Linking event-driven and communication-oriented business modeling

Hans Weigand[1] and Aldo de Moor[2]

[1]Infolab, Tilburg University, The Netherlands
weigand@uvt.nl,
[2]STARLab, Vrije Universiteit Brussel, Belgium
ademoor@vub.ac.be

Abstract

Event-Driven Architectures are a critical instrument for tomorrow's fast-acting and agile enterprises. However, to benefit maximally from the technological innovations, businesses need to link an event orientation to an organizational communication perspective. The objective of this paper is to arrive at a blended business modelling analysis method that combines event chain modelling with a communication-oriented approach.

Keywords: Event-driven architectures, communication modeling, event process chains

1 Introduction

As enterprises strive to cut costs and improve their responsiveness to customers, suppliers and the world at large, the concept of event-driven design is becoming more widely used. According to (Schulte, 2004), the new generation of application systems that will underlie tomorrow's fast-acting and agile enterprises will not be the same as traditional applications. The most important change will be Event-Driven Architectures (EDA). Event-driven business processes are primarily a business issue, but it has also implications for the IT architecture. To be event-driven, this architecture must have at least an appropriate message-oriented middleware, such as the message-driven beans supported by J2EE.

Another recent development is the realization of pervasive and ubiquitous computing, predicted by the visionary paper of Weiser (Weiser, 1991) which means that IT is getting smaller and smaller and increasingly becomes „invisible", attached to objects and humans, for example, in the form of an RFID chip. When such objects or humans meet, temporary connections are set up and executed. For example, containers get a unique RFID and when they pass the gate, this automatically triggers the generation of an „enter" event in the Information System. This event may trigger a message event to be passed to the driver in which he is informed about the exact location where the container is to be put, and may trigger internal tracking & tracing messages to update the information about the container as it is available to all parties involved.

In both developments, *events* play a key role. Events initiate procesess. Processes consisting of events and couplings between events. However, what does it mean in a business environment that an event *triggers* another event? How do we model event chains and how do we couple them to an organizational communication point of view? What are the links between (automated) control flows and (human) communication processes in an organization? To answer these questions, this paper starts from the language/action perspective (LAP) in which „action" has always been seen as more prominent than „representation" or „information". The main coordination mechanism in current LAP approaches is the

communicative transaction based on speech acts between human subjects; one question is how this perspective can be reconciled with a notion of anonymous event chains.

The structure of this paper is as follows. In section 2, we will give some background to event-based processing. Section 3 is a discussion of the role of events in the Language/Action Perspective. In section 4, we will present various ways to link event process models to communicative action models, and section 5 a communicative analysis of event models.

2 Event-based processing

2.1 The notion of Event-Driven Architecture

The goals of EDA are to increase the speed and agility of business processes. Real-Time-Enterprise (RTE) takes the role of timeliness to its logical extreme: zero latency, that is, all parts of the enterprise can respond to events as soon as they become known to any one part of the enterprise (or extended enterprise). The other goal besides speed is agility: to respond to exceptions and unanticipated events at any time, even when business processes are already under way.

What is an event? Usually, a distinction is made between business events and software events. A business event is a meaningful change in the state of the enterprise or of something relevant to the enterprise, such as customer order, an employee address change, the arrival of a shipment at a loading dock, or a truck breakdown. Software events are messages in the Information System that describe a business event. Software events are generated by an application program or some other software when they become aware of business events.

Direct triggering is a key characteristic of event-driven processes. A traditional, build-to-stock manufacturing approach also depends on events, but only indirectly. The order fulfilment process is typically event-driven, but not the manufacturing. In a build-to-order enterprise (such as Dell computers), the whole enterprise is event-driven, which implies that the business processes must run very fast: it is not possible to pick the order just from the warehouse, as there is no inventory. Of course, this build-to-order strategy is not practical for all products, but with more and more advanced IT (faster communication links, faster data processing, data-driven manufacturing devices), it becomes feasible in more and more situations.

In short, the main characteristics of EDA are (Schulte, 2004):

- handle events individually rather than in batch
- use push, not pull communication
- notify people selectively (management by exception, instead of the obligatory monthly reports and forecasts)
- relocate some business rules out of application programs
- dynamic IT architectures, based on message-oriented middleware capable of publish-and-subscribe behavior
- complex event processing (CEP)

Complex Event Processing (CEP), often related to Business Activity Monitoring (BAM), includes tools that monitor the events in the enterprise and are not only able to aggregate data into higher-level complex events but also to detect unusual event patterns that may need an alert (Luckham, 2002). Luckham argues that vast amounts of valuable information are latent in today's distributed information systems, and that a new breed of software tool is needed to collect and present the data. An example of the detection of unusual event patterns is the situation in which a creditcard holder in Amsterdam makes a routine purchase, and a few minutes later, the same cardholder attempts to make another purchase from Taipei, Taiwan. In such a case, the credit approval system should combine the two events and generate a „possible fraud event" and broadcast it as a message to the authorisation system and any other system that needs to know. CEP tools are now brought to the market by vendors like IBM and Tibco.

2.2 Service-Oriented Architecture

EDA is often related to Service-Oriented Architectures (SOA). SOA supports the assembly of distributed applications from loosely coupled services (Papazoglou, Georgakopoulos, 2003; Umapathy, Purao, 2004). It typically relies on synchronous interactions with a one-to-one request/reply style using web-standards such as WSDL and SOAP. Conceptually, this contrasts with EDA that typically promotes asynchronous interactions with a message push style (without needing or waiting for a reply). Nevertheless, the two can be combined in various ways. A SOA could be built on top of an EDA by modeling the request and the reply as two different events where the reply is triggered by the request. Conversely, an EDA could be built on top of a SOA by adding middleware that picks us events and translates them into service requests. The reply can be taken up then as a new event to be distributed to other services.

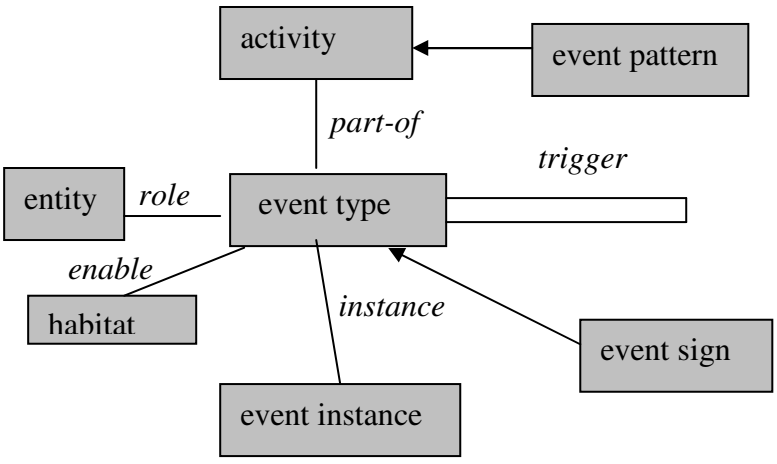


Figure 1: An initial event ontology; the arrows stand for isa-relationships

2.3 Events

A first approximation of an event ontology is presented in Fig 1. Events can be grouped together into activities. Events exist in time, they are typically enabled by a certain habitat (Andersen, 2005) and entities can play a certain role in an event (e.g. agent, object, instrument). A distinction must be made between event types and event instances, the latter

having a specific time stamp. In the following, we will usually use the word „event“ for „event type“. A key characteristic of events is the *triggering* relationship between them. Although this notion is used in many event modeling approaches (event process chains, petri nets, etc), it is a far from simple concept (cf. Dietz, 2001). In some cases, the triggering is a causal relationship based on physical laws, like one falling domino stone causing the next one to fall down. This case seems rather well-understood, although it is easily overlooked that for the triggering to be effective, often much more is needed than the triggering event. The domino effect takes not place when someone holds up the next stone with his finger, for example. In most business cases, the triggering is not causal at all. How does an order trigger an acknowledgment? How does the return of a book to the library trigger its being put on the shelf again? One possible answer is automation: in an EDI system, the order may trigger the acknowledgement by virtue of the fact that the receiving program works in that way. However, a more general answer is that events are related by social conventions or procedures that state that the event *should* be followed by the other event (Stamper, 2000). Such *norms* need not be explicit, and humans may not even be aware that they follow a convention. Sometimes the norm is laid down in an explicit procedure, and sometimes it is implemented into a program thereby avoiding the human intervention. However, it is essential to realize that there is an interaction between event models and organizational communication models. In short, we could say that event models focus on the *how*, and communication models on the *why* of organizational workflows. *Signs* play an important role as linking pins between the two types of models, and can be used in the diagnosis and resolution of breakdowns.

In general, the triggering of an event will not work without some signalling. The triggered event is only started after some actor has somehow become aware of the triggering event. We call this actor the *event recipient*. The *sign* associated with, for instance, an order (i.e. the incoming digital message) is what triggers the EDI-program to send an acknowledgement. Let us call this sign the *event sign*. When we talk about the event sign, we are talking about the sign in its function of signalling an event. In the case of the book having been returned, it is typically the visible presence of the book (on the front desk or some dedicated shelf) that causes a library assistant to pick it up and bring it into the library. We require all triggering events to have one or more event signs; in some cases, the event may be identified with the event sign (as in a prototypical speech act), so then the event sign could be said to *be* the event. The event and the event sign can be distinguished analytically, but because of their intimate connection, they are often treated as one whole practically. Also note that an event sign is only interpretable by an event recipient when occurring in the proper *sign context*. This is a set of conditions that needs to be present for a sign to be effective. For example, an e-mail may work very well as a trigger to get a researcher to submit his paper in time. However, when this actor is inundated with such requests, the message may lose its effectiveness, and the event chain may come to a premature halt. It may be clear that the semiotics of such event sign handling quickly become very complex, much more than acknowledged by regular workflow modelling methods, for example. Applying the language/action perspective may inform a systematic way to better understand what is happening in case of event signalling breakdowns.

3 Events from a language/action perspective

In the Language/Action Perspective, the „action“ has always been given priority over notions like „information“ and „representation“. Goldkuhl and Agerfalk (Goldkuhl, Agerfalk, 1998) argue that Information Systems should be viewed as instruments for organizations to perform

their actions. Information Systems present an action potential, a repertoire of predefined actions, summarized in the term „actability". In this section, we first trace back the action focus in LAP to the seminal book of Winograd and Flores; then we consider how it is used in the LAP method DEMO and the Extended Workflow Model. Finally, we propose a feasible way of reconciling an event and communicative action perspective.

3.1 Language as action

> „Popular accounts of language often portray it as a means of communication by means of which information is passed from one person (or machine) to another. An important consequence of the critique presented in the first part of this book is that language cannot be understood as the transmission of information.
>
> Language is a form of human social action, directed to what Maturana calls „mutual orientation". This orientation is not grounded in a correspondence between language and the world, but exists as a consensual domain – as interlinked patterns of activity. The shift from language as description to language as action is the basis of speech act theory, which emphasizes the *act* of language rather than its representational role" (Winograd & Flores, 1986:76).

As the LAP approach has become famous for its emphasis on commitments that are made by communicative actions (as made explicit in a system like the Coordinator (Medina-Mora et al, 1993), it is easy to forget that in fact it is rooted in a much broader action philosophy than Speech Act Theory only. A basic tenet of Winograd & Flores is that the primary function of language is in the coupling of activities rather than in the formation of representations or the exchange of such representations via messages. Two organizational communication modeling approaches that stress this action-focus are DEMO and the extended workflow loop paradigm.

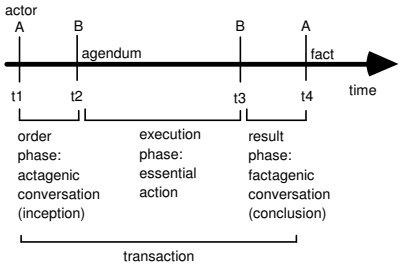


Figure 2: Transaction pattern (after Dietz, 1994)

3.2 DEMO: The OER-paradigm

DEMO (Dynamic Essential Modelling of Organizations) (Dietz, 1994) is a business process modelling method based on social theory, grounded in the philosophy of Searle and Habermas. The motivation behind DEMO is the strongly felt need to have a theory about the dynamics of activities in organizations for IS analysis.

According to Dietz communicative acts in business communication are related to each other according to a specific pattern, called the transaction pattern. The pattern consists of a communication part and an action part (see Fig. 2).

The transaction starts with a request of the initiator A (at time t1). The participants involved in the transaction, called actors, reach (at t2) a commitment for a future action, called agendum (thing-to-do), added to the agenda for the actor involved. Next, the action agreed upon is executed by the executor (t2-t3). Finally, the parties try to reach an agreement about the result of the action. When the initiator accepts the result, the transaction succeeds and a

fact is created (t4). The fact corresponds with the predication of the communication act as mentioned above. According to Dietz, the essence of the behaviour of an organization consists of the continuous accomplishments of such transactions between actors.
The DEMO methodology supports Actor Transaction Diagrams that models the interactions between the organization and its environment, as well as internal transactions, and a Process Step Diagram. The former can be seen as a representation of the „consensual domains", and the transaction pattern is indeed central there. The latter describes „the lawful sequences of events in the Coordination World and the Production World: the atomic process steps and their causal and conditional relationships" (Dietz & Habing, 2004). In our opinion, this model can be used for business process modeling, but it has some drawbacks. It uses the notion of causal relation for triggers from one transaction to another, in contrast to the trigger relationships within a transaction. We suggest to use one notion of „trigger" only. A more important concern is that the model seems to be predetermined by the transaction pattern rather than the actual workings of the system. For example, in the model of a care process (Dietz & Habing, 2004), several transactions are distinguished for the diagnosis, the establishment of policy options and the actual treatment. Each of these transactions is modeled according to the transaction pattern (so it is started by a request of the patient), although it is recognized in the text that this is not how it goes in practice: „in practice, the actual surface form will often be that the physician asks the patient for agreement [for the actual treatment]". The drawback of this approach is that the triggering that exists in practice, is not modeled, and therefore, the model can not help to determine whether at this point there is a potential deadlock (the doctor waiting for the patient's request) or not, and cannot be used as a way to optimize the actual process.

3.3 The extended workflow loop model

The extended workflow loop model (Weigand & De Moor, 2003) draws on the OER-transaction such as used in DEMO, but extends it with the notion of delegation. The agent (executor) that performs a certain service for a customer (initiator) in an organizational setting typically does this on behalf of the organization. It is the organization that offers the service, and part of the execution is delegated to the agent. For example, Fig. 3 pictures a pizza shop in which the owner has delegated the baking of pizzas to his daughter and the home delivery to a pizza boy. As the model shows, there is a chain of events starting with the phone call from the customer and ending with the delivery of the pizza. Particularly interesting is how these events are linked. Consider for example the interaction between daughter and pizza boy. When the daughter has finished the baking, the pizza is put on a shelf. When the pizza boy comes in, he takes the pizza and drives away. In the extended workflow loop model, this situation is interpreted as that the daughter initiates the execution of the delivery action (on behalf of the baker, who does evaluate the action later, but does not initiate it himself) and at the same time the pizza boy evaluates the baking action (again on behalf of the baker, who in this case did initiate the baking). In the case of breakdowns, it is the baker who solves the problem, and this follows naturally from the model as we are talking about delegated tasks, and so the baker remains responsible. In this way, the extended workflow model can account for workflow chains where one agent passes his results over to a next one, without one being the customer of the other, as is, in principle, the case in DEMO.

Let us come back to the interaction between daughter and pizza boy. From an event process perspective that remains close to the reality, it is the „pizza finished" event that triggers the „put on shelf" event, which again triggers the „take away pizza" event. The event sign in the first step may be a beep tone from the oven, and the event sign in the second case may be the visible presence of the pizza on the designated shelf. Let us assume that the pizza boy also

signs a list for each pizza that he takes away so that it is possible later to track the status of the process. This can serve as the event sign record of the „take away“ event that otherwise would leave no traces if it is performed unobserved by anyone apart from the pizza boy. Note that this event sign could trigger other events, for example, if the list is automated, it could trigger a „delivery notice“ event in the form of an email or sms message to the customer telling him that the pizza is on its way.

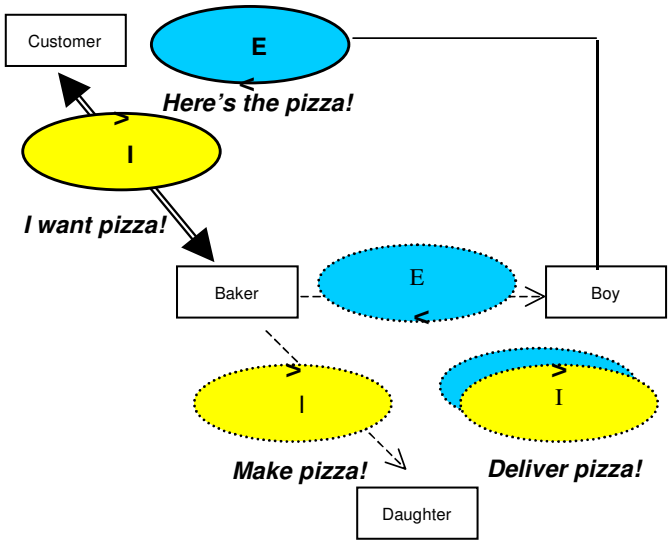


Figure 3: Pizza example of extended workflow loop

4 Linking event-driven and communication-oriented business modeling

The theoretical question is how the event process perspective is to be reconciled with the communicative action perspective, such as DEMO and the extended workflow loop model. We propose not to choose one perspective at the expense of the other, nor to mix the two perspectives, because that would mean a blurring of the differences between different abstractions. Rather, the two perspectives should be linked by mappings between events and communicative actions. The event process model can be seen as a surface structure that implements the „deep structure“ of the extended workflow model. Just as in language, the surface syntactic structure and the deep semantic structures have their own laws and neither one can be reduced to the other. We should therefore keep both layers, and identify the norms that are important in both, in order to assess the quality of the process. Within an overall methodology, meta-norms can be established that validate one model against the other, using the mappings between the event model and the communicative model. For example, for each communicative action identified in the extended workflow loop model there should be some event sign in the event process model that can be interpreted as the initiating action in the workflow model. So, in the pizzeria example, the „putting on the shelf“ event could be interpreted as initiating the (communicative) action for the pizza delivery. If this event sign would not be there, or if its sign context would be unclear (e.g., when the daughter puts the finished pizza somewhere in the kitchen, but it is not the designated place where the pizza boy has to look), then this is an indication that the event process model is lacking because it misses an initiating action.

By blending a communicative action model with an event-chain model, we also meet the critique of Goldkuhl and Agerfalk on event-based modeling approaches (Goldkuhl, Agerfalk, 1998). As they rightly say, in a social context, things most often don't `just happen'. An event is almost always the result of some communicative action performed by some actor, although the event chain triggered by that action may become very long. Although in a ubiquitous computing scenario, the role of anonymous events may grow, we do agree that for an IS design methodology, it helps to identify the communicative actions first and that any event chain must always have at least one mapping to a communicative action. However, we claim that an accurate modeling of events is also needed and that one model cannot be reduced to the other.

In the next section, we will provide some guidelines for analyzing event processes based on the theory developed sofar. The methodology, mappings, and meta-norms required to do so systematically are not formalized in this paper. For simplicity, we will concentrate on the internal processes of the organization rather than on the „consensual domains", the interactions with external agents, as this topic is already dealt with quite extensively in the literature (e.g.(Weigand, vd Heuvel, 1998; Johannesson et al, 2005)). Although we don't solve the problem of linking the two modelling worlds completely, the point we hope to make is that studying the links between event and communication model is a complex problem worth of in-depth future research.

5 Communicative Event Process Analysis

Suppose that we have an event process diagram consisting of events and triggers between them, as well a.communicative model of some kind. In this section, we (informally) present some checks that could be done to link the two modelling worlds. The analysis does not depend on the particular event or communicative modelling technique chosen (Petri Net, Event Process Diagram, Activity Diagram, BPEL, ...; and DEMO, XWL, ActionWorkflow etc., respectively). Their differences are not relevant for the analysis that we describe here.

5.1 Check for communication transaction completeness

For each service that is offered, check whether there is both a communicative and an event model. The first step is to determine the set of services. This can be taken from a diagram such as the Actor Transaction Diagram in DEMO, if this is available, to identify an actagenic conversation and a factagenic conversation, that is, the basic transaction pattern. It is also possible to analyze the event process model itself and sort out all events that are not speech events. The second step is to find event signs related to that service, and see whether they can be interpreted as factagenic or actagenic. If such an event sign cannot be found, then one needs to be added. Thus, this is one way to validate that communicative (trans)actions are properly implemented through (sets of) event chains. In the next example, we will annotate elements from event models with [EM] and from communicative models with [CM], to indicate more clearly their subtle interrelationships.

Example: suppose in a library we have the following event chain:
lend request → take book from store → lend event → return ...

The lend event [EM] is a service. The lend request [EM] (that could be a form filled in by the customer) can be interpreted as the start of an actagenic conversation [CM]. However, what

is missing in this event chain is the committing. Typically, this is done verbally, and that could be a reason why it was not included in the event chain so far. What is also missing is something that can be interpreted as the factagenic conversation [CM]. What can be added (and is actually used in many libraries) is that the lending event is accompanied by a small note that states that this book has been borrowed on this date and should be returned at that date [EM/CM: event sign record]. Often, this note is put into the book before it is handed over to the customer. So this leads to the following extension of the event chain:

lend request → „ok“ → take book from shelf → lend event → (paper note)
→ return ...

The paper note (put here between brackets to indicate that it is a record of an event sign) can be interpreted now as a record of an event as well as the completion of a factagenic conversation, i.e. a communicative action. But this is not all. It also functins as a return request from the library (relating the actagenic conversation to the return event). If the new return event process chain indeed takes place, then we can say that it satisfies the communicative transaction completeness. If not, then the suggestion must be made to add these steps in the procedure.

5.2 Check for event sign records

Every event should have event signs and event sign records. The event sign is needed to trigger the subsequent event in the chain. This is not needed when the two events are executed by the same human agent, such as an actor first taking a book from the shelf and then handing it over to the customer, but this is a special case. The event sign record is needed for internal control. If the event does not leave a trace, it is not possible later to verify that the event has actually happened. Nor is it possible for the agent to support his claim that he has performed the event. Such a control mechanism is needed for events in which some value exchange is involved. Taking the book from the shelf does not involve a value exchange, but handing it over to the customer (and therefore bringing it out of the control of the library) certainly is. In general, event sign records link the EM with the CM, since their creation generally is directly mapped to a particular communicative action. Note that the event sign record is not sufficient if it is not taken up by another event, for example, its storage in an archive.

Example: hand over pizza to customer → pay to pizza boy →

Neither the handing over of the pizza nor the payment are accompanied by event sign records. This introduces a control weakness. A form that is signed by the customer for having received the pizza and signed by the boy for having received the payment, with one copy for the customer and one for the boy, is a way of solving this weakness.

5.3 Failure handling

An event that is triggered may fail to execute. The general rule should be that each event failure triggers some other event that deals with the failure, often leading to a corrective communicative action (pair, such as an request for clarification, plus reply from the target actor). Of course, this cannot be done endlessly. Reasons for not working out the failure handling, may be that its occurrence is too rare, or too complex to specify in advance.

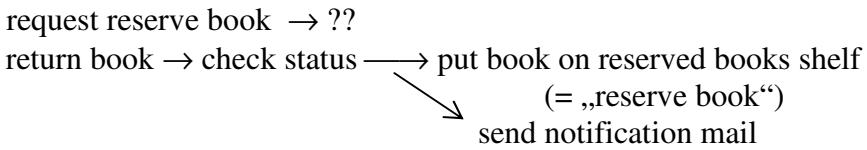
Example: lend request → „ok“ → take book from shelf [*failure*]

The „take book from shelf“ action may fail when the book is not there. This breakdown may trigger a question [EM] to the customer whether he wants to reserve the book.

In some cases, like the example shown here, the failure handling can considered to be a normal procedure. Often, it would make the model too complex if all failure handlings are included. Therefore, we propose to work them out in separate models. Although it is beyond the scope of the paper, we want to stress that in the light of the event-based approach, it is essential that these failure handling methods are linked to higher-level organizational patterns that observe the failures and try to learn from them. In short, failure handling will often contain complex interrelations between event chains and communicative action pairs or loops.

5.3 Detect potential deadlocks

First, it should be checked whether all endpoints in the event chains are indeed endpoints. For example, when the pizza boy has delivered the pizza and received the money, and the note is signed, is this the end of the chain? No, the note and the money need to be given to the baker for checking (this is part of the evaluation side of the delegation loop, i.e. [CM]).
Another situation that often occurs is when a certain event must wait another event. For example, in the library a reserve request [CM] has been made by a customer for a book that is not available. The reserve request would like to trigger the „put book on reserved books shelf“ event, but this cannot execute as long as the book has not been returned. When the book is returned, a check must be made to see if it is reserved, and this check triggers the „put book on reserved books shelf“ and the sending of a notification mail to the customer [CM]. So we have two event chains:



The first problem that can occur in such a situation is that the second chain has been forgotten, (returned books are not checked for their status and just put back in the library immediately) and so we have a reservation request that is waiting indefinitely, and a communicative action, i.e. reporting to customer, that never takes place. To prevent such a situation from happening, the designer should check that when there is a request for some action X, there is also somewhere an event chain containing the execution of that action X, linked to the appropriate communicative reporting action – as is the case in our second event chain. A second problem is that a kind of deadlock can arise when the book is never returned. For that reason, it is useful to have a time-out mechanism that after some period clears the request *and* sends the appropriate communicative action. For example

request reserve book → timer — [40 days] → check book status → inform customer that book has disappeared

Now the request triggers a timer that, after 40 days, triggers a check event that may trigger a notification (note that we have chosen an event-based approch, rather than a batch-oriented approach in which e.g. all open reservations are checked every night). For efficiency reasons, we may want to disable the timer when the book in question is returned. This would require an *interrupt* type of trigger that we don’t have in our event ontology yet as it is not commonly found in current event models.

6 Conclusion

When event-based business modeling will become more important, there will also be a need for proper modeling support. In this paper, we have shown how event chains can be put in an appropriate organizational context by taking advantage of a blended event-based/ communication-oriented approach. Our analysis of the examples was conceptual only. Our contribution is that we have argued that workflows should not be modelled either through events or through communicative modelling, but through both, in two different models. The interrelations between these two modelling realms are still very ill-understood, and will be an interesting domain of future research. One approach would be to develop sound and complete ontologies of both communication and event modelling, the possible mappings between their elements, and the meta-norms that govern the checking of mappings between these two types of models, such as prescribing that every event chain should be linked to at least one communicative action.

The method we propose differs from related approaches in the following sense:

- The difference with Action Workflow is that our event chains focus on the triggering relationship only (so they do not support analysis of customer satisfaction), and distinguish clearly between events and event signs.
- Event Process Chains (EPC) model event chains of processes connected to each other by means of events. In our case, the processes (that we call events) are connected by means of signs that must have a physical representation.
- The difference with the DEMO process model is that in our opinion, this model tries to combine causal relationships between transactions and logical sequencing relationships within one transaction in one model without making clear what is their common denominator. Instead, our event chains models limit themselves to the modelling of observable triggering relationship abstracting away from the transactional structures. In terms of the DEMO approach, they should be positioned on the "documental" or "forma" level, a level that has received relatively little attention so far.
- The COMMODIOUS approach (Holm & Ljungberg, 1996) also models activities and speech acts as special kinds of activities, just as we have events and sign events. It also recognizes the need for description events to report on the occurrence of non-communicative activities. However, there are also differences: the COMMODIOUS approach does combine all kinds of aspects into one model (such as actors and the kind of IT support) and, whereas we focus on the triggering relationship enabled by signs, the COMMODIOUS approach considers the processes from a conversation and discourse logic perspective.

We have concentrated on the analysis of event chains within organizations. Such an analysis should be complemented with a method for inter-organizational business process modeling. We have not dealt with the design question of how to reengineer event chains using automation, and the implementation question of how to map design models on an EDA messaging infrastructure. Another topic for future research is the inclusion of Complex Event Processing, and the question how to connect the operational event chains to meta-processes that can adapt the operational event chains smoothly on the basis of information gathered by signaling events.

References

Andersen, P.B. (2005) Things Considered Harmful. Proc. ALOIS 2005.

Dietz, J. (1994) Business Modeling for Business Redesign. Proc. 27th HICSS Conference, IEEE Society, 1994.

Dietz, J. (2001) DEMO: Towards a Discipline of Organization Engineering. European Journal of Operational Research 128(2), pp.351-363.

Dietz, J, N. Habing (2004) The notion of business process revisited. In: Meersman, R., Z. Tari, Proceedings CoopIS 2004, Agia Napa, Cyprus, Springer LNCS 3290, 2004, pp. 85-100.

Goldkuhl, G. Agerfalk, P.J. (1998) Action Within Information Systems: Outline of a Requirements Engineering Method. In: Proc. 4th Int. Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'98), Pisa, 1998.

Holm, P, J. Ljungberg J (1996). Multi-Discourse Conversations. In: Proc. Of the 4th European Conference on Information Systems (ECIS), pp.835-848.

Johannesson et al, (2005) Design solutions for interoperability using a process manager. Proc. INTEROP-ESA, Geneva, Feb, 2005.

Luckman, D. (2002) The power of events: an introduction to Complex Event Processing. Addison-Wesley, 2002.

Medina-Mora R., T. Winograd, R. Flores, F. Flores (1993) "The ActionWorkflow Approach to Workflow Management Technology". In: The Information Society, vol. 9, pp.391-404, 1993.

Nelson, R, S. Winter. (1982) An Evolutionary Theory of Economic Change. Harvard Univ Press, 1982.

Papazoglou, M., D. Georgakopoulos. (2003) Service Oriented Computing: An Introduction. Comm. of the ACM 46(10).

Schulte, R. (2004) Using Events for Business Benefit. Business Integration Journal, May 2004, pp.43-45.

Stamper, R. (2000) "New Directions for Systems Analysis and Design". In Filipe, J. (ed.), Enterprise Information Systems, Kluwer Academic Publ., London, pp.14-39, 2000.

Umapathy, Purao, (2004) Service-Oriented Computing: An Opportunity for the Language-Action Perspective?. M. Aakhus, M. Lind (eds) , Proc. 9th Int. WC on the Language-Action Perspective on Communication Modeling (LAP2004), Rutgers University, New Brunswick NJ, pp.41-58

Weigand, H. & Heuvel, W.J.A.M. van den. (1998) Meta-patterns for electronic commerce transactions based on the Formal Language for Business Communication (FLBC). International Journal of Electronic Commerce, 3(2), 45-66.

Weigand, H. & Moor, A. de. (2003) Workflow analysis with communication norms. Data & Knowledge Engineering, 47(3), 349-369.

Weiser, M. (1991) The Computer for the 21st Century. Scientific American 265,(3).

Winograd, T. & Flores, F. (1986) Understanding Computers and Cognition: A New Foundation for Design. Ablex Publishing, 1986.