Systems, Signs & Actions
An International Journal on
Communication, Information Technology and Work
Vol. 3 (2007), No. 1, pp. 77–92
http://www.sysiac.org/

Linking event-driven and communication-oriented business modeling

Hans Weigand [a] and Aldo de Moor [b]

[a] Infolab, Tilburg University, The Netherlands, weigand@uvt.nl
[b] CommunitySense, Tilburg, The Netherlands, ademoor@communitysense.nl

Abstract

Event-Driven Architectures are a critical instrument for tomorrow's fast-acting and agile enterprises. Event process models are essential instruments, but they are hindered by a lack of abstraction. To benefit maximally from the technological innovations in service-oriented computing, businesses need to link an event orientation to an organizational communication perspective. The paper introduces a blended business process analysis method that combines event chain modelling with a communication-oriented approach without blurring important distinctions between the two levels, and that is applicable both in the case of automated processes and in the case of manual or semi-automated processes. The method draws on a novel event ontology in which the signalling function of events is recognized and on previous work in the Language/Action Perspective. The method includes a number of design rules that can be used to evaluate and improve event process models. The use of the design rules is illustrated by a small example of a library information system.

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

This paper is developed from the previous publication: Weigand H., de Moor A. (2005) Linking event-driven and communication-oriented business modelling, In *Proceedings of the 10th International Working Conference on the Language Action Perspective on Communication Modeling* (LAP-2005), Kiruna

Accepting Senior Editor: Göran Goldkuhl

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.

A related recent development is the realization of pervasive and ubiquitous computing, as predicted by the visionary paper of Weiser (Weiser, 1991) which says 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 processess. Processes consist of events and couplings between events. However, what does it mean in a business environment that an event *triggers* another event? What is the intended semantics of event chains (chains of events linked by trigger relationships) and how do we relate these chains to a communication model? 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 practical relevance of this paper is that it contains guidelines on how to evaluate and improve event process designs. The paper also contributes to the communication modeling theory by introducing a novel approach on how to relate communication models at the social level to event process levels at the physical level and by making an argument for the autonomy of each layer.

The structure of this paper is as follows. In section 2, we give some background into event-based processing. Section 3 is a discussion of the role of events in the Language/Action Perspective. In section 4, we will indicate how to link event process models to communicative action models, and in section 5 we present a couple of design rules that can support the analysis of event process models and their alignment to communicative action models.

2 Event-based processing

In this section, we briefly introduce the notion of event-based processing and its relationship to Service-Oriented Computing

2.1 The 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 in this context? 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. The term „software event“ is used for messages in the Information Sys-

tem that describe a business event. Software events are generated by an application program or some other software when they are informed 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.

According to (Schulte, 2004), the main ingredients of EDA are:

- 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
- use dynamic IT architectures, based on message-oriented middleware capable of publish-and-subscribe behaviour
- explore complex event processing (CEP)

Complex Event Processing (CEP), often related to as 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 credit card 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 and Georgakopoulos, 2003; Umapathy and 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 service request and its reply as two different events and ensuring that the reply is triggered by the request. Conversely, an EDA with a publish/subscribe mechanism could be built on top of a SOA by translating the

publication of an event into a service request for the services that are subscribed to this event. If there is a reply, it can be lifted up into a new published event.

3 Events in the Language/Action Perspective

In the Language/Action Perspective, the „action“ has always been given priority over static notions like „information“ and „representation“. Goldkuhl and Agerfalk (Goldkuhl and 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.

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 and 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 LAP 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 Model paradigm.

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 Figure 1). The communication part covers the initiation and an evaluation of the action.

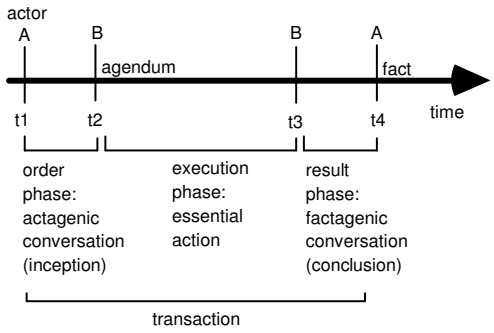

Figure 1: DEMO Transaction pattern visualizing the OER-paradigm (Order, Execute, Result)

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 model the interactions between the organization and its environment as well as internal transactions, and a Process Structure Diagram (PSD – previously called Process Step Diagram). The former can be seen as a representation of the „consensual domains" of the organization. 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). The notion of causal link is not explained, but apparently means that step A must or may (a so-called optional causal link) be followed by step B.

The PSD is based on the transaction patterns of the ATD 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 by the authors 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, in our opinion, 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 improve the actual process.

3.3 The Extended Workflow Loop model

The Extended Workflow Loop model (Weigand and 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, Figure 2 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 flow of events starting with the phone call from the customer and ending with the delivery of the pizza. Particularly interesting for the current paper 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 (also on behalf of the baker). 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 Loop 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.

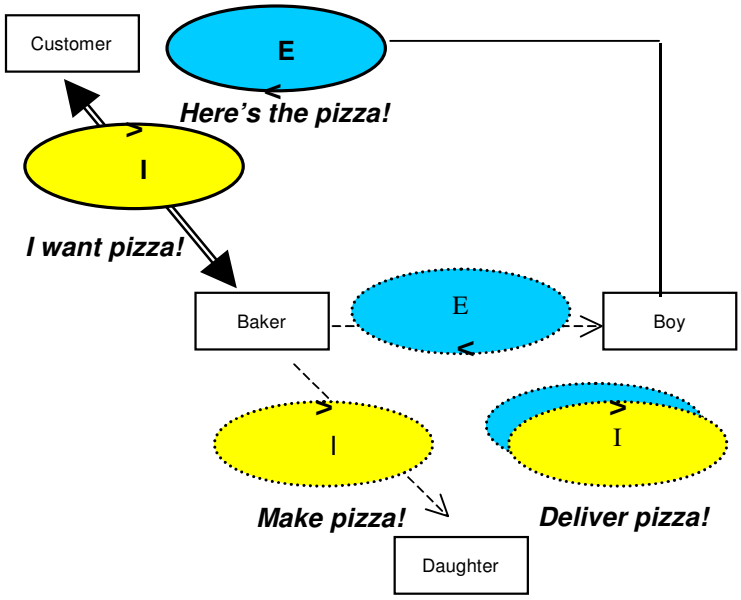

Figure 2: Pizza example of extended workflow loop. The Initiating (I) and Evaluative (E) parts of the transaction need not be performed by the same actor, as the actor can delegate actions. For more details about the figure and the pizza case, see (Weigand & De Moor, 2003)

The Extended Workflow Loop Model provides a meaningful interpretation of the interactions, but it does not explain how the event chain "works". From an event perspective, 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. Without the beep or some equivalent mechanism, the daughter would not come and take out the pizza, and without the pizza being on the designated shelf, the pizza boy would not take it away.

The Extended Workflow Loop draws on the same concepts as DEMO, but there are differences in how they reconcile the event processes and the interpretation of these processes in terms of communicative transactions. DEMO aims at highlighting the transactional structure of the process and for that reason prefers a more abstract process model. The Extended Workflow Loop aims at modeling both the actual event procesess and their communicative interpretation, and by introducing delegation it makes an attempt to reduce the distance between the two. However, both approaches are not explicit about how events, processes and causal links relate to communicative actions.

4 Linking event-driven and communication-oriented business modelling

The theoretical question is how an event process perspective is to be reconciled with a communicative action perspective (cf. Rittgen, 2006). We propose here not to choose one perspective at the expense of the other, as one cannot be reduced to the other exhaustively. Without drawing on the communicative logic, an event process model has no means to distinguish the essential and a particular realization of it. Conversely, a communicative interpretation alone cannot explain why one realization is better than another. It is not recommendable to mix the two perspectives, because that would mean a blurring of essential differences between two abstractions. The solution that we propose is to link the two perspectives by a mapping in analogy to the difference in language between surface structure and deep structure (semantic structure). The event process model can be seen as a surface structure that implements the „deep structure“ of the communicative model. The deep structure describes the essence, but does not determine the surface structure exhaustively. We should therefore keep both layers, and identify the norms that are important in both, in order to assess the quality of the model. 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 communicative model there should be some event sign in the event process model that can be interpreted as the initiating action in the workflow model.

By blending a communicative action model with an event-chain model, we meet the critique of Goldkuhl and Agerfalk on event-based modeling approaches (Goldkuhl and 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, and with the advanced use of RFID technology, the role of anonymous signs (as opposed to intentional speech acts) 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.

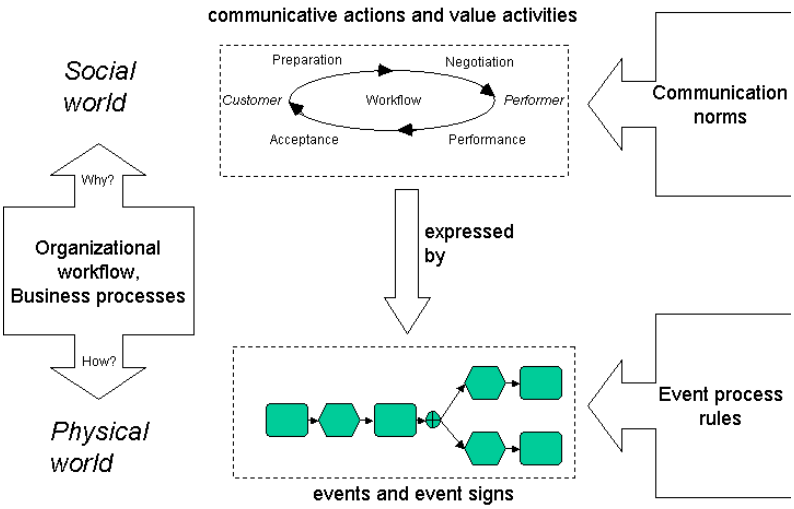

Figure 3: Communicative action and event process chains: a blended approach

Figure 3 presents the blended approach. Starting from business processes, we have to go to the social world to find the rationale behind the process (*why*), and we have to go to the physical world (including human actions and automata) to find out *how* they work. In the social world, we have communicative actions and value activities (the latter corresponding to what Dietz calls the Production World, which should not be confused with the physical world, as the most important aspect of the action at this level is its social value). Both levels are governed by their own norms or rules.

In the next section, we will provide some guidelines for analyzing event processes based on our blended approach. Before that, we introduce an event ontology that actually bridges the event and communicative action perspective (Figure 4).

Events can be *composed*. Most (not all) events cause a *transformation* of some state into another, for example, changing the location of an object. Events are typically enabled by a certain *habitat* (Andersen, 2006); for example, a shop enables buying goods, and a chess game enables certain specific moves. Entities of some domain can play a certain *role* in an event (e.g. agent, object, instrument). The ontology is focused on events and we do not address the issue when a certain event is to be seen as an action. A key notion of event models is the *triggering* relationship between events. Although this notion is used in many approaches (such as event process chains), it is a far from simple concept (cf. Dietz, 2001). A triggering B is more than A preceding B. 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 more is needed than the triggering event. The domino effect does not take place when someone holds up the next stone with his finger, for example. The ontological representation of "trigger" in Figure 4 is to be read as: "if <enabling condition> then, when <source event change>, because of <norm>, <goal event change>, unless <disabling condition>. For example: *if* the domino stones are placed correctly, then *when* the first stone is hit (hit event is finished), *because* of gravity law, all

stones fall down (fall event is finished), *unless* a counterforce keeps it upright. Note that the source and goal of triggers are not events, but event state changes, as we do not assume events to be atomic. Typical event state changes are the completion of an event, the failure of an event, its initialization and perhaps it pausing.

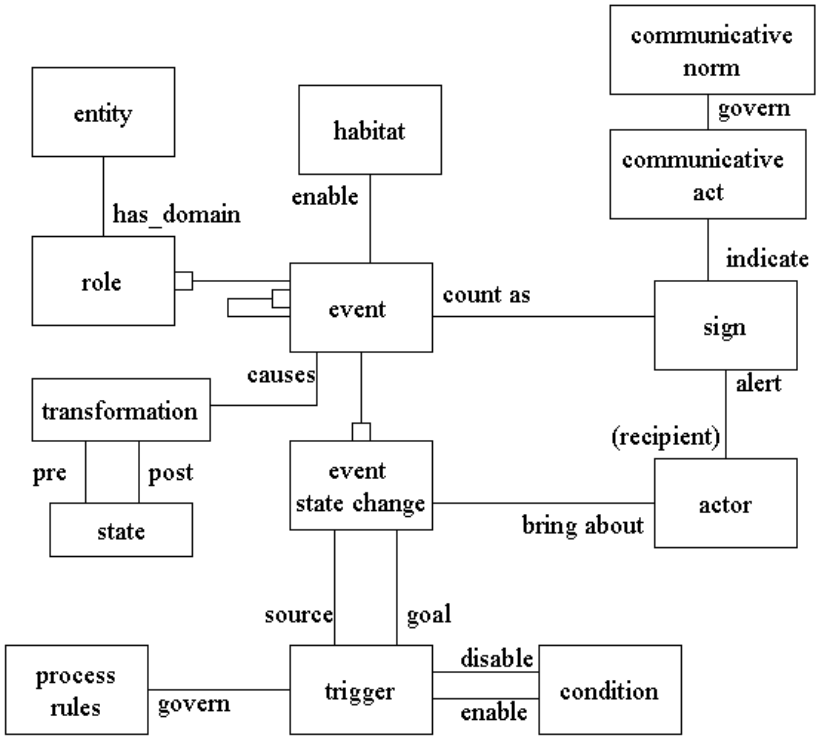

Figure 4: A basic event ontology

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. 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 *rules* need not be explicit, and humans may not even be aware that they follow a convention. Sometimes the rule is laid down in an explicit procedure, and sometimes it is implemented into a program thereby avoiding the human intervention.

In the human world, the triggering of an event will not work without some signalling. The triggered event is only started after some actor has become aware of the triggering event. We call this actor that the sign appeals to (Buhler, 1930) the *event recipient*. The *sign* associated with, for instance, a purchase order (i.e. the incoming digital message) is what triggers the EDI-program to send an acknowledgement. In the case of the pizza to be delivered, the sign is the visible presence of the pizza on the dedicated shelf. It is important to see that the sign has (at least) two basic func-

tions: to make the event recipient aware *that* he should act, and to indicate *how* he should act. For the latter function, the sign may draw on the shared background, the normative context, and even to such a degree that the sign contains no indication of what to do at all. Even in this extreme (but quite common) case, the sign keeps and must keep its former function, that of alerting. If the sign would be taken away, the event recipient does not know when to start. Therefore, we require all initiating acts in the social world to be indicated by some event in the physical world. This can be in the form of an explicit message that indicates a communicative act, but any other event with an alerting function can do as well (assuming the normative context is in place). The initiation cannot be performed tacitly, that is, without any sign.

In the event ontology, an event can *count as* a sign. Not all events count as signs, but a sign cannot do without a physical form. The sign may alert the event recipient. Apart from the alerting, the sign may *indicate* a communicative act, but this is not necessary. For example, the beep produced by the oven when the pizza is ready does not indicate a communicative act. The beep alerts the pizza girl and triggers an action.

What about the other communicative acts, such as the evaluative act? Here the situation is quite different. The evaluative act is not necessarily followed by some action, and if there is a next action, it will have its own initiating act, so a sign with triggering function is not necessary for that purpose. A certain sign can indicate an evaluative act, but in principle the act could be performed tacitly or it could "piggy-back" on another sign. However, alerting is not the only criterion. The problem with a tacit performance is that it does not leave a trace. Without a trace neither party in the transaction has evidence afterwards that the evaluation has been performed successfully. So a second function of signs is that they serve as event *records*. In that case, the sign must be pervasive. In this paper, we do not work out this function of signs as we concentrate on the trigger function.

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 for the same process. In this section, we present some important checks that could be made to analyze and improve event process chains. 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). The example process that we adopt is not an automated one and refers to physical signs between humans in a library organization. However, if the process model captures the design of an automated process, the same principles apply, the signs being typically messages between services. It does not matter whether the model is a descriptive model of a running process or the design of a new process; in both cases, the checks can be used to provide suggestions for improvement.

Four groups of checks are considered:

1. check for communicative transaction completeness
2. check for event triggering completeness
3. check for failure handling
4. check for potential deadlocks

5.1 Check for communicative transaction completeness

- Check whether the event model is complete with respect to the communicative model.

(a) every communicative action should have an expression in the event model

(b) *every production act should have an expression in the event model.*

The check is performed by first considering the signs at the event process level, and see whether they can be interpreted as factagenic or actagenic. In this way, every communicative action should have a corresponding event (an event may indicate more than one communicative action). If such an event sign cannot be found, then one needs to be added. This is one way to ensure that communicative (trans)actions are properly implemented through (sets of) event chains.

Example: suppose in a library we identify the following incomplete event chain:

Customer: fill in lend form → library assistant: take book from shelf → loan → return ...

The lend event is a service that provides value to the customer. So in the communicative model, there should be both an actagenic and factagenic conversation around it. The check to be made now is whether the event model is complete with respect to the communicative model. The lend request (that could be a form filled in by the customer) can be interpreted as the start of an actagenic conversation. However, what is missing in this event chain is the committing (the library granting the request of the customer). What is also missing is something that can be interpreted as the factagenic conversation. 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. Typically, 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, where we have annotated the events with their communicative interpretation:

Example:

Customer: fill in lend form (REQUEST) → library assistant: „ok“ (COMMIT) → library assistant: take book from shelf (EXEC) → library assistant: prepare book → library assistant: give lending note (EVAL+REQUEST)→ customer: return book (EXEC)

The lending note indicates the completion of a factagenic conversation, so we interpret it as a communicative action. At the same time, it functions as a return request from the library (as the returning is a second service in this example).

The completeness check often amounts to the analysis of whether all endpoints in the event chains are indeed endpoints. For example, returning to the pizza case: when the pizza boy has delivered the pizza and received the money, and the note is signed, is this the end of the chain? Obviously not, the note and the money need to be given to the baker who collects the money and checks the notes (this is part of the evaluation side of the delegation loop).

5.2 Check for event triggering completeness

- Every non-sign event should be triggered by a sign event
Restriction: when two subsequent events are executed by the same actor, a mediating sign is not necessary

As we have argued extensively above, the sign is needed to trigger the subsequent event in the chain when this event is to be executed by another actor. This leads to a variant of the rule well-known in the context of Event Process Diagrams that says, in our wording, that in the event process chain, there should be an alternation between events and signs (EPD uses the terms ‚process' and ‚event', respectively). Note that we do *not* require that every event (change) leads to a sign, so it is possible that a chain ends with an event. We also do not require that a sign *should* trigger a next event, so a chain can end with a sign as well. However, when it does not have an alerting function, there must be another motivation for this sign, such as a record function. We allow for the possibility that a sign event is followed by another sign event.

Example:

customer: loan request (S) → library assistant: take book from shelf (E) → library assistant: record loan (E) → library assistant: hand-over book (E/S) → customer: take book (E/S) → customer: return book (E/S) → library assistant: put back book on the shelf (E)

- We use E and S to indicate an non-sign event and a sign respectively. E/S indicates a combination: an event that causes a transformation and couns as a sign
- the handing over of the book is represented here as a combined event/sign. The non-sign event is the custody transfer of the book, the sign is the alerting of the customer to take the book. The sign function can only be effective when the customer is watching the library assistant. Otherwise there should be a separate sign, e.g. a verbal call.
- „take book". This event does not have an alerting function, but we hold that it still counts as a sign, as it indicates an accepting communicative act of the customer.
- „return book". Returning is represented here as a combined event/sign as well. The sign could be explicitly putting the book on a designated shelf of the library. This sign is the trigger for the subsequent action. As in the case of handing over the book, a separate sign may be used as well.
- It can be checked that the example above is complete from a triggering perspective

5.3 Check for failure handling

- Events should not have just event signs (event completion signs), but also event failure signs

An event that is triggered may fail to execute. Failure is one of the event states (Fig. 4). The general rule is that each event failure triggers some other event that deals with the failure, typically a corrective communicative action (or action pair, such as an request for clarification, plus reply from the target actor). Of course, this rule can-

not be applied 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:

customer: loan request (S) → library assistant: take book from shelf (E) [*failure*] → ??

The take-book event may fail if the book is not on the shelf. This breakdown should not be an end of the chain, but trigger for example a question to the customer whether he wants to reserve the book, but also internal actions in the library aimed to rediscovery of the missing 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 an 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. For example, if the number of missing books rises every month, management may decide to take extra security measures.

5.4 Check for potential deadlocks

- The process should have one or more starting event signs, and all non-starting events in the chain should be triggered by some event sign.

A starting event sign is needed to start the process (create a new process instance). During the process, a deadlock can occur, by which we mean here that a certain event should execute but it does not as it is not triggered (in some cases, this may be due to process conflicts around a shared resource, but we do not consider these in this paper).

A situation that often occurs is when a certain event must await another event. For example, in the library a reserve request has been made by a customer for a book that is not available. The reserve request aims to trigger the „put book on reserved books shelf“ event, but this event 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 then this check triggers the „put book on reserved books shelf“ and the sending of a notification mail to the customer. Suppose we model this by means of two event chains:

Example:

request reserve book (S) → file request (E)

return book (S) → check status (E) ⟶ put book on reserved books shelf (E)

(= „reserve book“)

send notification mail (S)

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 are just put back

in the library) and so we have a reservation request that is waiting indefinitely. As a consequence, there is also no evaluative communicative action from the library to the customer, as we would expect according to the rule 5.1 above. 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. But this is not sufficient. The book may never be returned, so that the second event chain is never executed. 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.

Example:

customer: request reserve book (S) → timer [40 days] (E) → system: alert message(S) → library assistant: clear request (E) → library assistant: inform customer that the book is lost (S)

Now the request triggers a timer that, after 40 days, triggers a check event that triggers a notification (note that we have chosen an event-based approach, rather than a batch-oriented approach in which e.g. all open reservations are checked every night, which would be an alternative solution).

6 Conclusion

When event-based business processing grows in importance, 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. We have also drawn attention to the alerting function of the sign, besides its function as indicator of a communicative act.

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 we do not support analysis of customer satisfaction), and distinguish clearly between non-sign events and event signs.
- Event Process Chains (EPC) model event chains of processes connected to each other by means of events. We have identified these events with signs. In this way, the EPC connections do indicate more than a temporal precedence relationship and become effective triggering relationships.
- The difference with the DEMO process model is that in our opinion, this model tries to combine causal relationships between transactions on the one hand and logical sequencing relationships within one transaction on the other hand in one model without making clear what is their common denominator. As a process model, it seems to be too much dominated by the communicative action logic. Instead, our event chains models limit themselves to the modelling of observable triggering relationship abstracting away from the transactional structures. In DEMO terminology, our event chains are intentionally positio-

ned on the "documental" or "forma" level, a level that has received relatively little attention so far.

- Rittgen (2006) describes an experiment in mapping DEMO models to UML models and concludes that this is possible, but not without a loss. As far as a mapping is established between the communicative action models and process models, there are similarities with our paper. However, we do not regard this mapping as a mapping between two (non-reconcilable) views, but as a mapping between different autonomous levels of the same communicative action, as pointed out under the previous bullet.
- The COMMODIOUS approach (Holm and Ljungberg, 1996) models activities and speech acts as special kinds of activities, just as we have events and event signs. 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.

There are many issues for future research. 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. We also have not worked out the issue of event records, pervasive representations of event signs that may be needed for internal control purposes. Another topic for future research is the inclusion of Complex Event Processing, and the question how to connect the operational event chains to management processes that can adapt the operational event chains smoothly on the basis of information gathered by signaling events.

References

Andersen, P.B. (2006) Activity-based design. *European Journal of Information Systems* 15, pp.9-25.

Dietz, J. (1994) Business Modeling for Business Redesign. *Proceedings of the 27th HICSS Conference*, IEEE Society.

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

Dietz, J, and Habing, N. (2004) The notion of business process revisited. *Proceedings of CoopIS 2004*, (MEERSMAN R and TARI Z, Eds), Springer LNCS 3290, pp. 85-100.

Goldkuhl, G. and 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, PJ. and Ljungberg J. (1996). Multi-Discourse Conversations. In: *Proc. 4th European Conference on Information Systems (ECIS)*, pp.835-848.

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

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

Papazoglou, M. and Georgakopoulos, D. (2003) Service Oriented Computing: An Introduction. *Communications. of the ACM* 46(10), pp.24-28.

Rittgen, P. (2006) A language-mapping approach to action-oriented development of information systems. *European Journal of Information Systems* 15, pp.70-81.

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 Publishers, London, pp.14-39.

Umapathy, K., and Purao, S. (2004) Service-Oriented Computing: An Opportunity for the Language-Action Perspective? *Proceedings of the 9th international working conference on the language-action perspective on communication modelling* (AAKHUS M and LIND M, Eds), pp. 59–76, Rutgers University, The State University of New Jersey, New Brunswick, NJ, USA, pp.41-58

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

Weiser, M. (1991) The Computer for the 21st Century. *Scientific American* 265,(3), pp.94-104.

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

About the Authors

Hans Weigand is Associate Professor at the Department of Information Management, Tilburg University. His research interests include communication modelling, business/IT alignment and adaptive service-oriented computing.

Aldo de Moor is research consultant at CommunitySense, the Netherlands. His research interests are evolution of virtual communities, communicative workflow modeling, argumentation support technologies, Language/Action theory, conceptual graphs, and ontology-guided meaning negotiation.