# Modeling Cooperative Work for Workflow Management

Jon Atle Gulla and Odd Ivar Lindland

Faculty of Electrical Engineering and Computer Science
The Norwegian Institute of Technology
N-7034 TRONDHEIM, NORWAY

**Abstract.** Characteristic to workflow management is modeling of workflow of manual coordination activities and automated production activities. Conceptual models are used to analyze and describe workflow, though most of these models are not very suitable for representing and relating both coordination activities and production activities. In the Input-Process-Output paradigm, workflow is modeled in terms of processes and data flow, whereas the Customer-Supplier paradigm defines conversation patterns between the actors. The rather intimate relationship between actor interactions and processual structures is recorded in neither of them. In this paper, we suggest to extend the IPO paradigm with concepts for coordination activities. We introduce actors and services as a separate model, and show how two-way flows, ports and rules help us model cooperative and manual aspects in data flow diagrams.

## 1 Introduction

Workflows describe the coordination and performance of work undertaken in businesses. Workflow management provides mechanisms for planning and controlling workflow, and has received a lot of attention the last few years. Workflow management should allow the analysis of current workflow in order to detect potential bottlenecks and the design of new workflow patterns so that shortcomings can be eliminated. Typically, this includes reorganizing, automating, and/or supporting activities in the current workflow. Business process re-engineering is an approach to organizational and technological change that has drawn on the achievements of this field [5].

In order to investigate and manipulate workflow in a business, a model of the current workflow is used for documenting, understanding, and communicating the coordinating business activities. Furthermore, the model is a natural basis for experimenting with and evaluating the consequence of introducing changes in the model prior to realization.

When doing so, it is important to recognize that workflow is linked to processes that business are performing. Two types of processes exist: *production processes* and *coordination processes* [10]. In production processes the business produces a result (information or material) to a customer. Since the result is expected to be of a certain value for the customer, the processes involves *value-adding* activities. The work in production processes can be manually and au-

tomatically performed. Gerrits denotes this process type business processes for either material production or information production [3]. Coordination processes controls the performance of the value-adding activities without directly adding value to the product. Typically, this process type the administrative layer of the production processes and involves manual coordination of people in order to carry out certain actions. Coordination processes are denoted information processes by Gerrits.

These definitions differ from the process definition of Medina Mora et al. [8]. They distinguish between material, information, and business processes. Material processes describe activities which involve physical and mechanical work to move, transform, manipulate, consume or combine materials. Information processes reflect electronic transfer, manipulation, etc. of information. Business processes coordinate actions carried out by people. In Flores and Winograds definition information and material processes are low-level and well-defined, whereas business processes are high-level, ad-hoc and involve a certain degree of uncertainty. They manifest themselves in communication between people and may trigger material and information processes.

Although different process terminology exists in literature, it is important to recognize that workflow involves and affects both production activities and coordination activities. In order to model the workflow in organizations two dominating "paradigms" exists [2]: The *Input-Process-Output (IPO)* paradigm and the *Customer-Supplier (CS)* paradigm. The paradigms are complementing in that the IPO paradigm is particularly suited for modeling the chain of production activities, whereas the CS paradigm is appropriate for modeling the coordinating structure within a business and between the business and its customers.

This paper further elaborates on integrating the IPO and CS paradigms. Section 2 briefly describes the characteristics of these paradigms. In Section 3 the shortcomings of the IPO paradigm with respect to workflow modeling is discussed. As such the section provides the rationale for the new concepts that are introduced in Section 4. In Section 5 a concluding discussion is offered. Directions for further work are also indicated.

## 2 The IPO and CS paradigms

The Input-Process-Output (IPO) Paradigm is used to model the workflow production processes. The structure of the IPO paradigm is shown in Figure 1(a). The workflow is regarded as a chain of activities that takes information and material as input and produces information or material as output. Complex activities can be decomposed into simpler and more structured activities. The management of the workflow chain is realized by control to and feedback from the process. In an IPO model it is expected that inputs, transformations, and outputs are well-defined. The IPO paradigm is suitable for modeling workflows in repeatable procedure-based processes. The commitments among the workers and in particular between the business and its customers in order to carry out the work is is not explicitly described in IPO models.

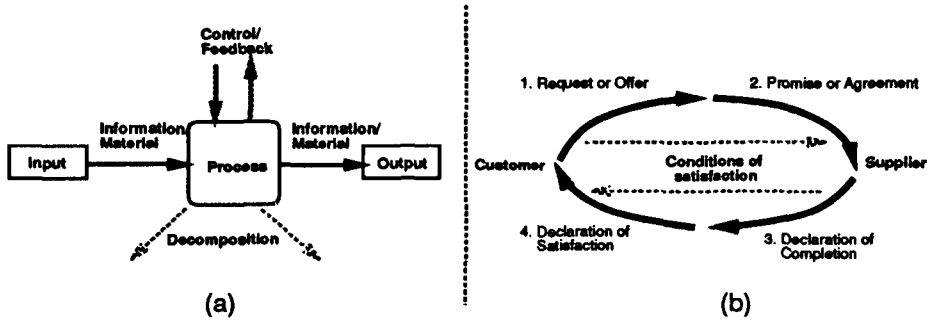


**Fig. 1.** (a) The IPO paradigm and (b) the CS paradigm to workflow modeling.

Whereas the IPO paradigm views the workflow as a chain of production activities the CS paradigm focuses on coordination among people. The paradigm has been established by Flores and Winograd [14] and is founded on Searle's *Speech Act* [15]. The basic structure is shown in Figure 1(b). Two major roles, *customer* and *supplier*, are modeled. Workflow is defined as coordination between these roles, and is represented by a conversation pattern with four phases [14]. In the first phase the customer makes a request for work, or the supplier makes an offer to the customer. In the second phase the customer and supplier aims at reaching a mutual agreement about what is to be accomplished. This is reflected in the contract *Conditions of Satisfaction*. In the third phase, after the performer has performed what has been agreed upon and completed his work, completion is *declared* for the customer. In the fourth and final phase the customer assesses the work according to the conditions of satisfaction and declares satisfaction or dissatisfaction The ultimate goal of this paradigm is *customer satisfaction*. This implies that workflow loops have to be *closed* and that the customer must acknowledge that the work has been satisfactory completed. As such, the paradigm is appropriate for explicitly modeling the chain of commitments that exists between people in order to satisfy the customer. The specific activities carried out in order meet the contract is not modeled. Also, the information and material needed in each production activity is not described.

In order to manage the workflow properly, we would like to a comprehensive overview of the workflow both in the coordination activities and in the production activities. Furthermore, we would to model the interdependencies between the coordination and the production processes. When doing so, the shortcomings in the coordination process might be explained by an insufficient production process. A comprehensive workflow modeling technique encompassing these interdependencies can be obtained in several ways. In this paper, we show how the IPO paradigm can be extended to capture customer-supplier relationships. We add new concepts for modeling actors and their cooperation within the IPO

paradigm. We are then able to model manual parts of coordination processes as well as automated parts of production processes.

## 3 Shortcomings of the IPO Paradigm

Although there are good reasons for adopting the IPO paradigm in workflow management, there are aspects of IPO that make it insufficient for modeling coordination activities. We address here three issues that concerns the cooperative aspects of workflow management: (1) the involvement of actors, (2) the exchange of information between actors, and (3) the structuring of cooperative work processes.

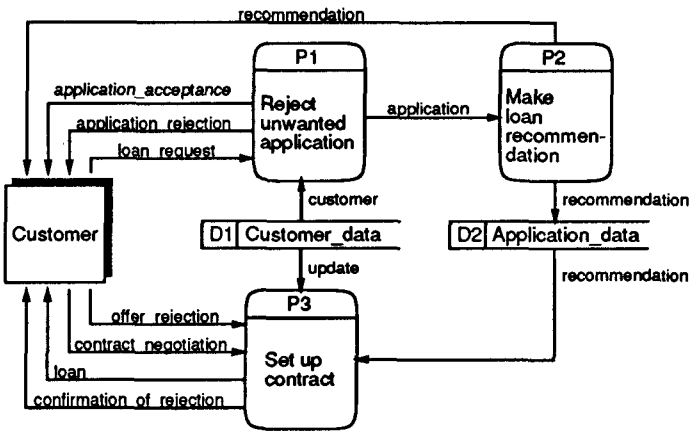


**Fig. 2.** Data flow diagram for loan processing.

### 3.1 Involvement of Actors

Consider the DFD model shown in Figure 2, which describes a rather simple processing of loan applications. A clerk checks that the customer is eligible for a loan, and either rejects the application or sends it to a loan consultant for further processing (process P1). The consultant suggests an appropriate amount to the customer, and if the customer accepts the offer, a loan contract is set up in cooperation between these two. In case of a rejection of the offered loan, the consultant just returns a confirmation that ends the whole process. However, neither the clerk nor the consultant is explicitly represented in the model, and we just have to know who is responsible for doing the tasks. For a simple model like this, that might not be a problem, but in a more general setting it

can be difficult to decide who the actors are. And looking at the characteristics of workflow management systems, we wee that this lack of information is rather problematic. Both manual and automated tasks are relevant in workflow management. Automated parts can — at least in theory — be specified all the way down to algorithms, and these can then be analyzed with respect to correctness, efficiency, and effectiveness. What manual parts are concerned, an analysis must also take into account who is designated to do the various tasks. Bottlenecks due to overloaded employees, for example, cannot be detected if there is no information about actors in the model.

### 3.2 Exchange of Information

Exchange of information is common both in production activities and in coordination activities. Consider for example process P3 in Figure 2, where the customer and the loan consultant are to work out a loan contract. Both actors contribute to the final contract, and the whole work is organized as a negotiation process where new ideas and suggestions are exchanged. However, the flow contract_negotiation, which should represent the costumer's contribution, only model the sending of one single piece of information and is unable to capture the responsive nature of negotiations like that.

What is needed here, is the ability to specify that the input flow's values at time t, for any t within the receiving process's active period, depend on the output flow's values at $t' < t$, and vice versa. As long as the flows are modeled as independent, this relationship cannot be made explicit in the model.

### 3.3 Structuring of Activities

Traditionally, process logic has been specified as algorithms [4], state transition diagrams [11, 13], Petri nets [7], and decision tables/trees [12]. These may work fine for automated processes, but in manual ones one cannot specify the process as a complete calculation — if that was possible, it would not have to be a manual process. Instead, notations for structuring the work itself is needed, in particular for coordinating the efforts of the actors involved. Looking back at process P3 in Figure 2, we see that there is little information represented that can be used to guide the manual execution of the process. The customer should either reject the offer or start negotiating the contract, but this is not clear from the model. Similarly, the result is either a loan or a confirmation that the offer is rejected, but the model just states that there are a number of possible input flows and a number of possible output flows. It is not feasible to specify that a flow depends on or excludes another one, or that the work has to be carried out in accordance to certain rules or constraints.

## 4 Concepts for Modeling Cooperative Work

Our objective is to extend the IPO paradigm with some basic concepts for modeling coordination activities and cooperative work. The concepts can be worked

out at different levels of formalization for integration with other formalized extensions of DFD, but we will here assume the traditional DFD notation (as described in [9]) as a basis.

A coordination activity involves actors and describe how these interact and cooperate with each other. They can be described on the basis of conversation theories like Speech Act though there is a great deal of variation among the activities. In some coordination activities, the actors are negotiating as independent customers and suppliers; in others, the cooperation is manifested more as a command or dependency relationship. In [1], the relationships between actors are classified as power relationships, peer relationships, and service relationships. Each is characterized by its own pattern of cooperation, and there is no general conversation theory that can easily account for them all. Still, there are some fundamental principles that underpin all these relationships, and these have to a large extent been neglected in the IPO paradigm used today. We define an additional model, the *Actor Service* model, that provides a cooperative point of view and is linked to the DFD model as part of the modeling process.

The DFD model itself is also supplemented with some extra concepts. These enable modeling of cooperative work processes and routing decisions.

### 4.1 Actor Service Model

The *Actor Service* model is a supplement to the DFD model, intended for the modeling of actor relationships in coordination activities. It describes how actors interact and cooperates in accordance to intentions and goals, but there is no control flow or data flow represented. The main concepts are — not surprisingly — *actor* and *service*, and these are briefly explained below.

- An *actor* $a$ is any entity that has the ability to provide a service to another actor. A hardware or software component can be classified as actors, though in this exposition the focus is rather on human actors. These can be external to the workflow management system, like external entities in DFD, but can also be employees working as part of it. Furthermore, an actor can be formed by grouping other actors, in which case it is more like a role played by these actors. Formally, the role actors are defined as sets of other actors $a_i$, $a = \{a_i\}$. An actor is assumed to have some kind of intention $i(a, s)$ that motivates her action; that is, $i(a, s)$ is $a$'s reason for taking the responsibility for bringing about $s$.
- A *service* is defined in terms of a *task*, a *result*, or both. Its boundaries relate it to actors, so that there are actors responsible for performing it as well as actors requesting or benefiting from its result. Formally, the service itself is represented as the tuple $s = \langle t, r \rangle$, where $t$ is a task and $r$ is a result specification. A service with its associated actors is called a *service constellation*. This constellation may also contain the intentions of the performing actors and the means for realizing the service, and is specified as a 4-tuple $c = \langle p, b, s, m \rangle$. Variable $p$ is the set of performing actors $p = \{\langle i(a_i, s), a_i \rangle\}$, where $a_i$ is an actor, $s$ is the service, and $i(a_i, s)$ is $a_i$'s intention of doing

$s$. Variable $b$ is the set of actors $\{a_j\}$ benefiting from or requesting service $s$, and $m$ is a set of subordinate service constellations $\{s_i\}$, DFD processes $\{p_i\}$, or DFD flows $\{f_i\}$. At the level of elementary services, thus, a service is realized as a number of elements in the DFD model.

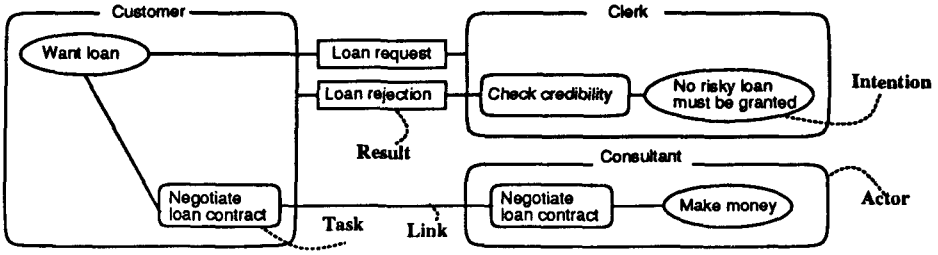


**Fig. 3.** Actors and services.

Graphically, we use the symbols shown in Figure 3. This figure is the Actor Service model for our loan processing system from Figure 2. The customer is responsible for sending the loan request to the clerk and is also negotiating with the consultant about the contract. The clerk rejects loan applications when the applicants do not have the necessary credibility.

At last, the Actor Service model can be worked out at different levels of detail. In our model in Figure 3, for example, we could decompose the contract negotiation constellation, introducing the consultant's offer as well as customer's rejection and acceptance as new services. Alternatively, we could specify the realization of the constellation as the set of DFD processes $\{P2, P3\}$. The constellation containing the task Check credibility is given a reference to DFD process P1 in its specification.

### 4.2 A Two-Way Flow

An ordinary one-way flow transports information from one location to another. A *two-way flow* denotes the exchange of information between two parties, such that information floating in one direction is followed by a response in the other direction. Conceptually, this corresponds to a conversation pattern where two or more actors work together and contribute to a common task.

Following the formalization of DFD indicated in [9], we assume flows to carry *items*, where each item is defined as a set of attributes. A flow transports an item from one location to another in zero time, without changing the values of the item's attributes[1]. It is defined as the tuple $f = \langle \pi_1, \pi_2 \rangle$, where item place

[1] Actually, this type of flow is referred to as an *ideal flow* in [9].

$\pi_i$ specifies a location of a chosen type of item, which means that an item at a location represented by $\pi_1$ is consumed and an item with exactly the same attributes is immediately produced at a location represented by $\pi_2$. Item places $\pi_1$ and $\pi_2$ represent locations for the same type of item, and the locations must be different from each other. Let us also define $\langle f, t \rangle$ to mean that a particular item is transported along flow $f$ at time $t$. Now, a two-way flow can be defined as the tuple $\phi = \langle f_1, f_2 \rangle$, where $f_1 = \langle \pi_1, \pi_2 \rangle$ and $f_2 = \langle \pi'_2, \pi'_1 \rangle$, where $\pi_1$ and $\pi'_1$ represent one location and $\pi_2$ and $\pi'_2$ represent another one.

In the loan processing example, two actors are negotiating a loan contract in process P3. The loan consultant is part of the banking system and is associated with P3 through the Actor Service model, but the clerk is also contributing to the manual work of the process. Since the clerk is an external entity, she cannot be held responsible for P3's execution, and her contribution must then be modeled by means of a two-way flow to P3, as shown by the flow contract_negotiation in Figure 5. A two-way flow is drawn as a thick line and counts as an input flow what ports are concerned (see next section).

### 4.3 Ports and Rule Modeling

An automated completely specified process can be described by its input flows, its output flows, and the functions determining the attribute values of the output flows' items on the basis of the input flows' items. Being a little more precise, we say that a process can consume items from a set of item places, $\Pi_I$, and produces items to another set of item places, $\Pi_O$, where all $\pi \in \Pi_I \cup \Pi_O$ represent the same location.

For manual processes, and incompletely automated ones, the functions may not be available, and one must use other means for describing their contents and internal structures. Rather than formulating functional relationships between inputs and outputs, we then specify constraints on the flows consumed and produced during process executions. These constraints serve as a structure for the work to be carried out and provide guidelines for the actors in cooperative work processes. We introduce *input port expressions, output port expressions,* and *constraint rules* for the specification of the constraints:

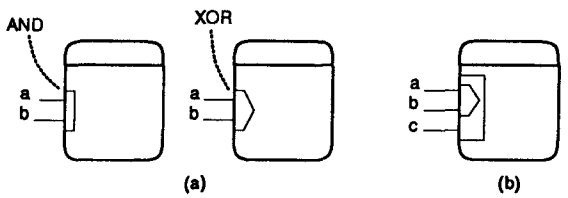


**Fig. 4.** (a) AND port and XOR port. (b) Nested ports.

- An *input port expression* lists all possible combinations of input flow items that can be consumed during one process execution. Formally, it is specified as the tuple $\langle \Pi_I, \Pi_{IC} \rangle$, where set $\Pi_I$ is as above and $\Pi_{IC}$ is a subset of $\Pi_I$'s powerset, $\Pi_{IC} \subseteq \mathcal{P}(\Pi_I)$[2]. $\Pi_{IC}$ identifies combinations of item places, from which items are consumed during a single process execution.
  In the DFD model, we add input port expressions using XOR ports and AND ports on input flows to processes. The AND port in Figure 4(a), which includes two flows with destination places $\pi_a$ and $\pi_b$, means that the process must consume both item $a$ from $\pi_a$ and $b$ from $\pi_b$. Correspondingly, the XOR port means that the process must consume either $a$ from $\pi_a$ or $b$ from $\pi_b$, but not both. Of course, ports may be nested, so that the port expression in Figure 4(b) specifies that the process consumes either $\{a, b\}$ or $\{b, c\}$ during each execution.
- An *output port expression* lists all possible combinations of output flow items that can be produced during one process execution. Formally, it is specified as the tuple $\langle \Pi_O, \Pi_{OC} \rangle$, where set $\Pi_O$ is as above and $\Pi_{OC}$ is a subset of $\Pi_O$'s powerset, $\Pi_{OC} \subseteq \mathcal{P}(\Pi_O)$. $\Pi_{OC}$ identifies combinations of item places, to which items are produced during a single process execution.
  Similarly to input port expressions, XOR ports and AND ports are used in the DFD model for the specification of the constraints.
- A *constraint rule* specifies the existence relationship between consumed items and produced items; that is, given that a process consumes a specific combination of input flow items, the rule determines which output flow items are produced. The values of the items' attributes are not involved in these rules, just their existence. Formally, we can describe the relationship $R$ as a subset of the Cartesian product of the sets $\Pi_{IC}$ and $\Pi_{OC}$, $R \subseteq \Pi_{IC} \times \Pi_{OC}$, where $\Pi_{IC}$ and $\Pi_{OC}$ are as described above.
  In the model, the constraint is specified as a number of constraint rules associated with a process. Each rule is of the form

  IF < *input flows* >
  THEN < *output flows* >,

  and specifies how the production of items to specific output flows depends on the consumption of items from specific input flows. Note that this notation is consistent with the rule notation in TEMPORA [6], so that our additions can easily be integrated with rule-based approaches to information systems development.

We can now describe the internal structure of manual processes more precisely. In process P3 in Figure 2, the process must either receive offer_rejection, or contract_negotiation and recommendation, and this is easily shown as an XOR port with an subordinate AND port in Figure 5. The output port expression says that an update is always produced, though only one of loan and confirmation_of_rejection can be produced during a particular process execution. The

[2] $\mathcal{P}(\Pi_I)$ is the collection of all subsets of $\Pi_I$.

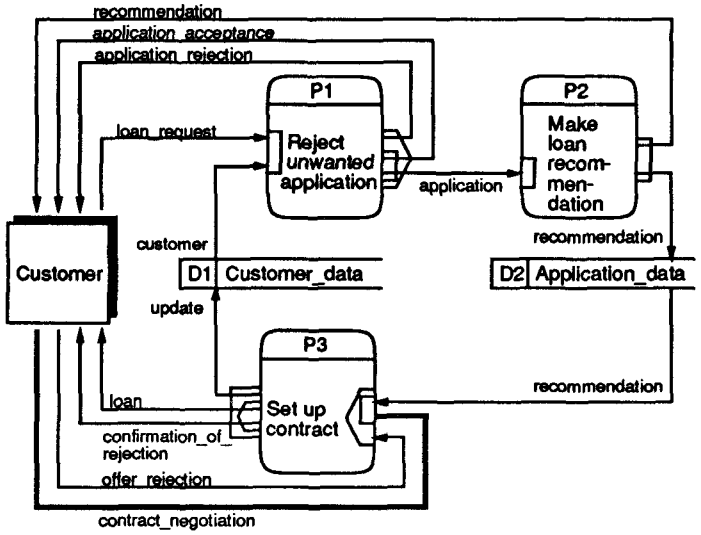


**Fig. 5.** Extended data flow diagram.

relationship between input flows and output flows is specified by means of the two constraint rules in Table 1. An offer_rejection is followed by the sending of confirmation_of_rejection, whereas an contract_negotiation leads to the generation of a loan. With these constraints specified, one can define tools and methods that structure the cooperative work of P3.

IF contract_negotiation AND recommendation
THEN update AND loan

IF offer_rejection
THEN update AND confirmation_of_rejection

**Table 1.** Rules associating input flows and output flows for process P3.

Additionally, we can restrict the values of the items carried by the flows. This is done by extending the constraint rule notation with a special WHERE field, which specifies the arithmetic or logical relationships between items consumed and produced during a process execution. For example, the flows recommendation, update and loan in Figure 5 all carry an item applicant, and the value of this item must be the same for all flows. Similarly, the amount item carried by loan

and update must have the same value. Assuming that the value of this amount has to be less or equal to recommendation's amount, we can now replace the first rule in Table 1 with the following extended rule:

```
IF      contract_negotiation AND recommendation
THEN    update AND loan
WHERE recommendation.applicant = update.applicant = loan.applicant AND
        recommendation.amount ≥ loan.amount AND
        loan.amount = update.amount
```

Finally, the rule can contain an initial WHEN field that specifies what flow — or combination of flows — triggers the execution of the rule. Looking at the rule above, we notice that the rule is triggered by the arrival of flow contract_negotiation: WHEN contract_negotiation. In the diagram, this information can be made explicit by marking that particular flow with a T.

## 5 Concluding Discussion

Our work relies on the distinction between workflows manifested in coordination and production processes. As indicated in Figure 5, the IPO paradigm is appropriate for modeling production processes, but lacks concepts for modeling coordination processes. For the CS paradigm the situation is the other way round. It is suitable for modeling coordination processes, but lacks direct links to production activities. Although both paradigms are flexible and can be extended with new concepts, it seems problematic to combine them as they are today. We find a focus on *control flow* in both paradigms — though differently modeled — and if these models are just combined, this redundancy of information would clutter the model and complicate consistency checks. Other problems are related to the cyclic and linear nature of these paradigms, and their different interpretation of flow content.

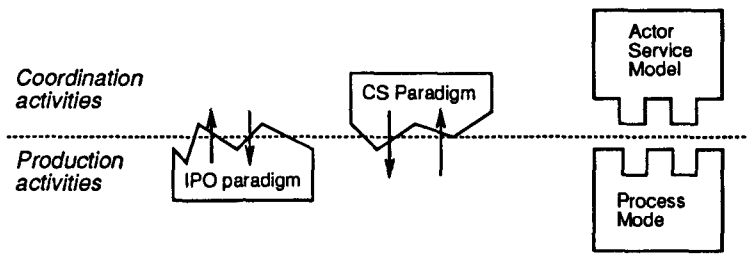


**Fig. 6.** Towards an integrated workflow modeling approach.

Our main contribution has been to provide a comprehensive framework for workflow modeling, where coordination activities and the production activities

are integrated. The IPO paradigm forms the basis, but we have now added concepts for the modeling of coordination and cooperative work. First, we introduced the Actor Service language which enable us to explicitly model the services provided by the actors, their intentions, and the relationships among the actors. Secondly, a two-way flow makes it possible to model exchange of information between two parties. Finally, ports and rules give us the opportunity to model the routing of work and the constraints imposed on it. In that sense, we have extended the modeling language to the realm of coordination activities and also established well-founded links between these too kinds of activities.

The Actor Service model records coordination patterns among people involved or affected by the organization. It captures many facets of the CS paradigm, though the following properties of our model should also be noted:

- A model like this does not represent a fixed conversation pattern, since such a pattern would include both a number of fixed types of interactions and a control flow specifying their sequencing. As the model is, we have the freedom to specify simplified interaction patterns when that is appropriate, whereas control aspects can be completely left out of the model.
- Contrary to the CS paradigm, we can here associate tasks and results with intentions of the actors, which in turn provides a mechanism for recording the rationale of DFD elements. Actors may cooperate in doing a task, but the model does not require them to have the same intentions.

In our current model, intentions are modeled in a rather simplistic way, but the basis idea now is to extend the notion of *intention* and relate it to other parts of the enterprise model. Intentions provide background information that can help us analyze communication breakdowns and predict both desirable and undesirable situations in organizations. More specifically, we are working on the relationship between intentions and constraint rules, letting the intentions influence on the organization of work among people.

The data flow model itself now includes concepts that enable a more precise specification of manual working procedures. Actor responsibilities and collaborative work are described by means of the new concepts and the links to the Actor Service model. Since these extensions are not in conflict with existing DFD concepts, the model is now well suited for models of both manual and automatic systems.

So far, our work is only theoretical. Further work should emphasize on applying the modeling language on case studies in order to evaluate its strengths and weaknesses. Tool support for model construction and model analysis should be provided. A natural basis here will be the experimental ICASE-environment PPP [4].

## References

1. A. J. C. Blyth, J. Chudge, J. E. Dobson, and M. R. Strens. ORDIT: A new methodology to assist in the process of eliciting and modelling organisational re-

quirements. In *Proceedings on the Conference on Organisational Computing Systems*, San Jose, November 1993.

2. P. J. Denning. Work Is a Closed-Loop Process. *American Scientist*, 80:314–317, July-August 1992.
3. H. Gerrits. Business Process Redesign and Information Systems Design: A Happy Couple? In N. Prakash, C. Rolland, and B. Pernici, editors, *IFIP WG 8.1 Working Conference on Information System Development Process*, pages 325–336, Como, September 1993. Elsevier Science Publishers B. V. (North-Holland).
4. J. A. Gulla, O. I. Lindland, and G. Willumsen. PPP: An Integrated CASE Environment. In R. Andersen, J. A. Bubenko jr., and A. Sølvberg, editors, *Proceedings of the Third International Conference on Advanced Information Systems Engineering (CAiSE'91)*, pages 194–221, Trondheim, May 1991. Springer-Verlag.
5. M. Hammer. Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, pages 104–112, July-August 1990.
6. J. Krogstie, P. McBrien, R. Owens, and A. H. Seltveit. Information Systems Development Using a Combination of Process and Rule Based Approaches. In R. Andersen, J. A. Bubenko jr., and A. Sølvberg, editors, *Proceedings of the Third International Conference on Advanced Information Systems Engineering (CAiSE'91)*, pages 319–335, Trondheim, May 1991. Springer-Verlag.
7. D. C. H. Kung. The Behavior Network Model for Conceptual Information Modeling. *Information Systems*, 18(1):1–21, 1993.
8. R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The Action Workflow Approach to Workflow Management Technology. In *Proceedings of CSCW'92*, 1992.
9. A. L. Opdahl and G. Sindre. Concepts for Real-World Modelling. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proceedings of the Fifth International Conference on Advanced Information Systems Engineering (CAiSE'93)*, pages 309–327, Paris, June 1993. Springer-Verlag.
10. M. E. Porter and V. E. Millar. How Information Gives You Competitive Advantage. In *Revolution in Real Time: Managing Information Technology in the 1990s*, chapter II-1, pages 59–82. Harvard Business Review, Boston, 1984.
11. G. Richter and B. Maffeo. Toward a Rigorous Interpretation of ESML — Extended Systems Modeling Language. *IEEE Transactions on Software Engineering*, 19(2):165–180, February 1993.
12. A. Sølvberg and D. C. H. Kung. *Information Systems Engineering.* Springer-Verlag, 1993.
13. P. T. Ward. The Transformation Schema: An Extension of the Data Flow Diagram to Represent Control and Timing. *IEEE Transactions on Software Engineering*, 12(2):198–210, February 1986.
14. T. Winograd. A Language/Action Perspective on the Design of Cooperative Work. *Human-Computer Interaction*, 3(1):3–30, 1987-88.
15. T. Winograd and F. Flores. *Understanding Computers and Cognition: A New Foundation for Design.* Ablex Publishing Corporation, New Jersey, 1986.