Applying Business Process Re-engineering Patterns to optimize WS-BPEL Workflows

Jonas Buys, Vincenzo De Florio, and Chris Blondia

University of Antwerp, PATS research group,
Middelheimlaan 1, B-2020 Antwerp, Belgium
{jonas.buys,vincenzo.deflorio,chris.blondia}@ua.ac.be

Abstract. With the advent of XML-based SOA, WS-BPEL shortly turned out to become a widely accepted standard for modeling business processes. Though SOA is said to embrace the principle of business agility, BPEL process definitions are still manually crafted into their final executable version. While SOA has proven to be a giant leap forward in building flexible IT systems, this static BPEL workflow model is somewhat paradoxical to the need for real business agility and should be enhanced to better sustain continual process evolution. In this paper, we point out the potential of adding business intelligence with respect to business process re-engineering patterns to the system to allow for automatic business process optimization. Furthermore, we point out that BPR macro-rules could be implemented leveraging micro-techniques from computer science. We present some practical examples that illustrate the benefit of such adaptive process models and our preliminary findings.

Keywords: business process re-engineering (BPR), service-oriented architectures (SOA), WS-BPEL, business processes, workflows.

1 Introduction

A cutthroat competition between enterprises is currently raging in which companies are compelled to constantly evolve so as to realize a competitive advantage. This head start is pursued by iteratively altering business processes[1] and strategies aimed at improving operational efficiency [1]. Business processes are thus continuously refined, mainly to resolve recurrent issues and as such rectify process performance. This concept is commonly referred to as business process re-engineering (BPR)[2].

[1] The notion of business process is defined as an orchestration of several activities carried out by computer systems or people within an enterprise with the objective of supplying a product or service to the customer.

[2] Because of the vague definitions found in most text books, the BPR acronym is commonly used interchangeably for business process re-engineering as well as business process redesign. The former has an evolutionary character, while the latter is revolutionary. For more information, we refer to [1].

M. Ulieru, P. Palensky, and R. Doursat (Eds.): IT Revolutions 2008, LNICST 11, pp. 151–160, 2009.

Information technology (IT) is actively deployed in large enterprises and this also applies to process automation. But regrettably most of these volatile business processes are enlaced in rigid IT systems and this imposes limitations with respect to the speed changes are possible with. In the beginning of this decade, this issue led to the concept of service-oriented architectures (SOA) in which IT is flexibly structured to better alleviate the re-engineering of processes by splitting up so-called business logic in a number of software components that are exposed as services [3]. Service (operations) should correspond to individual process activities that processes can be composed of by orchestrating and coordinating functionality comprised in those services. Actually this service-oriented computing paradigm comprises the best practices in distributed computing of - roughly estimated - the past twenty years, and commercially backed by major industry concerns, SOA keeps gaining popularity [4].

As one possible SOA implementation technology, web services have managed to become the *de facto* standard for enterprise software in which various distributed, heterogeneous software systems are integrated in support of corporate e-business and e-commerce activities [3]. The Web Services Business Process Execution Language (WS-BPEL) XML language is one of the standards that resulted from intensive standardization initiatives of industrial consortia, and shortly became a widely accepted standard for workflow modeling [6]. The benefit of the central BPEL orchestration component is that the process definition is no longer interwoven in the implementation code of the business logic. Because of this separation, SOA is said to alleviate the transformation and restructuring of business processes using highly reusable services that can easily be re-orchestrated into BPEL workflows [3].

The service-oriented paradigm turned out to be a giant leap forward in the construction of flexible IT systems indeed. XML-based SOA with BPEL further added to business agility, allowing for the quick development of new business processes leveraging service-wrapped legacy IT assets (*i.e.* business process redesign). But in spite of the popularity of BPEL and its clear separation of process and business logic, some shortcomings are not resolved thus far [5]. One of these issues is that a BPEL process definition is uttermost static: it is designed manually using some software tools and is then loaded into the BPEL engine. Since service orchestration and business processes are at the centre of SOA, it is imperative to continuously optimize BPEL process definitions to achieve system performance, besides the economic point of view in realizing a competitive advantage. Hence, this static BPEL workflow model should be enhanced to better sustain business process agility. Failing to do so would result in a somewhat paradoxical situation in which this static BPEL model is expected to drive the inherently dynamic processes required by the actual continual process evolution.

Although the BPR methodology originated in the early Nineties, until recently, businesses were still generally managed using an approach based on experience and intuition. But as BPR is gaining adherence, we are on the verge of uniting the IT-driven service-oriented paradigm and the BPR managerial methodology: automatically applying prevailing BPR principles to BPEL process definitions

can help in the further optimization of these process models, therefore sustaining process evolution.

One possibility to combining both disciplines is to build BPR intelligence into workflow design tools, transforming the process model before it is loaded into the BPEL engine. Alternatively, attributing business intelligence to SOA allows for the dynamic application of BPR principles to the original static BPEL process definitions with the goal of optimizing the process at runtime with respect to the system's current state and execution environment, and also allows to computerize more complex BPR patterns.

Leveraging techniques and practices from computer science to implement BPR patterns, adaptive process models arise, so that a more advanced form of SOA business agility can be attained. The ability to dynamically modify business processes such that the process semantics are preserved, while at the same time the process model is optimized, will positively impact SOA performance and add to the operational efficiency of a company striving for a competitive lead.

In this paper we illustrate the applicability of BPR patterns to BPEL workflow processes, and show how this can lead to performance improvements, such as a reduction in execution time.

2 Business Process Re-engineering and BPEL

Numerous best practices (principles, design patterns, heuristics) for BPR have been proposed in the literature [2], yet there has not been any thorough inquiry into combining IT and BPR so far. In order to support a higher level of process agility, we propose to design an intelligent system able to optimize the BPEL processes in accordance with these conceptual BPR principles. An overview of some potentially useful patterns for BPEL process improvement is shown in table 1.

Table 1. Some business process re-engineering patterns

BPR directives	**Basic techniques**	**Human-centric**
resequencing	data and control flow analysis [7],	No
parallelization	Tomasulo, scoreboarding [8]	No
exception	control flow and flow variable	No
knock-out (minimize process cost)	speculation [8]	No
order assignment and distribution	*chain of execution* [10]	Yes
flexible assignment	*nomination* [10,11]	Yes
specialist-generalist		Yes
split responsibilities	*4-eyes principle* [10]	Yes

The WS-BPEL XML language defines a set of primitives which business processes can be modeled with: basic activities can be set in order using control and data flow supported by structured activities [6,3]. These rudimentary structural activities turn out to be limited to the common control and decision structures available in most imperative programming languages. It is no surprise,

then, that we can spot similarities between the techniques for program optimization in computer science, operating on atomic units of instructions (which we refer to as the micro-level domain, or *micro* for short) and the BPR patterns for business processes, acting on coarser units of instruction blocks with a variable size: service operations (*macro*). Consequently, it is plausible to try and automate economic BPR patterns leveraging existing techniques from computer science.

Computerizing BPR turns out to be twodimensional: on the one hand, offline optimization techniques can preprocess the process model prior to its execution; on the other hand, runtime information can be used to adjust the BPEL process online depending on the system's current state and resource availability, either by affecting the process model or individual process instances[3].

In the following subsections, some examples will be shown to illustrate the benefit of applying BPR patterns to BPEL processes, and some suggestions for possible computer science techniques or practices for the implementation of these patterns will be motivated. For the purpose of clarity, the examples are presented in business process modeling notation (BPMN), a common graphical representation of the actual XML BPEL definition. This does not limit our contribution, as BPEL can easily be mapped to BPMN and vice versa [12].

2.1 Resequencing and Parallelization BPR Patterns

The execution of a BPEL process is essentially sequential, though the BPEL specification also provides constructs for executing activities in parallel [6]. As a primary BPR pattern, the execution order of process activities - *i.e.* service invocations - can be optimized by considering data flow dependencies so as to execute mutually independent activities in parallel [2]. The underlying idea of simultaneously executing activities and advancing activity initialization is that some time can be gained by avoiding performance-degrading stalls caused by dependencies. Throughout this paper, we assume an SOA-based environment aggregating a set of distributed IT systems allowing for optimization by parallel execution, but the amount of parallelism that can actually be achieved is limited by the number of resources (web services or employees) in the system.

Data dependencies can arise between different activities and relate to variables defined in the BPEL process definition. For instance, a service invocation activity has a read-dependency on whatever variable is used to hold the input message for the service to be invoked, and also a write dependency to the variable that will ultimately store the service's reply. Assignment statements normally construct and write to a variable after reading values from one or more other variables. Structural activities may also read certain variables during the evaluation of control flow variables.

Computer science techniques for dynamic instruction scheduling such as the Tomasulo approach and scoreboarding allow for an optimized, out-of-order

[3] BPEL processes are usually complex long-running processes. Every time the process model is triggered, the BPEL engine will construct a new process instance that runs in isolation from the other instances.

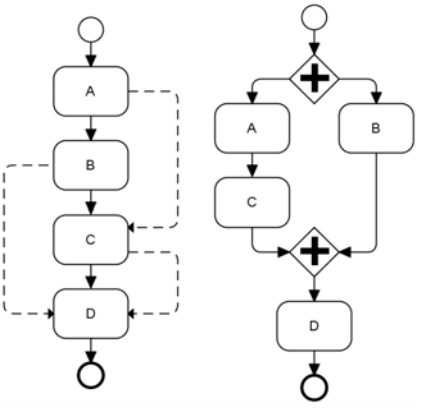


Fig. 1. On the left the original BPEL; on the right the optimized process. The rhombic symbols stand for parallel execution, *i.e.* AND-split/join.

execution of sequential streams of program instructions and could be used as the basis for individual process instances [8]. These techniques could be extended and applied to BPEL activities to avoid pointless waiting as the result of data dependencies. However, these approaches are limited to basic blocks of non-branching sequences of instructions - *cf.* one sequential scope in BPEL - and can benefit from techniques such as speculation to work around control flow statements and as such artificially increasing the number of instructions in these basic blocks. The Tomasulo approach exploits the knowledge on dependencies unravelled at runtime; thus it clearly outperforms all strategies that statically analyze the data and control flow of the BPEL process; nevertheless, such static analysis is easier to build up and can be used to restructure the overall BPEL process model [7].

In figure 1, we merely represent service invocation activities and assume that an unaltered output message from a particular service invocation is stored in a BPEL variable, which is used as input for invoking another service. Note that the dashed arrows representing these data dependencies are not part of the BPMN notation. The start event corresponds to the reception of a message that triggers execution of a BPEL process instance and the end event represents the process replying to its requester. The solid arrows in the diagrams indicate sequential flow. Supposing the respective execution times for activities (services) A, B, C and D are 10, 5, 20 and 7 seconds, the execution time in the original process would simply be 42 seconds, while the optimized version would result in an execution time of 37 seconds.

2.2 Exception and Knock-Out BPR Patterns

In re-engineering business processes, it is common to isolate exceptional flow from the normal process flow. Techniques like speculation are already applied to improve control flow, branches in particular [8]. This can be accomplished by conditionally executing the branch with highest probability, and compensating in

case of misprediction. Moreover, the amount of parallelism that one can exploit is also limited by control dependencies. Speculation is a technique that can be used to overcome the penalty of control dependencies in some cases by shifting highly probable activities to eliminate control dependencies so as to match the parallelism offered by the execution environment. To achieve speculation techniques in BPEL process definitions, scopes of activities might be shifted, provided that an estimation on the probability of each branch is available. This information must be gathered at runtime, during the execution of the program, using monitoring approaches as in feedback loops and autonomic computing [9]. An additional gain in execution time can be harvested by applying this pattern to a process because of further parallelization.

Consider for instance the example in figure 2. Imagine the left branch in the original process model has a 40% chance of being executed. For the service invocations A, B, C and X we suppose execution times being 10 seconds each. Then process execution would take 30 seconds for sequentially executing activities X, A and B, or 20 seconds for the other execution path comprising activities X and C. The time required to evaluate control flow structures, *e.g.* branch variables, is considered negligible. Speculation should restructure the model to always execute activity C (because of the 60% chance the containing branch is chosen) in parallel to X. The branching condition remains identical as long as the result of invoking service C is not written to a variable that is used for evaluating the branching condition. In case the execution proceeds accordingly, there is a considerable improvement: 10 seconds instead of 20 seconds in the original model (in this case a speedup of factor 2). However, speculation might deteriorate and delay process execution if the execution does not fit: because of the undoing, the best-case execution time of the speculated BPEL process is now 30 seconds. In the worst case, the performance degradation is given by

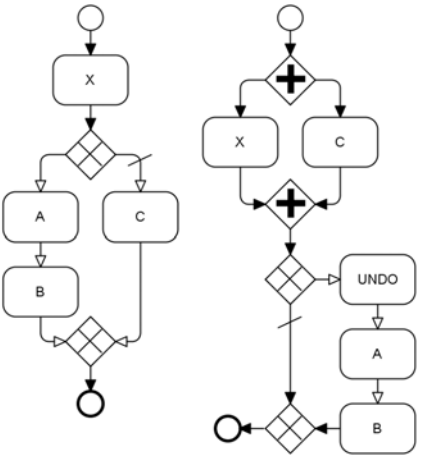


Fig. 2. On the left the original process; on the right the process optimized by speculation. The rhombic symbols in the original process denote conditional branches.

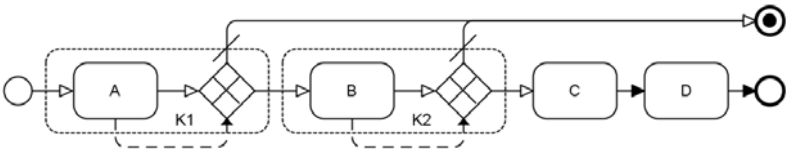


Fig. 3. An example to illustrate the knock-out BPR pattern

the overhead for undoing. Therefore, we suggest speculation to be applied only if the difference in probabilities of the alternative branches exceeds a minimal threshold. In that case, the overall performance might benefit from speculation, and the negative impact of non-fitting BPEL instances might be subdued.

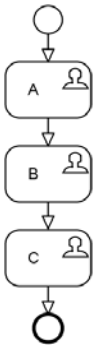
The knock-out BPR pattern, another application where speculation might fit in for implementation, is a special version of the resequencing pattern aiming to manipulate the process yielding on average the least costly execution. Knock-outs, conditional checks that may cause the complete process instance to cease, skipping all subsequent process activities, should be arranged in decreasing order of effort and increasing order of termination probability [2]. Upon occurrence, the BPEL process instance should be abandoned, possibly compensating in order to reverse the service invocations that were required for evaluating the knockout conditions. We illustrate this principle in figure 3. Suppose knock-out condition K_1 has a 40% probability of evaluating negatively and it takes 2 seconds to invoke service A and compute this branching condition. Likewise, for knock-out K_2 these values respectively equal 60% and 4 seconds including the invocation of service B. Then the ratio 0.40/2 for K_1 is higher then 0.60/4 for K_2. Hence, the arrangement of the knock-outs in this process model is fine.

As time goes by and more process instances have been executed, for both re-engineering patterns, the estimated probabilities of the branching conditions might change, which in turn might trigger new changes to the process model at runtime, resulting in adaptive business processes.

2.3 Human-Centric BPR in People-Oriented Business Processes

Human interactions frequently occur in business processes for the manual execution of tasks, *e.g.* an approval [10]. Expenses resulting from employment of people are still a major cost factor in enterprises. Therefore, an efficient allocation of the staff is imperative, and IT can also help to achieve this goal.

This brings us to another shortcoming of BPEL, which is rightly accused of being too automation-centric since it lacks the recognition of employees in process workflows [6,10]. The WS-BPEL4People and WS-HumanTask specification drafts - recently submitted to OASIS for ratification - allow for hybrid SOA in which human actors occur next to customary IT systems exposed as web services [10,11]. Consequently, we propose to automatically apply human-centric BPR patterns to people-centric BPEL4People processes. To our knowledge, this idea has not been previously investigated. Table 1 shows a few of these patterns




```
<extensionActivity>
  <b4p:peopleActivity name="B">
   <htd:task name="B">
     <htd:peopleAssignments>
       <htd:excludedOwners>
         <htd:from>
           b4p:getActualOwner("A")
         </htd:from>
       </htd:excludedOwners>
     </htd:peopleAssignments>
   </htd:task>
  </b4p:peopleActivity>
</extensionActivity>
```

```
<extensionActivity>
  <b4p:peopleActivity name="C">
    <htd:task name="C">
      <htd:peopleAssignments>
        <htd:potentialOwners>
          <htd:from>
            b4p:getActualOwner("A")
          </htd:from>
        </htd:potentialOwners>
      </htd:peopleAssignments>
    </htd:task>
  </b4p:peopleActivity>
</extensionActivity>
```

Fig. 4. Sample process comprising human tasks. At the right an excerpt from the BPEL4People process definition illustrating the language constructs for the concepts of separation of duties and chain of execution.

and points out relevant procedures defined in the BPEL4People specifications. Most of them deal with the issue of assigning the task to the best suitable person.

The *order assignment pattern* prefers the same employee to work on several process activities for a particular process instance. This is directly supported by the BPEL4People concept of chain of execution, where the actual owner that took care of the previous activity is selected as the sole potential owner for the task at hand (see figure 4, activities A and C). In addition, an escalation action should be defined to offer the task to the regular set of potential owners in case the default scenario would fail (*i.e.* the owner of the previous activity does not claim the task before the expiration deadline). Assigning several consecutive process activities to one person should result in a reduction of execution time as this person has got acquainted with the case. The side effect, however, is that the employee's workload will slightly increase compared to his or her colleagues.

Next, according to the *flexible assignment BPR pattern*, and supplemented by the *specialist-generalist pattern*, one should distinguish between highly specialized human resources and generalist employees that can be assigned to execute a diversity of tasks. The availability of generalists adds more flexibility to the business process and can lead to a better utilization of resources. Unfortunately, the generic human roles defined in the specifications are insufficient, and the people query facility and the organizational people directory that is searched by this query, both proposed in the above mentioned specifications, remain undefined. We should find a way to annotate people in this directory describing their skills, capabilities and permissions so that the system can reason about the degree of an individual's specialization. We envision an important role for techniques such as semantic processing and semantic matching in particular [13].

Lastly, assigning different tasks within a process to people from different functional units should be avoided (*split responsibilities pattern*). Again, enhancing the expressiveness of people queries and the structure of the people directory could allow the system to optimize the dispatching of human tasks to the

appropriate available human resources at runtime. Related to this pattern is the concept of separation of duties, also referred to as the 4-eye principle in which mutually independent individuals each perform an instance of the same task for the purpose of combating fraud and avoiding disastrous mistakes [11]. An example is shown in figure 4, where activity B may not be executed by whoever performed task A.

Combining human-computer interaction by means of the BPEL4People and WS-HumanTask specifications with BPR patterns for automatically and intelligently dispatching workload to human system resources is an exciting research challenge with the potential for a substantial performance improvement in process execution, which may lead to increased productivity and competitiveness. At the same time this also endorses the significance of BPEL4People in SOA, and we plead for a speedy ratification of the specification drafts.

3 Conclusion

We started this paper by briefly introducing BPR as a relatively new managerial methodology and SOA as a way to sustain the volatility of business processes resulting from the fierce competition in the market. It was pointed out that the static nature of BPEL process definitions is somewhat paradoxical with the necessity to quickly and easily re-engineer business processes in the quest towards operational efficiency. With SOA strongly embracing the principle of business agility, incorporating BPR system intelligence into the BPEL engine allows for the dynamic application of BPR principles to the original static BPEL process definitions with the goal of optimizing the process at runtime with respect to the system's current state and runtime environment. We propose an innovatory approach in which BPR principles are explicitly applied to WS-BPEL processes by means of established techniques and practices from computer science so that the process semantics are preserved and whereby at the same time the process execution is being optimized. It is expected that this will allow for a reduction in execution time, *e.g.* as the result of parallelization. This conjecture is corroborated by several small examples. Furthermore, we believe the new WS-BPEL4People standard might allow for the automation of human-centric BPR patterns in people-oriented business processes. This enables the system to intelligently dispatch human tasks to suitable human process actors.

Such BPR-aware SOA have the potential to turn static BPEL process definitions into adaptive workflows matching the current environmental and systemic conditions so as to make a more efficient use of these system resources in view of higher performance. The WS-BPEL specification need not be modified, which ensures a smooth transition in adopting these ideas. Complex BPR patterns can be implemented using runtime system information and possibly service metadata and annotations.

We are still in the early phase of elaborating on the ideas presented in this paper. We intend to develop a proof of concept illustrating the feasibility of the exemplified BPR patterns. Research on the introduced human-centric BPR

patterns will depend on the ratification process of the BPEL4People specification draft and the availability of compliant implementations. Moreover, it is worth remarking that the method proposed in this paper does not solely focus on higher performance, but could be extended to focus on other goals such as reliability and availability.

We conclude that BPR-aware SOA environments, automatically applying reengineering patterns to BPEL processes, result in adaptive business processes, which is a crucial requisite to achieve an enhanced form of business agility, better sustaining process evolution and as such mitigating the perceived paradox.

References

1. Reldin, P., Sundling, P.: Explaining SOA service granularity - how IT-strategy shapes services. Master's thesis, Linköpings Universitet, Sweden (2007)
2. Reijers, H.A., Liman Mansar, S.: Best practices in business process redesign - An overview and qualitative evaluation of successful redesign heuristics. Omega 33, 283–306 (2004)
3. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs (2005)
4. Stal, M.: Using Architectural Patterns and Blueprints for Service-Oriented Architecture. IEEE Software 23, 54–61 (2006)
5. Buys, J.: Business Process Re-engineering Leveraging Autonomic Service-oriented Computing. Technical report (unpublished) (2008)
6. Abode, et al.: Web Services Business Process Execution Language Version 2.0 (2007)
7. Klopp, O., Khalaf, R.: Reaching Definitions Analysis Respecting Dead Path Elimination Semantics in BPEL Processes. Technical report (2007)
8. Hennessy, J.L., Patterson, D.A.: Computer Architecture - A Quantitative Approach. Morgan Kaufmann Publishers Inc., San Francisco (1990)
9. Ganek, A.G., Corbi, T.A.: The dawning of the autonomic computing era. IBM Systems Journal 42, 5–18 (2003)
10. IBM, SAP, et al.: WS-BPEL Extension for People Specification, Version 1.0 (2007)
11. IBM, SAP, et al.: Web Services Human Task Specification, Version 1.0 (2007)
12. Stephen, A.: Using BPMN to Model a BPEL Process (2005)
13. Sun, H., et al.: Service Matching in Online Community for Mutual Assisted Living. In: Proceedings of the 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based Systems (2007)