# Modeling Inter-Organizational Workflows

Aija Sladek Antoni Wolski

VTT Information Technology
P.O. Box 1201, FIN-02044 VTT, Finland

e-mail: {Aija.Sladek, Antoni.Wolski}@vtt.fi

## Abstract

*Inter-organizational cooperation has many facets. One of them is information system (IS) cooperation which is a process of performing pre-defined cooperative activities using computer technology. IS cooperation can be modeled and implemented in the form of inter-organizational workflows involving (typically) pre-existing information systems of the participating organizations.*

*When participating information systems belong to autonomous organizations, they can at any time decide to exercise their autonomy by, for example, refusing to execute their part of the cooperative work. Modeling parallelism, asynchrony and exception handling policies emerge as major challenges. Traditional data and control flow based modeling approaches fail to capture these aspects of IS cooperation.*

*This paper reports on a case study of inter-organizational IS cooperation. A model of a real-world workflow spanning three autonomous organizations is presented together with the discussion of the encountered modeling difficulties and technical implications. Suggestions for improvement in the modeling approach are also included.*

## 1. Introduction

The EDI (Electronic Data Interchange) and workflow technologies have been in use since the 70's. The EDI technology provides for exchanging structured messages between organizations. Typically, an EDI system delivers EDIFACT-formatted messages to an organization's computer. A specialized EDI-application validates the data contents of the messages and inserts it into organization's internal data stores, and this is the farthest the EDI technology goes, at present. On the other hand, the traditional workflow technology deals with *production workflows* [GHS95] by automating intra-organizational circulating of tasks among workers (or, more specifically, worker roles such as a clerk or a manager). A worker does the designated share of the work and forwards the task to the next worker or a group of workers to be processed further on. A workflow is thus a collection of interrelated tasks. Workflow products enable to define a workflow as a type and instantiate it for each concrete work case. They provide the support for task routing and monitoring: beginning and ending conditions for tasks, deadlines for task finishing, alarming if a task was not finished in time, exception handling, etc.

These two well-established technologies are starting to mingle with each other in the inter-organizational IS cooperation: workflows try to span organizational borders and EDI systems start to have workflow features. The problem is that currently there is no sufficient support for either modeling or implementing this kind of systems.

Inter-organizational IS cooperation is often implemented with EDI technology only, which unfortunately means no support for routing, conditions, alarming and monitoring. Very often, in the process of IS cooperation modeling, the syntax of the data to be exchanged is the only well-defined aspect and everything else remains unspecified. As a result, the cooperation never meets its full potential.

Some workflow implementations span organizational borders. The EDI technology is used as the means to carry workflows across the organizational borders. However, it is not easy to integrate a workflow product with an EDI product and both with pre-existing applications possibly existing on heterogeneous platforms. Also, there is no methodology available for modeling of inter-organizational workflows.

What is needed is a new modeling methodology to capture the pattern of inter-organizational IS cooperation and even more importantly, to *encapsulate the intra-organizational aspects of this cooperation.* Note that the specification of intra-organizational processing practices may be a business secret and therefore, unlike in many modeling approaches, one does not want to go into detail with it!

Proc. Internat. Symp. on Applied Corporate Computing (ISACC'96), Oct. 30 – Nov. 1, 1996, Monterrey, Mexico, pp. 13-22.
(Best Paper Award)

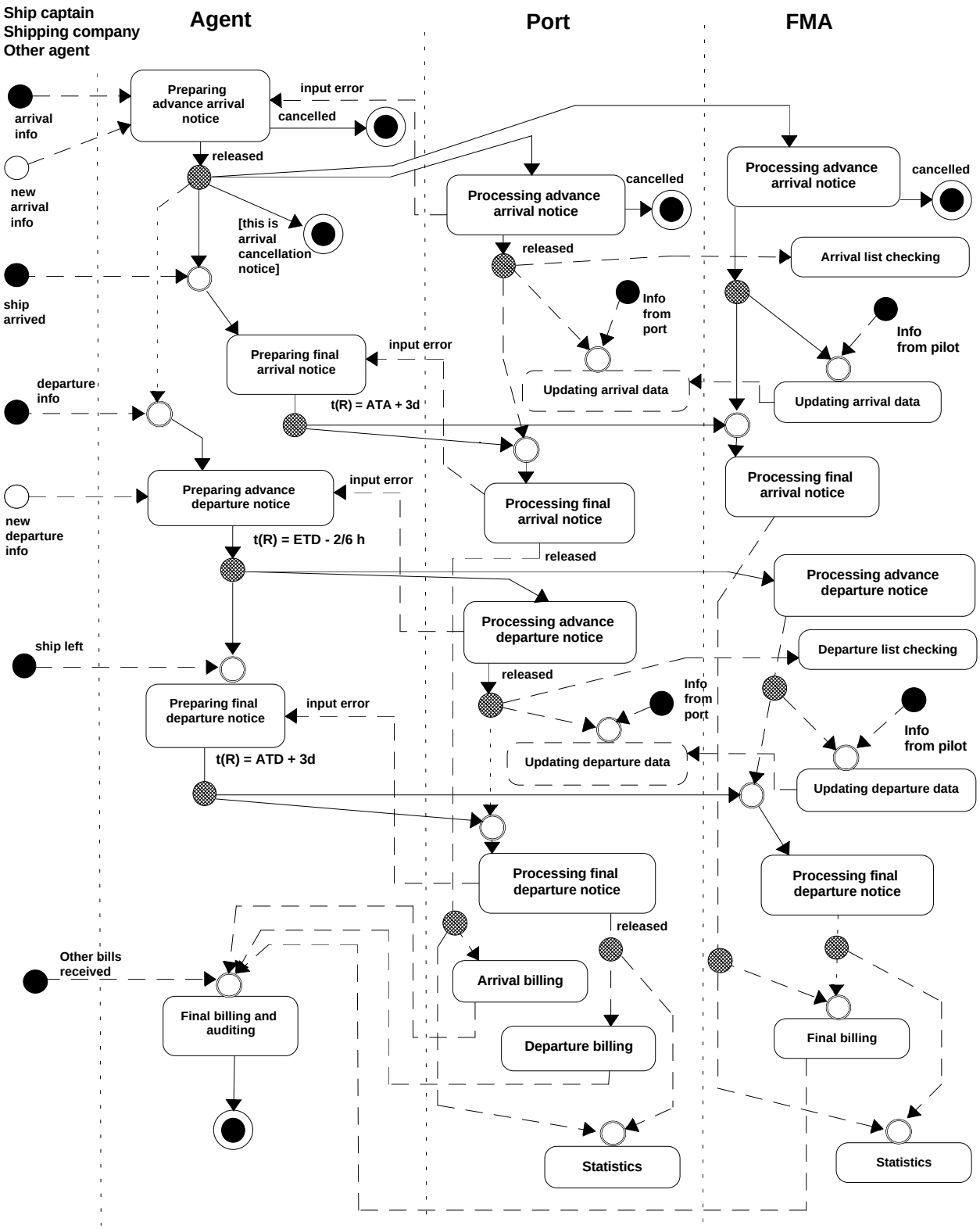


Fig.1. Transition diagram of an inter-oganizational workflow case study.

This article presents, in Section 4, a case study of the PortNet System* in which inter-organizational IS cooperation is implemented with the EDI technology. The modeling technique developed for the case study is presented in Section 3. Section 5 describes the problems encountered with this modeling approach and Section 6 the proposed new modeling principles. Section 2 describes the related work and conclusions can be found in Section 7.

The aspects we found most important in the modeling of inter-organizational IS cooperation are: specifying the expected response from the participants, specifying the inter- and intra-organizational business rules that affect the IS cooperation and defining the exception handling policies in case something goes wrong in the process of IS cooperation.

## 2. Related Work

Applying the CSCW *conversation for action* paradigm [FlLu80] to workflow modeling has been studied in [MWFF92], [Diet94a], [Diet94b] and [TeTe95]. This approach supports workflows with ad hoc nature and a high degree of human impact which, however, is not the case with the inter-organizational workflows whose degree of pre-determinism often is high.

The Workflow Management Coalition has defined the language WPDL for workflow process definitions [WMC96]. A workflow is composed of work units called activities. Participants are the users or organizational units performing the activities. A workflow type definition consists of the workflow name, description, version, workflow activity definitions, information about transitions from activity to activity, workflow participant definitions, workflow application definitions, the workflow relevant data definitions and other details. Applications are the programs to be invoked at run time to perform an activity. Workflow process relevant data defines the information that is to made available to a subsequent activity and thus may affect the choice of the next activity. The main purpose of the WPDL language standard is to make it possible to share workflow definitions among different workflow products. There is no explicit support for parallel executions. Inter-activity parallelism is implicitly allowed by making it possible to move control from one activity into more than one of other activities. However, no tools for synchronization, exception handling and distributed control, needed in inter-organizational environments, are included.

Exception handling in cooperative work has been studied in [KaRa90]. The proposed approach allows only cancellation of the currently executed task and the possibilities of canceling all the work done so far and/or starting a new kind of IS cooperation pattern are ignored.

Integrating business rules and processes to conceptual models is studied in [LoKa92]. In this so called TEMPORA philosophy, 1) the ER conceptual schemes are enriched with generalization/specialization hierarchies, complex objects and time modeling (the ERT model), 2) processes are specified with traditional data flow diagrams which the exception that processes access ERT model views instead of data stores and 3) the business rules are expressed in a relationship to the ERT model objects. Time modeling means that entities and relationships have existence and validity time periods. The rules may be static or transition constraints. The latter define valid state transitions. There are also event-condition-action rules in the WHEN...IF...THEN... syntax.

Transaction-based models have been lately proposed for workflows [GeHo94, GHS95, KrSh95, RuSe95]. The ideas call for using extended and relaxed transaction models to represent the variety of workflow processing requirements. As with transaction processing systems, a workflow would be supposed to guarantee certain "quality" properties of a workflow or its tasks, such as, e.g. atomicity and correctness. The approaches are, however, mostly based on the concept of a centralized control and are not well suited to autonomous environments.

The concept of the Information Carrier (INCA) [BMR96] involves a locale of control which is "traveling" from one executing participant to another. In the same time, required transactional characteristics are maintained at each temporary location. The concept is attractive from the point of view of autonomous participants, but it needs to be extended towards parallel executions within a single workflow.

A class of coordination languages [CaGe92] has been introduced for workflow-like "programming in the large". A C-language-based coordination language C&Co [FKB95] supports distributed execution and control, and also synchronous and asynchronous control flows. The language, however, does not take into account the autonomy aspects of the participation.

## 3. Modeling technique used

### 3.1 Model principles

The graphical description technique we used in the PortNet analysis is described in detail in [TeVe95]. In the sequel we will concentrate on its *transition diagrams* that were aimed to capture the IS cooperation pattern, i.e. the *workflow definition*. While searching for an appropriate

* The PortNet analysis work was done in the ESPRIT BRA project TransCoop (EP8012) which is partially funded by the European Commission. The partners of TransCoop are GMD (Germany), University of Twente (The Netherlands) and VTT (Finland).

diagramming technique we looked at some of the products. However, the resulting observation was in line with the accord of [Mie95] which states that "most diagramming tools rely on the flow diagram ..." and "although widely used, flow diagrams are relatively poor at supporting any formal reasoning without further semantics". We have taken a more formalized approach by applying Harel's statecharts used, e.g., in [Rum+91] as a tool for modeling object life cycles. We have enriched the statecharts with some elements of process algebras [BeKl84], namely to express forking to parallel threads and merging of the threads. Fig. 1 represents the PortNet System transition diagram: tasks are shown as boxes and transitions as directed edges.

In our technique, a transition is an instantaneous event of moving *control* from one node to another: a transition from task A to task B means that, when the execution of B is started, the execution of A is ceased. A complete workflow instance is a sequence of transitions. It starts at one or more entry points and ends at one or more exit points. The transition diagram specifies the workflow as a type, so all possible sequences are shown. However, following the state diagram semantics, at most one state may be valid at a time in a single thread of control. Of course, each workflow instance has its own life cycle (state history) and thread(s) of control.

### 3.2 Tasks

The meaning of a state symbol is "the task is being executed". There is always an participant associated with a task. The participant is responsible of performing the task.

The actual execution of the task may be automated or not. Non-automated tasks, i.e. tasks that do not have computerized support for execution of the work being defined, are drawn using dashed lines (see Fig.1).

Tasks may be *nested* and thus the task definition may be decomposed into lower level task definitions in a structured fashion.

For the purpose of stepwise decomposition of nodes, also the participants may be nested concepts, e.g. a department, a person role, etc. It is envisaged that tasks may represent certain pre-defined types of external behavior, e.g. there may be transactional and non-transactional tasks, tasks with compensation and tasks operating on streams [KrSh95].

There may be a precondition and a postcondition associated with a task. The conditions are Boolean functions defined over detectable events and workflow process relevant data. The execution of a task does not start before the precondition evaluates to true and it can not finish (i.e. generate exit events) before the postcondition evaluates to true.

### 3.3 Transitions

A transition is characterized by two components: an event stimulating a transition and a guard function (condition) which has to be satisfied in order for a transition to fire (i.e. take place). The event may be any event external to the system, or an event generated in the system, or an event of completing the execution of the source task. The condition is a Boolean function defined over the workflow process relevant data.

The notation *e1[c1]* is used to indicate that the transition takes place when event *e1* has happened and the condition *c1* has evaluated to true. If the condition part is omitted, it means that the event fires the transition unconditionally. If the event part is omitted, the default event of ending the source activity (the *released* exit event) is assumed. Therefore, an unlabelled transition between tasks T1 and T2 means: "the execution of T2 is started immediately after the execution of T1 has finished".

If many transitions point to a task, the first one to fire starts the execution of the task. At this point all other transitions leading to the task are disabled for the lifetime of the workflow instance, with the exception of possible "feedback" transitions leading from "down the stream". This brings the concept of looping which is achieved by introducing a transition loop within the same thread of control.

Transitions maintained by non-automated means are represented by dashed lines (see Fig.1).

### 3.4 Parallelism

A transition may be forked to generate parallel threads of control. A special fork symbol is used for this purpose (the gray circles in Fig.1). Each of the forked transitions may have its own condition that is AND-ed with the source transition condition.

Parallel threads of control proceed until a merge symbol is reached. There are two types of merges. The conjunctive AND merge (see the double circles in Fig.1) generates a transition when all the merging transitions are activated. This is used for synchronization of parallel threads. The other type is the disjunctive OR merge that generates a transition whenever one of the source transitions is activated. It is represented by several transitions leading to a task.

If the threads do not merge, they terminate independently. The workflow execution does not terminate before all parallel threads have terminated.

### 3.5 Entry and exit points

A transition diagram has to have at least one *entry point* (the black or white circles in Fig.1) and at least one *exit point* (the black-within-white circles in Fig. 1). An entry

point symbolizes the incoming transition, and an exit point symbolizes the stopping of processing within the transition diagram and generating the corresponding event. Entry points are unlabelled.

A black circle represents an entry point leading to a task activation. It is used to instantiate a workflow and to synchronize its progress with the external world. The white circle represents an asynchronous event which is accepted by the workflow regardless of the current state. As a result of the asynchronous event, the current state becomes the one the event points to. The currently executed task is "canceled" (with the semantics of canceling being defined for the task). For example, the effect of the asynchronous event "new arrival info" in Fig. 1 is that the currently executing tasks in the workflow are canceled, and the task "Preparing advance arrival notice" becomes the current one.

An entry transition is labeled with the name of the stimulating event and the condition, and the entry point is labeled with the name of the event.

Although any event may be generated at a task exit point, there are two standard exit events: one is *released* meaning a normal termination, and the other is *canceled* meaning an abnormal termination. The semantics of abnormal termination is specific to the activity being depicted. The assumption for an exit event is *released*.

# 4. PortNet Case Study

## 4.1 Background

In this section we describe the PortNet system [TeVe95]. The system has been in the making since 1992 and its main function is to exchange all necessary information about a ship's visit to a port. This information is expressed as notices of vessel arrival and departure that are sent to participants as EDIFACT messages.

The organizations involved with the system (the participants) are Finnish ports, governmental institutions, such as the Finnish Maritime Administration (FMA) and the National Board of Customs, and shipping agent companies. The goals of the system are to simplify the current business processes of the participants, to automate the input of ship visit information to various information systems of the participants, and to reduce costs.

## 4.2 Tasks of the participants

In relation to a visit of a foreign ship, there are a number of tasks to be carried out by the participants.

With a ship's arrival, the tasks include producing a declaration to the customs, ordering, reserving and performing of navigation and ice breaking services by FMA and ordering of other services the ship may need at the port, such as food and fuel supplements or help to unload the cargo and load in new cargo. In order to provide for the requested services, information about the ship, its arrival timetable and its current cargo has to be recorded into various information systems of the participants. The information serves as the basis of the future work planning.

The tasks related to a ship's departure include ordering and reserving navigation and ice breaking services from FMA and invoicing the ship's agent company for the port visit and the services received (supplements and loading).

## 4.3 The Agent, Port and Pilot Station

The involved organizations participate in different *roles*, for example as the Agent of the ship, as the Port the ship is going to visit or as the Pilot Station (a sub-role of FMA) which is responsible for navigating the ship in and out of the port. Each one of these roles has several instantiations whose internal behaviors vary slightly. In the sequel we take a closer look to the IS cooperation between the Agent, the Port and the Pilot Station/FMA (Fig.1)

Currently the IS cooperation is implemented with exchange of EDIFACT messages of type IFCSUM, S93.A (solid lines in the Fig.1) and by more informal communication using phone (dashed lines in the Fig.1). The IFCSUM message contains information of the ship itself (its name, radio call sign, machine power, etc.), timetable for the port visit, the cargo the ship is carrying, travel plan of the ship and the services the ship needs during the visit (navigation, ice breaking, food and fuel supplements, etc.).

## 4.4 Advance notices of arrival

The workflow is instantiated by an Agent who creates an Advance Arrival Notice (AAN) at the time they receive sufficient preliminary information about the ship visit. The execution of the Agent's task "Preparing advance arrival notice" (T1) can either be canceled before it is finished or the task can be successfully finished (i.e. released). In the latter case an EDIFACT message is submitted to other participants. In the former case the workflow is terminated (the black and white exit circle in Fig. 1).

There is also an asynchronous of T1 by the event "new arrival info". This means that, in case the Agent gets new information, they can update the data contents of the AAN and resubmit it to other participants. Effectively it means canceling of active tasks in any thread of the workflow instance and re-assigning control to task T1. In Fig.1 both the submissions and re-submissions are illustrated with the same solid lines from T1 to tasks "Processing advance arrival notice" at the Port (T2) and at the Pilot Station (T3). Note that if the new information involves cancella-

tion of the ship visit, the workflow will be terminated after both the Port and the Pilot Station have received and processed the cancellation AAN.

If the Port finds errors in the AAN during the task T2, the errors are corrected through phone calls to the Agent (the dashed line from T2 to T1 in the Fig.1). During T2, a Port List of ship visits is updated manually. This Port List is sent out weekly as a facsimile to the nearest Pilot Station (the dashed line in Fig.1).

After tasks T2 and T3, both the Port and the Pilot Station can find out more about the ship arrival locally. Very typically the Estimated Time of Arrival (ETA) of the ship is refined when the ship captain calls the Pilot Station to confirm ordering of the pilot. The new information triggers the execution of the tasks "Updating arrival data" at the Port (T4) and at the Pilot Station (T5). T4 is manual at the Helsinki port, but the T5 is partially automated at the Pilot Stations. When the Pilot Station updates an ETA of a ship, the updated information is sent to the Port automatically as a changed AAN.

### 4.5 Final notice of arrival

After the Agent has submitted at least one AAN and the ship has arrived at the port, the task "Preparing final arrival notice" (T6) can be started. The above condition is illustrated as a hollow merge circle in Fig.1. T6 has to be released (i.e. successfully executed) and a Final Arrival Notice (FAN) has to be submitted at latest 3 days after the Actual Time of Arrival (ATA) of the ship. This deadline is shown below the task T6 in the Fig.1. The Port cannot start the execution of the task "Processing final arrival notice" (T7) until after the execution of the task T2 is finished and a FAN from the Agent has been received. A similar rule applies to the Pilot Station's task "Processing final arrival notice" (T8).

| ID | Task name | Org. | Followed by |
|---|---|---|---|
| T1 | Preparing advance arrival notice | Agent | T2, T3, T6, Exit |
| T2 | Processing advance arrival notice | Port | T4, T7, Exit |
| T3 | Processing advance arrival notice | FMA | T5, T8, Exit |
| T4 | Updating arrival data | Port | Exit |
| T5 | Updating arrival data | FMA | T4, Exit |
| T6 | Preparing final arrival notice | Agent | T7, T8 |
| T7 | Processing final arrival notice | Port | T9 |
| T8 | Processing final arrival notice | FMA | T19 |
| T9 | Arrival billing | Port | T22 |
| T10 | Preparing advance departure notice | Agent | T11, T12, T15 |
| T11 | Processing advance departure notice | Port | T13, T16 |
| T12 | Processing advance departure notice | FMA | T14, T17 |
| T13 | Updating departure data | Port | Exit |
| T14 | Updating departure data | FMA | Exit |
| T15 | Preparing final departure notice | Agent | T16, T17 |
| T16 | Processing final departure notice | Port | T18, T20 |
| T17 | Processing final departure notice | FMA | T19, T21 |
| T18 | Departure billing | Port | T22 |
| T19 | Final billing | FMA | T22 |
| T20 | Statistics | Port | Exit |
| T21 | Statistics | FMA | Exit |
| T22 | Final billing and auditing | Agent | Exit |

Tab.1. Case study task summary.

The Port starts the execution of the task "Arrival billing" (T9) as soon as the task T7 has been released whereas FMA waits until both T8 and "Processing final departure notice" (T17) have been released before it starts the final billing (T19).

The FMA has an automated system for invoicing but the data exchange between the invoicing system and the PortNet system is still manual (by mail), therefore the dashed lines in Fig.1.

### 4.5 Notices of departure

The Agent can start the execution of the task "Preparing advance departure notice" (T10) only after T1 has been released and the departure information has been received. Note that there are no timing conditions between T6 and T10. In practice, the Advance Departure Notice can be sent out long before the ship actually arrives.

The processing of Advance and Final Departure Notices (ADN and FDN at T11, T12, T15 and T16) is almost identical to processing of notices of arrival. At the Port, the "Statistics" task (T20) is started only after both ADN and FDN have been received and processed.

The workflow is successfully finished when the bills from FMA and the Port have arrived at the Agent and the task "Final billing and auditing" (T22) has been released.

## 5. Problems with the chosen modeling approach

In the sequel we describe the problems the chosen modeling approach caused in the PortNet analysis.

### 5.1 Variations in local processing

It was soon discovered that we modeled the IS cooperation in too much detail. The local processing varied from port to port and pilot station to pilot station. In reality, the Fig.1 describes only the IS cooperation involving one Agent (the Finnmag Company), the Port of Helsinki and the Harmaja pilot station. Therefore, when modeling the IS cooperation at the level of the cooperating roles, such as the Pilot Station and the Agent, the modeling should have remained on a much higher level of abstraction. Modeling the slightly different processing practices at, for example, Port of Helsinki and Port of Turku would add unnecessary complexity and furthermore it would not give any better understanding of the *IS cooperation* these ports do with other organizations.

### 5.2 Managing complexity

Describing the local processing in too much detail also had its consequences when managing complexity. It would have been cumbersome to add, for example, more roles in the transition diagram of Fig.1. In reality however, there will be more players in the PortNet cooperation, and one would have to start thinking of the roles such as the Customs and the Ministry of Environment. Therefore, it would be essential to have the possibility of easily adding more roles and more messages into the diagrams.

A layered modeling approach could help to cope with the problem, and also with the problem of local variations, mentioned in the previous subsection.

### 5.3 Controlling parallel threads

The chosen model failed to represent the real nature of the distributed control in an inter-organizational workflow. In fact, the spirit of the diagram was that there was some centralized controlling entity–which, however, was missing in reality. Instead, each autonomous organization had its own controlling entity. It was incorrect to model the task activation across the organizational boundaries as a state transition because it meant that the requesting organization was loosing the control and thus its autonomy also.

To alleviate the problem, the inter-organizational communication should be modeled as a protocol, and the control of the workflow should be retained at each participating party in a form a protocol machine.

### 5.4 Asynchrony violates autonomy

It was an attractive idea to let asynchronous events to reset the global state of a workflow instance by canceling active tasks and assigning the control to a selected one, but it cannot stand the facts of reality. The proposed mechanism is simply too powerful–it would force the participating organizations to waive the total control over their participating activities to an external entity (in this case the workflow originating organization). However, organizations have policies and it may happen that interrupting of a thread is not allowed because it is too late or some uncancellable (uncompensatable) actions have been already performed (for example, you cannot stop the space shuttle once it became ignited). An asynchronous transition may lead to loosing some results of work done so far although it may be re-usable. For example, resetting the workflow shown in Fig. 1 with the asynchronous event "new arrival info" would possibly cause loosing of some information generated at the Port or FMA (like services requests, etc.) although the only new information would be a slight change in the estimated time of arrival.

Also, looping back from within a component thread into a main thread has the same effect of canceling all active tasks, in the current model. For example, invoking the input error transition from the task "Processing advance arrival notice" (T2) back to T1 in the main thread

would possibly result in loosing all the information already collected and generated at each participant organization, although the error might have been of minor nature (like misspelling) and the erroneous item may be easily corrected "on the fly".

It seems that dealing with asynchrony requires caution. It should be possible, when necessary, to engage the parties into negotiations each time an asynchronous event is sent over autonomy boundaries. Additionally, a semantic has to be defined for *interrupting* the workflow instance. Interrupting is different from canceling in that we know that the same workflow instance will be re-invoked with slightly different data.

### 5.5 Implementation

The model used in the case study would require support of a complex distributed system. The system would support not only delivering of task invocations and the relevant data (this can be achieved with current EDI technologies) but it would have maintain also the global state of all workflow instances in a failure-resilient way. For this purpose special workflow control protocols and the corresponding recovery techniques are needed. The main difficulty would be to implement a system for use in heterogeneous system environments and integrateable with legacy systems.

## 6. Suggestions for a workflow modeling framework

The PortNet analysis work convinced us that the following aspects would be essential in modeling inter-organizational workflows:

### 6.1 Add layers

A model should have a separate global layer for representing purely inter-organizational aspects of the IS cooperation. The organization-specific behavior would be encapsulated in the tasks visible at this level. The global workflow would be decomposed into organization level tasks called *subflows*. The next level would be devoted to decomposing the subflows from the point of view of global agreed-upon semantics. For example, it can be shown what it means to cancel or to interrupt a subflow. The next decomposition layers may be devoted to specifying organization-specific implementations.

### 6.2 Model IS cooperation as protocols

According to the layered approach, one should focus on defining the *IS cooperation interface* (an abstract service) and the related *IS cooperation protocol* (the messages and ordering thereof). The difference between this type of protocol and a typical on-line data communications protocol is in the time scale: waiting for a response to a request may take days instead of seconds.

The IS cooperation protocol would be used to implement "rendezvous points" such as the *agreement* and *acceptance* phases of the Workflow Loop model [MWFF92, MWF93]. Especially, dealing with asynchronous transitions should be supported by a "negotiation protocol" leaving the room for participant's organizational autonomy.

In a general case, there may be more rendezvous points dealing with e.g. status inquiries, cancellations of requested tasks, etc. The rendezvous points would be used to synchronize parallel activities with each party coming to a rendezvous point at its own pace.

On the basis of the PortNet experience, we have come to the conclusion that the message exchange protocol should be defined in a flexible, upwards scalable way, by using the *communicating finite state automaton* [Dant80]. The message exchange of each role is defined in isolation, in its own state automaton. The states are only the *externally visible* ones and transmitting or receiving a message moves the role into the next state. This way new roles and messages can be added and analyzed very easily.

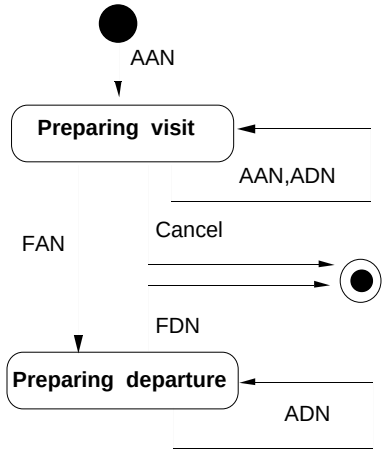


Fig.2. Automaton of the Agent.

Figure 2 shows the automaton of an Agent. The externally visible states are "Preparing a visit" and "Preparing a departure". Transmitting the first AAN moves the Agent into the "Preparing a visit" state. The Agent can update the arrival information by sending more AANs, in this state. The Agent can also already transmit ADNs. Transmitting a FAN moves the Agent into the "Preparing a departure" state in which the Agent can transmit several ADNs. Transmitting a FDN or a cancellation message terminates the automaton of the Agent.

### 6.3 The Two-Phase Transition protocol

A special negotiation protocol could be used to deal with asynchronous communications leading to workflow interruptions. We are proposing a protocol modeled on the Two-Phase Commit protocol (2PC) [GrRe92] used in transaction processing. The *Two-Phase Transition* protocol would operate in the following way: In the first phase of the protocol, the workflow coordinator (the participant responsible for the global workflow instance) would inquire whether the other participants are ready to accept the asynchronous transition in their subflows. If all the participants agree, the transition is issued, otherwise it is withhold and the coordinator has to find some other way to deal with the case. Figure 3 shows the states of a subflow at a participating organization. When a subflow is active, the organization exercises autonomy in the first phase (it may accept or refuse the interrupt request) and it has to comply with the final decision in the second phase, if it has accepted the original request. If the decision is INTERRUPT, the subflow remain in the state **Interrupted** until the actual asynchronous transition is issued or the flow is canceled by some reason (CANCEL may be caused by an asynchronous event as well).

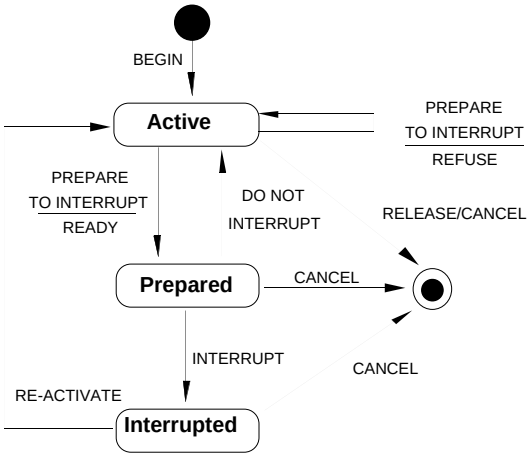


Fig. 3. The Two-Phase Transition Protocol: the subflow states.

The real implementation of the Two-Phase Transition protocol has to take in account that the time span of the protocol may be much longer than that of the 2PC protocol: it may, for example, take days to respond to the PREPARE TO INTERRUPT message.

### 6.4 Include consistency rules

Besides the data to be exchanged, it is crucial for the sender of a message to know which errors and inconsistencies in the data contents lead to rejection of the data at the receiving end. These *intra-organizational data consistency rules* should be defined and made visible to other cooperating parties.

Also *temporal rules* related to the arrival or processing time of the whole message or the validity times of its individual data items should be defined.

We find the TEMPORA philosophy useful when it comes to attaching the above rules to the data definition. For example, EDIFACT messages could be enriched to carry the relevant rules together with the data. This would be a straightforward solution to add more semantic knowledge to the messages.

### 6.5 Define exceptions

In a normal situation a workflow will be executed from the beginning to the end. However, in an inter-organizational environment the participating organizations are autonomous and can refuse to execute their abstract services at any given moment, for their internal reasons. Therefore, it is essential to carefully analyze the exception situations and to define policies how to cope with them. Note that recovery policies may involve more IS cooperation!

In order to find the deadlock situations, one can combine the state automata of the roles [Dant80] and look for dead ends in the paths. When there are several roles and/or transitions the combined state automata will become very large and complex. For this reason we suggest a pair-wise combination only (e.g. combine an Agent's state automata with the Pilot Station's and then combine the Agent's with the Port's and so on) which should be sufficient for analyzing purposes.

When defining the policy for exception handling, one should determine if the workflow (or a part of it) is supposed to be *atomic* or not. If it is atomic, either all or none of the services and message transmissions of the workflow definition are executed for each workflow execution.

In practice, maintaining atomicity means that *a compensating workflow* has to be designed for each atomic workflow definition. An execution of the compensating workflow will undo the executions of abstract services of the original workflow in the participating organizations, i.e. the compensating workflow is yet another IS cooperation pattern.

## 7. Conclusions

Our experience shows that modeling of inter-organizational workflows requires a new approach. The focus has to be on specifying the expected response from the participants, specifying the inter- and intra-organizational business rules that affect the IS cooperation and defining the exception handling policies in case something fails in the IS cooperation. Our future work includes developing a

graphical diagramming technique and a set of tools to model these aspects and to assist in finding the abnormal situations.

The most promising is a protocol-based approach whereby the workflow control is always retained at each participating site and the rendezvous points are used to synchronize the parallel and autonomous component workflows, and also to deal, in a bilateral way, with exceptions occurring in the course of workflow executions.

## Acknowledgments

The authors wish to express their thanks to Aarno Lehtola for his comments and for making the preparation of this paper possible within the DYNAWORK project at VTT Information Technology.

## References

[BeKl84] Bergstra, J. A. and , Klop, J. W., "Process algebra for synchronous communication", Information and Control, Vol. 60, No. 1–3, January/February/March 1984, pp. 109-137.

[BMR96] Barbará, D., Mehrotra, S., Rusinkiewicz, M, "INCAs: Managing Dynamic Workflows in Distributed Environments", Journal of Database Management, Vol. 7, No. 1, 1996.

[CaGe92] Carriero, N. and Gelernter, D., "Coordination languages and their significance" CACM, Vol. 35, No. 2, February 1992, pp. 96-107.

[Dant80] Danthine, A.S., "Protocol representation with finite-state models", IEEE Trans. on Communications, Vol. COM-28, pp.632-643.

[Diet94a] Dietz, J.L.G, "Business Modeling for Business Redesign", Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences, 1994, pp. 723-732.

[Diet94b] Dietz,J.L.G., "Modeling business processes for the purpose of redesign", in: Business Process Re-Engineering: Information Systems Opportunities and Challenges. Classon, B.C. et al (editors) Elsevier Science B.V. (North-Holland),1994 IFIP, pp. 233-242.

[Mie95] Miers, D., "Process product Watch: Modeling Tools report", Enix Limited, Richmond, Surrey TW9 1PL, U.K., 1995.

[FKB95] Forst, A., Kühn, E. and Bukhres, O., "General Purpose Work Flow Language", Distributed and Parallel Databases, Vol. 3. No. 2, April 1995, pp. 187-218.

[FlLu80] Flores, F.M. and Ludlow, J.J., "Doing and Speaking in the Office", in: Decision Support Systems: Issues and Challenges, Pergamon, Oxford 1980.

[GeHo94] Georgakopoulos, D. and Hornick, M., "A Framework for Enforceable Specifications of Extended Transaction Models and Transactional Workflows", Internat. Journal of Intelligent and Cooperative Information Systems, September 1994.

[GHS95] Georgakopoulos, D., Hornick, M. and Sheth, A., "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", Distributed and Parallel Databases, Vol. 3. No. 2, April 1995, pp. 119-153.

[GrRe92] Gray, J. and Reuter, A., "Transaction Processing Systems, Concepts and Techniques", Morgan Kaufmann Publishers, 1992.

[KaRa90] Karbe, B.H. and Ramsperger, N.G., "Influence of Exception Handling on the Support of Cooperative Office Work", in Multi-User Interfaces and Applications, Gibbs, S. and Verrijn-Stuart, A.A. (Editors). Elsevier Science Publishers B.V. IFIP, 1990, pp. 2-15.

[KrSh95] Krishnakumar, N., Sheth, A., "Managing Heterogeneous Multi-System Tasks to Support Enterprise-Wide Operations", Distributed and Parallel Databases, Vol. 3. No. 2, April 1995, pp. 155-186.

[LoKa92] Loucopoulos, P. and Katsouli, E. "Business Rules in an Office Environment". SIGOIS Bulletin, 13(2), February 1992, pp. 28-37.

[MWFF92] Medina-Mora, R., Winograd, T., Flores, R. and Flores, F., "The Action Workflow Approach to Workflow Management Technology", Proc. 4th Conf. on Computer Supported Cooperative Work (CSCW'92), New York, 1992, pp. 281-288.

[MWF93] Medina-Mora, R., Wong, H.K.T. and Flores, P., "Action Workflow (tm) as the Enterprise Intergration Technology", Bull. IEEE Data Engineering, Vol. 16, No. 2, June 1993, pp. 49-52.

[RuSe95] Rusinkiewicz, M., Seth, A., "Specification and Execution of Transactional Workflows", in: Kim, W. (ed.), Modern Database Systems, ACM Press and Addison-Wesley Publishing Company, 1995, pp. 592-620.

[Rum+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederik Eddy, William Lorensen., "Object-oriented modeling and design", Prentice-Hall, Inc., 1991.

[TeTe95] Teufel, S. and Teufel, B., "Bridging Information Technology and Business — Some Modeling Aspects", SIGOIS Bulletin, August 1995, Vol. 16, No. 1.

[TeVe95] Tesch, T. and Verkoulen, P. (eds), "Requirements for the specification language", TransCoop Deliverable II.2; TC Technical Report. GMD, University of Twente, VTT January 1995.

[WMC96] Workflow Management Coalition, "Interface 1: Process Definition Interchange", WfMC TC-0020 Draft 5.0., February 1996 (Confidential).