TOWARDS A STRUCTURED METHODOLOGY FOR EVENT-BASED ENTERPRISE FUNCTIONALITY MODELLING

Nancy Alexopoulou, Department of Informatics and Telecommunications, University of Athens, Greece nanci@di.uoa.gr

Mara Nikolaidou, Department of Informatics and Telematics, Harokopio University of Athens, Greece mara@hua.gr

Vasiliki Mantzana, Department of Informatics and Telematics, Harokopio University of Athens, Greece vasiliki.mantzana@hua.gr

Panagiotis Kanellis, Department of Informatics and Telecommunications, University of Athens, Greece kanellis@di.uoa.gr

Drakoulis Martakos, Department of Informatics and Telecommunications, University of Athens, Greece martakos@di.uoa.gr

Abstract

Business process agility has been recognized as a critical characteristic of modern enterprises that should exhibit flexibility to change. Traditional business process modelling approaches, however, fall short to provide the desired agility, especially regarding modifications during execution time or when unpredicted events occur. To this end, we have proposed an event-based enterprise modeling that enables on-the-fly business process execution; hence promotes business process agility. Event-based modeling requires the identification of meaningful events and actions. As this is not a trivial task, the objective of this paper is to present a structured methodology, named Actors-Actions-Events (AAE), that can guide the designer towards identifying the events and actions depicting enterprise functionality. The applicability of the AAE methodology is demonstrated through a simplified medical example.

Keywords: event-based modelling, on-the-fly business process composition, event-driven business processes.

1 INTRODUCTION

Nowadays, an increasing number of business processes are now conducted under the supervision of information systems driven by explicit process models. Automatic coordination of business processes has increased accuracy and efficiency in the execution of business processes (Marlon et al. 2005). However, as organizational environments grew more and more dynamic, concrete business process models executed by workflow engines proved to be inflexible to change, meaning that their adjustment to new requirements was an arduous procedure requiring much time and money. As such, business process agility was soon recognized as a critical feature for enterprises in the today volatile era. To this end, numerous research endeavours were conducted, proposing methods or techniques that could increase business process agility (ShuiGuang et al. 2004, Reichert & Dadam, 1998, Rinderle et al., 2004, Mangan & Sadiq, 2002).

Most endeavors ensure agility to some extent but they cannot efficiently support agility at run time or when unpredicted events occur that impose business process modifications. According to our point view, this shortcoming stems from the fact that in most methodologies, business process logic is

organized in integral action sequences that are predetermined at design time. Defining actions strictly within the context of a specific business process is an important reason for the restriction of the ability to respond efficiently change. In contrast, we have proposed an event-based approach (Alexopoulou et al. 2008) that it may prove more promising for the attainment of agility, as it enables the definition of autonomous actions at design time that do not pertain to a specific business process. However, in order to establish the applicability of such an approach, the identification of meaningful events and actions is required. To this end, this paper presents our effort to develop a structured methodology that may guide the designer in the application of the proposed event-based modeling approach.

The paper is organized as follows: Section 2 presents the basic concepts of the event-based description of the enterprise functionality. The methodology is described in section 3, while in section 4 the application of methodology is demonstrated through a simplified example taken from the medical world. Finally, section 5 wraps up the paper with some concluding remarks and discussion regarding future work.

2 EVENT-BASED DESCRIPTION OF ENTERPRISE FUNCTIONALITY

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 (Michelson, 2006). 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 started". 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* (Luckham, 2002). The notion of specifying and utilizing relationships (such as timing, causality, etc.) between events is included in *Complex Event Processing* defined by Luckham (Luckham, 2002).

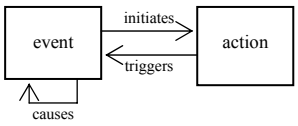


Figure 1. The basic concepts of event-based business process modeling

According to our perspective, the basic concepts of modeling organizational functionality are depicted in Figure 1. As indicated in Figure 1, 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 the action-centric approach.

In action-centric approaches, functionality is organized in specific units, i.e. actions, which are tightly coupled in predefined sequences, i.e. business processes, depicting the flow of action execution. In contrast, in event-based logic, emphasis is laid not only on actions but also on events initiating them as well as on events triggered by them. Essentially, in action-centric approaches events are implied. The sequential transition, for example, from an action to the next occurs after the first action has

completed, which implies a completion event signifying that the execution of the next action may begin. Likewise, conditions imply the occurrence of specific events determining the action paths that will be followed.

However, by explicitly modeling events and regarding them as separate entities, equally important to actions, it is possible to control action execution, ensuring at the same time loose coupling between actions. In fact, we consider each action as an autonomous unit being aware of only the events initiating it as well as the events it triggers. In that sense, the notion of business process is eliminated; the enterprise functionality is described in terms of autonomous actions, events and appropriate event combinations. The flow of actions is determined at run time by the triggered events. As such, specific event-action chains evolve during run time.

We believe that action independence promotes agility as it increases modularity in the way enterprise functionality is modeled. Thus, changes concerning the rearrangement of action sequence or the insertion/deletion of actions can be more efficiently accommodated. Furthermore, taking into consideration that in practice people usually perform an action in response to an event occurrence, an event-based approach seems more appropriate for modeling such human behavior, especially in case of dynamic environments where contingencies often generate the need for human reaction through ad hoc actions.

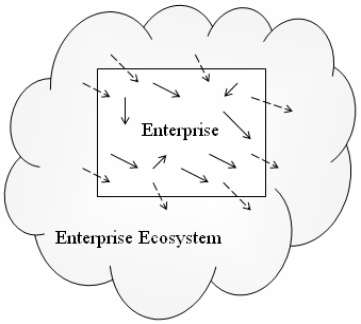


Figure 2. An enterprise interacting with its ecosystem through events

Actions are regarded as a primitive unit of functionality. As such, the direct identification of all actions depicting the enterprise functionality would be an arduous and ineffective procedure. In action-centric approaches, modeling usually starts from the identification of a business process as a black box which is then hierarchically decomposed into subprocesses until primitive actions are identified. In an analogous manner, to enhance the event-based modeling procedure, we have adopted a systemic perspective of an enterprise. In specific, as other researchers that have adopted a systemic approach for the description of an enterprise (Haeckel, 1999, Desai, 2005), we also consider an enterprise as a self-sustained system with well-defined boundaries. These boundaries demarcate the enterprise within its ecosystem, which comprises entities that interact with the enterprise such as, customers, suppliers, partners, etc. This interaction is depicted using events. As such, the enterprise may both sense external events from its ecosystem as well as diffuse to it internally generated events. These events are depicted in figure 2 through dashed arrows crossing the enterprise boundaries. Due to this fact, we call them *boundary events*. Boundary events may be either incoming or outgoing relative to the entrepreneurial environment. The solid arrows represent events that are generated internally and observed only within the enterprise.

An enterprise that is considered as a system may be analysed into subsystems characterized in an abstract manner as *units*. Units may be logical or correspond to physical departments of the enterprise. Each unit in turn may be further analysed into other units and so on. As such, an enterprise may be

ultimately decomposed to a unit hierarchy which in essence constitutes an allocation of the organizational functionality. The functionality encompassed by atomic units i.e. units of the lowest level that are not further decomposed is described through the definition of events, actions and their interrelations.

Events coming from the ecosystem are propagated down to the unit hierarchy and ultimately initiate actions. An external event may be propagated to more than one units of each level. Conversely the events that are intended to be diffused in the ecosystem are propagated the other way round. It should be noted that, equivalently, each unit identifies as its ecosystem the other units of the same level. As such, events received/sent by a unit from/to units of the same level are regarded as boundary events to the specific unit. While the hierarchical unit decomposition is different from the current trend of enterprise modelling that dictates a process-oriented approach (Joao & Francisco, 2005), is not however same as the older organizational hierarchies where the functions were distributed among several departments in an isolated manner, as in our approach, units may be both physical and logical and also are not modelled in isolation. On the contrary, interactions between units are explicitly modelled through boundary events. Furthermore, the unit hierarchy is essentially conceptual. Its role is to help the designer identify the events and actions that need to be modeled as it would not be possible for him/her to effectively conceptualize the whole entrepreneurial functionality directly through actions and events which constitute a low-level modeling. The definition of unit hierarchy breaks down the functionality range into smaller units that can be more easily manipulated and thus conceptualized through events and actions. The event-action chains cross unit limits and have an enterprise-wide spectrum, just as the business processes. The difference however is that business processes are predefined at build time, while event-action chains are formed during execution time and therefore better support agility.

3 A STRUCTURED METHODOLOGY FOR EVENT-BASED ENTERPRISE FUNCTIONALITY MODELLING

In an event-based approach, the identification of the meaningful events and invoked actions is a critical issue. However distinguishing the events affecting the function of the enterprise from a chaotic event cloud is a tough and possibly ineffective process. An indirect way therefore to identify events more effectively is through action modeling, which includes the definition of the events initiating the actions as well as the events triggered by the actions. In that sense, action modeling can guide the designer in the event identification. The problem then is transferred to action identification. For the detection of actions, an actor-oriented approach is used. We suggest that actors should be identified initially. As enterprise is viewed as a system, actors constitute an intrinsic part of the latter and are actually those that provide its functionality. Therefore actors constitute a very good source for information gathering regarding the tasks performed in the enterprise. Besides, actors are a more concrete concept than actions let alone events and thus are more easily identifiable. In doing so, the modeler will better understand actors' actions and will therefore support action identification and thus enterprise functionality modeling.

Consequently, we propose that actors should be initially identified which can lead to action and then event identifications; hence the proposed methodology has been named *Actors-Actions-Events* (*AAE)*. AAE methodology for event-based enterprise functionality modeling is grounded on the literature and consists of four phases which are presented in the following paragraphs.

Phase 1: Actor Identification

Actors are the user categories responsible for accomplishing the required tasks in a company. A plethora of researchers have focused on the issue of actor identification. As such, there are various methodologies proposed in the literature appropriate for actor identification (Mantzana et al., 2007, Pouloudi & Whitley, 1997). AAE does not specify the employment of a specific methodology or a specific list of actors, as both depend on the specific context and timeframe (Mantzana et al., 2007). Therefore, the decision is left to the modeler.

Phase 2: Unit Hierarchy Construction and Action Identification

Each actor may describe his/her everyday responsibilities. Based on these descriptions, both, units of functionality as well as primitive actions of each atomic unit may be deduced. Boundary events for the whole enterprise system as well as for each unit are also identified. Boundary event identification may be further refined during the next two phases.

Phase 3: Action Modeling

Each action identified in the previous phase is modeled separately as an autonomous unit and not as part of a business process. The modeling constituents of an action, depicted in figure 3, are described in the following:

1) Action is initiated when an event occurs. We define an event type called "Ready for Action X" that is used to initiate action X, where X is the id of the action being modelled (A_1 in figure 3). In fact, this is a conceptual event introduced in order to aggregate a number of real events which are those that actually initiate the action. Defining conceptual events to aggregate actual ones is a basic concept in complex event processing (Luckham, 2002). Through the conceptual events, complexity is hidden within events. Action does not need to be aware of complicated event combinations that cause its execution. Even if it is only one event, e.g. E_{10}, invoking an action, e.g. A_1, we still employ the event "Ready for Action A_1". Thus we define the relations E_{10} →"Ready for Action A_1" and "Ready for Action A_1"→A_1 instead of the relation E_{10} → A_1. Through this approach, any modification involving the actual events initiating an action is again hidden among events, as action A_1 for example will be constantly initiated by the event "Ready for Action A_1", even if modifications in the enterprise functionality impose the latter to be caused by a new event and no longer by E_{10}.
2) An action triggers one or more events upon its completion. For these events appropriate constraints have to be defined regarding the way they can be triggered.
3) An action requires specific data in order to be executed, i.e. input data, while during its execution, it produces output data.
4) An action is performed by a specific category of users represented by an actor, e.g. a doctor, a secretary etc.

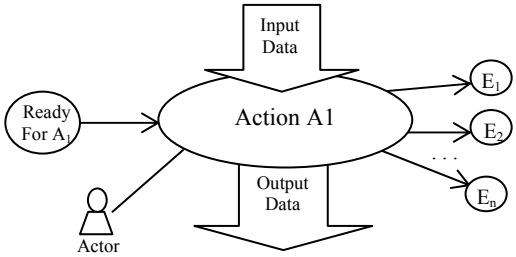


Figure 3. Constituents of action definition

Phase 4: Definition of Event Interrelations

Action modelling produces a number of events that are defined in respect to the action they initiate or the action that are triggered by. The next step involves specifying the interrelations among the defined

events. These interrelations determine the flow of actions at run time. The simplest way to relate events is through the causality relation, e.g. event E_1 causes event E_2. More complex causality relations may be constructed between events using logical operators such as, AND, OR, NOT, as shown in Table 1.

The simple causality relation may be used to express event chains. An event chain would be for example $E_1 \rightarrow E_2 \rightarrow E_3$, which means that event E_1 causes event E_2 and event E_2 in turn causes event E_3. Obviously, in this case, event sequence matters. If this is not the case, events can be related using an AND relationship. 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. If action A can be invoked either by event E_1 or event E_2, this would signify an OR relation. Practically, in an OR relation, the action caused will be initiated by the event that will happen first. However, 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. 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 can occur.

Event Relations	Description
$E_1 \rightarrow E_2$ (timing relation)	E_1 happens before E_2
$E_1 \rightarrow E_2$ (causality relation)	E_1 causes E_2
E_1 LO $E_2 \rightarrow E_3$	E_3 is caused by the combination of E_1 and E_2 through Logical Operators (LO), i.e. AND, OR, NOT, XOR.

Table 1. Combining events to produce complex events

Causality implies a timing relation, i.e. the caused event happens after the event causing it. However, a timing relation may not necessarily include the notion of causality. The fact that E_1 for example happens before E_2 does not necessarily mean that E_1 causes E_2. Timing relations may impose constraints between events such as that event E_1 for example must occur exactly after a four minutes lapse from the occurrence of event E_2 or that E_2 will occur in less than two hours after the occurrence of event E_1.

It should be noted that the above four stages are performed sequentially but not in isolation. This means that the designer may for example discover at the end of phase 4 that some actions need to be reconsidered and thus return to phase 3.

The definition of actions and event interrelations indirectly produces a number of possible event-action chains. With the support of an appropriate modelling tool, all possible event-action chains can be generated automatically. Such a capability is necessary as, according to the proposed approach, the modeller does not directly build process chains. However, by being offered the capability to view the generated process chains, the modeller can get a whole picture of the way actions are orchestrated to cooperatively offer the required functionality. Moreover, the modeller may view information flow and perhaps identify process discontinuities or other errors that will make him/her reconsider the currently modelled actions and/or event interrelations.

4 DEMONSTRATING THE PROPOSED MODELLING METHODOLOGY THROUGH A SIMPLIFIED MEDICAL EXAMPLE

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. Following, we provide a simplified system view of a hospital.

Phase 1: Actor Identification

As this demonstration is based on a healthcare organization, the authors decided to use IGOHcaps method proposed by Mantzana et al. (Mantzana et al., 2007). Using this method, a list of actors could be identified, such as doctors, managers, medical secretaries, patients, nurses, suppliers etc. However, in this paper we focus only on doctors and secretaries as (a) this is not a real case but an example and (b) its not our intention to perform an exhaustive identification of all actions and events of a hospital but merely to demonstrate our structured method.

Phase 2: Unit Hierarchy Construction and Action Identification

Based on the actors' responsibilities, the overall hospital functionality may be grouped as indicated by the unit decomposition presented in figures 4 to 7. Figure 4 illustrates the hospital as a system comprising four units, namely the *Inpatient Clinic*, the *Emergency Department*, the *Laboratories* and the *Imaging Department*. The arrival of a patient constitutes an incoming boundary event for the hospital, while when a hospitalized patient is discharged, an outgoing boundary event is generated.

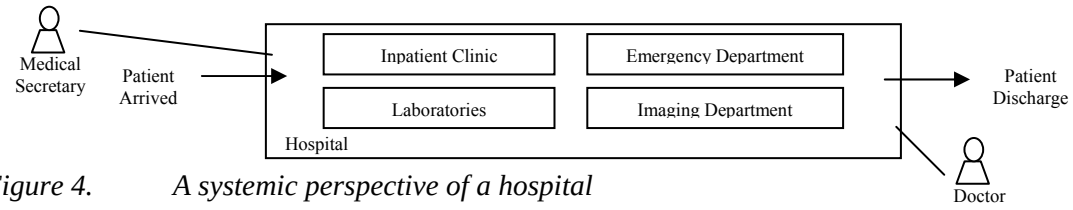


Figure 4. A systemic perspective of a hospital

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 the Inpatient Clinic. During the patient's hospitalization, a number of laboratory and imaging examinations are performed. Units and boundary events are presented in figure 5. The obvious coupling between events in figure 5 denotes the way the identified units intercommunicate. The actors presented in figure 4 also pertain to each of the four units but for simplification reasons they have been eliminated from each unit in figure 5.

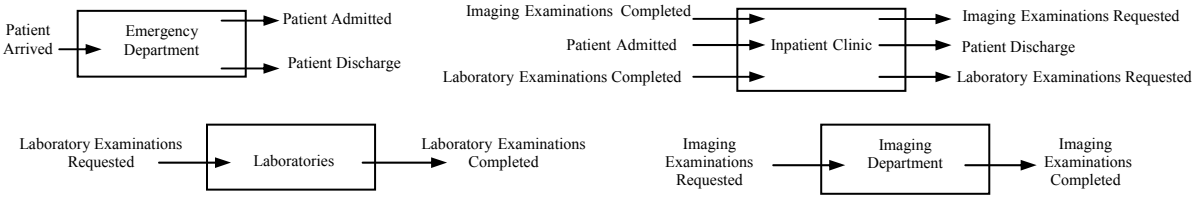


Figure 5. Boundary events for hospital's units

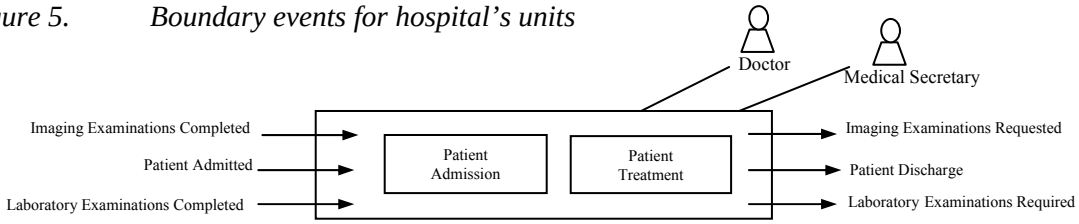


Figure 6. Decomposition of Inpatient Clinic unit

To demonstrate our modeling approach, we choose to further analyze the Inpatient Clinic unit. As shown in figure 6, this unit includes two other subunits that are regarded atomic, namely Patient

Admission and Patient Treatment, whose boundary events are presented in figure 7. Note that as opposed to the units of the first level (figure 4), which may exist in reality, the subunits encompassed in Inpatient Clinic unit constitute rather a logical functionality grouping. From figures 4, 5, 6 and 7, it can be deduced how events are propagated from the first level to the lowest and the other way round.

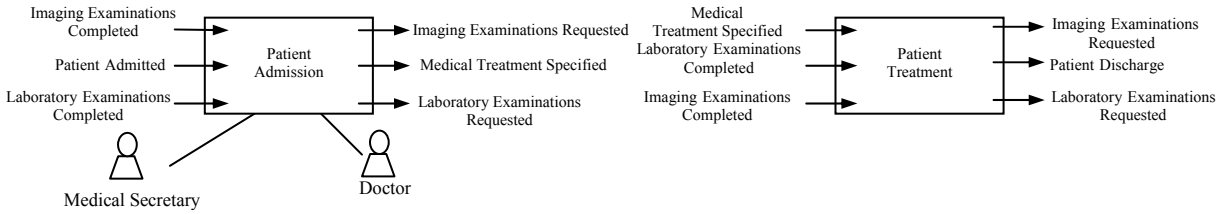


Figure 7. Boundary events of Patient Admission and Patient Treatment units

Based on actors' everyday tasks a number of primitive actions may be identified. Regarding patient admission, for example, the tasks performed by doctors and medical secretaries include *Ward and Bed Allocation, Patient File Creation, Performance of Clinical Examination, Findings Assessment, Medical Treatment Specification,* and *Specialist Consultation*. These six actions are appropriately modeled in the subsequent phase.

Phase 3: Action Modeling

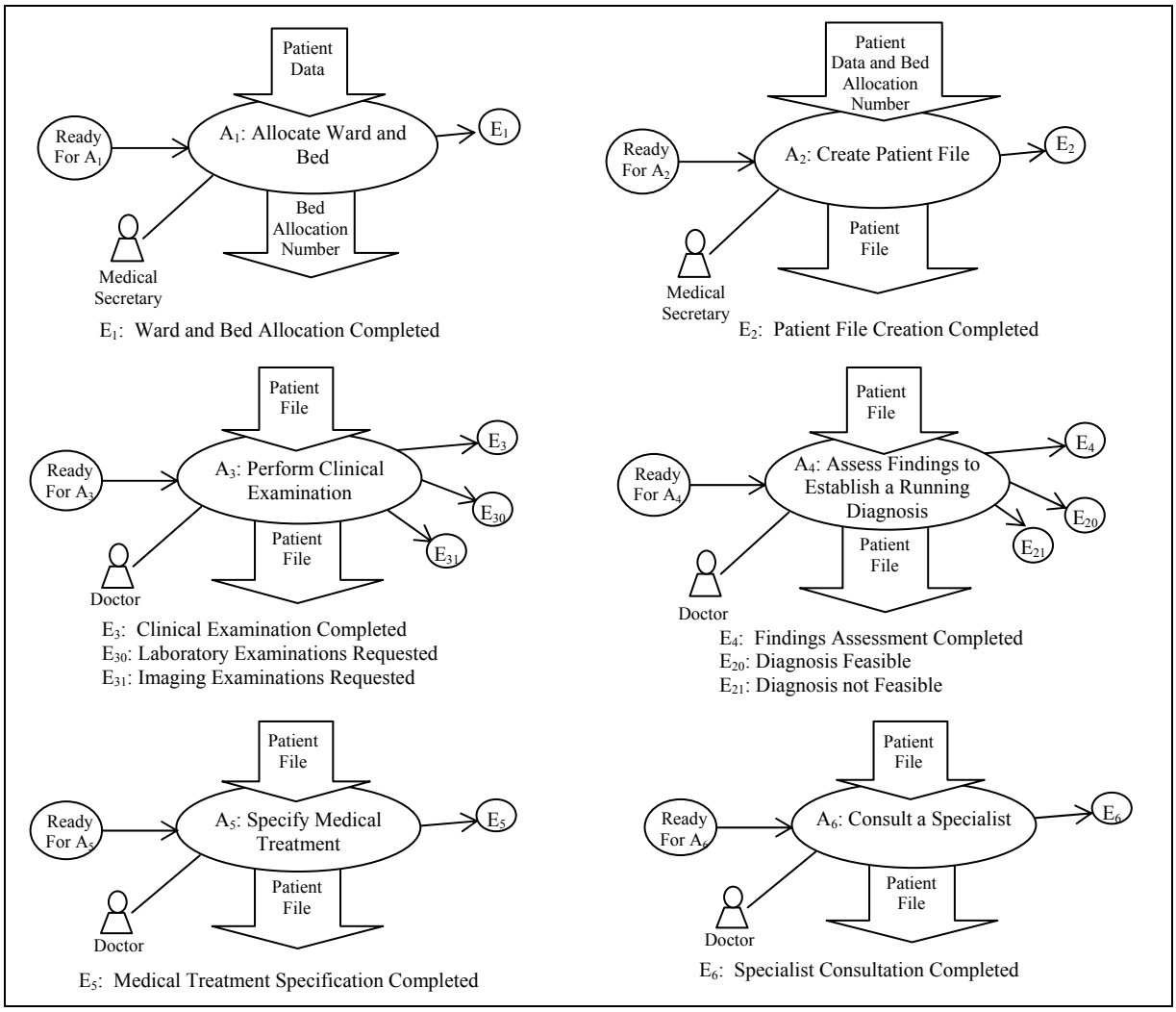
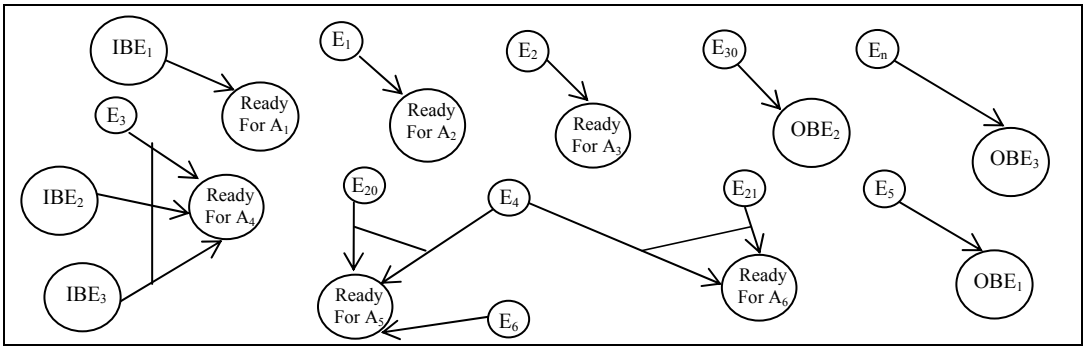


Figure 8. Modeling the identified actions of Patient Admission

Based on the information gathered from the previous phase, the identified actions can be modeled as illustrated in figure 8. Figure 8 does not include the constraints that should be defined between the triggered events. Regarding action "A_3: Perform Clinical Examination", for example, an AND constraint should be defined among events "E_3: Clinical Examinations Completed", "E_{30}: Laboratory Examinations Requested" and "E_{31}: Imaging Examinations Requested" as when a patient is admitted, he/she always submitted to a number of basic laboratory and imaging examinations. Concerning action "A_4: Assess Findings to Establish" a Running Diagnosis it triggers the events "E_4: Findings Assessment Completed", "E_{20}: Diagnosis Feasible" and "E_{20}: Diagnosis not Feasible". The constraint defined for these events may be expressed by the logical expression (E_4 AND E_{20}) XOR (E_4 AND E_{21}).

Phase 4: Defining Event Interrelations

Event interrelations are depicted in figure 9. The solid line is used to denote an AND relation. It is clear that no arrow ends in an incoming boundary event and likewise no arrow starts from an outgoing boundary event.



IBE_1: Patient Admitted
OBE_1: Medical Treatment Specified
IBE_2: Laboratory Examinations Completed
OBE_2: Laboratory Examinations Requested
IBE_3: Imaging Examinations Completed
OBE_3: Imaging Examinations Requested

Figure 9. Defining event interrelations

It should be noted that event "E_{31}: Imaging Examinations Requested" for example, semantically could be a boundary event itself. However, to ensure independence of boundary-specific information in action modeling, E_{31} is not defined as an outgoing boundary event. Rather, it invokes "OBE_3: Imaging Examinations Requested", which is the one considered an outgoing boundary event. In this way, changes in subsystem boundaries will not necessarily affect action definition.

Event interrelations presented in figure 9 may produce two possible event-action chains during run time (figure 10). For simplicity reasons "Ready for A" events can be eliminated from the representation of event-action chains as they do not offer any essential information regarding the action flow. The event-action chain depicted in part (a) of figure 10 is described as follows. When a patient is admitted to a hospital 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, a clinical examination is performed and then the findings are assessed, which lead to the establishment of a running diagnosis. As a result, a medical treatment is subsequently specified. In event-action chain depicted in part (b) the differentiation is that the running diagnosis is not feasible. Therefore, a specialist's consultation is acquired and then, according to his/her diagnosis, the medical treatment is specified.

As these chains are constructed on-the-fly during execution time and each action is an autonomous unit, how can the action instances referring to the same patient be correlated? To find a way to trace actions of the same process instance, yet without obliterating their independence, we considered the

analogy from the real world. In reality, two actions are associated through the common data they refer to. Instances of actions "A_3: Perform Clinical Examinations" and "A_4: Assess Findings to Establish a Running Diagnosis", for example, are associated if they refer to the same patient. To this end, actions are related to data structures containing specific patient information. As the reference to this data structure should be propagated to the related actions, we assume in our approach that this is accomplished by the events. As such, events encompass also information regarding data location which they pass to the actions they initiate.

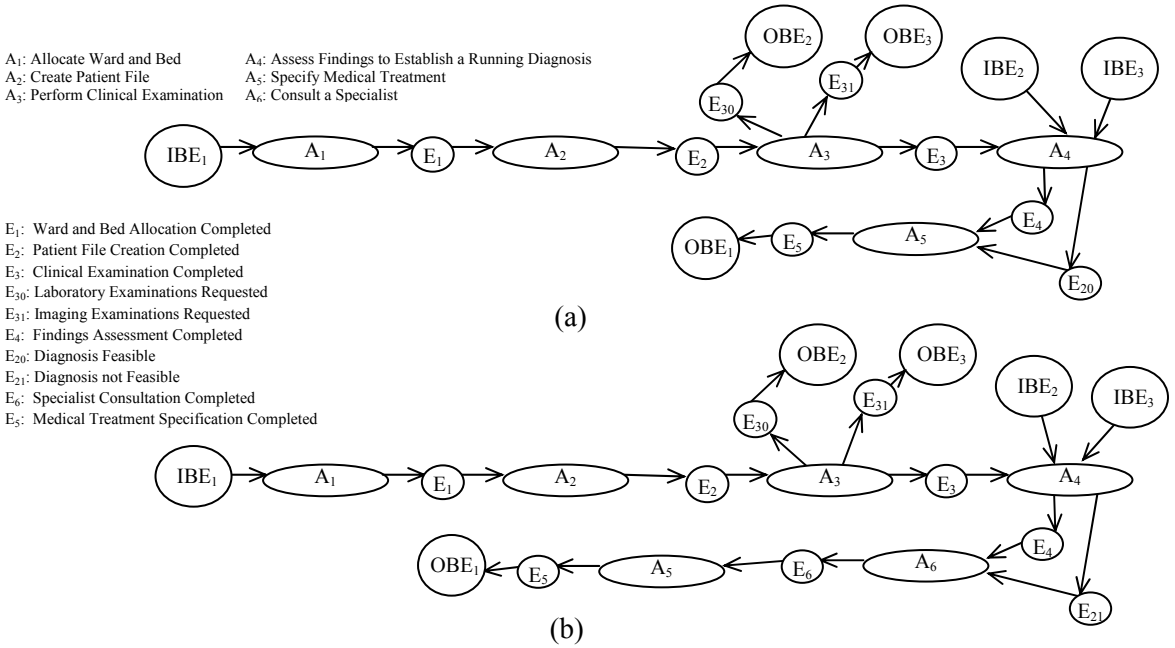


Figure 10. The possible event-action chains regarding patient admission

Lastly, it should be noted, that the differentiation between the event-chains depicted in figure10 lies in an implied condition, which checks whether a running diagnosis is feasible after the findings assessment. In our approach, conditions are implicitly modeled through the definition of two opposite events, namely, "E_{20}: Diagnosis Feasible" and "E_{21}: Diagnosis not Feasible". Each is combined with event "E_4: Findings Assessment Completed" (see figure 9) to initiate a different action. This indirect expression of conditions through opposite events that is introduced in this paper constitutes an Event-Action (EA) approach which is in contrast to Event-Condition-Action (ECA) approach (Dayal et al., 1990) originally applied within the active database community (Widom & Ceri, 1996, Paton & Diaz, 1999). In ECA model, when an event occurs, the condition is evaluated. If the condition is satisfied, the action is executed. ECA is a well-established and extensively applied concept in business process modeling and workflow approaches (Jan et al., 2006, Chen et al., 2006 and Bae et al., 2004). While ECA model is more expressive, we propose however EA model, since it may prove more efficient regarding business process agility. As EA model is based on only two entity types (event-actions) combined in autonomous pairs, it is characterized by higher homogeneity and less complicated structures than those combining event and actions through conditions. Apparently, changes are more easily performed in simpler structures. Moreover, the incorporation of business rules - expressed in ECA by conditions - within event processing, simplifies the way events and actions are interrelated. This leads to a more straightforward modeling procedure.

5 CONCLUSIONS

Efficient response to change may be attained through the adoption of an event-based modelling approach for enterprise functionality. To effectively applying such an approach, designers should be guided by an appropriate methodology. To this end, a structured methodology, called AAE, has been proposed in this paper. AAE facilitates the detection of events and actions starting from the identification of the actors operating within the enterprise.

AAE methodology makes an important contribution at both practical and conceptual level. At a practical level, it contributes towards a deeper understanding and simplification of the modelling process. At the conceptual level, it constitutes an efficient way for action and event identification that is required in an event-based approach. AAE increases the level of the analysis and provides a more detailed and systematic study of the event-based modelling. Moreover, it supports managers, designers and researchers in (a) understanding the importance and effect of actors, actions and events and (b) modelling actors, events and actions in a consistent manner. Consequently, it is suggested that this methodology might speed up and simplify the modelling process.

Future work includes testing the presented methodology by applying it in the practical arena. Subsequently, after further exploring implementation issues, we plan to develop an infrastructure based on the conceptual architecture proposed in (Alexopoulou et al., 2008) that will support the execution of event-driven processes.

References

Alexopoulou Nancy, Nikolaidou Mara, Chamodrakas Yannis, Martakos Drakoulis, 2008. 'Enabling On-the-fly Business Process Composition through an Event-based Approach', *Proceedings of the 41th Hawaii International Conference On System Sciences (HICSS 2008), Big Island – Hawaii, Jan 7-10, 2008*.

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

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

Dayal U., Hsu M., and Ladin R., 1990. 'Organizing Long-Running Activities with Triggers and Transactions'. *Proceedings of ACM International Conference on Management of Data (SIGMOD), pp. 204-214.*

Desai Anand, 2005. 'Adaptive Complex Enterprises'. *Communications of the ACM, Vol. 48, No. 5, pp.32-35.*

Haeckel, S. H., 1999. 'Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations'. *Harvard Business School Press, Boston*.

Joao M. Fernandes and Francisco J. Duarte, 2005. 'A reference framework for process-oriented software development organizations'. *Software Systems Model, Vol. 4, pp. 94-105*.

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

Luckham David, 2002. 'The Power of Events'. *Addison-Wesley*.

Mangan Peter and Sadiq Shazia, 2002. 'On Building Workflow Models for Flexible Processes'. *13th Australasian Database Conference (ADC2002), Melbourne, Australia. Vol. 5.*

Mangana V., Themistocleous M., Irani Z and Morabito V., 2007. 'Identifying HealthCare Actors Involved in the Adoption of Information Systems'. *European Journal of Information Systems, 16(01), pp.91-102*.

Marlon Dumas, Wil Van Der Aalst, Arthur H. M. Ter Hofstede, 2005. 'Process-Aware Information Systems'. *John Wiley & Sons INC.*

Michelson M. Brenda, 2006. 'Event-driven Architecture Overview'. *Patricia Seybold Group. http://dx.doi.org/10.1571/bda2-2-06cc.*

Paton N. W. and Diaz O., 1999. 'Active Database Systems'. *ACM Computing Surveys, Vol. 31, No. 1, pp. 63–103.*

Pouloudi A. and Whitley E. A., 1997. 'Stakeholder Identification in Interorganizational Systems: Gaining Insights for Drug Use Management Systems', *European Journal of Information systems 6(1), pp.1-14.*

Reichert, M.; Dadam, P., 1998. 'ADEPTflex – supporting dynamic changes of workflows without losing control'. *Journal of Intelligent Information Systems, 10, pp. 93-129*

ShuiGuang, D., Zhen, Y., ZhaoHui, W., LiCan, H. 2004. 'Enhancement of Workflow Flexibility by Composing Activities at Run-time'. *Proceedings of the ACM Symposium on Applied Computing, pp. 667-673.*

Rinderle Stefanie, Reichert Manfred, and Dadam Peter, 2004. 'On Dealing with Structural Conflicts between Process Type and Instance Changes'. *BPM 2004, pp. 274–289*.

Widom J. and Ceri S., 1996. 'Active Database Systems: Triggers and Rules For Advanced Database Processing'. *Morgan Kaufmann*.