Enabling On-the-fly Business Process Composition through an Event-based Approach

Nancy Alexopoulou[1], Mara Nikolaidou[2], Yannis Chamodrakas[1], Drakoulis Martakos[1]
[1]*Department of Informatics & Telecommunications, University of Athens, Athens, Greece*
[2]*Harokopio University of Athens, Athens, Greece*
nanci@di.uoa.gr mara@hua.gr, ihamod@di.uoa.gr, martakos@di.uoa.gr

Abstract

Agility is a key characteristic for modern enterprises operating in a highly dynamic environment. The attainment of agility is a difficult issue involving all organizational aspects, such as business processes, information systems, personnel, organizational structure, etc. The ability to respond rapidly to changes by utilizing agile business processes is an important constituent of an agile enterprise. The focus of this paper is business process agility outlining the benefits of event-based business process modeling. To this end, an event-based approach is proposed enabling on-the-fly composition of business processes. The introduced approach adopts the concepts of Complex Event Processing. A conceptual architecture for an information technology infrastructure supporting event-driven business process execution is also proposed.

1. Introduction

Nowadays, organizations operate in highly turbulent environments having to cope with a frenetic pace of change [1]. Globalization and continual technological evolution are the main drivers of this turbulence. Other change factors include political issues, deregulation, consolidation in the business network, etc. [2]. As firms continuously sense opportunities for competitive action in their product-market spaces, it is agility which underlies firms' success in continuously enhancing and redefining their value creation in highly dynamic environments [3].

Indeed, agility has been recognized as a key characteristic for a modern enterprise. It has been therefore the concern of numerous researchers [4, 5, 6], who try to specify the characteristics of an *agile enterprise* - or *sense-and-respond enterprise*, as alternatively occurs in the literature - and suggest methods or means that ensure agility.

The ability to respond rapidly to changing marketing opportunities by utilizing agile business processes is a critical attribute of an agile enterprise [7]. Enterprise agility from the business process perspective means that the enterprise is able to adjust its processes easily, in a timely and cost-effective manner, and efficiently execute them, in order to meet new market demands. This entails an appropriate method for business process modeling that will ensure agility, enabling on-the-fly composition of business processes. To this end, we propose an approach for an event-based business process description and explain how such an approach facilitates agile business process execution.

Business process agility is discussed in section 2. Section 3 outlines the benefits of event-based business process modeling. The event-based approach is introduced in section 4, while a conceptual architecture for an IT infrastructure supporting event-driven business processes is presented in section 5. Conclusions lie in section 6.

2. Business Process Agility

The ability to easily and cost-effectively modify a business process presupposes flexibility to modify the *business process definition*. Business process definition is the formal and precise description of the elements composing the business process. In particular, a business process includes a set of *actions* which may regard internal operations or business-to-business transactions. Each action may be hierarchically decomposed to sub-actions. Actions are executed by actors (users, systems, etc.) that take over specific *roles*. The sequence and execution of actions are governed by *events* which play the role of control signals, and *conditions*, which specify different paths of actions i.e. *routes*. During the execution of an action, each process both uses and produces *resources*. Resources involve anything necessary for the accomplishment of an action such as data, devices, even people. The resources used represent process input, while those produced represent its output. Every business process should always have a final output which constitutes the purpose of its execution.

Flexibility to modify the business process

This paper is part of the 03ED470 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (25% from the Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.U.-European Social Fund).

definition implies that the definition is agile itself, allowing even the radical change of the business process. Currently, there are several approaches for business process definition using various modeling techniques, aiming at the attainment of agility [8] [9].

Based on the business process elements mentioned above, we argue that agility in the business process definition means the ability to

- add new actions or delete actions that are no longer necessary,
- change process sequence by rearranging, eliminating or inserting routes,
- redefine roles in order for example to reflect changes in the organizational structure,
- modify conditions so as to apply, for example, new regulations or new policies adopted by the company,
- handle events to ensure smooth flow of the business process,
- use alternative resources, if those specified initially are not available, as well as map resources to roles in an ad hoc manner, as may be imposed by unexpected conditions.

The achievement of maximum agility however implies that the modifications mentioned above can take place at run time as well, i.e. while a business process instance is being executed. ShuiGuang et al. [10] proposed such a method. According to this method, a business process is composed of *general activities*, which are predefined in detail at design time and *flexible activities,* which are like a "black box", representing an undetermined sub-process without detailed specification at build time. In other words, flexible activities encapsulate the uncertain sub-process at run time. At run time, depending on current circumstances, a flexible activity can be replaced by a concrete sub-process composed of selected activities from existing or newly added ones (constituting a pool of activities), based on selection and composition constraints.

The above pool of activities implies the ability to change a business process at run time upon a predicted event for which a relative activity exists in the pool. The ability to change a business process at run time upon an unpredicted event reflects the highest grade of agility, as depicted in Figure 1. On the other hand, the lowest grade corresponds to the ability to change a business process at build time upon a predicted event. The other two cases (run time modification upon predicted events and build time modifications upon unpredicted events) are regarded of equivalent agility as the former has the parameter of run time while the second has the parameter of unpredicted. Obviously the greatest challenge is the implementation of *super agile* business processes.

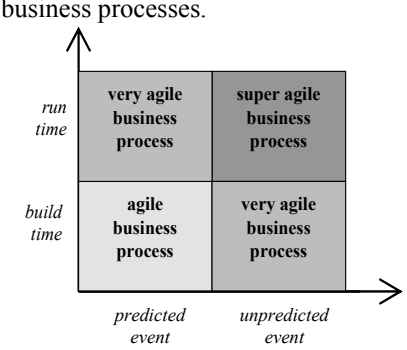

Figure 1. Scaling agility in business processes

Most methodologies, such as that previously discussed, may ensure agility to some extent but they cannot support super agile business processes, i.e. fulfill agility at run time upon unpredicted events. This shortcoming stems from the fact that in most methodologies business process description is action-centric. Consequently, business process logic is organized around a specific set of actions which are associated with roles and resources and are interrelated through conditions and events to produce specific flow routes. This set of actions is translated into an executable format as a single unit, hindering thus agility to a significant extent.

However, the dynamic environment organizations operate in is event-driven in nature [11]. Consequently, we believe that the way an agile business process should be conceived must be compatible with this nature and thus be event-driven. Therefore, instead of focusing on how to collect and organize the right actions to produce a business process, our focus is on what events occur and which actions are initiated by them. From the elements composing a business process definition, we distinguish events as playing critical role for attaining agility. Following an event-driven logic, execution would be on event-action basis and not on a whole set of actions. As a result, intervening in the execution of a business process could be greatly facilitated. In fact, the business process would be composed on the fly based on the events occurring. Breaking down a business process into a number of autonomous event-action pairs may increase business process modularity and facilitate thus the implementation of the modifications concerning the business process definition.

Event-based approaches for business process description are not yet as common as action-

centric approaches. However, their virtues in respect to agility are getting recognized [12].

3. Facilitating Agility through Event-based Business Process Modeling

The fundamental concept in an event-driven organizational environment is that of *event.* An event is a notable thing that happens inside or outside the enterprise [13]. Events are generated either as a result of the completion of an action or because other events have happened. As such, an event may either initiate an action, or cause another event, or both. The first case holds when the event needs to be handled somehow, while if this is not the case, it may just cause the occurrence of another event. Consider, for example, the event "Rain starts". This event may not necessarily require an action to be taken. However, it may cause the event "The courtyard flooded". The latter will probably initiate the action "Remove water from the courtyard". This example reveals that events can be interrelated through causality relations. Although it presents a simple causality relation, there can be more complex combinations among events. An event, for instance, may occur because two other events happened, or because two events occurred and another did not. In such cases, the generated event is referred as *complex event* [11]. The notion of defining and utilizing relationships (such as AND, OR etc.) between events is characterized as *Complex Event Processing* [11].

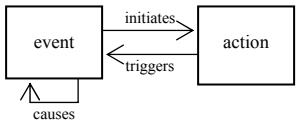

Figure 2. The basic concepts of event-based business process modeling

According to our perspective, the basic concepts of event-based business process modeling are depicted in Figure 2. As indicated in Figure 2, an event may cause another event either directly or indirectly through actions that it may initiate. The event-based logic may be applied for the development of a design technique for business process modeling completely different from that indicated by action-centric approach. Such a technique is presented in this paper.

Event-based business process modeling, as described in the following, involves identifying the meaningful events and explicitly relating them to actions they initiate. Every event-action pair constitutes an autonomous relationship and its definition range lies beyond a specific business process it participates in. This implies that an event is always associated with the same set of actions independently of the business processes in which it is involved. As it will be further explained, business processes are not explicitly defined, rather they evolve during run time depending on the events occurring, taking into account predefined event-action pairs. Such feature is particularly useful when modeling business processes that are highly volatile.

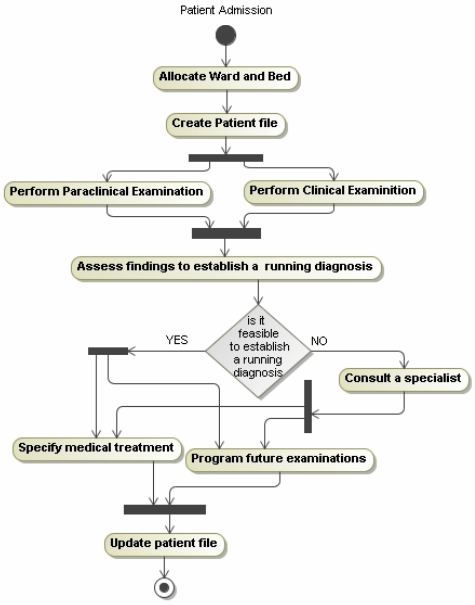

Figure 3. Action-centric description of "Patient Admission" medical process

Medical processes are a typical example of volatile processes. Due to their nature, they are characterized by high variability expressed by modifications in patient management and treatment, as well as by frequently arising emergencies which prevent the execution of regular steps. To elucidate benefits of Event-Action (EA) approach (see figure 2), a simplified medical process is used as an example. Patients usually arrive at the hospital's emergency department, where they get examined and it is decided whether they need to be hospitalized. If this is the case, the patients are admitted to an appropriate hospital clinic. Figure 3 presents a medical process involving the steps followed when a patient is admitted to a hospital clinic. As shown in Figure 3, when a patient is admitted to the clinic, he/she is allocated to a bed in a specific ward and then a patient file is created to keep all relative medical information. After that, clinical and paraclinical examinations are performed and then the findings are assessed so that a running diagnosis can be established. In

such a case, the medical treatment is specified, while future examinations are also programmed. If a running diagnosis is not feasible, a specialist's consultation is acquired and then, according to the diagnosis provided, the medical treatment is specified and future examinations are programmed. The process ends after the patient file is accordingly updated. In this process, some steps are carried out by humans, such as the clinical examination, while others are accomplished by software systems such as bed allocation.

Table 1. Events and Actions required to model "Patient Admission" medical process

Events	Initiated Actions	Triggered Events
E1:Patient Admission	A1: Allocate Ward and Bed	E2: Ward and Bed Allocated
E2: Ward and Bed Allocated	A2: Create Patient File	E3: Patient File Created
E3: Patient File Created	A3: Perform Paraclinical Examinations	E4: Paraclinical Examinations Completed
	A4: Perform Clinical Examination	E5: Clinical Examination Completed
E6: Paraclinical and Clinical Examinations Completed (where E6=E4 AND E5)	A5: Assess Findings to Establish a Running Diagnosis	E7: Running Diagnosis Feasible OR E8: Running Diagnosis not Feasible
E7: Running Diagnosis feasible	A6: Specify Medical Treatment	E9: Medical Treatment Specified
	A7: Program Future Examinations	E10: Future Examinations Programmed
E11: Medical Treatment Specified and Future Examinations Programmed (where E11= E9 AND E10)	A8: Update Patient File	E12:Patient File Updated
E8: Running Diagnosis not Feasible	A9: Consult a Specialist	E7: Running Diagnosis Feasible
E13: Patient admitted (where E13= E1 And E12)		

In a non event-based approach, the business process is modeled through activity-based diagrams, such as those provided by Unified Modeling Language [14]. Such a diagram is presented in Figure 3. In contrast, following an EA approach, business process functionality is modeled as a set of events and corresponding initiated actions.

In EA approach, the modeler concentrates on specifying autonomous event-action pairs not a whole business process. In such a case, the modeler may separately identify events and actions (specifying also for each action the triggered events) and then appropriately associate the former with the latter. After the definition of event-action pairs, the possible business processes could be automatically generated in the form of event-action sequences with the support of appropriate modeling software.

The required events for the medical process depicted in Figure 3 are listed in Table 1. Each event is related to one or more actions that have to be invoked upon the event occurrence. The relative actions are listed in the middle column of Table 1. The right column includes the events triggered by each action. As indicated in Table 1, event E6 is a complex event since it has been generated because of the occurrence of events E4 and E5. Likewise, E11 and E13 are also complex events. It should be noted that event E13 signifies the end of the process. It is a final event and as it will be explained in the following it does not initiate actions.

Tracking the event-action sequence extracted from Table 1, it is obvious that at run time, based on the events that will occur, two event-action sequences are possible to evolve, as presented in Table 2.

Table 2. Event-Action Sequences

Sequence 1	E1→A1→E2→A2→E3→(A3, A4)→E6→A5→E7→(A6, A7)→E11→A8→E12→E13
Sequence 2	E1→A1→E2→A2→E3→(A3, A4)→E6→A5→E8→A9→E7→(A6, A7)→E11→A8→E12→E13

The event-action sequences are characterized as event-driven process chains in the EPC (Event-driven Process Chain) modeling method [15]. The EPC method, as supported within ARIS framework [16], is used for modeling, analyzing, and redesigning business processes. Although the EPC method uses the concept of events, business processes are conventionally defined at design time, since the focus is not on run time business process agility. Therefore, it does not include the concept of autonomous event-action pairs. In the medical process of Figure 3, event E1 (*Patient Admission*) for example causes action A1 (*Allocate Ward and Bed*). However, if E1 appears within another business process, it can only cause action A1 again.

At run time, each event is independently processed by an event processing engine. In contrast, in traditional methods, the business process is translated as a single unit into a

machine executable language like Business Process Execution Language (BPEL) [17] and is executed using a business process engine. Business processing engine operates like a compiler while event processing engine acts analogously to an interpreter. In the latter case, the business process is composed on the fly during execution time, based on the events occurring. Therefore, changes at execution time are better supported when using the event-based approach. Consider for example the case that the medical process of Figure 3 needs to be modified by adding another step right after action A9 (*Consult a Specialist*) that would concern performing special paraclinical examinations. Such a change would require redefining action and event types, as well as event-action pairs. First, an action A10: *Perform Special Paraclinical Examinations* should be defined that will trigger event E7 (*Running Diagnosis Feasible*). Then the definition of action A9 should be updated so that A9, instead of triggering only event E7, will trigger either E7 or a new event E14: *Special Paraclinical Examinations Necessary*. Lastly, E14 must be related to action A10. It should be noted that these modifications can be accomplished even at run time after events E1 to E8 have occurred. Such a change, which concerns the definition of the business process, could not be performed at run time using the traditional methods, since in the latter, as already stated, the business process definition is executed as a single unit. Lastly, it should be stressed that changes in event-action pairs occurring in a business process will hold for every other business process in which these pairs are involved. If this is undesired, new event or action types may have to be defined.

Apart from run time changes at business process definition the event-based approach facilitates also changes that concern only a specific business process instance. Consider for example the case a patient that has been admitted, suddenly complains that he feels an intense chest pain. If this occurs for example by the end of action A2 (*Create Patient File*), the regular steps cannot be followed. Instead of actions A3 (*Perform Paraclinical Examinations*) and A4 (*Perform Clinical Examination*), a doctor - physician - should look into the case and if necessary the patient should be conveyed to the Intensive Care Unit. Such a case requires immediate human intervention. Using an appropriate graphical user interface, the responsible employee may generate the appropriate events through which the current evolving process will be terminated and a new process will commence, suitable for dealing with the arisen emergency. Apparently, as also implied in [18], human intervention is critical for the acquirement of business process agility. Consider another example where the system which performs bed allocation is down. In that case, a relative employee could do the allocation manually and then generate, through the aforementioned interface, the appropriate events so that the process can continue normally. In the worst case that the systems offering services to a specific business process are not available, their operations can be carried out exclusively by humans who will also have to generate the necessary events through such an interface. In highly dynamic environments, the humans' role in respect to automated business processes must be reconsidered.

4. An event-driven approach for business process modeling

The main idea of the introduced approach is that the relations between events and actions, though defined relative to a business process context, are essentially independent of the business process itself. In this sense, they are uniquely defined independent of the business processes they participate in. This, as already stated, means that an event is always associated with the same set of actions. Through this property, agility may be significantly facilitated as the notion of business process does not exist at build time. Instead, it is composed at run time based on the events occurring.

One could claim however that an event may cause different set of actions depending on certain conditions that may hold in a specific business process context. This is supported in the proposed approach in an implicit manner. Suppose for example that an event E1 initiates action A1 if condition a>b is true. In the proposed approach the latter case is modeled as follows: an action A2 that checks *a* and *b* is defined. If this condition becomes true, A2 triggers an event E2 signifying that *a* is greater than *b*, which, in turn, in conjunction with event E1 initiates action A1. This conjunction is expressed by a complex event, say E3. The EA-based model for business process modeling that we propose is in contrast to ECA (Event-Condition-Action) [19] model, originally used within the active database community [20, 21].

In ECA model, when an event occurs, the

condition is evaluated. If the condition is satisfied, the action is executed. In order to understand the difference between ECA and EA, consider the following example. When a patient has fever and is also coughing, then a chest x-ray is performed, unless the patient is a pregnant woman. According to ECA logic when the event "Patient has fever and is coughing" occurs the condition "Patient not pregnant" is examined. If the condition is satisfied, the action "Perform chest x-ray" is realized. On the other hand, following EA approach, action "Perform chest x-ray" is initiated by two events which are "Patient has fever and is coughing" and "Patient not pregnant" (note that the latter is essentially the opposite of event "Patient pregnant").

ECA is a well established and extensively applied concept in business process modeling and workflow approaches [22, 23, 24]. We are exploring EA model since it may prove more efficient regarding business process agility. As EA model is based on only two entity types combined in autonomous pairs, it increases modularity in business process description. Also, incorporating business rules - expressed in ECA by conditions - within event processing, simplifies the way events and actions are interrelated. This promotes flexibility in the integration with the underlying system infrastructure, as the latter would deal only with implementing the functionality for each action without considering any condition implementation. Lastly, it should be noted that expressing conditions through events further enriches the "event cloud", as referred by Luckham in [11], offering thus the opportunity to identify more event patterns that may be meaningful for the enterprise. As information is represented in a uniform manner (through events) it can be more effectively manipulated.

Since events and actions are not defined within the context of a business process, it must be ensured somehow that events/actions belonging to the same business process instance can be identified. This is accomplished through an id value common for all the events associated with a specific process instance as it will be explained in section 4.2. A new id value is assigned at run time to an event that has been defined as *initial.* Initial events are those that may start a sequence of events-actions constituting a business process instance. This id value is propagated to all subsequent events belonging to this sequence until a *final event* is reached. Final events are used to denote the completion of a business process instance and by definition do not invoke actions neither cause other events.

At run time, what actually occurs is an *event instance,* which instantiates an *event type* defined at build time. When an event instance happens, it always initiates the same action instances as explained earlier, specified by its type definition. This is not the case however with *action instances* which may trigger a different set of events. The possible sets of events that an action may trigger are specified by the *action type*. An action type defines mutually exclusive sets of events. For example, an action A1 may trigger either event E1 or E2 or both E1 and E2. Which of these cases will occur is determined at run time, based on the output of the service implementing the action. *Service* regards the implementation of an action and is introduced in order to abstract the implementation considerations from action definition, as adopted by the Object Oriented Paradigm [25]. This abstraction further augments agility, as it will be elucidated in section 5.

4.1 Applying the concepts of Complex Event Processing

Modeling the relations between events and actions is considerably simplified by adopting complex event processing. By employing the concepts of complex event processing, it can be ensured that each action is caused by a single event either actual or conceptual. The latter is introduced to aggregate a number of other events that have occurred and are those actually causing the action. These conceptual complex events encompass all the complexity stemming from action initiation by multiple events. The simplest way to relate events is through the causality relation, e.g. event E1 causes event E2. More complex causality relations may be constructed using the logical operators defined in Boole algebra [26]. As such, AND, OR, NOT, and XOR relations can be defined between events, as shown in Table 3.

Table 3.Combining events to produce complex events

Event Relations	Description
E1→E2	E1 causes E2
E1 LO E2 → E3	E3 is caused by the combination of E1 and E2 through Logical Operators (LO), i.e. AND, OR, NOT, XOR.

The simple causality relation may be used to express event chains. An event chain would be

for example E1→E2→E3, which means that event E1 causes event E2 and event E2 in turn causes event E3. Obviously, in this case, event sequence matters. If this is not the case, events can be related using an AND relationship.

An AND relation can be used to enforce each action to be initiated by one event. Suppose, for example, that an action A is initiated if both events E1 and E2 occur. This case is modeled as indicated in Figure 4. A conceptual event (E3) is introduced to aggregate the two events. This example reveals that the complexity of handling multiple events in order to invoke an action is enclosed exclusively between events.

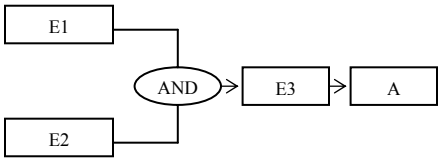

Figure 4. Introducing the conceptual event E3 for the initiation of action A

A specialization of this case is when an action is invoked after an event has occurred a specific number of times. This case corresponds to multiple identical events being related through an AND relationship.

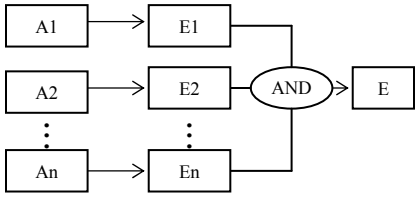

Figure 5. Modeling the case of an event E triggered by multiple actions

It should be noted that the concept of complex events is applied not only in case of events initiating actions but also in case of actions triggering events. Consider for example the case an event is triggered when multiple actions have completed. This case is not directly supported in the proposed approach, as for simplification reasons it is considered that an event is triggered by a single action each time. However, it can be implicitly expressed as shown in Figure 5. Every action is regarded to trigger a single set of events. In Figure 5, action Ai triggers event Ei. Event E is consequently generated as an aggregation of events E1, E2, .. En. This case reveals that all the complexity of the relationship between events and actions is enclosed within complex event processing.

If action A can be invoked either by event E1 or event E2, this would signify an OR relation, which is handled in a similar fashion as the one indicated in Figure 4. Practically, in an OR relation, the action caused will be initiated by the event that will happen first. However, for business process modeling, it is critical that events that will occur later will be ignored, and thus will not re-invoke the action, as this would lead to an erroneous business process execution.

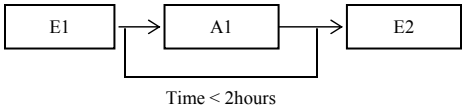

Figure 6. A timing guard between two events

The OR relation can be used for the realization of timing guards. Consider for example the case where an event must be followed by another event in less than 2 hours (Figure 6). This case is modeled as indicated in Figure 7. In this Figure, apart from action A1, another action is also initiated; the one that will keep track of time (A3). A different event (E5) had to be defined as an event always invokes the same actions independently of the business process that participates in. Events E1 (Figure 6) and E5 (Figure 7) are essentially the same event but are associated with different set of actions; E5 initiates A1 and A3, while E1 initiates only A1. As also shown in Figure 7, actions A1 and A3 trigger events E2 and E3 respectively. E3 will indicate that the time has elapsed. If it occurs prior to E2, then E4 will be identical to E3. Otherwise, it will be identical to E2.

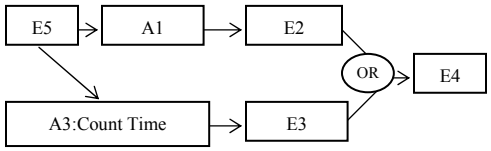

Figure 7. Modeling timing guards

In case the events that occur later are not to be ignored and rather produce an error, an XOR relation must be used instead of OR. An XOR relation dictates that only one of a set of events must occur.

4.2 A metamodel for event-based business process description

The basic concepts of the proposed event-based approach are depicted in the metamodel of Figure 8. The metamodel specifies four entities, namely *Event*, *Complex Event*, *Action* and *Service*, along with their interrelations.

Event is characterized by a *code_number* and a *description*, for example, "Request for Quote". Also, an event is characterized by its *type*, either *initial* or *intermediate* or *final*. As explained earlier, an initial event is used to initiate a business process while a final event is used to signal the completion of a business process. All others involved in a business process are defined as intermediate. The *id* attribute is used to uniquely characterize an event instance, while *process_instance* indicates the process instance the event instance belongs to. Without this information, the correlation of event instances at run time belonging to the same business process instance would not be feasible. *Time_stamp* denotes the time an event instance is created while *source* indicates the entity generated the event instance e.g. a specific component.

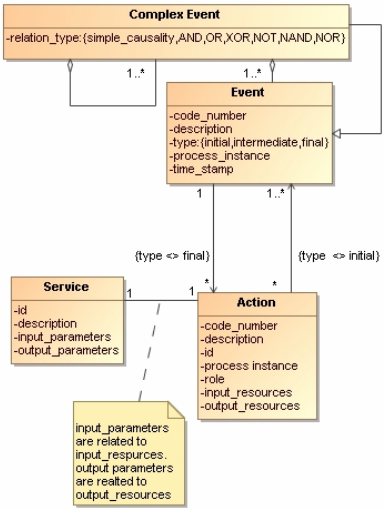

Figure 8. A metamodel for event-based business process description

The notion of an event causing another event is depicted through the aggregation relation between event and complex event entities. Note that the aggregation relationship is also defined between a complex event and itself. This signifies the recursive perspective of event aggregation. For example, events E1 and E2, may cause event E3, which in conjunction with E4 may cause event E5 and so on. Complex event is a specialization of event with the additional attribute *relation_ type*, which specifies how the related events are combined to produce the complex event.

Event is associated with action through two discrete relations. The first dictates that an event that is not final may initiate zero, one or multiple actions. Zero implies that an event causes only events and no actions. This relation specifies also that an action is initiated by a single event. The second association between event and action entities specifies that an action triggers one or more, not initial, events and also that an event is triggered by a single action each time. The same event may be generated by different actions.

Action is characterized by a *code_number*, a *description*, an *id* and a *process_instance* analogously to event. In addition, it includes the *role* attribute which denotes an actor category related to action's execution (e.g. secretary, inventory system, etc.). *Input_resources* and *output_resources* indicate the type of resources used and produced respectively by the action. *Input_resources* are passed to the *input_parameters* attribute of the service that will implement the action. Likewise, *output_parameters* of the service will be passed to the *output_resources* of the action. As indicated by the metamodel, there is a one to one relationship between service and action.

The metamodel does not specify conditions and routes, which were mentioned as constituents of a business process definition in section 2. Conditions, in particular, are implicitly modeled through actions and events as illustrated by the example provided in subsection 3.1. Routes, on the other hand, are not modeled at all, since they are not predetermined at build time. They are deployed at run time, based on events occurring. However to enable the designer to acquire the perspective of the whole business process at build time, the ability to view the event-action chain constituting a discrete business process should be facilitated.

5. A Conceptual IT Architecture Supporting Event-driven Business Processes

Event-driven business process execution requires the support of an appropriate IT infrastructure. A conceptual architecture for such an infrastructure is presented in Figure 9. *Event-handling infrastructure* is considered the heart of such an architecture. The responsibilities of event-handling infrastructure include:

- event manipulation by initiating the appropriate actions
- composition of complex events
- correlation of events belonging to the same business process instance
- monitoring of occurring events
- persistence storage of event models
- decoupling between actions used to define

business processes and their actual implementation

- communication with the appropriate services that will implement the initiated actions
- generation of the appropriate events after the implementation of an action has been completed
- human intervention through a user-friendly interface that will allow users both to manage and generate events.

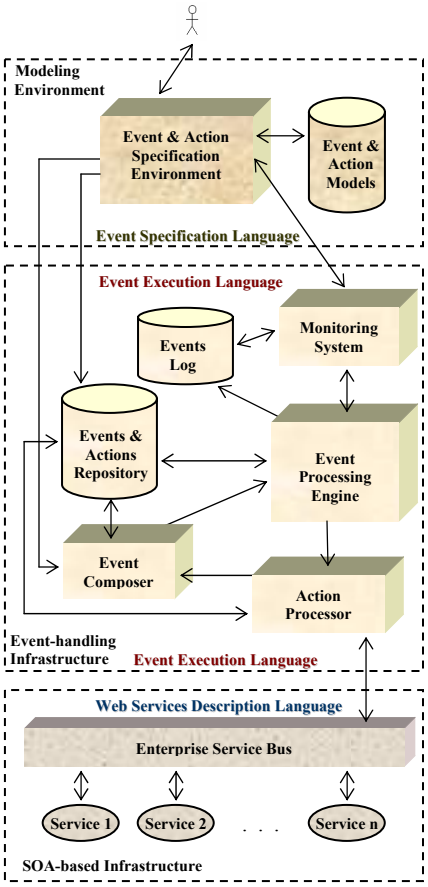

Figure 9. A conceptual architecture for an event-based IT infrastructure

As shown in Figure 9, event-handling infrastructure comprises four basic components, namely the *Event Composer*, the *Event Processing Engine*, the *Action Processor* and the *Monitoring System*. When an event instance occurs, it is received by Event Composer which is responsible for combining events to complex ones based on the information stored in the *Events and Actions Repository*. Event Composer examines whether the received event participates in any event combination using the complex event definitions stored in the repository. If it does not, Event Composer directly forwards it to the Event Processing Engine, while if it does, it keeps it in a record until the other events related to it arrive. When the related events arrive and the complex event is created, it is forwarded to Event Processing Engine. The latter, using the Events and Actions Repository, initiates the appropriate actions related to the event. In specific, it notifies Action Processor of the actions that must be accomplished. Action Processor invokes the appropriate services for action execution. In addition, it is responsible for producing the appropriate event instances based on actions' results. When Action Processor generates the event instances, it forwards them to Event Composer. Events instances, in order to be executed by Event Processing Engine, must be expressed in an *Event Execution Language*. Monitoring System, registers the event instances in the *Event Log*, meaning that it maintains information such as id, process instance, timestamp for every event instance. Through the Monitoring System, it is possible to correlate the event instances belonging to the same process instance, as Event Processing Engine merely executes events without being aware of the whole business process.

Event-handling infrastructure is independent of the underlying IT architecture. This independence is ensured due to the fact that actions constitute essentially a layer of abstraction allowing the event-handling infrastructure to be combined with different underlying IT architectures. In Figure 9, the event-handling infrastructure has been combined with Service-oriented Architecture (SOA) [27]. The harmonized symbiosis between the event-driven logic and SOA has already been identified [28, 29]. In this symbiosis, actions are associated with one or more services that will actually implement the functionality indicated by actions. As such, actions form an abstraction layer between events and services.

The *SOA-based infrastructure* is depicted at the lower part of Figure 9. The *Enterprise Service Bus* [27], which is a fundamental element of a SOA-based infrastructure, contributes to the acquirement of agility as it eliminates the direct dependence between service providers and service consumers.

At the upper part of Figure 9, the modeling environment is presented. It comprises the *Event and Action Specification Environment* through which action and events are modeled. Action and event modeling is performed using an *Event Specification Language*. Event Specification Language accommodates also the definition of the relations between events and actions as well

as the definition of complex events. The integration between event and action models created at build time and the respective executable instances is attained through the mapping between Event Specification Language and Event Execution Language. It should be noted that the interaction between this environment and Event Composer should also be facilitated so that users may intervene in the execution of a business process by generating the appropriate event instances.

6. Conclusions

Event-driven business processes, if properly modeled and adequately supported by the IT infrastructure, may constitute a promising solution for the attainment of the highest level of business process agility, which regards the ability of modifications at runtime upon unpredicted events. An event-based business process modeling approach was introduced in this paper. An analytical description of the main concepts was provided at all levels, including the benefits ensured by its adoption, the modeling perspective of event-driven business processes, as well as the IT infrastructure required to support such processes. Our next steps involve further studying the approach and testing it by applying it to specific business processes. Also, we intend to focus on the Event Specification Language by examining existing endeavors and exploring the capabilities of UML 2.0 [14] extension mechanism for the development of such a language.

7. References

[1] Rockart John, Earl Michael, Ross Jeanne. "Eight Imperatives for the New IT Organization". Sloan Management Review, Vol. 38, No. 1, 1996, pp.43-54.

[2] Oosterhout van Marcel, Waarts Eric and Hillegersberg van Jos. "Change factors requiring agility and implications for IT". European Journal of Inf. Sys. 15, 2006, pp.132-145.

[3] Sambamurthy, V., Bharadwaj, A., Grover, V. "Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms". MIS Quarterly, Vol. 27, No. 2, 2003, pp. 237-263.

[4] Dove Rick 2005. "Agile Enterprise Cornerstones: Knowledge, Values and Response Ability". IFIP 8.6 Keynote, Atlanta, May 2005.

[5] Haeckel, S. H. "Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations". Harvard Business School Press, Boston 1999.

[6] Evgeniou Theodoros. "Building the Adaptive Enterprise. Information Strategies For Successful Management of Complex, Global Corporations in Times of Change". INSEAD, June 2002.

[7] Goldman S.L., Nagel R.N. and Preiss K. "Agile Competitors and Virtual Organizations". New York: Van Nostrand Reinhold, 1995.

[8] Bhat M. Jyoti and Deshmukh Nivedita. "Methods for Modeling Flexibility in Business Processes" Sixth Workshop on Business Process Modeling, Development, and Support (BPMDS'05). Porto, Portugal, June 13-14, 2005.

[9] Ramanathan Jay. "Fractal Architecture for the Adaptive Complex Enterprise". Communications of the ACM, Vol. 48, No. 5, 2005, pp.51-57.

[10] ShuiGuang, D., Zhen, Y., ZhaoHui, W., LiCan, H. "Enhancement of Workflow Flexibility by Composing Activities at Run-time".Proc. of the 2004 ACM Symp. on Applied Computing, pp.667-673.

[11] Luckham David. "The Power of Events". Addison-Wesley, 2002.

[12] Schulte Roy. "Using Events for Business Benefit" Business Integration Journal, May 2004, pp 43-45.

[13] Michelson M. Brenda. "Event-driven Architecture Overview". Patricia Seybold Group 2006. http://dx.doi.org/10.1571/bda2-2-06cc.

[14] OMG Inc, 2007. "Unified Modeling Language: Superstructure". Version 2.1.1, 3/2/2007.

[15] G. Keller, M. Nuttgens, and A.W. Scheer. "Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK)". Ver¨offentlichungen des Instituts f¨ur Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrucken, 1992.

[16] A.W. Scheer. "ARIS-Business Process Modeling". 2nd ed. Berlin: Springer 1999.

[17] Andrews T. et al. "Business Process Execution Language for Web Services Version 1.1" IBM 2003, BEA Systems, International Business Machines Corporation, Microsoft Corp., SAP, Siebel Systems.

[18] Lin, J.Y.C.; Orlowska, M.E., "Partial completion of activity in business process specification". Proc. of IRMA 2005, San Diego, USA, pp. 186-189.

[19] U. Dayal, M. Hsu, and R. Ladin, "Organizing Long-Running Activities with Triggers and Transactions". Proceedings of ACM International Conference on Management of Data,1990,pp.204-214.

[20] J. Widom and S. Ceri. "Active Database Systems: Triggers and Rules For Advanced Database Processing". Morgan Kaufmann, 1996.

[21] N. W. Paton and O. Diaz. "Active Database Systems". ACM Computing Surveys, Vol. 31, No. 1, 1999, pp. 63–103.

[22] Julian Jang, Alan Fekete, Paul Greenfield, Surya Nepal . "An Event-Driven Workflow Engine for Service-based Business Systems", Proceedings of the 10th IEEE International EDOC Conference (EDOC 2006), Hong Kong, October 16-20, 2006.

[23] Chen Lin, Li Minglu and Cao Jian "ECA Rule-Based Workflow Modeling and Implementation for Service Composition". IEICE TRANS. INF. & SYST., Vol. E89-D, No. 2, 2006, pp. 624–630.

[24] Bae Joonsoo, Bae Hyerim, Kang Suk-Ho and Kim Yeongho. "Automatic Control of Workflow Processes Using ECA Rules". IEEE Transactions On Knowledge And Data Engineering, Vol. 16, No. 8, 2004, pp. 1010–1023

[25] Booch, Grady. "Object-Oriented Analysis and Design with Applications". Addison-Wesley, 1993.

[26] Mano M. Morris and Mano M. Morris. "Digital Design" Prentice Hall, 3 edition, 2001.

[27] IBM. "Patterns: SOA Foundation Service Creation Scenario". IBM. 2006, ibm.com/redbooks.

[28] Sriraman Badri and Radhakrishnan Rakesh. "Event-driven Architecture Augmenting Service Oriented Architectures". Sun Microsystems 2005.

[29] Gartner. Service-oriented Architecture Scenario.2003.