complementary disciplines: BPM is easier if applied using SOA and SOA adds more value in a BPM context. For example, Behara maintains: "SOA can exist without BPM, and BPM has flourished without a firm understanding of SOA. The combination of SOA and BPM is more powerful than either is alone." (Behara, 2006). True as that may be, it ignores the fact that we need BPM 2.0 in order to achieve substantial synergy between BPM and SOA. BPM 1.0 and SOA embody two different ways of thinking about what it is that the organization does, and these ways of thinking lead to different designs. Or, to put it more precisely, SOA – and BPM 2.0 - form a more rigorous approach than BPM 1.0, and consign to the scrapheap many designs that are quite acceptable in BPM 1.0.

Although BPM 2.0 has a high reliance on SOA in order to be viable, that does not make BPM 2.0 a technological concept. BPM 2.0 is all about new patterns of information exchange, and does not in any way consist of or comprise SOA or any other enabling technology. Where the 2.0 pattern can exist only by virtue of its enablers, it is natural and perhaps even productive to identify them, but the ever present danger in doing so is that the pattern is limited to that which the enablers can support.

Service automation

It is very easy to translate customer processes to IT services. Services that have been translated in this way can be connected very simply to BPMS processes. In this way, the synergy between BPM and SOA, as identified by authors such as Behara (see above) can be achieved. Existing services can be very easily exploited in new or changed BPMS processes.

Working from BPM process designs in order to develop services is generally not such a good idea. SOA experts have a saying: "a BPM makes bad SOA, SOA makes great BPM". Just google the quote in order to find how often this mantra is repeated. The problem is that identifying

services by having them drop out of a process design totally ignores the need for services with memory, and results in many services each performing only a small task in isolation.

A key feature of the SOA paradigm is that services are designed to be usable everywhere where the information they expose is needed. When SOA is applied as a way of thinking and not just as a technology, it becomes normal to develop services exposing each object which the organization collects information about.

Services remember what they need to

Within a service oriented architecture, services are more than request-response pairs, for they also remember the things that the provider needs to know in order to service future requests. In fact, the actual request-response pairs are, strictly speaking, not the service itself but *methods* which the service exposes to its environment, and the service is a whole consisting of the methods and the memory.

In a BPM 2.0 context the memory functionality of the services is vital, because the process itself is not a suitable vehicle for remembering any information that can be acquired or changed independently of the process, whether between steps or between instances. Consider, for example, a situation in which many customer order process instances use the same product specifications and prices, and changes to these specifications and prices need to be applied everywhere as soon as they are approved. In such a case, prompt application of the changes would be frustrated by process designs in which the process instance make a copy of this information early in the process and subsequently use the copy instead of the source. Whether the copy is stored digitally or on paper documents which, bundled together in a file, travel from desk to desk as the embodiment of the process instance, makes no real difference. A process is acceptable as a locus of memory only for those facts that have no relevance outside of the process instance in which they are registered.

Agreements between Requestor and Provider are Crucial

The need for agreements

BPM 1.0 is not strong on agreements between supplier and customer. They play only a minor role in IGOE. In Hammer's definition of a business process, the concept of an agreement between the supplier and the customer is lacking. The output must be of value to the customer, but need not necessarily be what he ordered. Until the final activity has concluded, it is not strictly necessary that the customer has any idea of what he is going to get, or even that the organization has a clue as to what it is going to produce.

Such vagueness is utterly foreign to customer processes. Each and every instance of such a process represents an agreement between the requestor and the provider. The requestor asks for something and the provider provides it. That may be just information – the answer to some question – but it might also involve changes to some material domain. In such cases the agreement is that the provider is going to make some material changes to the domain with which both requestor and provider interact (for example by replacing a flat tyre), communicate the results to the requestor and be recompensed by the requestor for his efforts. But whatever the customer process delivers, there is always an implicit or explicit agreement as to what the results of the customer process will be, and any change to that agreement requires the assent of both requestor and provider.

The customer process concept of all actions being based on agreements between the customer and supplier is an essential part of the BPM 2.0 pattern. That is because BPM 2.0 depends on the customer and the supplier being on an equal footing. The customer participates in 2.0, and that is not possible if he is regarded as an entity on the periphery of the process. In BPM 1.0 a process is something we do *for* the customer, in BPM 2.0 a process is something we do *together with* the customer.

It is helpful to regard a customer process as being first and foremost an agreement, and only in second instance a process. The agreement covers not only the result of the customer process, but also the conditions under which that result may be expected. In BPM 2.0, a customer process that is not governed by a Service level agreement (SLA) doesn't make sense.

Interrupts and steps

Once you see a customer process as an agreement, it is possible to adequately design mechanisms and steps to handle interrupts to the process. To do this, it is imperative that a distinction is made between events which result in termination of the agreement between provider and requestor, those which modify the conditions under which it is agreed to be carried out – for example, a delay – and those which leave the agreement intact. Each of these categories requires a different treatment. An interrupt that aborts the agreement - such as the customer going bankrupt or dying, or the occurrence of a major natural disaster – requires not only the cessation of the process but often also the reversal of already completed steps. An interrupt that changes the agreement requires that this be done explicitly, and may necessitate rescheduling. An interrupt that helps proceed the process – such as the arrival of new information – may change the flow of the process.

In the BPM 2.0 paradigm, it is natural to define a process as a set of steps, each with its own preconditions. The task of the BPMS is to determine whether the preconditions are satisfied and, if so, to present it to the provider for execution. This way of designing a process results in more robust and flexible process designs. It also makes it easier to change the process, because it allows those who are concerned with only a single step to optimize that step without having to worry about the rest of the process. This is a vital element of BPM 2.0; if the only people who can participate in optimizing a process are those who can see the whole process, almost all of the people involved in executing the process are excluded from participating.

Why preconditions improve processes

Whoever thinks seriously about these issues cannot avoid asking him or herself whether a view of a process as being composed of threads– each a sequences of steps – together with branches and joins, is not rather too limiting. Van Aalst c.s. (Aalst, Hofstede, Kiepuszewski, & Barros, 2002) empirically investigated the complexities that this view of processes leads to, and identified 30 different patterns in which discrete steps can affect each other's execution. Interestingly enough, almost all of this complexity vanishes when processes are considered to be composed of steps, each of which may be executed only when specific preconditions are satisfied and must be reversed or compensated for whenever certain other conditions are satisfied. In this view of processes, sequential execution of steps may (or may not) arise as a consequence of the preconditions, but the sequence is neither a structural nor a limiting aspect of the process design.

How the Provider Executes the Customer Process is up to the Provider

Functional freedom

One of the most liberating rules within the BPM 2.0 paradigm is that a provider is completely free to decide how he produces the response that the requestor has asked for. That way, the provider can, in principle, change his internal processes and information systems whenever and however he wants, providing only that the external interface – the request, the changes to the shared world of requestor and provider, and the response – remains intact. The reverse side of the coin is that the requestor can switch to another provider at will, providing that the new provider can support the same external interface.

In the BPM 1.0 paradigm this rule is regularly trampled upon. For example, it is tacitly and sometimes explicitly assumed that a single Business Process Management System (BPMS), or, worse, a single instance of the BPMS, should run the entire process, across all actors except the ultimate customer. This is a very limiting assumption, for many reasons. Not all of the providers will belong to the same organization, and some providers will support many requestors, each with their own BPMS choices. Not all parts of the process will lend themselves to the same level of

automation via a BPMS. Parts of the process may require a radically different style of BPMS – for example, case management. Often the process management is so closely tied to the process content that it is unwise to separate them by putting the process management in a BPMS, certainly if the process content is supported using packaged software. For all of these reasons, the single-BPMS assumption effectively limits the scope of process automation to at most a single organization and often to only a business function.

Process management freedom

The growing dissatisfaction with the 'single BPMS' ideal has led to BPM experts making a distinction between two levels of process management intimacy. The classical style of BPMS has been labelled 'orchestration', and a more hands-off style, in which the BPMS issues chunks of work to independent agents without concerning itself how they should be synchronized has been labelled 'choreography'. With choreography, it is the responsibility of the agents to ensure their actions are synchronized with those of the other participants, just as it is the responsibility of dancers in a choreographed ballet to ensure that their motions are synchronized with those of the other dancers. From a BPM 1.0 point of view that is undesirable, because the conception of the process as being an entirety that must be managed as a whole is diminished. From a practical view it is undesirable because the art of synchronizing choreographed agents is underdeveloped and is widely perceived as being risky.

It should not be a surprise that BPM 2.0 and choreography go hand in hand. Choreography allows a process to be executed by agents who decide for themselves how they are going to deliver what has been agreed. And defining process logic in terms of preconditions allows the synchronization between autonomous actors to be specified and implemented. BPM 2.0 defines a way of thinking and working together that allows choreography to be implemented without risk of unsound processes. With BPM 2.0 there is no need for orchestration involving multiple autonomous agents, because no single process will span multiple business domains and therefore require something spanning both in order to manage it. With BPM 2.0 the fault line between two business domains is always bridged by a customer process, and never by the transition to a subsequent step.

The BPM 2.0 paradigm enables each service provider to implement process management in whatever way he sees fit, providing only that a standard service call to inform the requestor of the current state of service execution is supported. Because the SOA response to this call is self-describing, the requestor does not need to build any knowledge of the provider's process into his own information systems in order to be able to make sense of the response.

The dynamics of freedom

The notion that the service provider determines how to execute the service can only work if the service provider is capable of identifying and implementing improvements. That is a natural assumption, because, in the ordinary course of things, the service provider has expert knowledge of his service, coupled with total control over the way in which it is executed. He sees every day how the service works, is confronted every day with how customers respond to the service, and is continually in a position to experiment with improvements. The changes he chooses to implement are more likely to be persistent, because they are supported by the people who have to apply them.

But what if the service provider is blind to potential improvements? Aren't there many improvements that can be adequately identified only by an expert process designer, overseeing the entire process? For example, isn't intervention by an expert process designer necessary to identify changes in the allocation of tasks to human resources, given that this allocation is typically subject to prejudices, turf fights and fears of reductions in the quality or quantity of work? After all, a process design in which a human resource does not need to switch on to a process instance more than necessary will never be developed if nobody sees or is willing to accept that distinct tasks can be performed by the same resource, and it takes an unbiased and perhaps daring person to defy the taboos and push through the change. There is more than a grain of truth in these objections, but this doesn't necessitate an expert process designer, but only an

expert process design facilitator. A good facilitator harnesses the creative energy within the organization, and transforms into a challenge the threat that underperforming service providers are liable to being replaced.

In BPM 1.0 it is often assumed that the people who carry out the process are incapable of improving it significantly, even when assisted by an expert process design facilitator. What this amounts to is the hypothesis that they are zombies. Interestingly enough, this zombie hypothesis appears to be built into many BPM 1.0 approaches, as is evidenced by the fact that they do not include any steps in order to identify how the people that execute the process may be mobilized to improve it.

Request data specify the request, and no more than that

The rule that a request contains only the data that is needed to specify the request is a direct consequence of the rule that the way a provider executes a service request is up to him. Information that the provider needs in order to carry out the request, but is not part of the customer order, may need to be gathered, but how and when that happens is the provider's problem, not the customer's. The dividing line here is concerned with the notion as to what constitutes a valid agreement between the customer and the provider; anything which does not need to be specified in order for the agreement to be binding to both parties can safely be determined at a later stage in the customer process.

The provider may have to ask the customer some additional questions, but could equally well maintain a database of customer preferences so that he doesn't need to ask the same information over and over again. In the BPM 2.0 paradigm, it is in the interests of the provider to make it as easy as possible for the customer to do business with him, so it makes sense to make the customer's experience optimal. Note that SOA as an IT approach greatly facilitates this separation between request data and implementation data. One of the foundations of SOA as an IT approach is that it is incumbent on every data owner to provide standard query services by means of which his data can be retrieved. A provider can use standard services to retrieve much, if not all of the implementation data, without inconveniencing the customer in any way.

This way of thinking radically changes the relationship between requestor and provider. For example, a requestor who asks a supplier to deliver goods has absolutely nothing to say about the resource management approach of the provider. The provider is free to switch from a statistical approach for reordering to an approach in which requirements are individually assessed and accounted for as they emerge in the business of its customers, without this needing a renegotiation of the agreement or other requests. The provider decides for himself how to run his own business. This approach of specifying only the request has the added advantage of leading to interfaces between information systems becoming much easier to standardize, which in turns results in increased ability to replace the information systems as circumstances dictate.

Process Boundaries

A process does only what the customer asked

With BPM 1.0 approaches, there are no particular criteria which determine exactly which activities form part of a process and which do not. Hammer's rule that the process should provide an output that is of value to the customer is not generally taken to mean that activities which do not contribute directly to that value should be excluded. As a result, different process designs for the same process will tend to draw the process boundaries differently, and are in consequence not interchangeable or even directly comparable. Given that processes that cannot be compared cannot be benchmarked, a lack of clear process boundaries would remove one of the pillars from under BPM 2.0.

With customer processes, on the other hand, it is always clear which activities belong to the process and which do not. Whatever we do on behalf of the customer is "in", the things we do on our own account are "out". By making this distinction, we can keep the customer process as simple as possible, in order to be able to replace it later without having to change all sorts of other

things as well. For example, we may manufacture products in order to satisfy an external service request, but maintaining the bookkeeping system is something we do for our own needs, not those of the customer. If we were to develop a customer process in which a customer order is translated into both the manufacturing activities and the bookkeeping, we would need to amend the customer process whenever we implemented a new bookkeeping system.

Including activities that don't belong to the process may not sound so bad, but when we consider all of the many things we do on our own account it is. Bookkeeping, logging, stocking data warehouses with up-to-date data, maintaining employee productivity data: all of these things and more are typically carried out using systems that differ from one organization to the next and from one time to the next, so that incorporating knowledge of them within a business process reduces interoperability and increases the difficulty of replacing the process with an implementation with another design or supported by another information system. We can avoid this problem by generating notifications whenever something that is important to these functions occurs, and letting them retrieve the information they need to process the event using the common services that SOA prescribes as a matter of course.

Distinct transactions are distinct processes

As discussed in the section on processes not being everything, each stage of a customer process results in some sort of an agreement or commitment. These agreements may be regarded as distinct transactions, each supported by its own process. In general, processes which result in commitments – for example the selling of an insurance policy – are distinct customer processes from those in which these commitments are honored. That applies structurally to generic intake of specific processes: accepting an application amounts to a commitment to making a decision on it. It also applies to processes in which resources are allocated or reserved before they are deployed.

The test as to whether these transactions should be included as part of the customer process is whether you will need to reverse them the customer request itself were to be retracted.[1] Typically, this includes activities such as ordering supplies because the customer order resulted in stock levels under the replenishment level, registering a new customer and updating the information of an existing customer. The execution of these activities is independent of the process outcome and should therefore be incorporated into separate customer processes. It is perfectly viable to link them together in the customer experience whilst executing them separately.

The BPM 2.0 top-down approach leads to more specific answers to the question as to what constitutes the boundaries of an orchestrated process. Typically, this is a question which leads to endless debates. For example, is the delivery of a confirmation message to the customer using an output management service something that should be orchestrated as part of a customer process, and if so, should it be a fire-and-forget affair or should the output management service report the successful completion of the action? In BPM 2.0 terms, all of these things are part of the purpose of the customer, and should therefore be orchestrated. Other actions, such as updating the data warehouse to take account of the new customer order or updating the general ledger, are clearly not part of that purpose and should not be included in the orchestration of the process, even if they are carried out by the same party.

Process performance is measured

External measurement

Because BPM 2.0 revolves around agreements between requestors and providers, some sort of measurement of the provider's performance is required. The measurement may be formal or informal, objective or subjective, but without measurement you may as well not agree anything in

[1] For a more detailed account of how the concept of transaction determines what should happen when a customer request is retracted, see (Wierenga, Information Security for SOA: why the Information Security Consultancy Industry Needs a Major Overhaul, 2010)

the first place. The more BPM-mature an organization is, the more systematic and intense the measurement.

The measurement should, like the agreement, include all aspects that matter to the customer. Typically, that includes not only the response times, but also the burden which the process places on him. A process in which the customer is continually nagged to provide anew information that he has provided in the past is likely to be less agreeable to him than a process in which he is asked to provide only information that is new for the current process instance.

Internal measurement

In BPM 2.0, as in BPM 1.0, a provider is well-advised to systematically measure each part of their process performance. However, there is a difference. In a typical BPM 1.0 environment, measurements are typically made from the point of view of the organization which executes the process, not from the point of view of the customer. As a result, there is a systemic bias towards process improvements that ignore the customer experience. With processes which have been designed as a BPM 2.0 hierarchy this is not the case. Each and every request and response pair can be measured from the customer's point of view as well as that of the provider. A key advantage of the BPM 2.0 approach is that the internal effort in order to execute the service can be measured at the same time as the external result, and the results can be compared. For such a comparison, measuring points at the boundary between the provider and the customer are required, and this is exactly what the BPM 2.0 hierarchy provides. The measurements of the performance of individual steps and threads makes much more sense when the measurements can be interpreted in the light of what they mean for the customer.

The BPM 2.0 hierarchy provides natural points of measurement at all levels within the process. Furthermore, it enables all aspects of the service that are relevant to the customer to be measured: not just response times but also the amount charged to the customer and the number and durations of the interactions with the customer by means of which the information required in order to execute the customer order was collected.

Benchmarking

Measurement is the basis of benchmarking. In BPM 2.0, benchmarking is made viable precisely because alternative implementations can be compared. Typically, benchmarking is applied when the same process needs to be carried out at many locations. The experiences at all of these locations can be compared, which enables best practices to be readily identified and strengthens the incentives to other locations to adopt them. BPM 2.0 makes benchmarking more widely applicable, both because it enables more useful measurements to be made, and because it makes it easier to compare processes with the same purpose but different business contexts.

It is always Clear who is Responsible

The provider is responsible for the service

One of the key advances of BPM over earlier approaches was its insistence that each process should have a process owner, responsible for the entire process. That way, it is in principle always clear who should be held accountable for a suboptimal process, and, in consequence, who should be given the mandate, the political power and the instruments which are required to optimize it. With this concept, it became possible to transform a fact of life – different parts of the organization each have their own view of things and their own agenda – into a problem that could be discussed and eventually solved. In our opinion, much of the success of BPM 1.0 can be attributed to this insight.

Unfortunately, BPM 1.0 did not go all the way and insist, as the BPM 2.0 customer process hierarchy compels you to, that each and every action has an owner, responsible for the entire action. In BPM 2.0, it is unthinkable that anything could happen without somebody asking for it from some agent who is entirely responsible for ensuring that it happens. In BPM, the notion that somebody should be responsible is all too often downplayed, because it is not perceived to be an

essential part of the paradigm. This translates into dysfunctional process designs. It is, for example, not at all unusual to encounter processes in which a front-office hands over a customer query to the back-office to handle. From a BPM 2.0 point of view that is wrong, because the customer issued the query to the front-office but the front-office is not entirely responsible for delivering an answer to the query. If the back-office should take too long to respond, or perhaps lose track of the query altogether, the front-office can neither detect that this is the case nor remedy it. Such processes are dysfunctional for the same sort of reason that lepers are: because a leper cannot feel pain in an affected part of the body, he or she does not immediately detect and respond to situations in which it is burned, bruised or cut. Any process design in which nobody within the organization feels pain when the organization fails to deliver on a promise made to a customer is just as unhealthy.

Blame cultures

Note that the BPM 2.0 concept of responsibility is incompatible with a blame culture. In a blame culture, if somebody else is responsible, you are not. If his failure leads to your failure, then it is his fault and you are blameless. In the BPM 2.0 customer process hierarchy, each provider is responsible for his own customer process. If some party way down in the hierarchy fails to deliver, resulting in failures all the way up the hierarchy, each failing party is one hundred percent responsible for its failure. That is the way it should be in order for the process to be made more reliable, for this ensures that each party has both an incentive and the mandate to select and manage its suppliers better so as to remove the cause of the failure. Because it ensures this, the BPM 2.0 concept of responsibility is vital to good process governance. Note that the rule that a customer process does only what was requested (see above) enables suppliers to be replaced more easily.

RA(C)I

The BPM 2.0 concept of responsibility matches the conventional distinction between the parties that are responsible, accountable, consulted or informed (RACI) concerning an action, with the exception of “consulted”. In BPM 2.0, the requestor is accountable, the provider is responsible and any parties that should be informed of an event are informed, but it is meaningless to consult someone unless you implement that consultation as a service, with as minimum output “Approved” or “Not approved”. A process design with a vague intention to inform someone so that he cannot claim that a product arose without his cognizance is utterly foreign to BPM 2.0; the provider is always totally responsible for the delivery of the product. Rather than let other parties influence the result, it is better to ensure that the provider is equipped to deliver the product correctly the first time around.

The BPM 2.0 concept of responsibility restricts the parties that are “informed” to those who need to act as a result of the process step. In BPM 2.0 the people who want or need to know something are enabled to ask: all information is at all times accessible using services. That way you don’t get e-mails informing you of everything, just so that you don’t overlook something important. The practice of informing everyone in sight is a symptom of not daring to accept responsibility, and it results in us all suffering from an information overload.

Standardization and Infrastructure

Standardization

Standardization is the process by means of which the number of valid solutions to a problem is by mutual agreement artificially reduced in order that autonomous agents, each working on a part of the problem, produce results that are compatible with each other. That trapeze artists can find each other in the air and that we all drive on the same side of the road are just as much standardization as the http-protocol.

SOA is all about standardization. [2] For BPM 2.0 that is a big advantage, for three major reasons. Firstly, it enables a fairly noise-free comparison of similar activities carried out in different times and places and by different agents. In other words, it makes the results of benchmarking more meaningful and more useful. Secondly, it enables organizations to outsource and insource more or less at will. Thirdly, it enables the infrastructure which is used to support standardized processes and subprocesses to be acquired as commodities, instead of this having to be tailor-made. These advantages are so considerable that they result in a mindshift: the first thing you ask yourself when applying BPM 2.0 is how you can reduce the agreements which the process embodies to their essentials. The step to standardization is then a small one, as SOA demands only that the agreements are the same and not the implementation. Added value is much easier to standardize than business functions, because it takes less information to specify. In principle, everything that is non-standard is taken out of the request interface. There are so many factors fewer on which agreement needs to be reached.

Convergence

Once you have reduced the service request to its essentials, it becomes easier to compare it with commercial offerings and decide whether they are acceptable or perhaps even better than what you have arrived at. This leads to a convergence of service requests. Standardization is the key enabler of an ecosystem of service providers and service requestors. But will the reduction SOA brings in the number of factors on which agreement must be reached be sufficient to generate an ecosystem? Perhaps enough factors remain in order to ensure that almost all services need to be tailor-made. If that would be the case then there would be serious barriers to outsourcing and insourcing at will. This question can be answered only by investigating particular areas in which sufficient standardization appears to be possible.

To start off with, almost any task which is carried out in the same way for many organisations and sectors can be made available as a service. That includes things like output management, in which the added value consists of taking information which has been produced by the customer, choosing a medium and an address which is appropriate for communicating this information to the addressee, converting the information to a form – such as a printed letter – that can be understood by the addressee and then sending it. The request itself consists merely of the information which is to be sent, the identification of the addressee and the identification of the template which is to be used for converting it to the send format. It is up to the provider to decide whether to ask the customer for the addressee data, the medium preferences and the template. A change in any of his choices will not change the service call. The service call will not even change when the service is greatly enhanced, for example to send messages as text to mobile phones.

Similar approaches permit many tasks in primary processes to be called using standardized interfaces. For example, the customer order stage model (see Figure 2) allows each stage to call subsequent stages using standardized service calls. The variation in the data required to specify the products is handled by placing the orders themselves in the customer order register, so that the service interfaces contain only the order numbers. That way an order for spare parts can use the same services as an order for a holiday on Tahiti. The same strategy allows resource management to be standardized using services to allocate, deploy, dispose of, maintain, enhance and acquire instances of the resource (see Figure 3). This model has the additional advantage that the services that are most exposed to the rest of the world – resource allocation and resource deployment – are fairly universal. For example, resource allocation for airline seats is different to resource allocation for pallets of bricks, but asking for an airline seat to be allocated to you is not basically different than asking for a pallet of bricks to be delivered to you. As a result, it is feasible to outsource the management of some resources, for example the particular resources which you do not wish to own, whilst retaining the customer order processes in which these resources are utilized. The ability to outsource reinforces the standardization process.

[22] For a more detailed discussion as to what that standardization involves the reader is referred to "10 SOA Commandments" (Wierenga, 10 SOA Commandments, 2010).

Services as infrastructure

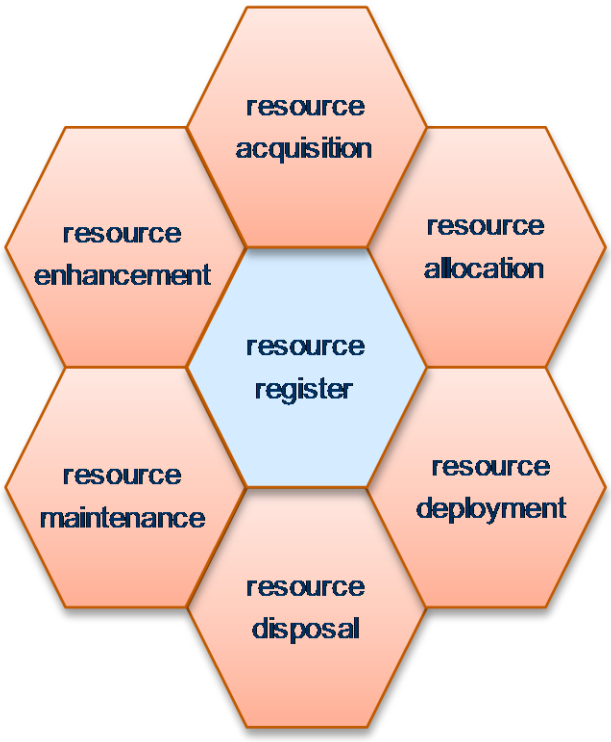

Figure 3: resource management services

SOA is all about separating infrastructure from business, not just in the functional sense, but also in the project management sense: a project manager should be able to concentrate on the business added value of the processes that he is responsible for developing, without having to worry about the infrastructure. It is no less absurd that a business project manager should have to organize a new output management service as part of a project to support a new insurance product, than that he should have to install a new telephone exchange. To burden the project manager with such chores results in an increase in the number of things that he has to coordinate, which results in turn in a disproportionate increase in the complexity of his task. Even letting him wait until some other project manager delivers the infrastructure is something that should be avoided where at all possible.

The step from convergent services to infrastructure is a small one. Once a service has been defined in such a way that it can serve any number of masters, there is no need to wait until there is a context in which it is required in order to make it. Nor is it necessary to build it or even operate it yourself. Given that projects with double the number of elements to coordinate cost four times the project management effort, it becomes advantageous to develop (or acquire) and operate the service as standard infrastructure. That shortens project development times and lowers project risks. And once the solution is live, only one service needs to maintained, not as many variants as there are situations in which it must be applied.

To be fair, there are also disadvantages. Using one service in many different contexts poses problems when a new version of the service must be implemented. How do you test that it will continue to work well in all the different contexts in which it is applied? Well, if you do need to test everything, it probably wasn't such a good idea to separate it out and apply it in many places. Do you test all your information systems every time your word processor is updated? Do you test all of your business processes every time you update your mail center? As a general rule of thumb, if you can outsource a function at a business level, you can treat it as infrastructure and can rely on conformance testing done by your provider for a functionally sound integration with your own processes.

Heterogeneity

Avoiding the monomania of BPM 1.0

Because BPM 1.0 requires you to consider the entirety of a process in order to be able to optimize it, it has an inbuilt bias towards uniform, one size fits all solutions. A single entirety tends to result in one designer who is made responsible for the solution, and one designer means one design, based on one view of the problem. If you start off by thinking that there is one problem,

you are likely to identify one solution, especially when multiple solutions are perceived as being more difficult to sell, let alone implement. There is an ever present temptation to ignore those variations in the problem domain that can best be supported by variation in the solution.

In order to facilitate implementation of multiple solutions, it is necessary to apply the BPM 2.0 paradigm – including the IT support using services - as outlined in this paper. Only if the request and response are limited to the absolute minimum is it possible to maintain several solutions next to each other, and replace current solutions without having to change the customer processes which they serve. Otherwise there will tend to be so much complexity built into the interface that the barriers to replacement and variation become almost insurmountable. Only when all interactions with other parties are interactions between, rather than within customer processes does it make sense to avoid the straight jacket of a single BPMS.

Often it makes sense for organizations to maintain some of the resources they need within their own ranks and hire additional resources when necessary in order to deal with peak volumes or unusual requirements. The BPM 2.0 solution to this type of requirement is illustrated in Figure 4: How BPM 2.0 deals with mixed sourcing.

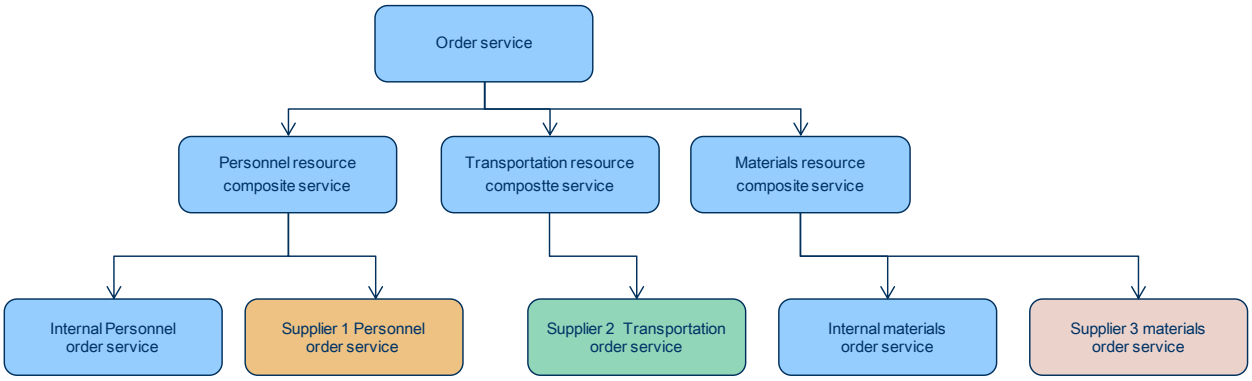

Figure 4: How BPM 2.0 deals with mixed sourcing

There are many factors which result in variations in problem domains that are best supported by multiple solutions, and hence unsuited for BPM 1.0. We shall examine some of these factors in the following paragraphs.

Varying levels of sophistication

Often the same process needs to be carried out with different levels of sophistication with regard to both the process steps and the IT support, due to differences in scale or subject area. For example, a process step which aims to capture an incoming customer order will differ greatly depending on the nature of the products being ordered. Ordering commodities requires less interaction with the customer and fewer data items to be captured than, say, ordering spare parts which are needed in order to repair complex machines In BPM 2.0 this is not a problem, because the request and the response of the actual order placement remain the same, even though the dialogue preceding the order placement differs.

Variation in time

In administrative environments, it is common that old customer orders are processed using the rules that applied when the orders that were accepted, and a change in the rules is applicable only to new orders. In long tail businesses such as life insurance, in time there is such an accretion of rules that this becomes a major administrative burden, to the extent that it is often preferable for companies to induce customers to accept modifications of the customer order than to continue to support old rules. As is the case with varying levels of sophistication, in BPM 2.0 the request and the response remain the same, even though the execution differs.

Varying levels of aggregation

In any organization which has a geographical subdivision in the form of branches or an ownership subdivision in the form of franchises, customer processes often comprise not only steps that are carried out in the individual units but also steps that are carried out centrally. These steps may alternate. For example, determining whether there are sufficient resources to fulfill the order may be carried out locally, determining whether the credit risk is acceptable centrally, the actual order delivery locally and the billing centrally. Because the customer processes typically start in the units and fall under their responsibility, solutions are needed in which local units can access centralized functionality and centralised units can ask a single question to all units. BPM 2.0 has no difficulty with processes like this, nor even with more complicated situations in which functions are concentrated but not centralized, or with shared service centres in which functions are shared between organizations.

Multiple providers

The more difficulty a customer has in switching between providers, the more power his current provider has over him. The balance of power shifts toward the customer if the switching costs are so low that the customer can choose the provider with each and every deal. SOA and BPM 2.0 are all about reducing transaction costs in order to make this possible. In particular, the insistence that a request contains only the information that defines the request enables the same interfaces to be used for many providers: you don't need to replace your IT or change your processes every time you switch to a new provider.

Process Status follows Product (or service) Status

The BPMS and process status

One of the major delusions of BPM 1.0 is the proposition that the BPMS moves a process to a new status. That is simply not true: it is the material progress in the process that moves it to a new status, and the BPMS merely detects or infers that this new status has been attained. This can be demonstrated by an investigation as to who is right if there is a difference between what the BPMS thinks and the material progress. Consider, for example, a process in which an order confirmation step requires the customer to have a positive credit rating. Suppose that the BPMS initiates a service to produce a credit rating, resulting in a positive rating, but then the credit agency withdraws the rating before the order has been confirmed? Should the order confirmation step still be carried out? Only a BPMS-purist could consider that a good idea. If the rating changes spontaneously, then the process must be interrupted.

A BPMS can be wrong about the process status and therefore give a wrong process continuation for a number of reasons. The handover from the material world to the BPMS can fail even though the material progress has been made. The material world can change after the handover. Manual steps can be carried out without the BPMS being informed of their inception, their completion or both. Note that the SOA paradigm automatically corrects for each of these conditions: a failed handover leads to progress being queried, a change after the handover is handled by separating the request as such from the information which may be required to carry it out, and the insistence that nothing is done without somebody asking for it to be done and expecting to be informed of the result make it totally unnatural for the inception or completion of a step to remain unreported.

The BPMS process conversion problem

One of the major benefits of designing processes in terms of steps with pre-conditions is that this makes it possible to convert running process instances to new designs and even new BPMS tools. With the conventional application of BPMS tools, such a conversion is a nightmare; so much so that almost everybody lets the running process instances go the full course. To give a trivial example, if the new process checks the customer's creditworthiness first and then reserves the products, whereas the old process does it the other way around, then there is no valid conversion for a process instance in which the products have been reserved but the customer's creditworthiness has not. It cannot be started anew, because then the products will be reserved

twice. It cannot recommence with the step after the reservation of the products, because then the customer's creditworthiness will not have been checked. The more the sequences of the old and the new processes differ, the more complicated this becomes. Typically, only the new process instances run according to the new process design, and there is a long and messy interregnum. This, of course, makes a mockery of the claim made by BPMS proponents that BPMS facilitates process flexibility. In fact, it is quite the reverse: processes that are not supported by BPMS are generally much easier to change than those that are. The conventional application of BPMS makes BPM 2.0 nigh on impossible, because it results in major barriers to process change, particularly to process change by the process users themselves.

Querying process progress

Another benefit of this approach is that it provides a standard mechanism to query the progress of a customer process, regardless of how many levels of subprocesses it involves. For each step, it is possible to determine whether it has been executed, is available for execution or is not yet ready to be executed. A standard query, applied recursively, will generate a complete picture of the status of any customer process. The result is a tree, which can be collapsed or expanded according to the choice of the requestor. The mechanism fails only when consecutive steps are not connected by means of a persistent intermediate product. But in that case, if just one of the steps has been carried out when process execution is suspended, a subsequent execution will need to redo the step. In other words, no persistent progress has been made.

Implications for management of subprocesses

When a business process applies BPMS in the BPM 2.0 manner, the need for the whole process to be managed by a single BPMS instance evaporates. This, in turn, allows us to integrate process management into applications, rather than having to place it above them. That makes it possible for problem domains to be completely supported by an application and still be integrated perfectly into the wider process context in which they are applied. For example, in the BPM 2.0 paradigm there is nothing wrong with having a purchasing application manage the purchasing subprocess as part of – say – a customer process.

Optimization is a Way of Life, not a Project

Continuous improvement

Continuous process improvement methods have been around for as long as there have been market economies. In the past decades such approaches – most notably Kaizen, JIT, TQM, Lean and Six Sigma, although there is a host of others – have proven themselves time and time again, so much so that almost all the Fortune 500 companies apply them. They are at their most successful when they become part of the culture of the organizations that apply them. Their success demonstrates that the project oriented thinking of BPM 1.0 is not a prerequisite for success.

BPM 2.0 borrows many traits from the established continuous process improvement methodologies, just as they do from each other. For example, BPM 2.0 thanks its preference for handling each request individually from the Kaizen concept that the ideal batch size is 1. Its absolute insistence on everything adding value for the customer is a key Lean concept. Its emphasis on measurement has strong affinities with Six Sigma. Its focus on a community that works together on continuous process improvement can be traced to TQM. To these traits it adds an opportunity to involve all participants by presenting them with a part of the whole that they can improve without prejudicing the rest, and a drive to standardize service calls in such a way that comparison and competition become the norm rather than the exception.

For continuous process improvement methodologies to become part of the organization's culture, more is needed than just that people think it is a good idea. It must become a natural way of working, supported in myriad ways and embedded in the organization's culture, so that applying the approach takes less energy and delivers better results than the alternatives. Then it becomes a way of life. BPM 2.0 has that potential.

Making an approach to be a way of life, rather than a project, generates enormous advantages. Projects typically take considerable amounts of corporate willpower to initiate and conduct, and they destroy almost as much as they produce. Markvoort, in a very perceptive article entitled "Deliberately choosing simplicity" (Markvoort, 2010) argues that it is a sign of weakness to have many projects, and that the reason why so many organizations rely on projects in order to make changes is that there is a systemic bias in the way problems are identified, which makes it seem natural to solve them by means of a project. He argues further that changes which have been introduced without recourse to a project are more likely to be supported by the stakeholders within the organization, make better use of its strengths and are more persistent than changes made by means of a project.

However, before BPM 1.0 practitioners move to BPM 2.0, there is still a major question that needs to be answered. This question arises from the cornerstone of BPM 1.0, the proposition that you need to oversee the entire process in order to be able to identify and apply measures to improve it. The question is: "Can BPM 2.0 identify and solve problems which require oversight of the entire process in order to be detected and fixed in the same way that BPM 1.0 can?". To answer this question, we shall discuss two major process interventions that appear to fall into this category: resource optimization and the optimization of the customer experience.

Resource optimisation in BPM 2.0

It is illustrative to consider how BPM 2.0 as a way of life facilitates a key process improvement strategy of BPM 1.0: resource optimization. From a BPM 2.0 viewpoint, resource optimization is a good thing, too good to be left to the process designer, for he cannot sense all the opportunities and is not intrinsically motivated to exploit them. It is far better to arrange things so that the provider has every incentive to optimize. That is what BPM 2.0 is all about.

However, having an incentive to optimize is meaningless if the provider does not oversee the resource usage within the process. In BPM 1.0 this is not a problem, but in BPM 2.0 the process is effectively divided up into a hierarchy of black boxes. What if dividing up a process according to a customer process hierarchy leads to the same employee having to switch on to a process instance many times, when this could have been done maybe just once? For example, an employee performs the order intake, another does the credit check, then the first employee checks whether all the goods are in stock, whereas letting the first employee do both the intake and the stock check at once would be far more efficient. Don't you need an overview of the process in order to see that?

Actually, you don't. In BPM 2.0, processes are not modelled in terms of sequences of steps, but as a set of steps, each with its own preconditions. In the above example, the service provider who is asked to perform both the order intake and the stock check will see that both steps become ready for execution at the same time. Because he understands what is needed in order to carry out the steps, he can also see if it makes sense for one employee to perform both of them in a single sitting. If he is provided with incentives to carry out these steps as efficiently as possible, he will be likely to identify and apply the improvement, even if the process designer missed it. In fact, the process designer is more likely to miss it, because the artificial imposition of a sequence on a set of process steps often leads to the sequence being perceived as a given that should not be changed. A further advantage of BPM 2.0 is that the service provider doesn't need to consult with the whole world before making the change, nor modify anybody's information systems excepting his own.

Optimization of the customer experience in BPM 2.0

One of the major advances of BPM 1.0 was that it provided techniques by means of which all the interactions with the customer could be made visible in a single diagram, thereby both suggesting the question: "How can we improve this?" and making it easy to answer it. It would be a serious objection to BPM 2.0 if it did not prompt this question or if it made it more difficult to answer.

Fortunately, BPM 2.0 actually improves on BPM 1.0 in this domain. Not only does it make the customer experience visible and amenable to analysis and redesign, but it also empowers people to make the necessary process improvements. In BPM 2.0, whichever agent accepts the

customer order is responsible for all interactions with the customer, regardless of where in the process the information garnered from the customer is used. This agent 'owns' the customer experience and part of his performance metrics is concerned with how happy the customer is with it. In consequence, dividing up a process according to a service hierarchy will not normally lead to the customer being requested for new information at every step and turn.

Owning the customer experience does not necessarily mean conducting all communication with the customer yourself. As outlined earlier in this paper, there is a big difference between reaching and implementing the agreement with the customer. The agent must carry out all the communication which is concerned with reaching or updating the agreement. For communication that doesn't change the agreement, the agent is responsible but may delegate the execution to another party. For example, the precise loading dock at which the order should be delivered is typically communicated by the customer to the carrier, because it is not part of the agreement between the customer and the provider, but merely an implementation of it. It is important to note that there are a number of reasons why making this distinction between the agreement and its implementation is so extremely helpful in optimizing the customer experience. Firstly, it enables the provider to think about how the implementation information can be made available without having to ask the customer every time. Much of this information can be maintained in a database of customer preferences, often using a self-registration system. Secondly, it provides an antidote to the tendency to perceive the customer as a monolith. Having one lane labelled 'customer' in a swimming lanes diagram rarely does justice to the way corporate customers operate. There is much to be said for a process design which places the responsibility for conducting additional communication as close as possible to the activities in which the information thus acquired is used.

Conclusions

We have shown that BPM 1.0 suffers from severe limitations. It has an inadequate concept of continuity between process instances. Its translation into IT by means of BPMS results in a loss of process agility rather than a gain, and inhibits large scale applications. Its concept of customer is rather too simple, and its concept of transaction doesn't make business sense. The measurements it generates are inadequate for benchmarking. Its concept of process leads to an expert-driven approach which fails to harness the energies and insights of process practitioners.

However, these deficiencies do not mean that BPM 1.0 cannot be used as a point of departure from which BPM 2.0 can be reached. We have shown that the application of the SOA paradigm to BPM 1.0 not only highlights its problems, but also points the way to improvements. It enables a new way of perceiving processes, which enables process practitioners to continuously improve their part of the process without affecting the rest of the process negatively. It motivates improvement using meaningful benchmarking. It turns process improvement into a way of life, and makes it progressively easier by calling into existence a wide range of process building blocks from which processes may be constructed and improved. Applying the SOA paradigm is the way to achieve BPM 2.0.

BPM 2.0 differs from BPM 1.0 on so many fronts that they must be regarded as completely separate entities, each with its own dynamic behavior. BPM 2.0 is more precise, more focused, more flexible, more aware of its limitations, and greatly more dynamic. If BPM 1.0 is like wading knee-deep along the shore, then BPM 2.0 is like swimming amongst the waves, with the whole body one single coordinated unity in which each and every limb is engaged, in which the power of the whole muscle system is unleashed, and nothing stands between the will to act and the action. Try it!

Author

Hans Wierenga is a consulting i-planologist – the IT equivalent of a city planner. He works for the Altran group and is based in Amsterdam. He is one of the two founding authors of the NORA, the business and IT reference architecture which guides the development and business alignment of the IT of all layers of government in the Netherlands. He is a key author of the GEMMA information architecture, in which the principles of the NORA are worked out in detail for all Dutch municipalities.

BPTrends Linkedin Discussion Group
We created a BPTrends Discussion Group on Linkedin to allow our members, readers and friends to freely exchange ideas on a wide variety of BPM related topics. We encourage you to initiate a new discussion on this publication, or on other BPM related topics of interest to you, or to contribute to existing discussions. Go to Linkedin and join the **BPTrends Discussion Group.**

References

Aalst, W. v., Hofstede, A. t., Kiepuszewski, B., & Barros, A. (2002). *Workflow Patterns; QUT Technical report. FIT-TR-2002-02.* Brisbane: Queensland University of Technology.

Behara, D. G. (2006, 5 26). *BPM and SOA: A Strategic Alliance.* Retrieved 5 6, 2011, from Bptrends: http://www.bptrends.com/publicationfiles/05-06-WP-BPM-SOA-Behara.pdf

Ghalimi, I. (2008, 10 26). *BPMlab.* Retrieved 5 6, 2011, from BPMlab: http://www.bpmlab.org/2008/10/26/what-is-bpm-20/

Hammer, M. a. (1993). *Reengineering the Corporation: A Manifesto for Business Revolution.* Harper Business.

Long, K. (2009, Vol 10, No. 9 (Sep 2009)). The Invisible Process (where are your processes?). *Business Rules Journal* .

Markvoort, D. (2010, December). *Deliberately choosing simplicity.* Retrieved 5 18, 2011, from Bptrends: http://www.bptrends.com/publicationfiles/ONE%2012-07-10-ART-Deliberatelly%20choosing%20simplicity-Markvoort%20_3_-final.pdf

Silver, B. (2006, 3 27). *BPM watch: make way for BPM 2.0*. Retrieved 5 6, 2011, from BPM Institute: http://www.bpminstitute.org/articles/article/article/bpms-watch-make-way-for-bpm-2-0.html

Wierenga, H. (2010, 5 10). *10 SOA Commandments.* Retrieved 5 17, 2011, from Infoq: http://www.infoq.com/articles/10-soa-commandments

Wierenga, H. (2010, 8 9). *Information Security for SOA: why the Information Security Consultancy Industry Needs a Major Overhaul.* Retrieved 5 18, 2011, from Soamag: http://www.soamag.com/I42/0810-1.php

