

Integrating Complex Events for Collaborating and Dynamically Changing Business Processes

Rainer von Ammon¹, Thomas Ertlmaier¹, Opher Etzion², Alexander Kofman²,
Thomas Paulus¹

¹ CITT GmbH, Konrad-Adenauerallee 30, D-93051 Regensburg, Germany

² IBM HRL, Haifa University Campus, IL-31905 Haifa

[\[Rainer.Ammon, Thomas.Ertlmaier, Thomas.Paulus\]@citt-online.com](mailto:Rainer.Ammon,Thomas.Ertlmaier,Thomas.Paulus@citt-online.com)

[\[Opher, Kofman\]@il.ibm.com](mailto:Opher.Kofman@il.ibm.com)

Abstract. Business processes must become agile, respond to changes in the business environment in a timely manner and quickly adapt themselves to new conditions. Event-Driven Business Process Management (ED-BPM) is an enhancement of Business Process Management (BPM) by concepts of Service Oriented Architecture (SOA) and Complex Event Processing (CEP). The most important enhancement is the integration of services accessible via the Internet that fire events into global event clouds. The events can be processed by event processing platforms for aggregating the information into higher value complex business events. These events can be modeled in a business process execution language within a process driven Business Process Management System (BPMS) to trigger changes in control flow of a process or start other services. A reference model and a reference architecture for ED-BPM are presented, based on the NEXOF Reference Architecture. A taxonomy for classifying changes to process flow is proposed. Enhancements have to be applied to the existing standards in the BPM field, including both the design-time and the runtime. A scenario from the banking domain illustrates the main concepts and principles.

Keywords: Business Process Management, Complex Event Processing, Business Activity Monitoring, Software as a Service, Event Driven Architectures, Business Process Execution Language, Business Process Modeling Notation

1 Introduction: Future Internet, Internet of Services and the (Re-) Active Internet of the Future

In the last years a number of voices have been arising around the topic of the so called “Future Internet” [1, 2]. Not only the foreseen limitations of the current Internet makes us think about its future evolution, but also new concepts and applications appearing recently will put the Internet to its limits. Among these new concepts is the “Internet of Services” [3]. In a world in which everything is connected to the network, the next step is thinking of the network as a worldwide trusted ecosystem of service providers, consumers and brokers buying, selling and composing services for

different needs. This associates one of the most complex problems to be solved in the future Internet: How can all those elements and services be coordinated?

This paper discusses a coordination based on events. Each element or service can generate different events providing information to other elements as well as each element can consume events from other elements. In this context each actor is able to react in a number of manners to the received information leading to a new concept that we call the “(Re-) Active Internet”. To achieve this, the concept of business processes is applicable to the “(Re-) Active Internet”, whereas a business process itself can be considered as a service that sends and reacts on events.

Defining a number of business processes we can set up how the elements within the Internet react to the different events they receive. We have to deal especially with the problem that it is not feasible to define a business procedure for every possible situation. Therefore business processes have to deal with e.g. unexpected situations. Taxonomies have been defined for process flexibility like in [4] in which the authors differentiate among four types of flexibility in business processes: flexibility by design, deviation, underspecification and change. Descriptions, examples and a deep analysis of each type of flexibility are provided. An appropriate taxonomy will be discussed in the following sections. In Sec. 2 we first describe the model of Event-Driven Business Process Management (ED-BPM) and also a reference architecture as the basis of realizing ED-BPM platforms. In Sec. 3 a taxonomy for use cases as examples from different domains is presented. Sec. 4 provides an overview on current and future business process modeling and execution standards which may influence ED-BPM. In Sec. 5 we investigate the problem to map the process models to a workflow engine in a runtime system for executing and for dynamically changing the flow of a business process or for coordinating collaborating processes accordingly. For this reason WS-BPEL as a standard for the execution of business processes has to be enhanced. The conclusion in Sec. 6 also shows the next related steps in our research work.

2 ED-BPM as the Basic Technology Platform

A business process is a structured set of activities designed to produce a specified measurable result for a particular customer or market. The common understanding behind Business Process Management (BPM) is that each company’s unique way of doing business is captured in its business processes. They are seen as the most valuable corporate asset.

The term “Event-Driven Business Process Management” is a combination of actually two different disciplines: Business Process Management (BPM) and Complex Event Processing (CEP) [5]. In this context BPM means a software platform which provides companies the ability to model, manage and optimize their processes for significant gain. As an independent system, Complex Event Processing (CEP) is a parallel running platform that defines, analyses, processes events. The BPM- and the CEP-platform interact via events which are produced by the BPM-workflow engine, by the IT services and other event sources that have an influence on the business process. The definition of ED-BPM and its historical background is described in [6, 7, 8]. In

the following, we roughly outline the basic components needed for operational ED-BPM systems. As shown in the reference model [9], basic elements can be taken from BPM platforms as well as CEP applications. ED-BPM comprises two modeling layers for business processes as well as for events. Basically this connects the pre-operational abstraction of process modeling with the runtime observations of business process-related events occurring at execution time so the principle of how an ED-BPM platform works is on the basis of events.

With this architecture in mind and given the recent availability of commercial CEP platforms, there will be two different kinds of ED-BPM specialists in the future: Workflow modelers responsible for designing business processes and event modelers responsible for identifying and designing events as well as complex event patterns with the aim of detecting relevant situations occurring in business processes. Events and event patterns are designed on the basis of an Event Processing Language (EPL). At present no standard for an EPL is available although different commercial solutions exist on the market. They imply an SQL-like language or provide a rule based approach. Currently the appropriate standard is discussed by the CEP community (e.g. [10, 11]).

The combination of service oriented and event-oriented thinkings are drawn closely together by the emergence of event driven architectures (EDA), hence dealing with different issues and problem solving approaches. Within this movement the currently most known proposal for reference architecture is the Networked European Software & Services Initiative (NESSI) approach called NESSI Open Service Framework – Reference Architecture (NEXOF-RA) [12, 13, 14]. This approach has been enhanced with event processing capabilities in each level of the layered architecture as shown in [15]. Event processing itself may be layered to display increasing abstractions of events from a low level (technical layer) to higher level (business process layer) as proposed by David Luckham's event hierarchies [5]. The extensions on the existing NEXOF-RA have been proposed in aspects of integrating event processing and are described in detail in [15].

3 Challenges Regarding Collaborating and Dynamically Changing Business Processes

The need for process flexibility has already been recognized since the middle of the nineties of the last century as a critical quality of effective services or business processes in order to provide organizations the ability to adapt to changing business circumstances [16, 17, 18]. In this section we show that the challenges in the case of collaborating and dynamically changing business processes respectively dynamically starting, stopping, terminating or changing Internet services need another taxonomy.

3.1 Related Work regarding Taxonomies of Service and Process Flexibility

Effective services and business processes must be able to accommodate changes to the environment in which they operate, e.g. new laws or changes in business strategy.

The ability to encompass such changes is termed process flexibility [4]. The notion of flexibility is often viewed in terms of the ability of an organization's processes and supporting technologies to adapt to these changes [19, 20]. An alternate view advanced by [21] is that flexibility should be considered from the opposite perspective, i.e. in terms of what stays the same not what changes. Indeed, a process can only be considered to be flexible if it is possible to change it without needing to replace it completely [22]. Hence flexibility is effectively a balance between change and stability that ensures that the identity of the process is retained [21, 23]. There have been a series of proposals for classifying flexibility, both in terms of the factors which motivate it and the ways in which it can be achieved within business processes.

According to [16, 24, 25], flexibility can be classified with respect to the types of changes it enables. The taxonomy presented in [22] includes three orthogonal dimensions: the abstraction level of the change, the subject of change, and the properties of the change, which include extent, duration, swiftness, and anticipation.

[4] take a different look into the various ways in which flexibility can be achieved and propose to distinguish "flexibility by design" for handling anticipated changes in the operating environment, where supporting strategies can be defined at design-time. A different category is called "deviation" for handling occasional unforeseen behavior, where differences with the expected behavior are minimal. Another category is "underspecification" for handling anticipated changes in the operating environment, where strategies cannot be defined at design-time, because the final strategy is not known in advance or is not generally applicable. The last category of "change" aims either for handling occasional unforeseen behavior, where differences require process adaptations, or for handling permanent unforeseen behavior.

The knowledge model of the S-Cube project [26] defines an adaptable service-based application as an application augmented with a run time control loop that monitors and modifies itself on the basis of adaptation strategies designed by the system integrators. An adaptation can be performed either because the monitoring of these services has revealed a problem or because the application identifies possible optimizations or because its execution context has changed. The context can be defined by the set of services available to compose the service-based applications, the parameters and protocols being in place, user preferences, and environment characteristics (e.g. location, time). Examples of adaptation strategies are re-configuration, re-binding, re-execution or re-planning (see also [27]).

The S-Cube project is contributing to the Nessi NEXOF-RA in which - during its preparation phase - our approach enhanced the ED-BPM-components (see chap. 3 as well as [28] and [29]). In the sense of S-Cube a service-based application is composed by a number of possibly independent services available in a network, which perform the desired functionalities of the architecture. The services can be provided by third parties and not necessarily by the owner of the service-based application. S-Cube defines a service-based application as a profound difference with respect to a component-based application, because the owner of the component-based application owns and controls all its components, while in contrast the owner of a service-based application does not own, in general, the component services, nor can he control their execution.

The approach of this paper is more comprehensive and follows a broader understanding. As the following use case shows exemplarily, a concert of independent

applications and their services or processes is managed and controlled by the analysis of global event clouds (as „posets“ (partially ordered set of events)) or event streams (as „tosets“ (totally ordered set of events)) as described in [6, 7, 8].

3.2 A Sketch of a Taxonomy of Collaborating, Dynamically Invoking and Changing Processes

The following section describes a first sketch of a taxonomy that deals with the dynamic changes in collaborating processes. In future work it will be further elaborated taking additional criteria into consideration. The purpose of this taxonomy is twofold. First of all, it helps to systematize different criteria and requirements involved in driving dynamic changes in collaborating processes. Second, we use this taxonomy to investigate the gaps in modeling and execution process languages and to propose the corresponding enhancements.

The main focus of such process collaborations are events generated by the single process steps respectively by the service invocations and sent to a global event cloud (see e.g. [9]). For this aim processes and services have to be enhanced appropriately or the upcoming Enterprise Services Bus (ESB)-generation will be able to automatically generate such events only by configuration (see e.g. [11]). In the following we don't differentiate between the terms "process" and "service", in fact a business process can also be perceived as a coarse grained service in the Internet. Depending on the defined event patterns according to the use cases of a specific domain, we distinguish the following situations:

- an existing process instance is {stopped, continued, terminated, changed}, a new process instance of the {same, different} process type is {instantiated, new defined},
- a task in an existing process instance is {stopped, continued, aborted}.

It could also make sense to include more criteria e.g. for subprocesses; such questions need still to be investigated (see e.g. the Semel-approach of the European projects SUPER (FP6-026850) and SOA4All (FP7-215219) [30]).

3.3 Use Case "Fraud-Management" in the Banking-Domain

In order to make it more explicit, Fig. 1 shows a reference model for ED-BPM-based fraud management for different domains like banking, insurance or retail. We use this model throughout the paper to explain possible scenarios involving event-driven collaboration between loosely coupled business processes.

The following scenario is describing a "cash withdrawal" process derived from [31], which in connection with a potential 'fraud' event pattern and related processes exemplifies the ED-BPM principle:

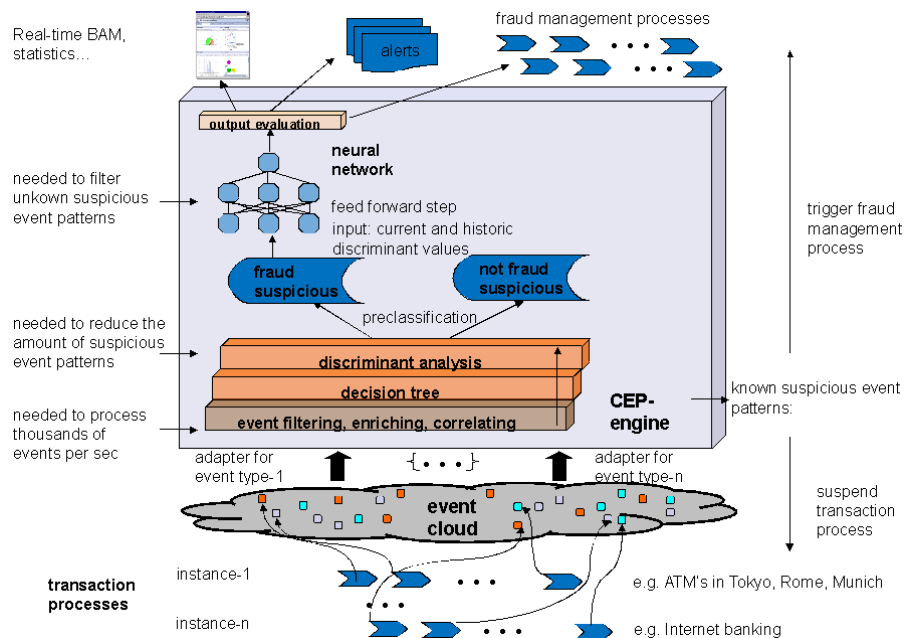


Fig. 1. A reference model of ED-BPM-based, non-deterministic service and process adaptiveness, e.g. fraud management in different domains like banking, insurance or retail [31].

1. An instance of a transaction process starts to process a withdrawal at a certain ATM.
2. A lot of process instances of the same type are instantiated in a certain timeframe at different ATM's.
3. Each process step generates an event that contributes to the event cloud using some messaging infrastructure (e.g. JMS publish/subscribe).
4. The global event cloud is analyzed in real-time by the CEP-system and optionally by some "intelligent" components like discriminant analysis and neural networks. A suspicious event pattern is detected because the same credit card is used more than once at different locations within a certain period of time. The following describes the behavior that should take place as a result of this fraud attempt detection.
5. All running instances of the same process type "cash withdrawal" for the suspicious card must be stopped as well as all their corresponding subprocesses.
6. A new process instance of the type "alert an affected branch" has to be started in order to check the customer at the ATM.
7. A new process instance of the type "fraud management" will be started. This process calls the appropriate subprocesses to update the statistics of the real-time BAM dashboard, to lock/disable the suspicious card, etc.

8. The fraud suspicion must be examined and an employee must make a decision regarding the following procedures: If no decision is made within the specified time period, a new process instance must start to escalate the problem to a higher layer in the decision making hierarchy.
9. If the decision is evaluated as a “false positive”, an ‘unlock card’ process will be instantiated, and the execution of all the process instances that were stopped as a result of the ‘fraud’ situation must be resumed.
10. If the “false positives” are too frequent (e.g. the percentage of ‘false positives’ is greater than some predefined threshold) the process type has to be modified. The process type can be changed automatically or manually. Changes may contain modifications to the workflow (e.g. adding or removing activities) or changes to values used by business logic, or both. An example of changing a workflow would be: “If the frequency of false positives is above a certain value, add a new activity of asking the client a special question”. An example of business logic modification would be: “If the frequency of the false positives is above the certain threshold, increase the maximal allowed distance between ATMs by 5 percents.” An event serves as a trigger of a process change procedure (see pattern 5 in Sec. 5.4).

There are a lot of different and even more sophisticated event patterns of fraudulent withdraw trials which would have to be modeled accordingly [32]. Similar event patterns of fraud scenarios are also known in other domains like insurance [33] or retail [34, 35].

4 Current and Future Movement on Standards

With the development of ED-BPM (Event-Driven Business Process Management) a number of existing and future standards as well as different research projects, whose coherence is shown in Fig. 2. will play a major role. The coherence between the different standards and research projects are shown in Fig. 2. In January 2009 OMG released BPMN 1.2 – a business process modeling notation. Thereupon requirements and developments were leading to a new major release. While this paper was written the OMG was about to release BPMN2.0 [36]. BPMN provides a special symbol for every event type and BPMN provides some enhancements for choreography and conversations for their modeling at design time [37]. BPMN 2.0 does not provide a facility to model process external event patterns and how to react on them.

The HPI (Hasso Plattner Institute) introduced BEMN (Business Event Modeling Notation) [38] in 2007, a graphical notation for modeling complex events in business processes. According to a statement from the HPI, this project has also reached a final state and will not be enhanced anymore. Nonetheless this work may have some influence on BPMN – not concerning BPMN 2.0 but a next major release – as well as on EMP (Event Metamodel and Profile) [39].

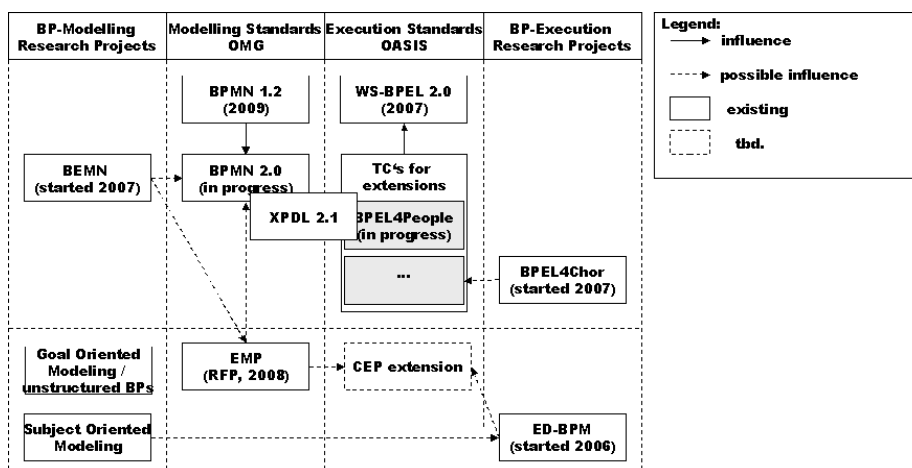


Fig. 2. Existing standards and current research projects related to ED-BPM

Another interesting approach regarding ED-BPM is called “subject oriented modeling”, a BPM modeling notation based on [40, 41]. This notation may be used due to conceptual characteristics like direct execution of modeled processes for ED-BPM technologies. First prototypes of ED-BPM are developed using this method in 2009 [42]. Nowadays are also some published approaches for the so called flexible unstructured business processes or Goal Oriented BPM, based on already existing BPM-tools [43]. A similar distinction was already known in the middle of the nineties of the last century in order to categorize workflow engines according to production workflows, ad hoc-workflows and administrative workflows. These BPM tools for unstructured BPM will not be based on the mentioned standards, but those platforms will be based on their own proprietary notations for their special focus.

A common standard for executing business process models is the WS-BPEL 2.0 standard. The actual version 2.0 is a final release and was released in April 2007. It will not be enhanced anymore and the TC (technical committee) has finished its work [44]. Further development and enhancements to WS-BPEL will be accomplished by particular BPEL extensions that will be maintained by autonomous TCs.

BPEL4People is an extension to expand WS-BPEL – in this case to support human interactions. The development of this extension is currently in progress and will be arranged by an OASIS TC [45]. Thus the extensions are enhancements to the standardized WS-BPEL.

BPEL4Chor as a research project from IAAS [46] might influence another extension of WS-BPEL regarding the challenge of choreography. Interconnecting information systems of independent business partners requires the specification of the interaction behavior the different partners have to adhere to. Choreographies define such interaction constraints and obligations and can be used as starting point for process implementation at the partners sites. The BPEL4Chor project showed how the BPMN and WS-BPEL can be used during choreography design. Step-wise refinement of choreographies to the level of system configuration is supported through different language extensions as well as a mapping from BPMN to BPEL4Chor.

A corresponding modeling environment incorporating the language mapping was also investigated. Complex choreographies as intended in the use cases of our approach cannot be created within a single step. Whenever many different business partners and many interactions are involved, choreography design must be split up

into different phases, each addressing different issues of the model as described in [47] and [48]. This approach can also be evaluated in our work.

XPDL is another standard for executing business process models. The WfMC (Workflow Management Coalition) Steering Committee voted on 23 April 2008 to approve version 2.1 of XPDL [49]. This current release includes all new functionality that was accepted by the working group. Also included is new functionality to update the BPMN to version 1.1. As mentioned, BPMN is a visual process notation standard from the OMG, endorsed by WfMC. The BPMN standard defines only the look of how the process definition is displayed on the screen. How those process definitions are stored and interchanged is outside the scope of the BPMN standard. XPDL provides a file format that supports every aspect of the BPMN process definition notation including graphical descriptions of the diagram, as well as executable properties used at run time. With XPDL, a product can define a process definition with full fidelity, and another product can import and reproduce the same diagram that was sent. XPDL is used today by more than 80 different products to exchange process definitions.

BPEL and XPDL are entirely different yet complimentary standards. BPEL is an "execution language" designed to provide a definition of web services orchestration. It defines only the executable aspects of a process when he is dealing exclusively with web services and XML data. BPEL does not define the graphical diagram, human oriented processes, subprocesses, and many other aspects of a business process, but it is going to be enhanced accordingly. Also XPDL has to be enhanced for the integration of CEP in future XPDL-based ED-BPM platforms.

5 ED-BPM-Based Enhancements for BP-Execution

Business process execution can be implemented on the basis of so called imperative or declarative implementation languages (see e.g. [4]). The "imperative" style means to implement exactly "how" a process has to be executed while the "declarative" approach describes "what" the process shall do. In our work, we concentrate on the implementation language WS-BPEL as an imperative language. The challenges of Sec. 3 are tried to get realized by appropriate enhancements of WS-BPEL. Our studies will also investigate declarative languages and the application of "constraints" respectively rules that can be inserted in a process e.g. as placeholders instantiated at runtime (see [4]), but this approach is not in the focus of the paper.

5.1 Requirements for ED-BPM Execution

As a result of the taxonomy described in Sec. 3, new demands are recognized regarding to the business process execution languages. An event-driven process execution language must provide appropriate language constructs to implement the behavior patterns listed in para. 3.3. Additionally, in order to receive events of interest, the business process execution environment (including the business process execution language) must allow *subscribing and unsubscribing* to various events. To

support business processes as event producers, a process execution language should provide corresponding constructs for explicit *event reporting*.

Although in many cases it is better to decouple the complex event definition from the process specification. In some cases it would be beneficial to incorporate event patterns into the business process definition. Therefore, event-driven process execution languages should allow incorporating event pattern definitions into the process definition. Different programming languages provide different levels of abstraction [50]. Depending on the abstraction provided by the language, different languages suit better or worse for solving different problems in different domains. The right abstraction for event-driven process definition languages should express aspects of event-driven behavior regarding to the process workflow.

5.2 WS-BPEL as ED-BP Execution Language

WS-BPEL [51] is a common language for process specification and execution for SOA. It is also called a “language for business process orchestration based on web services” [52] because the distributed entities participating in BPEL processes are collaborating by using web service interfaces. In this paper the final version BPEL 2.0 is discussed.

WS-BPEL provides some event-driven behavior capabilities e.g. event handlers, fault and compensation handling mechanisms. However in its standard form it can not completely support event-driven process execution. Interactions between partners in BPEL are peer-to-peer by their nature – even asynchronous ones [51]. The event in BPEL is, in fact, an asynchronous request issued by one of the collaborating parties. The binding between physical entities is performed based on partner link definitions; thus no subscription mechanism is available. There is no construct for publishing or broadcasting a message over the network without targeting any specific consumer in the standard language. Hence no event reporting mechanism is available. The use of events and event patterns in a business process is impossible with BPEL because of the lack of appropriate constructs and mechanisms as well as a semantic mismatch of the event concept. Last but not least, the level of abstraction provided by the WS-BPEL language corresponds to the aspects of web services orchestration rather than event-driven behavior.

Although WS-BPEL in its standard form can not execute event-driven processes, developing a completely different execution language for ED-BPM would be problematic because the adoption costs for existing enterprises that currently use WS-BPEL might be too high.

5.3 Enhancing WS-BPEL for ED-BPM

The following explains an approach of enhancing WS-BPEL for supporting event-driven business processes.

Event Subscription: To provide event subscription, we introduce a new element called *eventSubscriptions*, under a *scope* or a *process* element. This element contains a list of subscriptions to be activated before starting the first activity enclosed in the

scope. Each *subscription* element specifies the event source, expiration period, and an optional filter expression to narrow the stream of the incoming events down to those of interest in the current scope.

Event Reporting: We introduce the *reportEvent* extension activity in order to allow a business process to report events. It explicitly reports about events that occur at the “business level”. The *reportEvent* element specifies the qualified name of the reported event, as well as how the event contents should be built. The event message can be composed based on a process variable, a part of a variable, or an expression involving multiple variables.

Event Patterns: The *eventPattern* element declares an event pattern inside the process definition. The WS-BPEL standard allows using pluggable expression language in expressions. We adopt this approach in regard to the expression languages for event patterns or event filter expressions (used in the *eventSubscription* element). In the absence of a standard common language for event processing this approach is even more appropriate.

The *expressionLanguage* attribute specifies the expression language used to define the pattern. The *pattern-expr* element specifies the pattern expression using the corresponding expression language. The list of (optional) *to-spec* elements allows initializing variables with the values derived from event instances that made the pattern.

```
<edbpel:eventPattern expressionLanguage="anyURI">
  pattern-expr
  to-spec*
</edbpel:eventPattern>
```

Other enhancements: We further use the enhancements introduced above to extend other elements of BPEL. All the message-handling activities (e.g. receive, pick) as well as event handlers are extended to allow the specification of an event or event pattern instead of an incoming request to serve as a trigger for the activity.

```
<onEvent>
  <edbpel:eventPattern
    expressionLanguage="http://example.com/evtPatternLanguage">
    <all>
      <event name="et:TradingOpportunity" alias="to" />
      <where>to.instr.name == "Example"</where>
    </all>
    <to variable="TradOptInstrValue">
      <query queryLanguage=
        "http://example.com/evtPatternLanguage">
        to.instr.value
      </query>
    </to>
  </edbpel:eventPattern>
</scope ...>...</scope>
</onEvent>
```

5.4 Implementing the ‘Fraud Management’ Use Case

In para. 3.3 we described the ‘Fraud Management’ use case. This use case serves as basis for transforming it into a set of executable processes implemented in WS-BPEL enhanced as described above. Details of the implementation for all of the patterns of the event-driven behavior shown in the scenario are provided in our future work. Instead, we demonstrate five patterns as a short illustration, leaving the rest for a separate discussion.

Pattern 1: Stop one or more running processes upon event

This pattern implements item 5 of para. 3.3. This could be achieved by using the fault handling mechanism available in WS-BPEL [51]. In order to activate the corresponding fault handler we can use the event handler that is activated by a complex event occurrence. The event handler throws a fault that stops all the currently running activities. A strong benefit of using a loosely coupled event-driven solution is the ability to stop any number of running processes upon a single event: all the processes subscribed to this event will be affected. This is opposed to a ‘conventional’ peer-to-peer approach that would require explicit sending of a request to each one of the processes that should be stopped.

Pattern 2: Start a new process instance upon event

This pattern implements items 6, 7, and 9 of para. 3.3. Standard WS-BPEL allows starting a new process instance upon an incoming message (via the *receive* element). We extend the *receive* element with the ability to specify an event name or an event pattern instead of a message. In order to activate a new process instance the enhanced *receive* version could be used. If there are several possible events or messages that should start the same process the *pick* element can be used in a similar manner.

Pattern 3: Activate a task or a process in the absence of the expected event within some time period

This pattern implements item 8 of para. 3.3. The absence of some event is an event pattern by itself. It can be specified in an event processing language and detected by an event processing system. Therefore, this pattern is a special case of the above one (start a new process upon event).

Pattern 4: Suspend a process and resume upon event

This pattern can be used to suspend processes until a certain event occurs as described in items 7 and 8 of para. 3.3. This pattern can be implemented using one of the synchronous BPEL constructs, i.e. *receive* or *pick*. For this specific use case (‘fraud’) *pick* is preferable because it allows handling time out situations, that is what happens if the ‘false positive’ decision is never made.

Pattern 5: Start a process modification/adaptation upon event

This pattern solves case 10 of para 3.3. The pattern can be implemented using the enhanced version of the event handler mechanism. The event handler is activated upon an instance of the specified trigger event (e.g. ‘Too Many False Positives’). The enclosed activity can modify the BPEL variable that participates in the ‘fraud detection’ pattern calculation, or submits an update request to the event processing system. If the case requires modifying the workflow, the workflow adjustment could be considered as a separate process that is started upon this event. As soon as the new process version is available, it will be activated. The process and possible issues of

activating a modified process in presence of an older version of the process is out the scope of this paper.

6 Conclusions

This paper provided an impression of the components and requirements needed for ED-BPM which is elaborated by providing a reference model and reference architecture. After analyzing the presented model and architecture it followed the identification that current business process execution languages need extensions to provide the required capabilities as disclosed in an exemplified financial use case for event-driven behavior. This example serves on the one hand to prove the concept of the (Re-) Active Internet of the Future. On the other hand the BPEL enhancements as an example of an execution language provide a first step into the direction of the (Re-) Active Internet, whereas many other issues have to be solved in processing these events. Past, current and future movements within the according standards and their possible influences show a broad interest both on the modeling and the execution side. The ED-BPM reference model, the enhanced NEXOF-RA, the necessary involved extensions of business process modeling notations and business process execution languages serve as the basis for integrating complex events providing the ability for influencing and dynamically changing business processes. Yet the similarities and diversities of different application domains need to be identified and investigated in the future.

References

- [1] European Future Internet Portal, <http://www.future-internet.eu>
- [2] The Future Of Internet, <http://www.fi-bled.eu>
- [3] Li, M.-S., Crave, S., Müller, J. P., Willmott, S.: The Internet of Services: Vision, Scope and Issues. In: eChallenges e-2008 Paper No. 143; eChallenges e-2008 Paper No. 143: <http://ssrn.com/abstract=1293722> (2008)
- [4] Mulyar, N.A., Schonenberg, M.H., Mans, R.S., Russell, N.C., Aalst, W.M.P. van der.: Towards a taxonomy of process flexibility (extended version). BPM Center Report No. BPM-07-11, Brisbane/Eindhoven (2007)
- [5] Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems, Addison-Wesley Professional (2002)
- [6] Ammon, R. v., Emmersberger, C., Springer, F., Wolff C.: Event-Driven Business Process Management and its Practical Application Taking the Example of DHL, http://icep-fis08.fzi.de/papers/iCEP08_8.pdf, Future Internet Symposium, Vienna (2008)
- [7] Ammon, R. v., Emmersberger, C., Springer, F.: "Event-Driven Business Process Management" - Eine neue Technologie und erste Projekte am Beispiel der DHL, OBJEKTSpektrum 06/2008, http://www.sigs.de/publications/os/2008/06/ammon_OS_06_08.pdf, SIGS-DATACOM (2008)
- [8] Ammon, R.v.: Event-Driven Business Process Management. Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag (2008)

- [9] Ammon, R. v., Silberbauer, C., Wolff, C.: Domain Specific Reference Models for Event Patterns – for Faster Developing of Business Activity Monitoring Applications. VIPSI 2007, Lake Bled, Slovenia, 8-11 October 2007
- [10] Albek, E., Bax, E., Billock, G., Chandy, K. M., Swett, I.: An Event Processing Language (EPL) for Building Sense and Respond Applications. In: Proceedings of the 19th IEEE international Parallel and Distributed Processing Symposium (Ipdps'05) - Workshop 2 - Volume 03 (April 04 - 08, 2005). IPDPS. IEEE Computer Society, Washington (2005)
- [11] Brandl, H.-M., Guschakowski, D.: Complex Event Processing in the context of Business Activity Monitoring. An evaluation of different approaches and tools taking the example of the Next Generation easyCredit. Diploma thesis, Preworkshop DEBS07, http://www.citt-online.de/downloads/Diplomarbeit_BaGu_Final.pdf (2007)
- [12] Networked European Software & Services Initiative, NESSI Brochure, Retrieved February 15, 2009 from http://www.nessi-europe.eu/Nessi/Portals/0/Nessi-Repository/Publications/Flyers/2005_09_NESSI_Brochure.pdf (2005)
- [13] Corte, P., Desideri, D.: NEXOF RA, Definition of an architectural framework & principles, Retrieved February 24, 2009 from http://www.nexof-ra.eu/sites/default/files/D7.2_Definition_of_an_architectural_framework_principles.doc (2008)
- [14] NEXOF RA, NEXOF Reference Architecture, Retrieved February 23, 2009, from <http://www.nexof-ra.eu/?q=node/1> (2008)
- [15] Ammon R. v., Emmersberger C., Ertlmaier T., Etzion O., Paulus T., Springer F.; Existing and Future Standards for Event-Driven Business Process Management. DEBS 2009, Nashville (2009)
- [16] Heintz, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., Teschke, M.: A comprehensive approach to flexibility in workflow management systems. In: WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration, pages 79–88, ACM, New York (1999)
- [17] Regev, G., Wegmann, A.: A regulation-based view on business process and supporting system flexibility. In: Workshop on Business Process Modeling, Design and Support (BPMDS05), Proceedings of CAiSE05 Workshops, pages 35–42, Porto (2005)
- [18] Reijers, H. A.: Workflow flexibility: The forlorn promise. In: 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), pages 271–272. IEEE Computer Society, Manchester (2006)
- [19] Daoudi, F., Nurcan, S.: A benchmarking framework for methods to design flexible business processes. Software Process Improvement and Practice, 12:51–63, Trier (2007)
- [20] Snowdon, R. A., Warboys, B. C., Greenwood, R. M., Holland, C.P., Kawalek, J. P., Shaw, D. R.: On the architecture and form of flexible process support. Software Process Improvement and Practice, 12:21–34, Trier (2007)
- [21] Regev, G., Bider, I., Wegmann, A.: Defining business process flexibility with the help of invariants. Software Process Improvement and Practice, 12:65–79, Trier (2007)
- [22] Regev, G., Soffer, P., Schmidt, R.: Taxonomy of flexibility in business processes. In: Proceedings of the 7th Workshop on Business Process Modeling, Development and Support (BPMDS'06), Luxembourg (2006)
- [23] Regev G., Wegmann, A.: Business process flexibility: Weick's organizational theory to the rescue. In: Proceedings of the 7th Workshop on Business Process Modeling, Development and Support(BPMDS'06), Luxembourg (2006)
- [24] Bider, I.: Masking flexibility behind rigidity: Notes on how much flexibility people are willing to cope with. In: Proceedings of the CAiSE'05 Workshop, p. 7-18., Porto (2005)
- [25] Soffer, P.: On the Notion of Flexibility in Business Processes. In: Proceedings of the CAiSE'05 Workshop, p. 35 – 42, Porto (2005)
- [26] S-Cube, <http://www.s-cube-network.eu/>

- [27] Kazhamiakin, R.: Adaptation and Monitoring in S-Cube: Global Vision and Roadmap. Proceedings of the Mona+ Workshop, ServiceWave2008, Madrid, p. 67-76, <http://www.s-cube-network.eu/news/s-cube-mona-proceedings-published> (2008)
- [28] Ammon, R.v., Etzion, O., Paschke, A., Stojanovic, N.: Event-Driven Business Process Management. ED-BPM Workshop, ServiceWave2008, Madrid, Retrieved September 20, 2009 from: <http://www.nessi-europe.com/Nessi/AdminP/ArchivedPages/ServiceWave2008WorkshopBPM/tabid/468/Default.aspx> (2008)
- [29] 1st European Conference on Software Services and SOKU technologies, <http://www.eu-ecss.eu/contents/conference/exhibitors-ssoku09/view>
- [30] Liu, D., Pedrinaci, C., Domingue, J.: Semantic Enabled Complex Event Language for Business Process Monitoring. Proceedings of the 4th SBMN-Workshop, collocated with ESWC 2009, 31 May to 4 June 2009 on Crete, Greece (2009)
- [31] Ammon, R. v., et al., 1st European Conference on Software Services and SOKU technologies: http://www.citt-online.de/downloads/Flyer_SSOKU_BankingFraud.ppt, SSOKU09, Brussels (2009)
- [32] Widder, A., Ammon, R. v., Schaeffer, P., Wolff, C., Identification of suspicious, unknown event patterns in an event cloud, ACM International Conference Proceeding Series; Vol. 233, Proceedings of the 2007 International Conference on Distributed Event-Based Systems, Rome (2007)
- [33] Widder, A., Ammon, R. v, Hagemann, G., Schönefeld, D.: An Approach for Automatic Fraud Detection in the Insurance Domain, AAAI 2009 Spring Symposia / Intelligent Event Processing, March 23-25, Stanford (2009)
- [34] Paulus, T., Zacharias, R., Scheider, H., Wolff, C.: Fraud Management and Notification Event Architecture for Retail (NEAR) Standard and First Experiences with Point of Sales/Point of Services at Wincor Nixdorf. Service Wave 2008, Madrid, (2008)
- [35] Paulus, T.: Tutorial on Event Processing Use Cases, ED-BPM in the Retail Domain - Taking the Example of Fraud Management. 3rd ACM International Conference on Distributed Event-Based Systems 2009, Nashville (2009)
- [36] OMG, Business Process Management Initiative, <http://www.bpmn.org>
- [37] Allweyer, Th.: BPMN – Business Process Modeling Notation. Einführung in den Standard für die Geschäftsprozessmodellierung. Books on Demand GmbH, ISBN 978-3-8370-7004-0 (2009)
- [38] Decker, G.; Grosskopf, A.; Barros, A.: A Graphical Notation for Modeling Complex Events in Business Processes, Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International, Volume , Issue , 15-19 Oct. 2007 Page(s):27 – 27, Annapolis (2007)
- [39] Object Management Group, Event Metamodel and Profile (EMP) Request For Proposal, Retrieved February 24, 2009 from: <http://doc.omg.org/ad/2008-9-15> (2008)
- [40] Fleischmann, A.: Distributed Systems – Software Design and Implementation, Springer-Verlag, Berlin (1994)
- [41] Schmidt, W., Fleischmann, A., Gilbert, O.: Subjektorientiertes Geschäftsprozessmanagement. Praxis der Wirtschaftsinformatik, HMD, Heft 299, dpunkt.verlag, Page 52, Heidelberg (2009)
- [42] 2nd ED-BPM Workshop at the Service Wave 2009, 24-27 November 2009, Stockholm, <http://www.icsoc.org/> (2009)
- [43] Sinur, J./Gartner, Getting Painted in a Corner by Structured Business Processes, Retrieved September 20, 2009 from: http://blogs.gartner.com/jim_sinur/2009/08/06/getting-painted-in-a-corner-by-structured-business-processes/ (2009)
- [44] OASIS, Web Services Business Process Execution Language (WSBPEL) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- [45] OASIS, WS-BPEL Extension for People (BPEL4People) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bpel4people

- [46] Decker, G., Kopp, O., Leymann, F., Pfitzner, K., Weske, M.: Modeling Service Choreographies using BPMN and BPEL4Chor. Proceedings CAISE 2008, Montpellier (2008)
- [47] Weske, M: Business Process Management: Concepts, Languages, Architectures. Springer Verlag, Berlin, ISBN 978-3540735212 (2007)
- [48] Barros, A., Decker, G., Dumas, M. Multi-staged and Multi-viewpoint Service Choreography Modeling. Technical Report 4668, Queensland University of Technology, Brisbane, Australia (2006)
- [49] XPD L Support and Resources, <http://www.wfmc.org/xpdl.html>
- [50] Abelson, H., Sussman, G., Sussman, J.: Structure and Interpretation of Computer Programs. Second edition, The MIT Press (1996)
- [51] OASIS, Web Services Business Process Execution Language Version 2.0, Committee Draft. January 25, 2007 Retrieved February 14, 2009 from: <http://docs.oasis-open.org/wsbpel/2.0/> (2007)
- [52] OASIS, Web Services Business Process Execution Language Version 2.0 Primer. May 9, 2007. Retrieved February 14, 2009 from: <http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf> (2007)