Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Data & Knowledge Engineering 47 (2003) 349–369
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Workflow analysis with communication norms

Hans Weigand *, Aldo de Moor

Infolab, Department of Information Systems and Management, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Received 19 February 2003; received in revised form 26 February 2003; accepted 26 February 2003

Abstract

The language/action perspective (LAP) as orginally introduced by Winograd and Flores has inspired several tools and information system design methodologies. The goal of this article is to make the communication norms underlying various LAP workflow loop models (DEMO, ActionWorkflow) explicit and to contrast them with the auditing norms of internal control. It appears that the communicative action paradigm embedded in DEMO and the customer satisfaction orientation of ActionWorkflow lead to norms which resemble the ones required by internal control, but there are some important differences. For that reason, we propose an extended workflow loop model that distinguishes between customer relations and agency relations. Whereas current LAP approaches do not take agency relations explicitly into account, the extended workflow loop model allows us to analyze the effects of delegation on communicative processes. A framework is offered for the normative analysis of workflows based on a number of formalized communication norms.

Keywords: Language/action perspective; Communication norms; Workflow

1. Introduction

During the last two decades, the language/action perspective (LAP) as orginally introduced by Winograd and Flores in 1986 [21] has inspired several tools and information system design methodologies. ActionWorkflow [12] and dynamic essential modelling of organizations (DEMO) [7,8] are two approaches that offer a special modelling method for business processes based on

* Corresponding author. Tel.: +31-13-466-2806; fax: +31-13-466-3069.
E-mail addresses: h.weigand@uvt.nl (H. Weigand), ademoor@uvt.nl (A. de Moor).

0169-023X/$ - see front matter
doi:10.1016/S0169-023X(03)00064-8

LAP. In contrast to data-oriented methods such as state transition diagrams, or UML interaction diagrams, LAP modelling is based on the notion of *communicative action*. This means that communication is viewed at the level of social relationships. For example, a request is aiming at the performance of a certain action, but it is also an action itself. A successful request creates an obligation, and although the physical world has not changed yet, the social world has. Viewing communication at the level of social action means that the focus is not on the efficient transmission of some data content from one place to another, but on organizational coordination. This is in line with the two major philosophical sources of LAP: on the one hand John Searle, who has placed his speech act theory explicitly in the context of human institutions [16], and on the other hand Jürgen Habermas, who has developed a theory of communicative action as part of a sociological critique on modern coordination structures [10].

From a coordination perspective, communication processes are more than sequences of communicative acts. The LAP perspective imposes a certain structure on communication processes. In the case of DEMO, this is the transaction paradigm, in the case of ActionWorkflow, the ActionWorkflow loop (see Section 3). This imposed structure excludes certain "ill-formed" processes. Data-oriented approaches do not impose much: it is not difficult to draw a use case diagram that is syntactically correct, but does not make any sense as communication. Some process-oriented approaches in business process modelling are based on Petri Nets. Petri Nets have the advantage that formal verification techniques can be used to test certain properties. However, a Petri Net in itself does not impose more communication structure on the process than a data-oriented approach.

An important advantage of the LAP approaches—the structure they impose—is sometimes also a point of criticism. According to Suchman and others, the ActionWorkflow loop is too restrictive [18]. It is said that in practice the analyst is confronted with situations that do not adhere to the workflow loop principle. The crucial question is of course not whether such situations occur, since that is obviously the case, but how to evaluate such a deviation. If the deviation turns out to be a distorted communication process, then it is a virtue that the LAP model indicates how this process must be redesigned. However, in order to make a strong case for the advantage of such a normative application of the model, it is essential that the normative principles underlying it are explicated. *Why* is it so important that the "loop is closed", as ActionWorkflow claims?

The critique of Suchman was particularly aimed at the *imposition* of norms in systems such as the Coordinator. Users would be forced to make all their commitments explicit, and this would introduce just a new form of bureaucratic control. Whether the Coordinator was as intruding as Suchman suggested, is a matter of discussion. What the argument makes clear, however, is that the *recognition* of certain norms should be distinguished from the *imposition* of these norms. In this article, we want to analyze the communicative norms in workflow situations. These norms can be used, for example, to diagnose practical situations and explain why some situations are problematic. The norms can also be used to suggest alternative structures. However, the question whether the systems allow norm-conflicting behavior, or whether the organization allows norm-conflicting behavior, must be addressed in its own right. In doing so, the *costs* (economic, social, personal) related to deviations must be taken into account, as wells as the *benefits* of adhering to the norms. In addition, the *feasibility* of imposing these norms needs to be considered. Finally, there is also a *cultural* aspect; after the Enron case in the US and similar cases in other countries, the question of norms in business is receiving more positive attention than a few years ago.

Talking about communication processes in the field of Information Systems means talking about workflow and business processes. According to the Workflow Management Coalition, a workflow is "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". For us, the automation is not essential. However, the fact that documents, information and tasks are passed from one participant to another means that workflow contains communication. In the majority of workflow approaches, this communication is considered only at the data level and the norms that are considered are also on that level (e.g. optimizing the critical path). The LAP approaches claim to address communication at the social level. We therefore take LAP as our starting point.

The objective of this article is then to explicate the norms inherent in the LAP models, in particular, DEMO and ActionWorkflow. In that respect, it takes a meta-modelling approach. Section 2 introduces the notion of norm analysis based on Stamper's semiotic approach. Section 3 provides a brief overview of the mentioned LAP models. In Section 4, an overview is given of communication norms derived from internal control theory used in accountancy. In Section 5, we make the norms underlying LAP workflow models explicit and compare them with internal control norms. Section 6 introduces our formal framework for the normative analysis of the workflow loop paradigm, combining elements from the approaches discussed. Section 7 is the conclusion.

2. The role of communication norms in workflow modelling

Our view on communication is based on Habermas's theory of communicative action [10]. Habermas stresses the cooperative nature of communication when it is used to coordinate actions. Speakers do not just forward information, but make validity claims: for example, by an assertion that company X produces product Y, they claim that company X exists, and that it is *true* that this company produces Y. A claim can be challenged. The hearer may disagree, or may not yet be convinced. The communication is successful when, perhaps after some discussion, the partners agree on the claims. Evidently, this framework presupposes that the parties already have some common pool of facts and norms that can be used to back up a claim, or challenge it. This background knowledge does not need to be explicit, and should not be seen as something static and fixed.

Today's Internet-age information systems are much more communication than computation systems. There are many applications that support complex communication processes, like discussion and group decision making, as well as many kinds of collaborative work such as group authoring. The semiotics of these systems are often much more complex than in traditional information systems, particularly because the intended semantics and pragmatics are not under the control of one single organization, and therefore often remain un(der)defined. This entails that often the meaning of information produced and responsibilities for system use and specification are not clear.

In order to deal with such problems, we need to move away from the traditional information flow paradigm, in which positivistic modelling aimed at producing automated solutions is central. Instead, an *information field* paradigm is needed [17]. At the core of this semiotic paradigm are

fields of *norms*, binding together groups of people. The norms allow meaning and responsibilities to be clearly specified, thus fostering the active construction of social reality, shared understanding and mutual commitments. A norm should be understood here as "a principle of right action binding upon the members of a group and serving to guide, control, or regulate proper and acceptable behavior" (Merriam Webster). Communication presupposes that the communication partners agree (or can arrive at agreement) on certain norms: not only norms on the language that is used, but also norms for valid inferences (e.g. modus ponens), norms for interpretation, and norms for perception. For example, if trading partners talk about the arrival of goods in the harbor on a certain date, there must be some agreement on how this is perceived (is it the customs report that is conclusive?), and how it is to be interpreted when the goods have arrived at the customs at a certain date, but only cleared a few days later. Can we infer from the fact that they arrive at a certain date, that they will be delivered at most one week later?

Norms need not be explicit at all in order to be operative. In normal (sic!) circumstances, we take rules for granted and we apply them automatically. However, when a *breakdown* occurs—for example, two communicating parties disagree on the meaning of a term, or on the classification of the product—it becomes clear that the rules are not self-evident, and they are revealed as norms (cf. [13]). Since they are norms, they can be violated, even if that would be exceptional, and the communicating parties can be held responsible. Note that "norm" should not be interpreted in the narrow sense of laws or ethical rules imposed by some society or institution. A norm is something we apply in our daily practice and that we expect others to apply as well. When a norm is made explicit, it typically takes the form of a rule (a prescribed guide for conduct of action, Merriam Webster).

Communication is a very complex phenomenon, in which many levels can be distinguished. From all the norms in the information field that can play a role in communication, we focus in this article on one specific but important group: the group of norms regulating communication *processes*, and even more specifically, communication processes consisting of several interactions. We are interested in grinding a diagnostic lens specifically designed for detecting workflow complexities and deficiencies. For that reason, we do not consider linguistic norms (such as the norms of correct business English), nor the norms that regulate single interactions (such as Grice's maxims—"be relevant!", "avoid ambiguities!" etc. [9]). We also do not consider the very high-level norms that Habermas has discussed under the term "discourse ethics". In order to find norms for communication processes, we start with a normative analysis of LAP-based workflow modelling methods. In the following, we will use the term "workflow" to designate a communication process consisting of several interactions aimed at some particular business goal.

3. Business modelling—DEMO and ActionWorkflow

To build a normative framework for workflow processes, we analyze the two above-mentioned well-known LAP-business modelling approaches: ActionWorkflow and DEMO.

3.1. ActionWorkflow: the customer orientation

ActionWorkflow [12] is a theory about the organization of work taking a LAP and relies on theoretical work of [21]. ActionWorkflow can be seen as generic business framework, or a

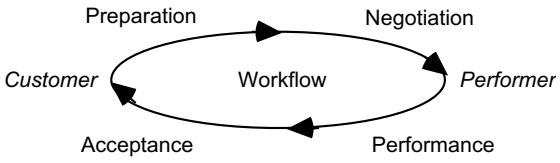

Fig. 1. ActionWorkflow.

business process and workflow analysis and modelling method, and is also the name of a supporting software tool. It uses the 'work is a closed loop' idea (Fig. 1). According to [5], traditional workflow management methods have been *production*-centered, focusing on efficiency and control, whereas their approach is *satisfaction*-oriented, with a central focus on commitments, conditions of satisfaction and timely completion.

Business processes are split up in elementary transactions between a customer and performer and consist of four steps: preparation, negotiation, performance, and acceptance. The first two steps aim at the establishment of a commitment of the addressee to perform an action. The last two steps aim at the establishment of the performed action. The action itself is not modelled, only its results. In both parts there is negotiation aimed at mutual agreement of what has to be established. The ActionWorkflow theory (with its roles and phases) can be seen as a generic blueprint for the organization of work.

ActionWorkflow is based on the recognition that, in practice, the loop is often not closed, and this is denounced as a source of problems. "Incomplete work flows invariantly cause breakdowns, and if they persist, they give rise to complaints and bad feelings that interfere with the ultimate purpose of work—to satisfy the customer". It is stated that "many of the problems that plague organizations are connected with persistently incomplete work flows" [5].

"Closing the loop" is a central underlying norm of the ActionWorkflow approach. The approach incorporates some important principles: (a) work in organizations is done *for or on behalf of* somebody, (b) task assignment should be followed up by task evaluation, and (c) facts derive from actions having been performed, so fact creation (as part of the task evaluation) is preceded by task assignment. We will come back on these principles in Section 5.

3.2. DEMO

DEMO [7,8] 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 Information System 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 an *order* 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 ($t4$).

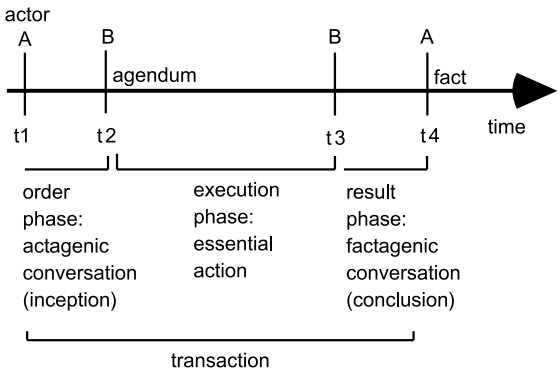

Fig. 2. DEMO transaction pattern (after [7]).

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.

From a distance, the ActionWorkflow loop looks very similar to the DEMO transaction pattern. However, there is an important difference in perspective. Whereas ActionWorkflow motivates the use of workflow loops by the problems that occur when the loops are not closed, according to DEMO, the transaction pattern just models how things are. If an actagenic conversation is found followed by some execution, but without factagenic conversation, the DEMO approach concludes that there is no transaction in this case, but DEMO makes no statement on whether this is good or bad. On the other hand, the DEMO paradigm considers it the normal case that communication is realized in transactions. For that reason, the DEMO handbook [7, p. 51] can state that often some of the communicative actions making up a transaction are missing. ''But if there is at least one communicative action, the other must be there also, in one way or another''. This ''must'' is not meant as a strong norm, but as a design heuristic.

Allthough DEMO itself does not want to attach normative value to the transaction pattern, a norm analysis must conclude that the same underlying principles that we mentioned in the above are applied. However, the norms are weak. The fact that DEMO allows communicative actions to be missing or implicit, shows that apparently, other norms, such as efficiency, may overrule the communication norms.

A feature of the DEMO approach, and particularly the transaction pattern, is that it is highly abstract. This abstraction is very useful in a certain phase of the design process, but it can go at the expense of concrete guidance on how to improve communication structures. For example, the roles of initiator and executor can be fulfilled by one or more subjects. Therefore, it is possible that the initiation (starting the actagenic conversation) is performed by one subject, and the evaluation (closing the factagenic conversation) by another—they are performing together the role of ''initiator''—and it is also not excluded that initiator and executor roles are assigned to the same subject. It is not easy to say whether DEMO prefers the initiator and evaluator to be the same subject—what ActionWorkflow strongly promotes—although the fact that there is only one role suggests that there must be some close connection.

To conclude: we can identify workflow norms in both ActionWorkflow and DEMO. ActionWorkflow applies them in rather straight-forward way, thereby running the risk of being too

restrictive, whereas DEMO seems to be rather liberal about the application. What is needed, in our opinion, is not a discussion of one approach against another, but a careful analysis of each individual norm. To analyze the norms on a generic level, a kind of meta-model is needed. In order to arrive at a truly generic meta-model, we first consider the wider context of internal control theory that represents an old tradition of configuring information and communication processes in organizations.

4. Internal control theory

The information system in an organization provides the information to its actors to execute and coordinate their tasks. According to audit theory, an organization does not only need an information system, but also an internal control system to secure trustworthiness of the registered information and to control potential errors. According to [15], internal control is needed when an organization has a delegated task structure which allows agents to establish commitments on behalf of the organization, to employ certain funds, goods or products. Delegating an activity does not mean that the responsibility for this activity is delegated as well. Instead, it introduces a control task for the principal that delegated the task to the agent. This involves communication: since in most cases, the principal responsible for that control task cannot personally observe the performance of operational tasks, he must rely on documentary evidence (evidence function). At the same time, to protect himself, the executing party (the agent) must be able to prove the completion of an activity (preventative function).

Chen [3] has listed a couple of control principles based on his review of the accountancy literature that are worth citing:

1. If an agent-based operational task exists, its corresponding control task should exist as well and should follow the operational task.
2. If a control task exists, it must be furnished by supporting documents. These supporting documents should be the result of a preceding control task that directly witnesses the activity to be controlled.
3. Supporting documents should be generated by a source independent of the source which generates the document to be verified.
4. If a control task uses a supporting document, this should be transferred directly from the preceding control task which generated it.
5. An operational task and its corresponding control task should be segregated into two different organizational positions and into two different agents.
6. An operational task and its corresponding control task must not be lower in the formal power hierarchy than the position responsible for the operational task.
7. The agents responsible for the operational task and its corresponding control task should be socially detached.

Control tasks can be divided into two categories: control tasks that make direct statements about the operational tasks (such as witness reports), and control tasks that evaluate the resulting document and draw conclusions based on them [1].

The principles of Chen focus on control tasks from the principal's perspective. Starreveld (ibid) states the fundamental principle that agents should render account of the tasks delegated to them. This is not only an imperative from the side of the principal—since the principal delegated certain authorizations and access to resources to the agent—but also from the side of the agent there is a need to to be able to "finish the job" and obtain discharge from his responsibility.

The principles above assume an intra-organizational context (e.g. number 6). An inter-organizational context, such as international trade, makes use of some of the principles above. It also needs some additional rules, in particular concerning the contracting process that results in certain obligations for the operating party. Dewitz [6] has defined the following rules:

1. The role issuing the promise should be responsible for the primary activity being promised.
2. The beneficiary of this activity should receive the document (with the promise).

The basic motivation for these rules lies in the absence of a hierarchical relationship between the parties involved. For example, the seller may hire a shipper for the transportation, but the relationship between the shipper and the seller is a contractual one, not a hierarchical one. Bons formulates his own general principles of inter-organizational controls based on the lists of Chen and Dewitz, in which he explicitly pays attention to the implicit or explicit outsourcing of activities. Another addition that he makes has to do with the reciprocal character of a business contract. A contract requires some rules on the performance of one's own activity in relationship to the counteractivity. In this paper, we do acknowledge the fundamental character of reciprocity in inter-organizational settings, but we focus on individual workflows and hence will not consider the specific contractual norms.

5. Making the workflow loop paradigm norms explicit

We are now in a position in which we can analyze the norms inherent in the DEMO paradigm and ActionWorkflow approach (together called workflow loop paradigm in the following) and relate them to principles of internal control. For that purpose, we will formulate the workflow norms in the terminology of internal control theory. This gives us a rather explicit formulation. Whether the norms of internal control theory, focusing on delegation, and the workflow loop norms, focusing on customer satisfaction and communication, can be unified, is a question that we want to address after we have achieved the explicit formulation first.

5.1. The workflow loop norms

In a neutral terminology, the workflow loop paradigm incorporates at least the following normative principles:

1. For any activity, a distinction must be made between the *operational task* and the *control* task. These two tasks are executed by different roles and different subjects.
2. If an operational task exists, there should be a corresponding *initiating* control task and the operational task should follow the initiating task.

3. If an operational task exists, its corresponding *evaluative* control task should exist as well and should always follow the operational task.
4. The initiating task should contain a request for action from a role (initiator) independent of the role performing the task.
5. The role issuing the initiating task (initiator) should be the same as the role responsible for the (evaluative) control task.
6. The initiating task should be closed with a commitment (promise) from the role performing the operational task.
7. The evaluative control task should be furnished by supporting documents. The supporting documents should originate from the role performing the operational task.
8. The evaluative control task should be closed with a performative statement from the role performing the evaluative control task.
9. The performative statement of the evaluative control task should be received by the role that performed the operational task.

Ad 1. The first principle corresponds to principles 1 and 5 of Chen. In DEMO even more clearly than in ActionWorkflow, the transaction contains an execution part and a control part. There is a clear distinction between the initiator and executor roles. The principle that these two roles are filled by two different subjects, is not stated explicitly, but is, in our opinion a logical consequence. In this way, evidence is produced for the initiator that the action has been performed, and the executor is discharged from the obligation (and can prove that later).

Ad 2. The second principle is derived from the occurrence of the Order phase in the DEMO transaction pattern that precedes the Execution phase. This principle is not found in Chen's list, but can be found in the additional principles of [6]. The principle can be defended since it is not desirable that an agent performs a certain task without being requested to do so. Also, for the agent himself it is beneficial, not to say necessary, that he is backed up by an explicit instruction when he is later asked why he did perform this particular action. Think in particular of an agent who is made responsible for the access to some goods, such as in inventory manager or a treasurer, and the action consists in giving away these goods. The reason that Chen omits this principle is probably his exclusive focus on evaluative control tasks.

Ad 3. This principle is derived from the occurrence of the Result phase in the DEMO transaction pattern that follows the Execution phase, and corresponds to principle 1 of Chen.

Ad 4. This principle is a refinement of principle 2. Although the term "initiator" suggests that this role takes the initiative, we have formulated the principle a bit more general: the task should contain a request, but this may also have been prompted by an offer of the executor.

Ad 5. This principle is central to the customer-orientation in ActionWorkflow, but less obvious from a control point of view. From the control point of view, it is sufficient that the task is (independently) evaluated by the organization, but it does not matter who performs the evaluation. For the customer-orientation of ActionWorkflow it *does* matter; if the evaluation is performed by a third party, or the boss, this may be sufficient for preventing fraud and for discharging the agent, but it may not be sufficient for the customer being satisfied.

Ad 6. This principle is also rather central in the ActionWorkflow approach. It is also found in the principles of Dewitz. For the principal/beneficiary, a commitment is important, and it is

natural that the commitment is given by the agent that performs the action. However, it is not immediately clear that this commitment must be as explicit *within* an organization as it must obviously be *between* organizations. In the formal model developed later, we will relax this norm somewhat.

Ad 7. This principle is about the Result phase. It seems to contradict principle 3 of Chen that not the agent himself, but another party should provide evidence of completion. However, the contradiction can be resolved by observing that the account provided by the agent himself is not taken at face value, but is evaluated and verified in a next step. In this final evaluation step, the evaluator can use third-party evidence. The organization should have certain norms that mediate between the execution of the material act and the verbal report of that act. These norms should foster objectivity and prevent fraud.

Ad 8. This principle has some relation to principle 2 of Chen, but is stronger. It stresses the importance of discharging the executor in the closing part of the workflow loop. However, it is not evident that this statement has to be explicit in all circumstances.

Ad 9. This principle is open for discussion. It is important that the executor is discharged, but it could be sufficient if the performative document (the evidence) is passed to some independent party. The practical advantage of the principle is that the executor can now consider the action as closed, instead of keeping the file open.

It is good to point out that with the list of workflow loop norms above, we have achieved two things. Firstly, we have made explicit the norms underlying the workflow loop paradigm. The list should be seen as a first approximation. In the next section, we will improve the presentation in several respects. The second result is that it has become clear that there are interesting commonalities between the normative principles underlying the workflow loop paradigm and the internal control principles of Chen. However, some internal control principles are missing, such as the ones related to conflicts of interests, while on the other hand, some principles of workflow loop model are stronger, due to its customer-orientation.

6. A formal framework for normative workflow analysis

In this section, we present a formal framework that can be used to analyze the normative content of workflow loop models. However, as we have seen, the customer-orientation in these models leads to a certain bias. Therefore we see it as our goal to develop a model that takes both customer-orientation and agency problems into account.

Agency means that a relation between an agent and some principal exists, where the agent executes work on behalf of a principal, with some autonomy. According to Taylor, an agent has a responsibility for the conduct of an operation [19]. From an economic point of view, an interesting question is how the principal can make sure that the agent acts in the interests of the principal. What Taylor adds to this is that he makes a distinction between principal and beneficiary. For example, the sales agent has a principal (the sales manager), but the beneficiary of his actions is the customer. So the agent has a double responsibility: he acts *on behalf of* someone and he acts *for* (to the benefit of) someone.

According to Taylor, every organizational event is, communicatively speaking, doubly embedded—in the universe of the agent, with his or her preoccupation with instrumentality on the

one hand, and, on the other, in the universe of experience of the beneficiary for whom the patient (the good or service offered) is not just a body to be regulated and operated on, but something affecting his or her well-being.

In practice, we will often encounter the following situation. There is an employee (agent) who performs some service to a customer (beneficiary) on behalf of an organization represented by a manager (principal). The beneficiary can be inside or outside the organization. In some special cases, like a secretary performing jobs for the boss who hired him, the beneficiary and manager/principal can be the same subject.

Example: the pizza delivery case

Suppose that a pizza baker originally bakes and delivers his pizzas himself. In that case, the communication between him and a customer can be easily modelled using a standard workflow loop (within a contract relation, but we will focus here on the baker as performer). Now the baker hires a boy to deliver the pizza to the house of the hungry client for him. Then there exists an agency relation between the baker and the boy: the baker plays the manager/principal role, the boy the employee/agent role. The workflow loop now seems distorted, since the new pizza delivery workflow loop performer is no longer one subject. Say the hungry client calls the baker on the phone. In an actagenic conversation, part of the workflow loop, the baker agrees to bake and deliver a pizza. After calling the boy, the baker orders the boy to bring the pizza to the client. The boy takes the pizza, drives to the house, rings and starts a factagenic conversation in which the hungry client accepts the pizza, perhaps after having signed a note. The boy returns to the baker and reports the successful delivery, possibly with handing over the note as evidence.

By focusing on customer satisfaction, the ActionWorkflow approach tends to ignore the agency relationship between manager and employee and the accompanying communication needs. Of course, customer satisfaction can be more important than production orientation and increase of efficiency. Usually, however, both will be important to some extent. Our conclusion must be that the workflow loop is useful but must be supplemented with other models that focus on the production optimization (the information flow between principal and agent, consisting of setting production norms, task descriptions as well as accounts, and progress reports). Note that in all cases where optimizing a certain value is important, some kind of feedback loop is fundamental—whether this value is customer satisfaction, efficiency, quality or whatever. However, the feedback loop must be specialized for each such objective.

6.1. Outlining the framework

The framework proposed here takes the *service relationship* between two actors as its starting point. In most cases, this is a symmetric relationship where a *service* or object of value is exchanged against some financial compensation. The service has a *beneficiary*, which usually but not necessarily coincides with the *customer* of the service. Service relationships are typically found at the organizational borders, but they may also be explored within organizations. The service relationship is fundamental, but it can be complemented with *delegation* relationships. In this paper,

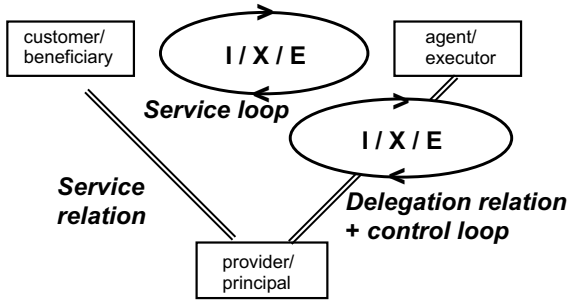

Fig. 3. The extended workflow loop model.

we will focus on delegation on the side of the provider, but delegation at the customer's side is possible as well. Although the task, or part of it, can be delegated to an agent, the delegating actor still keeps a responsibility to the other party, [1] that is, the service relationship itself is not delegated. We call this relationship a *contractual* relationship, whether there is a written contract or not. The two roles in this contractual relationship are the service customer and the service provider.

Closely related to a service relationship is a *service loop*, that is, the service plus the communication loop around the service. This service loop should not be identified with the service relationship. It is a necessary part of it. The service provider can delegate the performance of the service itself, but also the accompanying communication loop. Consider for example the pizza case above: the baker has delegated part of the performance of the service (the delivery of the pizza, but not the baking), and also part of the communication loop (the factagenic conversation, but not the actagenic conversation).

From DEMO, we take the workflow loop *roles* of *initiator* and *executor*, to which we add the *evaluator* role. Instead of using the specific DEMO (order, execution, result) and ActionWorkflow (preparation, negotiation, performance, and acceptance) workflow loop phases, we use the neutral terminology of initiation (I), execution (X), and evaluation (E). These are tasks, but in our formal representation we will only use the corresponding roles of initiator, executor and evaluator, as there is a one-one mapping between the task and the role responsible for this task. The tasks are interconnected by *conversations*: the actagenic conversation and the factagenic conversation. Tasks (task roles) and conversations together are constitutive of the service loop.

From internal control theory, we use the distinction between operational and control tasks. We define the functional role of *principal* to be responsible for the control tasks (initiation and evaluation), and the *agent* for the execution part. The control loop is very similar to the service loop, which makes it possible to view them as two kinds of the so-called *communication loop*.

Our extended workflow loop model is presented graphically in Fig. 3. Note that the agent has two executor roles, but there is a slight difference between the X-role of the agent in the service

[1] Cf. the basic principle of subcontracting laid down in Article 8.107 of the "THE PRINCIPLES OF EUROPEAN CONTRACT LAW—1998" stating that *a party who entrusts performance of the contract to another person remains responsible for performance*.

loop and the X-role of the agent in the control loop. From a control perspective, the agents' performance may consist of executive tasks, but also conversations with the beneficiary (so his overall performance in the service loop is what counts).

6.2. Formalizing the framework

A normative analysis quickly becomes very complex. To deal with this complexity, formal representation and reasoning can be of great help. Having introduced the concepts underlying the extended workflow loop, we are now ready to start our formalization. It consists of two main parts: (1) a *formal ontology* to precisely define the meaning of the extended workflow loop concepts, and (2) a set of *extended workflow loop norms* based on these definitions.

6.2.1. The extended workflow loop ontology

The formal ontology is now represented as follows:

Basic sets include: **Signs**, **Actors**, **Speech Act Types**, **Service types**. These need not be not analyzed further here.

A **speech act** is represented as a tuple <Speech act type, Speaker, Addressee>, where *Speaker* and *Addressee* are Actors. Obviously, this is a very rudimentary representation, but sufficient for the present purpose.

Furthermore, we posit a relationship **count_as** between signs and speech acts, that says that this particular sign *can* be used to perform a certain speech act. Signs are transmitted over some medium that may be digital.

A **conversation** is a tuple <Messages, Participants, Constraints> where *Messages* is a set of Speech Acts and *Participants* is a set of actor roles involved in the conversation. A conversation may or may not have a certain structure, and this structure can be described using the Constraints. In this article, we will not work out that part.

A **service** is a tuple <Service type, Executor, Object, Beneficiary>, where *Service Type* is some predicate designating a service type. The *Object* is the object of value, or can be left empty when the service does not involve an object. *Executor* and *Beneficiary* are actor roles with respect to the service. Services can be decomposed, so we also need a **subtask** relationship Service $\times$ Service, represented with the <-symbol.

A **service relation** is a tuple <Service, Customer, Provider>, where *Customer* and *Provider* are actor roles, and *Service* is some service as defined above.

The **service loop** for a certain service is a tuple <Service, Exec, Init, Eval, Acta, Facta>, where *Exec*, *Init* and *Eval* are actor roles and *Acta* and *Facta* are conversations. The Exec role of the service loop of a service is by definition the same as the Executor of the service.

A **delegation relation** is a tuple <Principal, Agent>, where *Principal* and *Agent* are Actors. For the time being, we omit what exactly has been delegated (that is, task roles and/or conversations). The transitive closure of the delegation relationship is called the *delegation line*.

A **control loop** for a certain delegation relation is a tuple <Delegation, Exec, Init, Eval, Acta, Facta>, where Init and Eval are actor roles and Acta and Facta are conversations. The Exec role of the control loop of a delegation is by definition the same as the Agent of the delegation, whereas the Init and Eval role belong to the Principal, unless he has delegated these roles as well.

Example: the pizza delivery case

In the elementary case that the pizza baker works alone, there is a service relationship between the pizza baker and customer. The service can be expressed as "providing pizza", so the object of value is the pizza, the Executor is the baker and the Beneficiary is the customer. The accompanying service loop consists of an actagenic and a factagenic conversation between the two. There is no delegation.

After the baker has hired a boy to deliver the pizza, the situation has changed. A delegation relation is added between the baker (Principal) and the boy (Agent). The task of the boy is twofold. First, he performs the service of delivering the pizza; this is a subtask of the service of providing pizza, but in this case, the boy is the Executor, and the Service Type is "to deliver". The Beneficiary is still the customer. Secondly, the boy takes over the factagenic conversation from the service loop, that is, he reports the fulfillment of the service on behalf of the baker. The service relationship itself is still between the baker and the customer. The delegation relation gives rise to a control loop between the baker and the boy, consisting of two conversations: the conversation in which the baker instructs the boy to deliver the pizza, and the conversation in which the boy reports the successful delivery (or failure). Note that the subtask of delivering the pizza is modelled as a service, but this service is part of the main service and does not require a separate service loop.

6.2.2. The extended workflow loop norms

Using the service ontology, we are now able to express the workflow norms in a formal way. Each norm expression will be followed by a short comment.

$$\forall s{:}\text{service}\ \exists y{:}\text{service\_relation}\ y.\text{service} = s \wedge s.\text{beneficiary} = y.\text{customer}$$

This norm states that services are always embedded in a service relationship, and that the beneficiary of the service is involved in this relationship (as customer). This seems quite natural, but the norm is not always satisfied. Consider the case that someone books a certain service for somebody else, for example, Mary orders a pizza to be delivered to her children while she is at work. One way of modeling this situation is by saying that Mary does this on behalf of her children (as an agent of the customer). In that case, not Mary, but the children are both beneficiary and customer. This would not violate the norm. Alternatively, if Mary is assumed to be the customer, and the children are the beneficiary, the norm is violated. Although this norm violation may be considered not too bad, the evaluation responsibility is now ambiguous, which may cause breakdowns. Below we will state an additional norm that says that the customer does the evaluation. However, if the customer is not the beneficiary, he may delegate this evaluation task to the beneficiary. Note that even in that case, the transaction is still with the customer.

$$\forall s{:}\text{service}\ \exists w{:}\text{service\_loop}\ (w.\text{service} = s) \vee (w.\text{service} = s' \wedge s < s')$$

This norm states that services are not only embedded in service relationships, but also that there is there needs to be a service loop (initiating, evaluating). The second part of the disjunction is a kind of waiver in the case of complex services: it is sufficient that the composite service has a service loop. It is not excluded that the subservice has one as well.

$$\forall s{:}\text{service\_loop}\ (s.\text{init} \in s.\text{Acta.Participants} \wedge s.\text{eval} \in s.\text{Facta.Participants})$$

$$\forall s{:}\text{service\_loop}\ \forall r{:}\text{service\_relation}\ (s.\text{service} = r.\text{service}) => (s.\text{init} = r.\text{customer} \wedge s.\text{eval} = r.\text{customer})$$

These two norms do have the effect that the customer is involved in both the initiation and the evaluation conversation. They are the direct expression of the "closing the loop" principle. By way of refinement, it could be allowed that the customer delegates the initiation and/or evaluation. For that reason, we have split up the principle in two norms, where the first norm just states that initiator and evaluator are involved in the service loop conversations, which is almost an analytical truth, and the second norm requires that the customer is both initiator and evaluator—which reflects a strong commitment to customer orientation that may be violated for other reasons. Note that we do not require that only one actor is involved in the service performance, the service may be provided by several actors, as in the second pizza example.

$$\forall s{:}\text{service\_loop}\ \exists m{:}\text{message}\ (m \in s.\text{Acta.Messages} \wedge m.\text{type} = \text{request} \wedge m.\text{speaker} = s.\text{init}) \wedge \exists m{:}\text{message}\ (m \in s.\text{Facta.Messages} \wedge m.\text{type} = \text{accept} \wedge m.\text{speaker} = s.\text{eval})$$

The previous norm only said that the customer is involved in the actagenic and factagenic conversations. This can be made more specific by saying that the customer makes an explicit expression of his interest (request) and his satisfaction (accept). However, other configurations could be imagined as well, therefore we state this norm separately, so that it is possible to satisfy one norm and violate the other, in certain circumstances.

$$\forall s{:}\text{service\_loop}\ \exists m{:}\text{message}\ (m \in s.\text{Acta.Messages} \wedge m.\text{type} = \text{commit} \wedge m.\text{speaker} <> s.\text{init}) \wedge \exists m{:}\text{message}\ (m \in s.\text{Facta.Messages} \wedge m.\text{type} = \text{assert} \wedge m.\text{speaker} <> s.\text{eval})$$

This norms makes a precision by requiring that the actagenic conversation contains a commit message, and the factagenic conversation contains an assertive message (reporting the completion of the service). Obviously, these messages cannot be created by the customer. The violation of this norm leads to situations where commitments and accomplishments are left implicit, and therefore are prone to failure.

The next set of norms deals with delegation.

$$\forall d{:}\text{delegation\_relation}\ \exists c{:}\ \text{control\_loop}\ (c.\text{Delegation} = d \wedge c.\text{init} <> d.\text{agent} \wedge c.\text{eval} <> d.\text{agent})$$

The first one says that delegation must be accompanied by a control loop. The initiator and evaluator of this control loop (the principal, unless the tasks are delegated) are separated from the agent (cf. Section 5.1, principle 1).

$$\forall c{:}\text{control\_loop}\ (c.\text{exec} \in c.\text{Acta.Participants} \wedge c.\text{exec} \in c.\text{Facta.Participants} \wedge \exists m{:}\text{message}\ (m \in c.\text{Acta} \wedge m.\text{type} = \text{request} \wedge m.\text{speaker} <> c.\text{exec}) \wedge \exists m{:}\text{message}\ (m \in c.\text{Facta} \wedge m.\text{type} = \text{assert} \wedge m.\text{speaker} = c.\text{exec})$$

From the theory of communicative action it follows, that a typical conversation will contain both a request and a commit, or both an assert and an accept. However, in an intra-organizational context where the initiator, evaluator and executor all have a delegated authority, it often happens that the request is assumed to be obligating even if the addressee does not express his commitment explicitly. Similarly, the acceptance of the finalization of the task can be left implicit. Therefore, the norm only requires that there is an initiating message (request) and a report message (assert).

The norm is intentionally vague about what exactly is requested or reported. This is because it may an individual task (''bake this pizza'') or a generic task (''bake a pizza when a customer orders you so'').

$$\forall s{:}\text{service}\ (s.\text{agent} <> s.\text{service\_relation.provider} \Rightarrow \exists d{:}\ \text{delegation\_line}\ d{<}s.\text{service\_relation.provider}, s.\text{agent}{>})$$

If a service is not performed by the service provider himself, he must have delegated that task (directly or indirectly). The same can be said about the participation in the service loop. If the service provider does not participate himself, he must have delegated his role.

The last norm that we consider concerns the communication itself.

$$\forall c{:}\ \text{speech\_act}\ \exists s{:}\ \text{sign counts\_as}(s,c)$$

All the conversation, service and control structures remain abstract when they are not related somehow to concrete messages. It is said sometimes that communicative acts can be "implicit". A norm-based communication analysis must be precise about what this implicitness means. In our framework, a speech act would be implicit if there is no observable sign by means of which the speech act is performed. The norm above states that we do not allow implicit speech acts.

However, silence can be a communicative act, so a distinction must be made between silence and no sign at all. If the addressee has the opportunity to react, if he has a free slot, so to say, then his leaving this slot empty has communicative significance. What is decisive is that the communicative actors are aware of this significance, and know that the other party is aware (mutual awareness). How to realize that is very much dependent on the medium and context, and although the question is important, it is beyond the scope of this article. In our framework, we treat a significant silence as any other sign. Note that this does not mean that silence is always "as good" as any other sign: for legal reason, for example, a more explicit consent can be necessary.

Another kind of implicit communication occurs when the communication is accompanied by a material transfer. For example, taking a newspaper from the tobacco shop desk and putting down some money is sufficient to express the intention to perform an economic transaction. In that case, it is justified to assume that the material act itself is a sign with communicative value. What is decisive, again, is that the material act is mutually perceived as communicative act.

Apart from cases like these, we think that the notion of "implicit communicative act" should be avoided. If there is no sign of commitment, then there is no communicative act of commitment. This is the only way to keep communicative action theory empirical.

6.3. Applying the extended workflow loop model

The advantage of our formal framework is that complex communication situations as encountered in practice can be modeled. To illustrate this, we continue the pizza case.

The pizza case continued

Another layer of agency complexity is introduced when the baker not only delegates the delivery of the pizza but also the baking itself, let us say, to his daughter. After having received the phone call, he instructs his daughter to bake the pizza. Then, after the baking, the daughter may report back to the pizza baker. That would create a customer/performer workflow between baker and daughter. However, it is more efficient that she directly instructs the boy to deliver the pizza.

What about the conversation between daughter and the boy? Let us assume that this takes the form of a simple dialogue:

Daughter:	Pizza is ready! You can go.
Boy:	OK.

This conversation has in fact two functions. On the one hand, it triggers the boy to deliver the pizza. The boy's OK is then the expression of his commitment to do it. On the other hand, the

daughter reports that the baking is finished. It is not sure yet that the customer will be satisfied, but it is fact that a pizza has been baked, and the boy agrees.

So the boy does some evaluation and we prefer to interpret this evaluation as a control loop evaluation. The baker, being the principal, delegates part of the evaluation of his agent (daughter) to somebody else, namely, the boy. The evaluation has a control perspective rather than a customer satisfaction perspective. Note that the conversation between daughter and boy, if documented on paper or otherwise, can be used later to provide evidence to discharge the daughter. So it is a factagenic conversation, and together with the actagenic conversation between the baker and the daughter, it "closes the loop" of the control process (of the daughter). At the same time, it is an actagenic conversation with the boy which together with the factagenic conversation between the baker and the boy (after he has returned and submits the delivery sheets), closes the boy's control loop (see Fig. 4).

An important remark has to be made about the granularity of the delegated task. In most cases, the baker does not delegate individual tasks ("bake this pizza"), but a *role* (baking pizzas, initiating control loops, etc.). The control loop corresponding to this role delegation consists of a role assignment conversation and a role evaluation conversation. The principal will not evaluate each instantiated performance of that role. There are several ways to work out the role evaluation, but typically it is based on aggregated data or exception reports such as complaints. This is no problem, as long as the principal is able to make a proper evaluation, that is, as long as the aggregated data together with other data give sufficient confidence that the role performance has been in accordance with the stated norms.

It is worth noting that that there is no unique realization. The baker could have realized the relationship with his daughter as a service loop. In that case, he becomes a beneficiary and should express his satisfaction after the baking. The present communication lines would not be sufficient then: either the shortcut between daughter and boy should be removed or the boy should have the delegated responsibility to evaluate the baking of the pizza. Which realization is better cannot be

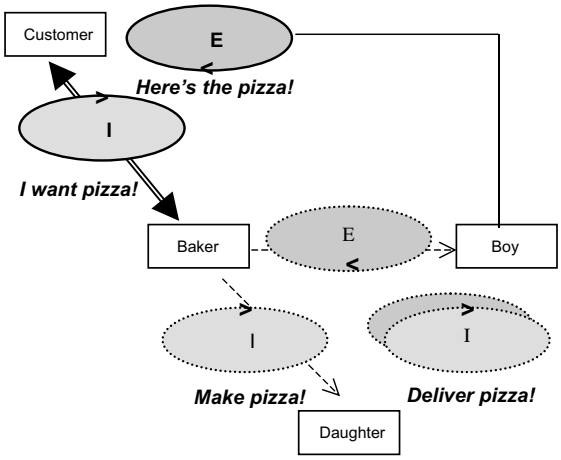

Fig. 4. Complex agency in the pizza-delivery case. The delegation lines are indicated as dashed lines. The service relation is indicated as a two-arrow double line. The I or E inside a loop indicates the type of conversation (initialization, evaluation)—so these are in fact loop halves. Control loops are rendered with dotted lines.

decided on the basis of the present norms. Additional norms and preferences are necessary that take into account the efficiency of the communication and other factors such as employee satisfaction.

6.4. Production workflow and workflow management systems

The example illustrates how a production workflow chain [11] can be created on the basis of delegation. The chain is still short with only a boy and a daughter, but it is not hard to see that it can be made longer, for example, when the baker also delegates the initiating conversation of the service loop (the order acceptance). The baker is still the service provider, and he remains responsible to the customer when something goes wrong, because the service relationship itself cannot be delegated. The advantage of a chain is that the agents can focus on one task (functional specialization), but from a service perspective, a disadvantage is that they do not interact directly with the beneficiary of their work. Note that the workflow chain as a whole is embedded in a service loop (the customer is still at the front and at the end), and belongs to one delegation domain (all tasks are delegated, and ultimately, there is one responsible principal). What does this mean for cross-organizational workflow? This cannot be modeled as one workflow chain, as there are different organizations with their own delegation domains. It should be modeled as two workflow chains that are connected through a service relation (cf. [20]).

If a Workflow Management System is used, it can be given a passive and an active function. A passive system acts as a communication channel between the agents in the chain. An active system takes a (delegated) role in the chain itself. In that case, the actagenic and factagenic conversations are conversations between the human agents and the WFMS. The WFMS has the (delegated) responsibility to initiate the next agent; if this is not successful, the system has the responsibility to escalate the problem to its principal.

7. Conclusion

In this paper, an analysis has been made of the norms underlying LAP workflow loop models. Norms implicit in those models have been made explicit and contrasted with explicit norms from internal control used in accountancy. A framework for the (meta)analysis of workflow loop models was created. The framework consists of an extended workflow loop model and a set of reconstructed LAP workflow loop norms.

We claim that an extended workflow model that considers both customer relations and agency relations is needed to chart complex organizational communication situations. LAP, internal control and possibly other norms can be applied to assess the situation. Quality management activities as described in [4] may be used to improve upon the current communication situation. Thus, this framework may prove to be a helpful tool in optimizing organizational communication patterns.

There are many things that are still to be done. At some point, we may need a practical way of modelling (diagram technique). The DEMO model abstracts from the delegation, and Action-Workflow diagrams do not distinguish between delegation and workflow loops. Even more important is the way of working, the way that the model is built up. Theoretically, there should be a

recursive method based on the principles of *task decomposition*, *delegation* (introducing new agency relations) and *outsourcing* (introducing new customer relations). For a certain organization or department, one can start with identifying the contract relations with all stakeholders in which the organization is the performing actor. Then the model can be worked out by applying delegation and outsourcing transformations. Each transformation should preserve the validity of the communicative norms, or at least a warning should be given when some norm is violated for some reason. In this way, only meaningful and valid communication structures can be derived. The model can also be used for diagnosis [14] which works in a bottom-up fashion. The goal of diagnosis is to model the current situation and to analyze actual or potential flaws by linking them to communicative norm violations. The diagnosis should result in recommendations for improvement. The reengineering process description should indicate how the new situation can be reached from the current situation by retracting existing delegation and service relations and introducing new ones.

So in contrast to other LAP approaches and to most of the current business modeling approaches (e.g. BPML, [2]), our framework not only contains explicit communication norms for workflow processes, but it also gives a starting-point for modeling (organizational) *change* processes. This is an interesting area for further research.

Acknowledgements

The authors want to thank the participants of LAP-2001 and LAP-2002 for useful comments on the first version of this paper, in particular Goran Goldkuhl, Michael Lind and Jan Dietz.

References

[1] R. Bons, Trustworthy trade procedures, Ph.D. thesis, Erasmus University, Rotterdam, 1997.
[2] See http://www.bpmi.org.
[3] K.T. Chen, Schematic Evaluation of internal accounting control systems, Ph.D. thesis, Univ of Texas at Austin, 1992.
[4] A. de Moor, H. Weigand, Towards a semiotic communications quality model, in: K. Liu et al. (Eds.), Organizational Semiotics—Evolving a Science of Information Systems, Kluwer, 2002, pp. 275–286.
[5] P. Denning, R. Medina-Mora, Completing the loops, Interfaces 25 (3) (1995) 42–57.
[6] S.K. Dewitz, Contracting on a performative network: using information technology as a legal intermediary, Ph.D. thesis, Univ of Texas at Austin, 1992.
[7] J. Dietz, The atoms, molecules and fibers of organizations, this issue.
[8] J. Dietz, V. van Reijswoud, DEMO Modelling Handbook, version 2, 1998.
[9] H.P. Grice, Logic and conversation, in: P. Cole, J.L. Morgan (Eds.), Syntax and Semantics 3: Speech Acts, Academic Press, New York, pp. 41–58.
[10] J. Habermas, The Theory of Communicative Action, I. Beacon Press, 1984.
[11] F. Leymann, D. Roller, Production Workflow: Concepts and Techniques, Prentice-Hall, New Jersey, 2000.
[12] R. Medina-Mora, T. Winograd, R. Flores, F. Flores, The ActionWorkflow Approach to Workflow Management Technology, in: The Information Society, vol. 9, 1993, pp. 391–404.
[13] W. Orlikowski, J. Yates, Genre Systems: Structuring Interaction through Communicative Norms, Working Paper, Centre for Coordination Studies, MIT.
[14] F. vd Poll, H. Weigand, A. De Moor, Communication Diagnosis of a Financial Service Process, in: Proceedings of the 7th International Workshop on the Language/Action Perspective, Delft University, 2002.

[15] R.W. Starreveld, Bestuurlijke Informatieverzorging. Deel 1, Algemene Grondslagen. Samson, Alphen a/d Rijn, 1997 (Foundations of Management Information Systems, in Dutch).
[16] J. Searle, The construction of social reality, Penguin Philosophy (1995).
[17] R. Stamper, New Directions for Systems Analysis and Design, in: J. Filipe (Ed.), Enterprise Information Systems, Kluwer Academic Publishers, London, 2000, pp. 14–39.
[18] L. Suchman, Do Categories have Politics? The Language/Action Perspective Reconsidered, in: Computer Supported Cooperative Work (CSCW), vol. 2, 1994, pp. 177–190.
[19] J.R. Taylor, Rethinking the Theory of Organizational Communication, Ablex, New Jersey, 1993.
[20] H. Weigand, W.J. van de Heuvel, Cross-organizational workflow integration using contracts, Decision Support Systems 33 (2002) 247–265.
[21] T. Winograd, F. Flores, Understanding Computers and Cognition: A New Foundation for Design, Ablex Publishing, 1986.

Hans Weigand is associate professor at the Department of Information Systems and Management of Tilburg University (Infolab). He holds a Ph.D. in Computer Science from the Vrije Universiteit in Amsterdam (1989). His research interests include communication modelling, language/action perspective, negotiation support, and agents. He has supervised several Ph.D. projects in these areas and has been involved in a number of industrial projects such as the E.C. supported project MeMo (Mediating and Monitoring Electronic Commerce). Since 2002, he is a member of the editorial board of Data and Knowledge Engineering.

Aldo de Moor is an assistant professor at the Department of Information Systems and Management of Tilburg University. He holds a Ph.D. in Information Management from this university and has been a visiting researcher at the University of Guelph, Canada, and the University of Technology, Sydney. His research interests include the evolution of virtual communities, communicative action theories, user participation in systems development, normative system specification methods, and the impact of ICT on society. Projects in which he has participated include the development of an electronic journal, a business negotiation support system, and various virtual community development projects.