Rule Based Business Process Execution with BPEL+

Adrian Paschke and Kia Teymourian
(Freie Universität Berlin, Germany
AG Corporate Semantic Web
{paschke, kia}@inf.fu-berlin.de)

Abstract: While WS-BPEL addresses the industry’s need for rich and standard service orchestration semantics it provide only limited expressiveness to describe (business) decision logic and conditional reaction logic. In this paper we propose a heterogeneous service-oriented integration of rules into BPEL to describe rule-based business processes and implement a distributed rule inference service middleware using rule engines which are deployed as distributed web-based inference services on an ESB. This declarative rule-based Semantic BPM (SBPM) approach aims at agile and adaptive business process execution via rules.

Key Words: BPM
Category: I.2, I.2.4, I.2.11

1 Introduction

Business Process Management (BPM) is considered as an effective approach to achieve business agility and flexibility. BPM solutions support simulating, modeling, executing, monitoring, and managing key business processes by abstracting the underneath IT infrastructure, IT service and application logic, and other aspects of IT. BPM can be realized in a top-down approach, where processes are first modeled and simulated and then deploy into the runtime environment. Or, in a bottom-up approach, as part of a Service-Oriented Architecture (SOA) transformation of the enterprise applications into modular business services, that can be easily integrated and reused in business processes - thereby creating a flexible, adaptable IT infrastructure. The explicitly modeled business processes can be reused, changed easily in response to new business requirements, and they enable real-time business activity monitoring.

Over the years, the Web Services Business Process Execution Language (WS-BPEL 2.0 or BPEL for short) has been broadly adopted as automated execution standard for the orchestration of Web Services into business processes. BPEL describes processes as activities performed by web services within a control flow which is orchestrated by the BPEL execution environment. However, typical business processes often include a number of decision points which effect the process flow; for example, a customer’s spending in the last year may determine whether he or she will get a discount or not. These decisions are evaluated based on certain events, conditions and facts, which may be internal or external to
the business process, and may be predefined company policies or business rules. BPEL’s expressiveness for representing such conditional decision logic is rather limited to simple qualifying truth-valued conditions (without e.g. declarative rule chaining, variable binding, etc.).

In recent years we have seen the rise of a new type of software called business rule management systems (BRMS) which allow enterprise architects to easily define, manage, update and run the decision logic that directs enterprise applications in a Business rules engines (BRE) without needing to write code or change the business processes calling them. This addresses an urgent need businesses do have nowadays: to change their business rules in order to adapt to a rapidly business environment, and they contribute to agility in SOAs by enabling reduced time to automate, easier change, and easier maintenance for business policies and rules.

Early manifestations of BREs which have their roots in the realm of artificial intelligence and inference systems were complex, expensive to run and maintain and not very business-user friendly. New rule standards such as OMG SBVR, OMG PRR, W3C RIF / RuleML and improved technology providing enhanced usability, scalability and performance, as well as less costly maintenance and better understanding of the underlying semantics and inference systems makes the current generation of BREs and rules technology more usable. Business rules have been used extensively in enterprises, e.g., to implement credit risk assessment in the loan industry (what is the interest rate for my car loan?), yield management in the travel industry (what price to sell a ticket?), operations scheduling in manufacturing (what should we build today to maximize throughput and keep customers happy?), etc.

In this paper we describe a heterogeneous service-oriented combination of BPM technology and rule technology which incorporates BREs as inference / decision web services into the BPEL process logic (BPEL + Rules). While BPM enables automated and flexible business processes, declarative rules enable decision logic, reaction logic for processing complex events (CEP) and for reacting to them, and various types of constraints. While WS-BPEL addresses the industry’s need for rich and standard service orchestration semantics, declarative rules address event-based, message driven, decentralized choreography semantics. The inference services, which run a BRE, are deployed on an enterprise service bus (ESB) and can be invoked by the BPEL process as semantic web services (SWS), using one of the many transport protocols supported by the ESB middleware.

This service-oriented rule-based approach has the potential to profoundly change the way IT services are used and collaborate in business processes. Akin to multi-agent systems (MAS) the rule-based logic inference layer which wraps the existing web services and data sources allows for semi-autonomous decisions and reactions to complex events detected by the rule-based event processing
logic. The combination of rules with ontologies gives precise semantics to used concepts such as processes, events, actions, tasks, state, time etc. Different rule inference services can communicate in (agent-based) conversations via messaging reaction rules, which enables them to build agent-based choreography workflows, that can be used as external subprocesses in the main orchestrated BPEL process. Potential application scenarios of BPEL+Rules for BPM are, e.g.:

- Dynamic processing
  - Intelligent routing
  - Validation of policies within process
  - Constraint checks
- Ad-hoc Workflows
  - Policy based task assignment
  - Various escalation policies
  - Load balancing of tasks
- Business Activity Monitoring
  - Alerts based on certain policies and complex event processing (rule-based CEP)
  - Dynamic processing based Key Performance Indicator (KPI) reasoning

In this paper we contribute with a declarative rule-based service-oriented business process execution approach exploiting WS-BPEL in combination with Corporate Semantic Web (CSW) technologies (semantic rules, ontologies and events/actions) and a scalable ESB middleware to operationalize such a distributed rule-based approach. Rule-based reaction logic for event processing and event messaging, and rule-based decision logic in combination with ontologies play a central role in connecting the various IT resources, human and knowledge resources, and Web-based services to agile business service networks. Ultimately, our novel rule-based design artifact might put the vision of highly flexible and adaptive Internet of Services supply chains / enterprize service networks into large scale practice and will lead to a highly flexible and agile BPM.

This combination of semantic technologies, i.e. declarative rules, ontologies and semantic event/action logics, with BPM, which leads to Semantic Business Process Management (SBPM) promises:

- enhanced automation in semantic discovery, configuration and composition of appropriate process components, information objects, and services (including human agents)
automated mediation between different heterogeneous interfaces (e.g. semantic web services) and abstraction levels

targeted complex queries on the process space and the process flow and state

much more agile business process management via expressive rule-based decision and reaction logic

The key benefits of our approach are:

- use of complementary technologies: semantic technologies for representing rules, events, actions, state, etc. + BPM technologies

- standard-compliant use of WS-BPEL for orchestration of services as well as people and partners implemented as external semantic web services or semantic agents (having a precise meaning/semantics defined for their interfaces)

- clear separation of concerns:
  - BPEL is used for orchestration in the business process
  - rules for declarative specification of constantly changing business policies and regulations
  - rules focus on decision making, policies and rule-based event processing for building complex events of higher level of abstraction and relevance for the business process
  - rules can be used to integrate choreography sub-workflows in orchestrated BPEL processes

- enables business users to participate in business processes and in adapting business rules

- modify and apply new business rules without redeploying processes

- declarative semantics for processing events, actions, rules, state etc. (including logic for transactions, compensations etc.)

The rest of the paper is organized as follows: In section 2 we give an overview of the relevant concepts. In section 3 we discuss what is missing in BPM with respect to semantics and rules. In section 4 we introduce the heterogenous service-oriented integration approach for adding rules to BPEL. In section 5 we implement this integration approach with a highly scalable and efficient rule-based inference ESB middleware. Section 6 concludes this paper and gives an outlook on our ongoing efforts to integrate humans by means of rule-based agents.
2 Background

In this section we give an insight into the applied technologies and define relevant concepts and technologies.

2.1 Service Oriented Computing

The computing paradigm that utilizes services for developing applications in open distributed environments such as the Web is known as Service Oriented Computing (SOC). The vision is to build large scale service supply chains (also known as enterprize services networks) which enable enterprizes to define and execute web services based transactions and business processes across multiple business entities and domain boundaries using standardized (web) protocols. Web services, which emerged in the last time as the prevailing technology for implementing IT services on the web, have received a great commitment from industry. A service-oriented architecture (SOA) expresses a perspective of software architecture that defines the use of loosely coupled software services to support the requirements of the business processes and software users. In a SOA environment, resources on a network are made available as independent services that can be accessed without knowledge of their underlying platform implementation. A SOA is not tied to a specific technology. It may be implemented using a wide range of technologies, including REST, DCOM, CORBA or Web Services. Service Component Architecture (SCA) is a set of specifications which describe a model for building applications and systems using a SOA. Recently, semantic web services (SWS), which provide an approach for semantically representing the functionality and interfaces of Web Services with the help of semantic technologies (ontologies and rules), attracted much attention from industry.

2.2 Complex Event Processing

Complex Event Processing (CEP) allows real-time detection and processing of complex events from typically huge amounts of smaller (raw/atomic) events (e.g. from an event cloud or event stream). Rule-based CEP (aka intelligent CEP) exploits reaction rule technologies for event processing using, e.g., event algebras to describe the complex event types which trigger reaction rules (e.g. Event-Condition-Action (ECA) rules or event-driven production rules). Rule-based CEP often supports situation detection (detecting transitions in the universe that requires reaction either “reactive” or “proactive”), pre- and post-conditions, and (transactional) action logics (complex actions).

2.3 Declarative Rule Programming

Rule systems have been investigated comprehensively in the realms of declarative programming and AI over the last two decades. Using declarative (inference) rules
has several advantages: reasoning with rules is based on a semantics of formal logic and it is relatively easy for the end user to write rules. The basic idea is that users employ rules to express what they want, the responsibility to interpret this and to decide on how to do it is delegated to an interpreter (e.g., an inference rule engine or a just in time rule compiler). There are different types of rules reaching from derivation rules, transformation rules to reaction rules and integrity rules / constraints.

2.4 Semantic Technologies

Semantic technologies aim at the formal representation of knowledge with clear, typically logic based semantics which allows for inference and reasoning, hence providing machine-readability and interpretability of data. The two major building blocks in semantic technologies are:

1. Ontologies which represent the conceptual knowledge of a domain and its meaning (semantics). Ontologies can be used, e.g. to represent business domain vocabularies, meta-models and meta-data

2. Rules which derive conclusions from given knowledge (derivation rules), which are logical constraints, or rules which process events/actions/state on the basis of a formal logic-based semantics

3 State of the Art in non-semantic BPM

Current BPM languages, such as BPMN 1.2 for modeling and WS-BPEL 2.0 for execution, are mostly syntactic process specification languages without a precise declarative formal logic semantics. This allows for ambiguities in the interpretation of models and specific functionalities such as compensations or exceptions. It hinders the integration of, e.g., rule inference services or humans via people link interactions. Modeling languages and models become incompatible and are no longer interchangeable. For instance, there are problems with model-to-model transformations and round-tripping, e.g. between BPMN and BPEL and vice versa. There is also a semantic gap between the business world and the IT world which use different vocabularies to describe the models and have different requirements on the abstraction level. The inner semantics of activities and their effects cannot be represented semantically in the current BPM approaches, which makes it hard to analyze and understand the business process behavior and the effects on the process states, beyond the pure functional description of the orchestration flow and the invoked web service interfaces. Agile business decisions in terms of business rules, complex event-driven behavior, and choreography interactions cannot be semantically integrated into BPEL.
What it missing for the semantic enrichment of BPM, i.e. for SBPM, is the integration of ontologies for events, processes, states, actions, and other concepts that relate e.g. to change over time into the BPEL definitions. With ontologies the processes and their causalities and roles would be precisely defined by logic. This would also make rules or logic that govern processes or react to events truly declarative knowledge which can be used in the processes for decisions and ad-hoc reactions. It would increase the ability to interchange semantic models across major BPM and BRMS vendors and would allow reusable, enterprise-relevant knowledge such as business policies and business vocabularies to be used in BPM models.

In this paper we will mainly focus on the heterogenous integration of rules technology with BPEL and will not specifically address ontologies for BPM and CEP which must relate processes, events, states, and actions, must include tense (past, present and future; perfected and progressive), time intervals with quantities (units, duration, composition, conversion and accuracy), etc.

4 Integration of Rules in Business Processes Execution

In this section we describe a heterogenous integration approach of rules into service-oriented BPEL processes. Rules are integrated as external distributed rule inference services. This means they are not directly part of the BPEL process (homogenous integration), but are integrated by invoking the inference service (heterogenous integration) which runs a rule engine and executes the rule logic.

Figure 1 illustrates this integration approach. A BPEL decision activity invokes a rule inference service sending event or fact data (e.g. queries) to the rule engine. The rule engine processes the received data according to its internal
rule-based logic. The results might be, e.g. derived answers (e.g. decisions) on the queries of the BPEL process, or they might be actions which are triggered by the reaction rule-based processing of the events from the BPEL process.

The concrete steps in this approach are

1. Create a rule inference service with rule repository and define the semantic interface description of the inference service

2. Create a new inference service Partnerlink, choose a rule connection and choose an interaction pattern and the parameter bindings

3. Add a rules Activity which invokes the rule inference service; Bind BPEL variables to parameters of the Partnerlink

During the process execution the BPEL engine will invoke the rule inference service mapping the BPEL variables to the input facts / events and the result from the rule inference service. It is possible to implement different stateful and stateless interaction patterns between the BPEL process and the inference service.

5 Implementation of the BPEL+ Rules Approach

In this section we will implement the proposed heterogenous BPEL+Rules integration approach and describe the main technical components.

5.1 Rule-based Semantic Web Services

In order to use rules for BPEL a BRMS, which manages and executes the rule logic with an internal BRE, is provided as Semantic Web Services (SWS). A SWS is a Web Service which in addition to its syntactic interface description (WSDL) is extended with semantic information, for semantic discovery, selection, invocation and composition. SWSs have received quite a lot of attention in the last years and many different approaches exist, e.g. OWL-S (former DAML-S), WSDL-S, SAWSDL, SWWS / WSMF, WSMO / WSML, Meteor-S, SWSI, etc. In [Pas07] we have implemented a rule-based SWS approach which in addition to the semantic interface descriptions of the service allows representing non-functional properties of the service, in terms of a Rule-based Service Level Agreement (RBSLA) - see figure 2.

For instance, a rule set from a Service Level Agreement might define three different IT Service Level Management schedules (see ITIL ITSLM process):

\begin{align*}
\text{if current time between 8am and 6pm then prime schedule.} \\
\text{if current time between 6pm and 8am then standard schedule.} \\
\text{if current time between 0am and 4am then optional maintenance schedule.}
\end{align*}
if prime schedule then the service level "average availability"
has a low value of 98%, a median of 99% and a high value of 100%
and a response time which must be below 4 seconds.

... if standard schedule then the responsible role for service outages
is the second admin.

As shown in the RBSLA project [Pas07] such rule sets can be adequately
represented as logic programs and easily extended with further rules, e.g. rules
describing escalation levels and deontic norms such as certain obligations if ser-
vice levels are missed.

The additional semantic information provided by RBSLA service descriptions
is used for effectively invoking the inference services from the BPEL process, e.g.
invoke a inference service which provides the requested decision functionality and
the required quality of service (QoS).

5.2 Prova - A Semantic Rule Engine

Prova (http://prova.ws) [KPS06, Pas07] is both a Semantic Web rule language
and a highly expressive distributed rule engine which supports declarative (back-
ward reasoning) decision rules, complex reaction rule-based workflows, rule-
based complex event processing (CEP), and dynamic access to external data
sources such as databases, Web services, and Java APIs. It has a typed logic ap-
proach which allows to use external ontologies or Java class hierarchies as type
system for the rules. Prova follows the spirit and design of the W3C Semantic
Web initiative and combines declarative rules, ontologies (vocabularies) and inference with dynamic object-oriented programming and access to external data sources via query languages such as SQL, SPARQL, and XQuery. One of the key advantages of Prova is its elegant separation of logic, data access, and computation as well as its tight integration of Java, Semantic Web and event processing technologies. It semantics draws on backward-reasoning logic programming (LP) concepts to formalize decision logic in terms of derivation rules and combines them with forward-directed messaging reaction rules for distributed event and action processing in order to exploit the benefits of both worlds.

Messaging reaction rules in Prova describe processes in terms of message-driven conversations between inference agents / services and represent their associated interactions via constructs for asynchronously sending and receiving event messages. Choreography interaction flows between distributed Prova inference agents/services are defined by the order of sending and receiving message constructs in messaging reaction rules. Messaging reaction rules maintain local conversation states which reflect the different activity flows and support performing them within in simultaneous conversation or sub-conversation branches.

The main language constructs of messaging reaction rules (as implemented in the rule engine Prova [KPS06]) are: sendMsg predicates to send outbound messages, reaction rcvMsg rules which react to inbound messages, and rcvMsg or rcvMult inline reactions in the body of messaging reaction rules to receive one or more context-dependent multiple inbound event messages:

\[
\begin{align*}
&\text{sendMsg}(XID, Protocol, Agent, Performative, Payload | Context) \\
&\text{rcvMsg}(XID, Protocol, From, Performative, Payload | Context) \\
&\text{rcvMult}(XID, Protocol, From, Performative, Payload | Context)
\end{align*}
\]

where \(XID\) is the conversation identifier (conversation-id) of the conversation to which the message will belong. \(Protocol\) defines the communication protocol such as JMS, HTTP, SOAP, Jade etc. \(Agent\) denotes the target party of the message. \(Performative\) describes the pragmatic context in which the message is send. A standard nomenclature of performatives is, e.g. the FIPA Agents Communication Language ACL or the BPEL activity vocabulary. \(Payload\) represents the message content sent in the message envelope. It can be a specific query or answer or a complex interchanged rule base (set of rules and facts).

For instance, the following messaging reaction rule waits for an inbound query which might come from the invoke request of a BPEL activity: (variables start with upper case letters)

\[
\begin{align*}
&\% \text{ receive query and delegate it to another inference agent} \\
&\text{rcvMsg}(CID, esb, Requester, acl\_query-ref, Query) :- \\
&\quad \ldots \text{ some conditions finding e.g. another inference agent} \ldots, \\
&\quad \text{sendMsg}(Sub-CID, esb, Agent, acl\_query-ref, Query),
\end{align*}
\]
rcvMsg(Sub-CID,esb,Agent,acl_inform-ref, Answer),
... (other goals)...
sendMsg(CID,esb,Requester,acl_inform-ref,Answer).

Via logical unification the data from the received inbound message is bound to variables and is used by the conditions in the body of the rule, which act as goals on other derivation rules in the local rule base (so called rule chaining). In the example the rule sends the query in a new sub-conversation (with the unique id Sub-CID) to another agent and waits for the answer. That is, the rule execution blocks at the rcvMsg condition in the body until an answer message is received in the inlined sub-conversation (or some timeout occurs), which activates the rule execution flow again, e.g., to prove further subsequent conditions (subgoals) of the rule or start other sub-conversations. Data from the received answers is bound to variables including backtracking to several variable bindings as usual in logic programming. Finally, in this example the rule sends back the answer to the original requesting party, which would be the BPEL process and terminates (finitely succeeds).

That is, the declarative rule-based approach provides an expressive and compact declarative programming language to represent complex event processing logic in arbitrary combination with conditional decision logic implemented in terms of derivation rules. Via sending and receiving event messages it is possible to implement choreography workflows that span several communicating (messaging) rule inference agents / services. With that approach it is possible to implement e.g. petri-net style conversation flows, and all complex event-based workflow patterns (as described, e.g. by Van der Aalst et al. [vdAtHKB03]) such as Join, Simple Merge, Cancel Activity, Multi-Choice, Structured Loop, Milestone can be implemented including also patterns like Deferred Choice which cannot be implemented by orchestration flows.

In summary, the major benefit of the described rule-based approach is the tight integration of standard derivation rules for implementing complex decision logic and messaging reaction rules for implementing behavioral event message based reaction logic, conversation-based interactions and choreography workflows. Complex (business) decision logic, such as business rules, can be implemented in a declarative way in terms of derivation rules (as logic programs) and used to prove conditional goals mixed in arbitrary combinations with send and receive message constructs in messaging reaction rules.

5.3 Enterprise Service Bus Middleware

To deploy and execute distributed rule-based inference agents as semantic web services an enterprise service bus middleware is used. We focus on the technical
aspects of the middleware and on the machine-to-machine communication between automated rule inference agents. Figure 3 illustrates the architecture of the ESB middleware.

Several Prova rule engines [KPS06] are deployed as distributed semantic web-based services. Each service runs a local rule base which implements the decision and reaction logic. The rules have access to local applications, data sources (e.g. web services, Java object representations such as EJBs, databases etc.), etc. and use them as local fact bases. The rule agents react to incoming events messages (requests) according to the defined rule-based execution logic. To communicate between the rule inference agents, Reaction RuleML [PKB+06] [Bol06], the current de-facto standard language for reactive web rules, is used as common rule interchange format. Translator services translate from Prova execution syntax into Reaction RuleML and vice versa.

To seamlessly handle asynchronous message-based interactions between the Prova rule inference services and with other applications and web-based services an enterprise service bus (ESB), the Mule open-source ESB [Mul06], is integrated as communication middleware. The ESB allows deploying the rule-based agents as distributed rule inference services installed as Web-based endpoints in the Mule object broker and supports the Reaction RuleML based communication via arbitrary transport protocols such as JMS, HTTP, SOAP (more than 30 protocols are supported) between them. That is, the ESB provides a highly

**Figure 3: Rule-based Middleware**
scalable and flexible event messaging framework to communicate synchronously but also asynchronously between services.

6 Conclusion

In this paper we have proposed a declarative rule-based approach for executable business processes descriptions in BPEL, where complex conditional decisions, complex event processing (CEP) logic, and choreography workflows are implemented in terms of rules. The rules are heterogeneously integrated into the BPEL process by invoking and executing them as semantic inference services. The rule based logic language combines derivation rules as means to represent conditional decision logic such as business rules with messaging reaction rules to describe conversation based process flow between agents. The conversational interactions take place via event messages. We have implemented a rule-based ESB middleware which deploys Prova rule engines as distributed inference services. At the platform-independent level it uses Reaction RuleML as a compact, extensible and standardized rule and event interchange format between the platform-specific Prova services. A highly scalable and efficient enterprize service bus is integrated as a communication middleware platform and web-based agent/service object broker. The realization of business processes by means of rules provides an expressive orchestration language which forms the technical foundation to integrate the business rules technology with the enterprize service and BPM technology. In summary the key benefits of our rule-based SBPM approach are:

- BPEL for orchestration of services, people and partners
- Rules focus on decision making and policies
- Rules can be used to integrate choreography sub-workflows in orchestrated BPEL processes
- Declarative specification of constantly changing business policies and regulations
- Enables business users to participate in business processes via adapting business rules
- Modify and apply new rules without redeploying processes

The agent model of our rule-based approach can be applied to human agents and provides promising means to integrate people into the BPEL process. In a next step we will implement user-friendly interfaces such as a semantic Process WIKI tool for the integration of humans. The users will communicate with the rule-based agents by adding knowledge-intensive information via the easy to
use WIKI interface, and the rule-based agents, which implement the particular agent role of the human user in the business process, will process and send this knowledge to the BPEL business processes.

Acknowledgements

This work has been partially supported by the “InnoProfile-Corporate Semantic Web” project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Entrepreneurial Regions.

References


