Information Systems Vol. ??, No. ??, pp. ??-??, 1999

Printed in Great Britain
0306-4379/98 $17.00 + 0.00

PROCESS-ORIENTED ARCHITECTURES FOR ELECTRONIC COMMERCE AND INTERORGANIZATIONAL WORKFLOW

W.M.P. VAN DER AALST†

Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands,+31 40 2474295/2733, wsinwa@win.tue.nl

Abstract — The Internet's World Wide Web has become the prime driver of contemporary Electronic commerce (E-commerce). Although the emphasis has moved from Electronic Data Interchange (EDI) to the Internet, the focus is still on the technology required to exchange information rather than supporting business processes crossing organizational borders. E-commerce is not just about facilitating individual business transactions, it also comprises the management of the causal relations between these transactions. This paper stresses the process aspect of E-commerce by relating it to workflow management. Traditional workflow management systems assume one centralized enactment service and have problems dealing with dynamic changes and local variations. Since E-commerce is characterized by interorganizational workflows distributed over autonomous business units, these systems tend to be useless in the context of E-commerce. This paper reviews new and existing architectures to enable interorganizational workflow. The presentation focuses on two approaches to partition an interorganizational workflow over multiple business partners. Both approaches are evaluated. One of the key concerns in this paper is the possibility to verify the correctness of the interorganizational workflow. The dynamics of the marketplace, with rapid changing business processes and relationships, underlines the need for verification tools.

Key words: Electronic commerce, workflow management, trade procedures, Petri nets, message sequence charts

1. INTRODUCTION

Traditional *E-commerce* [13,18,30,36,45,47,51,53] mainly using *Electronic Data Interchange* (EDI) is rapidly moving to the Internet. Moreover, E-commerce is moving from long-lasting well-defined business relationships to a more dynamic situation, also named open E-commerce [27,35], where parties having no prior trading relationship engage in a common business process. Consequently the operational boundaries between organizations have become fluid. As a result, it is difficult to separate interorganizational business processes from the intraorganizational ones. E-commerce has complicated the management of business processes. The processes are scattered over multiple organizations and are subject to frequent changes. Unfortunately, research efforts mainly focus on the business transactions rather than the business processes.

To position our work, we use the standard classification of E-commerce. *Consumer-oriented commerce* comprises business-to-customer applications such as remote shopping, banking, and infotainment-on-demand. *Business-to-business commerce* is the typical application domain of EDI where business partners join forces to enact interorganizational business processes. *Intraorganizational business* is the third category. Intraorganizational business facilitates the sharing of business information, maintaining business relations, and conducting business transactions *within* the company. The definition of a company is continuously shifting (e.g., consider the emergence of virtual companies). Therefore, it is not useful to distinguish between the last two categories (business-to-business commerce and intraorganizational business). We will use the term *interorganizational* instead. The main focus of this paper is on interorganizational workflow, i.e., workflows crossing organizational boundaries inside the company or

† Part of this work was done at the LSDIS Lab (University of Georgia, USA) and the CTRG Lab (University of Colorado, USA) during a sabbatical leave.

between companies. Although the results are relevant for consumer-oriented commerce, we concentrate on the two other categories because there the need for process support is most prominent.

Present E-commerce applications focus on the *front side* of business documents [35]. EDI based developments such as UN/EDIFACT, XML/EDI, and open EDI [27] concentrate on the representation of transaction data. In this paper, we focus on the *back side* of business documents. The back side of a business document reflects the business process rather than individual transactions or transaction data. To illustrate the significance of an explicit notion of a business process in the context of E-commerce, we use an example taken from [35]. "Consider only a simple postpayment contract for goods. The buyer assumes that an invoice will be sent after delivery to trigger the payment obligation. The seller, on the other hand, abides by the practice that payment becomes due from the time of delivery, and does not send an invoice. Thus, the goods arrive, and the buyer does not pay, waiting for an invoice. Meanwhile the seller becomes irked, and initiates collection proceedings. This is an example of the so-called *battle of the forms*. Each party utilizes standard documents such as a purchase order, delivery agreement, etc. which contain (typically on the back side, in small print) the terms and conditions that are their style of doing business." Supporting E-commerce without specifying the "way of doing business" creates many problems. The only way to avoid such problems is to make interorganizational workflow explicit.

Until recently there were no generic tools to support workflow management. As a result, parts of the business process were hard-coded in the applications. For example, an application to support task X triggers another application to support task Y. This means that one application knows about the existence of another application. This is undesirable, because every time the underlying business process is changed, applications need to be modified. Moreover, similar constructs need to be implemented in several applications and it is not possible to monitor and control the entire workflow. Therefore, several software vendors recognized the need for *workflow management systems*. A workflow management system (WFMS) is a generic software tool which allows for the definition, execution, registration and control of workflows (cf. [33,50]). At the moment, many vendors are offering a workflow management system. This shows that the software industry recognizes the potential of workflow management tools. Unfortunately, today's workflow management systems are designed to support the workflow *within one business unit* rather than between business units. Moreover, workflow management systems of different vendors have problems cooperating. The Workflow Management Coalition (WfMC, [33,50]) is working on standards for workflow interoperability. However, the real issue is not to connect systems but to develop fundamentally new concepts and architectures to support interorganizational workflows. Today's workflow management systems also have problems dealing with both ad-hoc changes and evolutionary changes [6,15]. As a result, they are not used to support dynamically changing workflow processes or the workflow process is supported in a rigid manner, i.e., changes are not allowed or handled outside of the workflow management system.

This paper focuses on two architectures for interorganizational workflows, i.e., two ways to organize information systems to support several business partners that are involved in shared workflow processes. We restrict ourselves to structured processes with a predefined set of tasks and routing constructs. In many cases, where the coordination structure and the interaction between the business partners are not specified explicitly, this is not a realistic assumption [39]. Nevertheless, there are numerous situations where the organizations participating in a shared workflow processes feel the need to specify the coordination structure explicitly. The *(extended) case transfer architecture* (CTA) uses a vertical decomposition of the workflow, i.e., cases (workflow instances) are partitioned over the business partners involved. The *loosely coupled architecture* (LCA) uses a horizontal decomposition where the process itself is partitioned, i.e., each business partner has a private workflow process which is connected to the workflow processes of some of the other partners. We will compare both architectures and identify criteria to select a suitable architecture.

In an interorganizational workflow, business partners operate essentially independently, but have to synchronize at certain points to ensure the correct execution of the overall business process. Synchronization of parallel processes is known to be potential source of errors (e.g., deadlock and livelocks). Therefore, it is difficult to establish the correctness of a complex interorganizational workflow. In [1,3] a notion of correctness is introduced which is called *soundness*. This notion can be extended to interorganizational workflows [2,4]. For both architectures (CTA and LCA), concepts and analysis techniques to verify the correctness of an interorganizational workflow are discussed. Because processes are a dominant factor in interorganizational workflow, it is important to use an established framework for modeling and analyzing workflow processes. In this paper, we use Petri nets to represent workflow processes [1,8,17,20,40,41]. The graphical representation and clearly defined semantics of Petri nets, allows us to express the concepts in a compact and unambiguous manner.

The remainder of this paper is organized as follows. First we introduce the various ways of interoperability and review the functionality of today's workflow management systems. Then we focus on the two architectures. For each of the architectures, we provide a clear definition, survey the technical aspects, and discuss verification issues. Since the CTA is quite unconventional, we describe a real application (Sagitta 2000: handling up to 70,000 cases a day and distributed over 50 business units!) using this architecture. Finally, we compare both architectures.

2. INTERORGANIZATIONAL WORKFLOW

As indicated in the introduction, this paper focuses on interorganizational workflows, i.e., several business partners are involved in shared workflow processes. From a technical point of view these issues are also of concern to the WfMC [33,50]. The WfMC is working on standards to enable *workflow interoperability*. These interoperability standards address the technical issues but not the content of the coordination structure. It is our belief that possible conceptual architectures for truly supporting interorganizational workflows should be explored before solving the technical issues (mainly syntactical). Therefore, we start by reviewing the various forms of interoperability and their usefulness in the context of E-commerce (business-to-business and intraorganizational).

2.1. Capacity sharing

The first form of interoperability is *capacity sharing*. This form of interoperability assumes centralized control, i.e., the routing of the workflow is under the control of one workflow manager. The execution of tasks is distributed, i.e., resources of several business partners execute tasks. Figure 1 illustrates this form of interoperability.

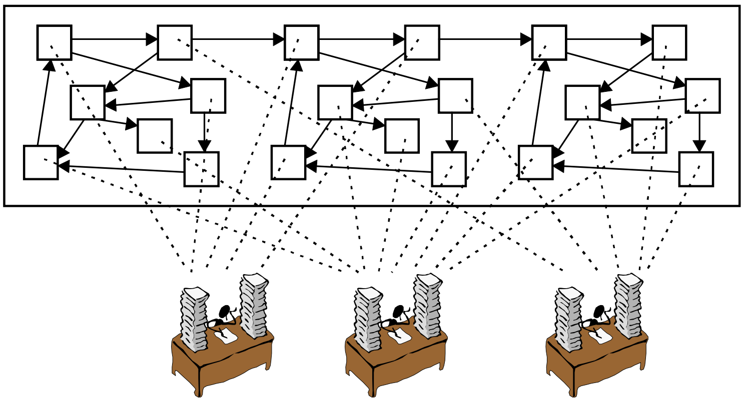

Fig. 1: Capacity sharing.

2.2. Chained execution

Figure 2 illustrates the second form of interoperability. If *chained execution* is used, then the workflow process is split into a number of disjunct subprocesses which are executed by different business partners in a sequential order. This form of interoperability requires that a partner transfers or initiates the flow for a case after completing all the work. In contrast to capacity sharing, the control of the workflow is distributed over the business partners.

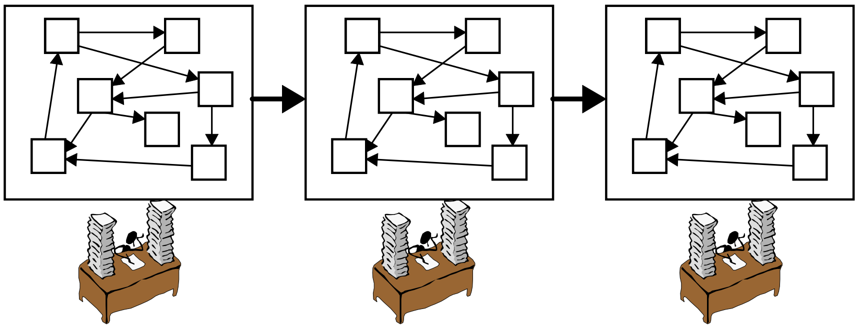

Fig. 2: Chained execution.

2.3. *Subcontracting*

The third form of routing is *subcontracting*. There is one business partner which subcontracts subprocesses to other business partners. Consider for example Figure 3 where two subprocesses are subcontracted. For the top-level business partner the two subcontracted subprocesses appear to be atomic. For the two business partners executing subcontracted work, the subprocesses can be very complex. Note that the control is hierarchical, i.e., although there is a top-level actor, the control is distributed in a tree-like fashion.

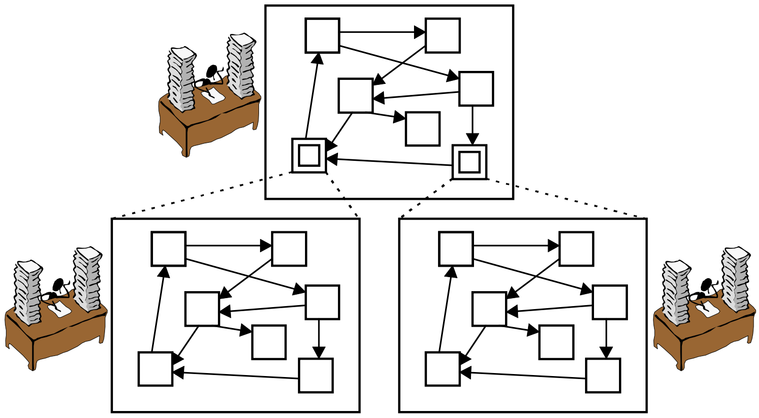

Fig. 3: Subcontracting.

2.4. *Case transfer*

Figure 4 illustrates the fourth form of interoperability: *case transfer*. Each business partner has a copy of the workflow process description, i.e., the process specification is replicated. However, at any time, each case resides at exactly one location. Cases (i.e., process instances) can be transferred from one location to another. A case can be transferred to balance the workload or because tasks are not implemented at all locations.

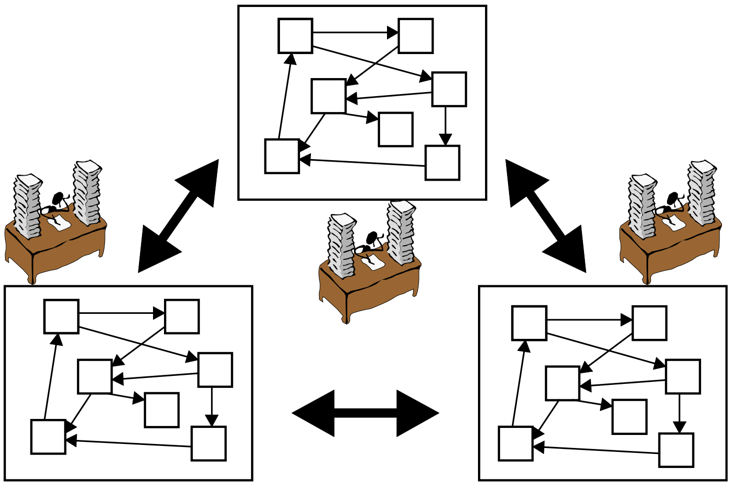

Fig. 4: Case transfer.

2.5. *Extended case transfer*

In Figure 4 it is assumed that each of the business partners uses the same process definition. However, it is possible to allow local variations, e.g., at a specific location the process may be extended with additional tasks. It is important that the extensions allow for the proper transfer of cases. This means that the extensions are executed before transferring the case or that there is some notion of inheritance which allows for the mapping of the state of a case during the transfer.

2.6. *Loosely coupled*

The last form of interoperability is shown in Figure 5. For this form of interoperability the process is cut in pieces which may be active in parallel. Moreover, the definition of each of the subprocesses is local, i.e., the environment does not know the process. Only the protocol which is used to communicate is public for the other business partners.

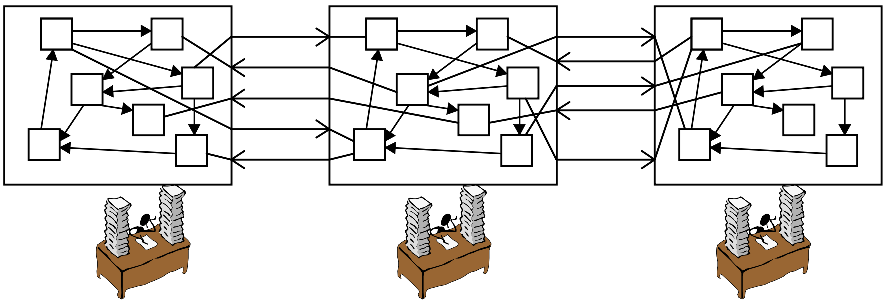

Fig. 5: Loosely coupled.

Note that capacity sharing uses centralized control. The other forms of interoperability use decentralized control. However, note that in case of subcontracting and (extended) case transfer part of the control is (can be) centralized.

Capacity sharing is the only form of interoperability which does not require some partitioning of the workflow. The other five forms of interoperability partition the workflow in various ways. Basically there are two partitioning dimensions: the *case dimension* and the *process dimension*. *Vertical partitioning* uses the case dimension to distribute the workflow, i.e., the cases are distributed over several business partners but the process is not cut into pieces. *Horizontal partitioning* is based on the process dimension, i.e., the process is cut into pieces and cases are not allocated to specific business partners (in fact several business partners may be working on the same case at the same moment in time). Figure 6 illustrates both partitioning dimensions. The partitioning dimensions are orthogonal and each characterized by pro's and con's. An advantage of a vertical partitioning is that at any time the whole case resides at one location, i.e., many concurrency problems are avoided. Another advantage is the fact that the interaction between the enactment services of the organizations involved is simple and can be realized using today's generation of workflow management systems: The only mechanism that is needed is a swap-in/swap-out facility. The vertical partitioning also has some drawbacks. The same process description, or variants of the same description, needs to be replicated over all business partners involved. Moreover, the degree of potential parallelism is reduced because a case cannot be processed by two organizations at the same time. Horizontal partitioning allows for the parallel processing of one case in multiple organizations at the same time. Moreover, each organization is only confronted with the relevant part of the workflow process. However, the coordination costs of horizontal partitioning can be high. Causal relations between tasks in different organizations need to be maintained, exceptions become more difficult to handle, and all kinds of concurrency problems can occur (e.g., conflicting decisions in different organizations).

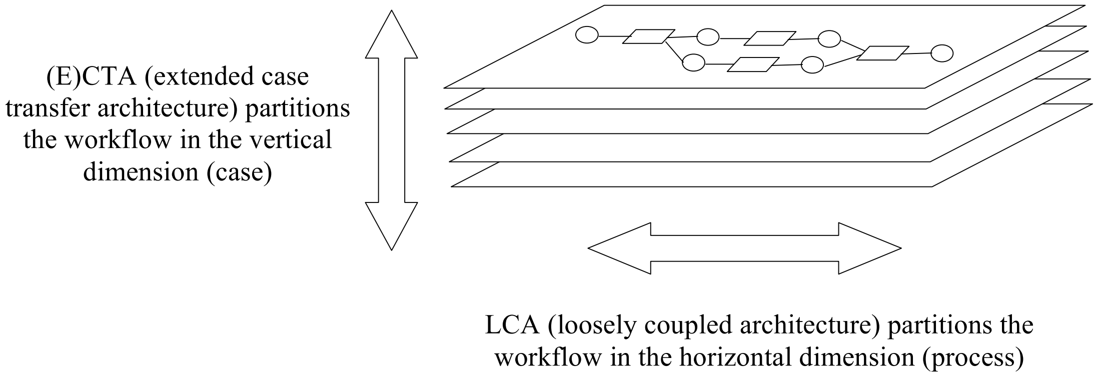

Fig. 6: The essential difference between both architectures.

Subcontracting and loosely coupled use a *horizontal partitioning* of the workflow, i.e., the process is cut into pieces. (Extended) case transfer uses a *vertical partitioning* of the flow, i.e., the cases are distributed over the business partners. Chained execution is a mixture of horizontal and vertical partitioning.

Which forms of interoperability are useful in the context of E-commerce? Capacity sharing assumes that there is one centralized workflow manager, i.e., the control is not distributed. Therefore, it is not useful for E-commerce applications. Chained execution is only useful for applications where the process is composed of sequentially ordered parts (each part is executed by a different business partner). In E-commerce applications there is often interaction between the business partners following a protocol which is much more complicated than just sending the case. Therefore, chained execution is not a realistic option. Subcontracting is more useful in the context of E-commerce. In traditional E-commerce applications based on EDI, the hub-and-spoke model, with a dominant business partner (the hub) gradually surrounding itself with suppliers, customers and collaborators (the spokes), is used [53]. Subcontracting can be useful for such applications. (Extended) case transfer and loosely coupled are more flexible than the other forms of interoperability. Therefore, we conclude that the subcontracting, (extended) case transfer, and loosely coupled forms of interoperability are the interesting propositions for E-commerce.

Although subcontracting is a suitable form of interoperability for many E-commerce applications, we choose not to describe the corresponding architecture in this paper. Subcontracting can only be applied in an environment with clearly hierarchical structured business partners, the architecture is fairly straightforward and can be realized using conventional technology (for the top-level business partner the subcontracted processes can be seen as normal tasks). We focus on the two remaining architectures, because these are more interesting from a research point of view.

The remainder of this paper focuses on (extended) case transfer and loosely coupled workflow processes. If (extended) case transfer is used, all business partners share a common workflow process but cases are routed from one partner to another. A *transfer policy* is used to determine when to transfer to which partner. In a loosely coupled workflow process, each business partner has a private workflow process which is connected to the workflow processes of some of the other partners. The communication mechanism that is used for interaction is *asynchronous communication*. Loosely coupled workflow processes operate essentially independently, but have to synchronize at certain points to ensure the correct execution of the overall business process. Before we discuss the architectures needed for both form of interoperability, we start with some preliminaries.

3. WORKFLOW MANAGEMENT (SYSTEMS)

The term workflow management [20,25,29,32,33] refers to the domain which focuses on the logistics of business processes. There are also people that use the term *office logistics*. The ultimate goal of workflow management is to make sure that the proper activities are executed by the right person at the right time. Although it is possible to do workflow management without using a workflow management system, most people associate workflow management with workflow management *systems*. The *Workflow Management Coalition* (WfMC) defines a workflow management system as follows [33,50]: *A system that completely defines, manages, and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic*. Other terms to characterize a workflow management system are: ‘business operating system’, ‘workflow manager’, ‘case manager’ and ‘logistic control system’. On the one hand, it is a pity that workflow management is often associated with workflow management systems, because it limits the application domain of workflow management in an unnecessary manner. (It is possible to do workflow management without using a workflow management system.) On the other hand, workflow management systems give concrete form to the essential concepts, techniques, and methods for workflow management.

Workflows are *case-based*, i.e., every piece of work is executed for a specific *case*. One can think of a case as a workflow *instance*. Examples of cases are a mortgage, an insurance claim, a tax declaration, an order, or a request for information. Cases are often generated by an external customer. However, it is also possible that a case is generated by another department within the same organization (internal customer). The goal of workflow management is to handle cases as efficiently and effectively as possible. A workflow process is designed to handle similar cases. Cases are handled by executing *tasks* in a specific order. The *workflow process definition* specifies which tasks need to be executed and in what order. Alternative terms for workflow process definition are: ‘procedure’, ‘flow diagram’ and ‘routing definition’. Since tasks are executed in a specific order, it is useful to identify *conditions* which correspond to causal dependencies between tasks. A condition holds or does not hold (true or false). Each task has pre- and postconditions: the preconditions should hold before the task is executed, and the postconditions should hold after execution of the task. Many cases can be handled by following the same workflow process definition. As a result, the same task has to be executed for many cases. A task which needs to be executed for a specific case is called a *work item*. An example of a work item is: execute task ‘send refund form to customer’ for case ‘complaint sent by customer Baker’. Most work items are executed by a *resource*. A resource is either a machine (e.g., a printer or a fax) or a person (participant, worker, employee). In most offices the resources are mainly human. However, because workflow management is not restricted to offices, we prefer the term resource. Resources are allowed to deal with specific work items. To facilitate the allocation of work items to resources, resources are grouped into classes. A *resource class* is a group of resources with similar characteristics. There may be many resources in the same class and a resource may be a member of multiple resource classes. If a resource class is based on the capabilities (i.e., functional requirements) of its members, it is called a *role*. If the classification is based on the structure of the organization, such a resource class is called an *organizational unit* (e.g., team, branch or department). A work item which is being executed by a specific resource is called an *activity*. If we take a photograph of a workflow, we see cases, work items and activities. Work items link cases and tasks. Activities link cases, tasks, and resources.

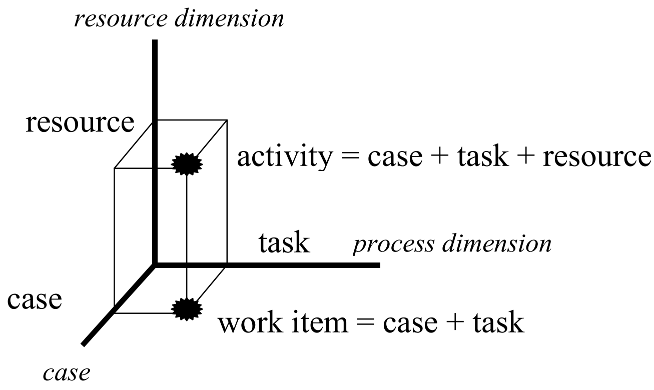

Fig. 7: A three dimensional view of a workflow.

Figure 7 shows that a workflow has three dimensions: (1) the case dimension, (2) the process dimension and (3) the resource dimension. The case dimension signifies the fact that all cases are handled individually. From the workflow point of view, cases do not directly influence each other. Clearly they influence each other indirectly via the sharing of resources and data. In the process dimension, the workflow process, i.e., the tasks and the routing along these tasks, is specified. In the resource dimension, the resources are grouped into roles and organizational units. We can visualize a workflow as a number of dots in the three dimensional view shown in Figure 7. Each dot represents either a work item (case + task) or an activity (case + task + resource). Figure 7 shows that workflow management is the glue between the cases, the tasks, and the organization. In the remainder, we focus on the first two dimensions, i.e., we concentrate on the workflow process which is defined to handle cases. We will not discuss human resource aspects such as the classification of resources (organizational modeling) and the mapping of resources to work items (scheduling) in much detail.

In this paper, we focus on workflow management systems as defined by the WfMC, i.e., systems for handling potentially large volumes of cases according to a predefined workflow process definition. Many vendors and users of such workflow management systems have joined the WfMC to identify the common characteristics of these tools, to standardize terminology, and define standard architectures and interfaces. One of the first results achieved by the WfMC was the definition of a reference model for the architecture of a workflow system. However, before we describe the reference model, we stop at the subtle difference between the term ‘workflow management system’ and the term ‘workflow system’. A *workflow management system* is a generic software product which can be applied in many organizations, e.g., Staffware (Staffware plc) or COSA (COSA Solutions). However, if the workflow management system is not installed, configured, and filled with data on process definitions and applications, it cannot be used. Therefore, we reserve the term *workflow system* for the whole of the installed workflow management system, the process definition data, the organizational data, the applications, the application data, the database management system, the configuration files, and other software components in the periphery of the actual workflow management system.

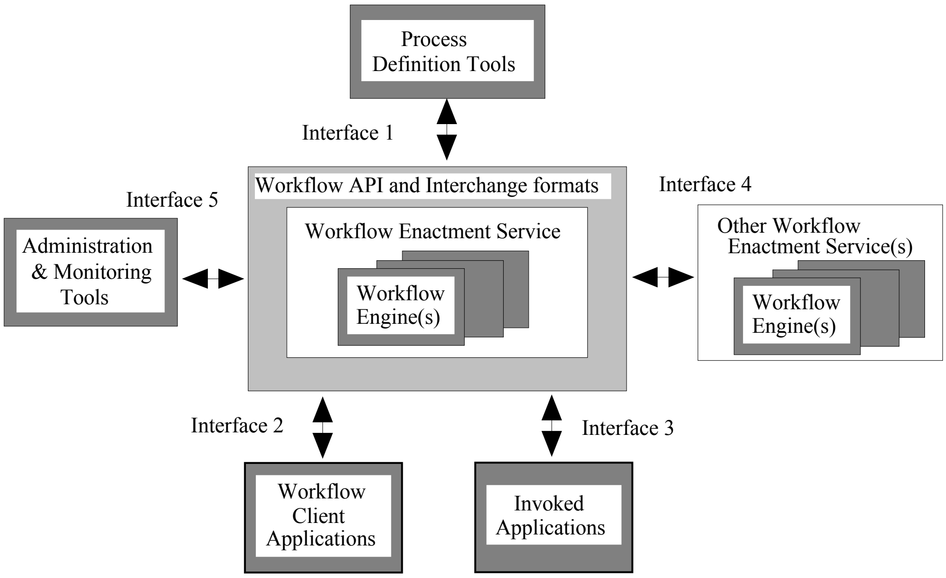

Fig. 8: Reference model of the Workflow Management Coalition (©WfMC).

Figure 8 shows an overview of the WfMC reference model. The reference model describes the major components and interfaces within a workflow architecture. The core of any workflow system is the *workflow enactment service*. The workflow enactment service provides the run-time environment which takes care of the control and execution of the workflow. For technical reasons or managerial reasons the workflow enactment service may use multiple *workflow engines*. A workflow engine handles selected parts of the workflow and manages selected parts of the resources. The *process definition tools* are used to specify and analyze workflow process definitions and/or resource classifications. These tools are used at design time. In most cases, the process definition tools can also be used as a BPR-toolset. Most workflow management systems provide three process definition tools: (1) a tool with a graphical interface to define workflow processes, (2) a tool to specify resource classes (organizational model), and (3) a simulation tool to analyze a specified workflow. The end-user communicates with the workflow system via the *workflow client applications*. An example of a workflow client application is the well-known *in-basket*. Work items are offered to the end user via such an in-basket. By selecting a work item, the user can execute a task for a specific case. If necessary, the workflow engine invokes applications via Interface 3. The *administration and monitoring tools* are used to monitor and control the workflow. These tools are used to register the progress of cases and to detect bottlenecks. Moreover, these tools are used to set parameters, allocate people and handle abnormalities. Via Interface 4 the workflow system can be connected to other workflow systems. To standardize the five interfaces shown in Figure 8, the WfMC aims at a common *Workflow Application Programming Interface* (WAPI). The WAPI is envisaged as a common set of API calls and related interchange formats which may be grouped together to support each of the five interfaces (cf. [33,50]).

The architecture shown in Figure 8 has been adopted by most vendors. For most of the interfaces, standards have been proposed. Unfortunately, these proposed standards are at a technical level with the emphasis on syntax instead of semantics. There is no consensus at a conceptual level. Consider for example Interface 1. The syntax of the exchange format has been defined without formalizing the meaning of states and essential building blocks such as the AND/OR-split/join. Similar remarks hold for Interface 4. Therefore, there is a need for a conceptual standard which clearly specifies the semantics of the basic constructs needed.

4. MODELING OF WORKFLOW: PRELIMINARIES

In this paper we use Petri nets [17,41] to model workflows. There are several reason for using Petri nets (cf. [1,20]). However, the results presented in this paper do not rely on Petri nets. They are only used for the unambiguous vendor/tool-independent representation of workflow processes. In this section, we review the basic terms and notations. Moreover, some basic correctness notions are introduced.

A *Petri net* is a network composed of squares and circles. The squares are called *transitions* and correspond to tasks that need to be executed. The circles are used to represent the state of a workflow and are called *places*. The arrows between places and transitions are used to specify causal relations. Figure 9 shows two Petri nets. A place p is called an input place of a transition t iff there exists a directed arc from p to t. Place p is called an output place of transition t iff there exists a directed arc from t to p. At any time a place contains zero of more tokens, drawn as black dots. The state of the net, often referred to as marking, is the distribution of tokens over places. For each of the Petri nets shown in Figure 9, only place i contains a token. The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net: They change the state of the net according to the following firing rule:

1. A transition t is said to be *enabled* iff each input place p of t contains at least one token.
2. An enabled transition may *fire*. If transition t fires, then t consumes one token from each input place p of t and produces one token for each output place p of t.

By using this rule it is possible to determine which transitions can fire and in what order. Consider the left Petri-net (a) shown in Figure 9. In the initial state, there is just a token in place i. Hence transition A is the only transition enabled. Therefore, A will fire consuming one token from place i and producing two tokens: one for $p1$ and one for $p2$. Then three transitions are enabled: B, C, and E. Firing B, will disable E and result in the state with a token in both $p2$, $p3$ and $p5$. Firing E, will disable both B and C and result in the state with a token in $p4$. Firing C, will disable E and result in the state with a token in both $p1$ and $p4$.

Recall that workflows are case-based (see Figure 7), i.e., every piece of work is executed for a specific case. The goal of workflow management is to handle cases as efficient and effective as possible. A workflow process is designed to handle similar cases. Think of a workflow process as a type. Cases are the instances of this type. Cases are handled by executing tasks in a specific order. The workflow process definition specifies which tasks need to be executed and in what order. In the workflow process definition, building blocks such as the AND-split, AND-join, OR-split and OR-join are used to model sequential, conditional, parallel and iterative routing [33,50]. Clearly, a Petri net can be used to specify the routing of cases. Tasks are modeled by transitions and causal dependencies are modeled by places. In fact, a place corresponds to a condition which can be used as pre- and/or post-condition for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/incoming arcs. Moreover, in [1,3] it is shown that the Petri net approach also allows for useful routing constructs absent in many workflow management systems.

A Petri net which models the process aspect of a workflow, is called a *WorkFlow net* (WF-net). It should be noted that a WF-net specifies the dynamic behavior of a single case in isolation.

Definition 1 (WF-net)

A WF-net is a Petri net with one source place i (i.e., a place with no incoming arcs) and one sink place o (i.e., a place with no outgoing arcs), and every node (i.e., a place or a transition) is on a path from the source place to the sink place.

A WF-net has one input place (source) and one output place (sink) because any case handled by the procedure represented by the WF-net is created if it enters the workflow management system and is deleted once it is completely handled by the workflow management system, i.e., the WF-net specifies the life-cycle of a case. Moreover, any node should be on a path from the input place to the output place. This requirement has been added to avoid 'dangling tasks and/or conditions', i.e., tasks and conditions which do not contribute to the processing of cases.

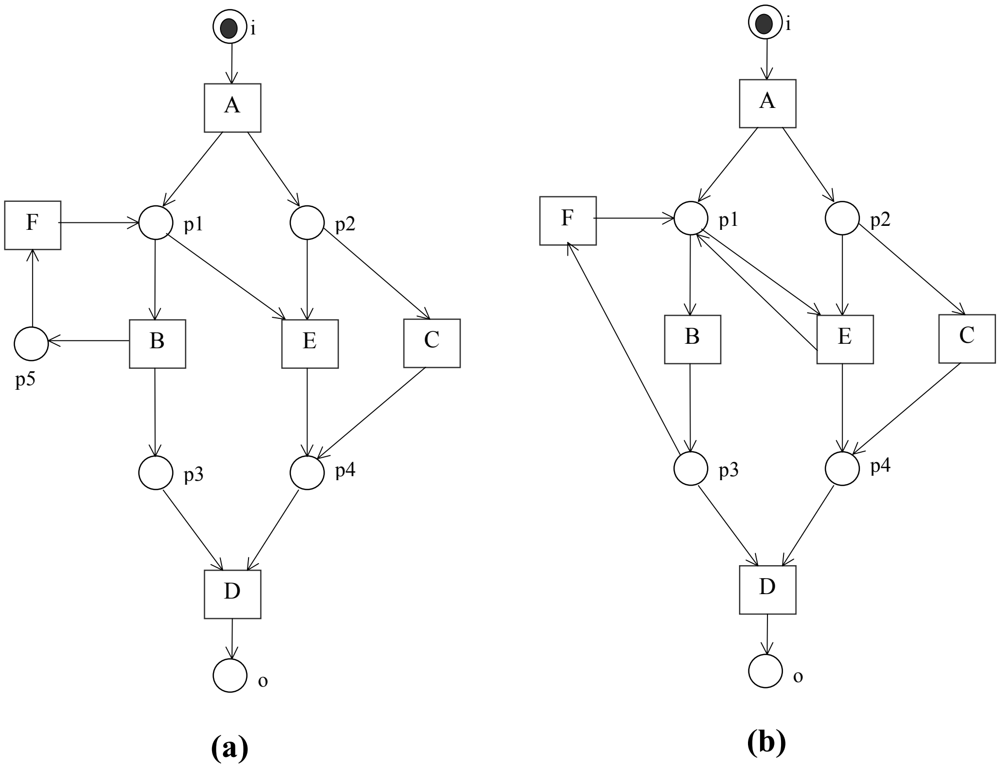

Fig. 9: Two WF-nets: one is sound (b) the other is not (a).

Both Petri nets shown in Figure 9 are WF-nets. However, as is illustrated by the left WF-net in Figure 9, a WF-net can cause serious problems if it contains dynamic errors like livelocks or deadlocks. Figure 9(a) shows an incorrect workflow process definition. The execution of task *E* before task *B* and task *C* will result in a deadlock because the case gets stuck in the state with just a token in *p4*. If task *C* is executed, then it is possible to execute *D* but a livelock is created because it is not possible to escape from the cycle formed by *B* and *F*. The definition of a WF-net only relates to the syntax of the process definition. Therefore, we need a stronger notion. The *soundness property* relates to the dynamics of the workflow process definition.

Definition 2 (Soundness)
A workflow net is *sound* if the following requirements are satisfied:

- For any case, it is possible to terminate, i.e., it is possible to reach a state with at least one token in the output place *o* (the source place of the WF-net).
- The moment the case terminates (i.e., a token appears in *o*), there are no tokens left behind in the workflow net. This means that there will be no dangling references.
- There are no dead tasks, i.e., starting with a token in the input place *i*, it should be possible to execute an arbitrary task by following the appropriate route through the WF-net.

Figure 9 (b) shows a sound WF-net. Soundness is the minimal property any workflow process definition should satisfy. Note that soundness implies the absence of livelocks and deadlocks. Soundness can be verified using Petri net techniques. In fact, we have developed a workflow verification tool, named *Woflan*, to decide soundness [1,24,48]. For a given workflow net, Woflan is able to decide whether it is sound. For this purpose, Woflan uses an interesting relation between soundness on the one hand and liveness and boundedness on the other hand. A workflow net is sound, if and only if, the net obtained by connecting *o* and *i* via an additional transition *t** is live and bounded (see [3]). Although soundness can be decided in polynomial time for certain subclasses (e.g., by using the Rank theorem for free choice nets

[17]), Woflan constructs the reachability graph to verify whether the workflow net is live and bounded. For normal workflow process definitions, the size of the reachability graph is not a restricting issue. Woflan can cope with workflow process definitions having more than 200.000 states.
The application of Petri nets to the modeling and analysis of workflows within one organization has been reported in [1,3,8,20,40]. The application of Petri nets to interorganizational workflows has been discussed in [2,4]. In this paper, the focus is not on the modeling of interorganizational workflows, but on architectures to support workflows distributed over several business partners.

5. (EXTENDED) CASE TRANSFER

In the remainder of this paper, we focus on two architectures: (extended) case transfer and loosely coupled. These architectures where briefly introduced in Section 2. In this section, we first present the *case transfer architecture* (CTA). Then we focus on the *extended case transfer architecture* (ECTA) where local extensions of the common workflow process definition are allowed. In the next section, we discuss the *loosely coupled architecture* (LCA).

5.1. Definition of CT-IOWF

The CTA assumes that the business partners involved in the interorganizational workflow share a common description of the workflow process definition (i.e., one definition is replicated). This does not mean that all partners can execute all tasks! Each business partner can execute specific parts of the process. However, it is possible that multiple business partners can execute a given task. The interorganizational workflow is partitioned vertically. This means that at any moment a case (i.e., a workflow instance) resides at exactly one location: one business partner has the case in its possession. Under certain circumstances cases jump from one location to another. There can be several reasons for such a transfer. For example, if a business partners has the case in its possession but is unable to execute any task and there are tasks which need to be executed by other business partners, then the case is transferred to one of the other business partners for further processing. Other reasons for a case transfer are issues such as load balancing or proactive reconfiguration. Basically, there are two important control rules: (1) *when* to transfer a case, (2) *where* to transfer a case to. Before we discuss the transfer policy and the technical aspects, we focus on the definition of an interorganizational workflow in the context of the CTA.

Definition 3 (CT-IOWF)
An *Interorganizational Workflow with Case Transfers (CT-IOWF)* is a tuple *CT-IOWF = (B, WF, T, task_map),* where:
1. *B* is the set of business partners,
2. *WF* is a WF-net describing the interorganizational workflow,
3. *T* is the set of tasks in the WF-net,
4. *task_map* is a function mapping business partners to sets of tasks, i.e., $task\_map \in B \rightarrow 2^T$. For each business partner $b \in B$, *task_map(b)* is the set of tasks *b* can execute.

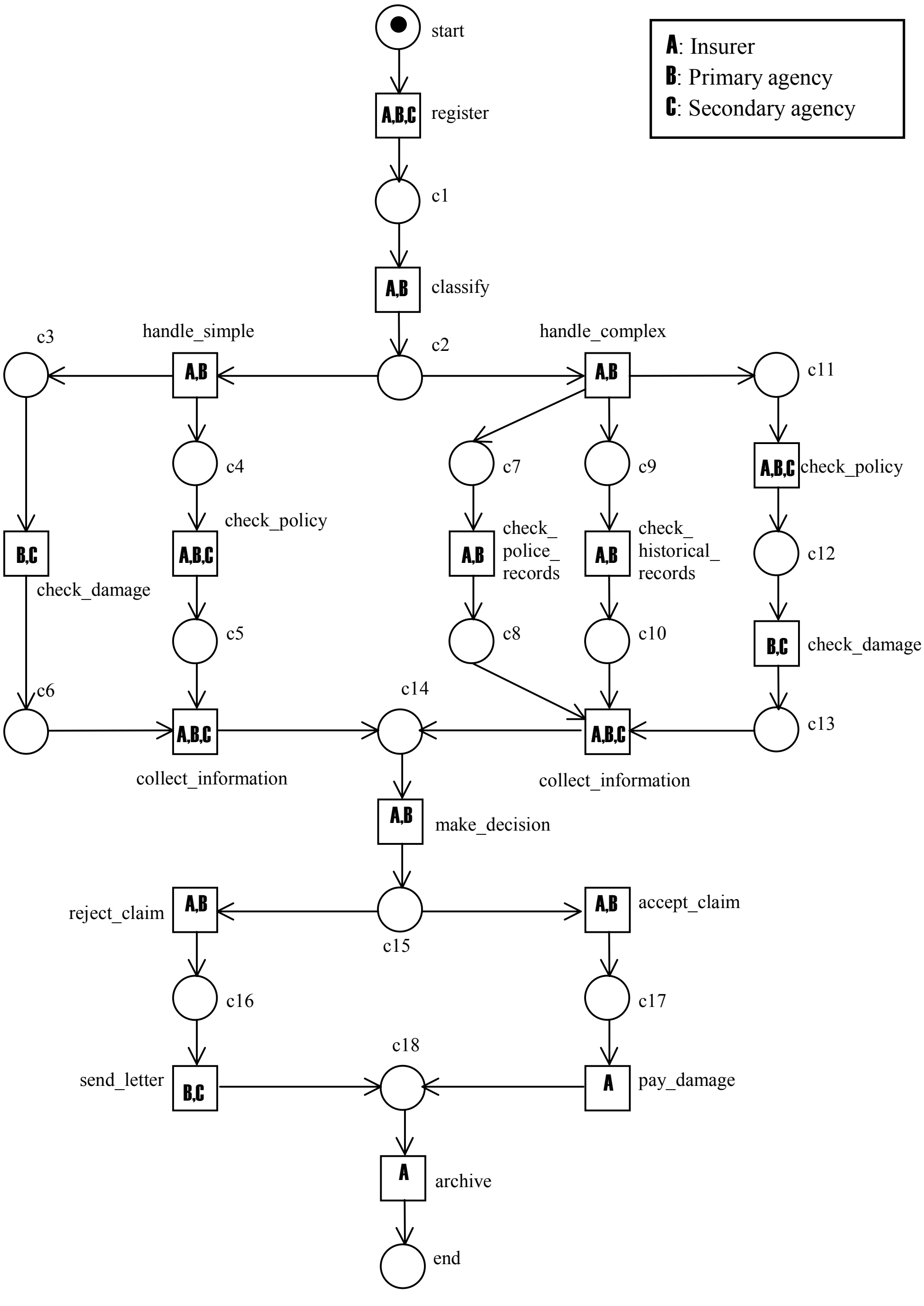

Fig. 10: A CT-IOWF.

Fig. 10 shows a fictive CT-IOWF. It describes the process of handling insurance claims. There are three types of business partners: (A) the insurance company, (B) primary agencies, and (C) secondary agencies. The insurance company is responsible for handling the claims and final payments. However, local agencies are used to establish contact with the customers and to execute certain steps in the claim handling process. Primary agencies are larger agencies which are certified to execute various tasks in the process ranging from registering claims to checking police records. Secondary agencies are generally smaller and have a smaller set of tasks they can execute. Both types of agencies are independent of the insurance company. In fact, they typically do business with multiple insurance companies and offer other services

related to mortgage loans and tax declarations. The agencies are paid for the tasks they execute. For each task there is a standard amount that is transferred from the insurance company to the agency.
The CT-IOWF shown in Fig. 10 uses inscriptions to indicate which business partners are allowed to execute the tasks. Consider for example the task *classify*. This task can only be executed by the insurance company and the primary agencies. A secondary agency is not allowed to do the classification. The process itself is quite straightforward and simplified for the presentation. All claims are first registered (task *register*) before further processing. Basically, there are two ways to register. The customer can report the claim directly to the insurance company via the Internet or telephone or the customer visits one of the agencies (primary or secondary) to file the claim. All claims are classified into simple or complex depending on different attributes of the claim (e.g., estimated damage). As indicated in Fig. 10, this task (*classify*) can only be executed by the insurance company and the primary agencies. For simple claims, two checks are performed: *check_damage* and *check_policy*. These tasks are in parallel and can be executed in any order. Note that the damage is always checked by some agency and not by the insurance company itself. After execution of both tasks the information is collected and a decision can be made. For complex claims, the procedure is more involved. The two checks *check_damage* and *check_policy* are executed in sequential order and in parallel with two additional checks: *check_police_records* and *check_historical_records*. The two additional checks cannot be executed by a secondary agency. For complex claims, the information gathered in the four checks is collected before a final decision is made. Decisions can only be made by the insurance company or primary agencies. The work of primary agencies is tested at random by the insurance company. If certain errors are detected, the agency looses its status as primary agency. If the claim is accepted, the damage is paid by the insurance company. Otherwise, the claim is rejected and the customer receives a letter from a local agency. Finally, the claim is archived by the insurance company. Clearly, Figure 10 specifies a CT-IOWF as defined in Definition 3. The network shown is a WF-net (see Definition 1) and the function *task_map* is given by the inscriptions.

In a CT-IOWF each business partner uses the same workflow process definition, i.e., the insurance company, the primary agencies, and the secondary agencies use the WF-net shown in Figure 10. However, some tasks can only be executed by certain business partners and a case always resides at exactly one location. Figure 11 shows the view of a secondary agency on the workflow process definition. The shaded tasks cannot be executed by a secondary agency. The state shown in Figure 11 is the state after checking the policy for a task classified as complex. Each of the places *c7*, *c9*, and *c12* contains a token. Suppose that the case is residing at secondary agency *C1*. In this paper we use *C1:c7+c9+c12* to denote the state where the case resides at *C1* and the places *c7*, *c9*, and *c12* contain a token. In state *C1:c7+c9+c12* there are two possibilities: (1) task *check_damage* is executed by C1 resulting in state *C1:c7+c9+c13,* or (2) the case is transferred. If the case is transferred to the insurance company, the resulting state is *A:c7+c9+c12*. In this state, there are two possibilities (assuming the case is not transferred): (1) task *check_police_records* is executed or (2) *check_historical_records* is executed. Suppose that the insurance company executes both. This results in state *A:c8+c10+c12*. In this state, there is just one possibility: a transfer to a primary or secondary agency. Note that in this process transfers are unavoidable. Consider the following scenario: a customer reports a claim at a secondary agency, the claim is classified a complex, the claim is rejected, and only the insurance company and the secondary agency are involved in the process. For this particular scenario, at least five transfers are necessary to complete the case.

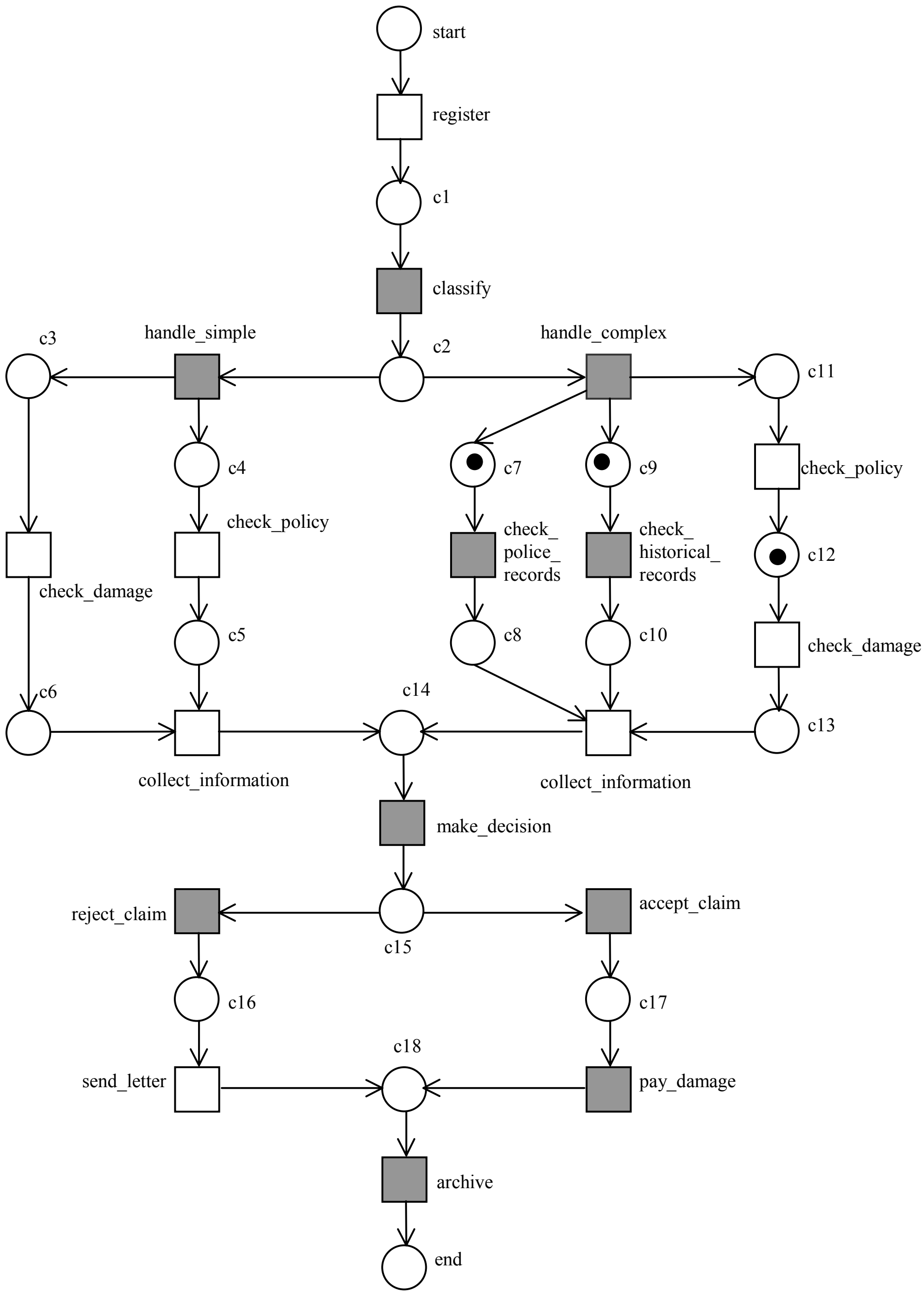

Fig. 11: The view of a secondary agency on the workflow: the shaded tasks are not available for secondary agencies.

5.2. Transfer policy

A CT-IOWF only defines the shared process and the mapping of business partners to tasks. What is missing is a transfer policy. The state shown in Figure 11 illustrates the need for such policy: the case is either transferred or task *check_damage* is executed. The transfer policy specifies *when* to jump (i.e., in which states the case will be transferred to another business partner) and *where* to jump (i.e., the new

business partner the case will be transferred to). The following four issues are important for a transfer policy:
1. *Minimize the number of transfers*: transfers consume time (both transfer and setup time) and occupy the network,
2. *Balance the load over business partners*: overloading leads to congestion and other problems,
3. *Minimize the throughput time:* the total flow time of cases should be as small as possible, and
4. *Minimize costs:* tasks should be executed at the location where processing costs are minimal.

Based on these issues it is possible to define performance criteria. Some of these criteria are conflicting, e.g., minimizing processing costs can lead to many transfers.
To discuss possible transfer policies, it is important to recognize that a case has four possible states:
1. *Active*: a task is being executed,
2. *Ready*: at least one task is enabled and is not waiting for an external trigger,
3. *Waiting*: at least one task is enabled but all enabled tasks are waiting for an external trigger, and
4. *Blocked*: no tasks are enabled.

Note that, throughout this section, we use the term 'enabling' for the situation at a given site, i.e., we only consider tasks that are available for the business partners where the case is residing. *C1:c7+c9+c12* and *A:c8+c10+c13* are examples of ready states. *C1:c1* and *A:c8+c10+c12* are examples of blocked states. Since we did not specify any triggering, there are no waiting states. A trigger is an external condition which leads to the execution of an enabled task. The execution of a task for a specific case starts the moment the task is triggered. A task can only be triggered if the corresponding case is in a state which enables the execution of the task. There are basically two types of triggers: (1) message triggers and (2) time triggers. A message trigger is an external event which activates an enabled task. Examples of messages are telephone-calls, fax messages, e-mails and EDI messages. Time triggers are used to activate enabled tasks by a clock mechanism, i.e., the task is executed at a predefined time. For example, the task 'remove document' is triggered if a case is trapped in a specific state for more than 15 hours. Suppose that the task *check_damage* requires a message trigger (e.g., a written statement of a damage expert), then state *C1:c7+c9+c12* shown in Figure 11 is an example of a waiting state (until the trigger arrives).
Based on the classification of states, we identify a number of transfer policies.
- In the *persistent* transfer policy, a case is transferred, if and only if, it is in a waiting or blocked state and a transfer leads to a ready state at another location.
- In the *strongly persistent* transfer policy, a case is transferred, if and only if, it is blocked and a transfer leads to a ready or waiting state at another location.
- In the *time-out* transfer policy, a case is transferred, if and only if, it has not been active for a specified time and a transfer leads to a ready or a waiting state at another location.
- In a *load-balancing* transfer policy, cases that are ready, waiting, or blocked are transferred to another location if they are not-blocked at the new location and the resulting workload at the new location is smaller.
- In a *cost-driven* transfer policy, cases are transferred to the location where the processing costs for the next task are minimal.
- *Intelligent* transfer policies use a mixture of the above strategies and also anticipate future developments in their calculations (e.g., the workload in the next period).

These policies indicate that the CTA results in a very flexible environment where the workload can be dynamically redistributed over the organizations involved. Note that the vertical partitioning used by the CTA is particularly attractive if tasks are typically supported by multiple organizations, i.e., based on various criteria, the work can be distributed dynamically. This flexibility is not possible in architectures based on a horizontal partitioning. (At least not without explicitly modeling this flexibility in the workflow process definition.)
In the past decade many research groups have been working on research prototypes with distribution aspects as the focal point. (See also the section on related work.) These groups have evaluated the performance of distributed enactment services using various architectures and partition strategies. In [12] the effect of server migration is investigated using a quantitative model. In [42] the impact of several synchronization schemes on the communication costs has been evaluated using an approach based on the partitioning of state charts. A detailed qualitative/quantitative study of the impact of the CTA on the communication costs is outside the scope of this paper. Moreover, the goal of the CTA is not to tackle

performance issues. The benefits of the CTA are: (1) a simple robust architecture without the need for complex concurrency control, (2) to be able to integrate workflow management systems of different vendors using a simple swap-in/swap-out mechanism, and (3) a reduction of coordination (not communication) costs at run-time. In fact, for many applications the communication costs of handling the control flow are negligible compared to the costs of transferring case data.

5.3. Extended case transfer and the use of template repositories

In the case transfer architecture (CTA), every business partner uses the same process definition. The sharing of a common business process brings many advantages, but it also complicates things. In some situations, one business partner can force the other business partners to use their business process (hub-and-spoke model). For example, the insurance company can force the agencies to use the WF-net shown in Figure 10. However, in many situations this is not realistic. For example, if the process is new to all partners, there will be a common design effort. Note that this is not specific for the CTA architecture. Inter- and intra-organizational E-commerce require the formulation of (trade) procedures by definition. The only difference is that a business partner sees parts of the process that would normally be hidden behind the organizational boundaries of the other partners. To support the design of common process definitions, central repositories with *process templates* could be useful. The templates are tailored towards electronic markets and may be based on best practices or the characteristics of the organizations involved. At the moment, the use of such repositories with process templates may appear to be unrealistic. However, the use of process templates is already supported by today's leading Enterprise Resource Planning (ERP) systems (e.g., SAP R/3 and BaanERP). These systems typically offer references models containing process templates to configure the enterprise information system. Therefore, the use of templates for interorganizational E-commerce is not as impractical as it may seem. Template processes can be seen as trustworthy trade procedures which are offered by organizations specializing in E-commerce support. Note that this agrees with recent efforts of several international organizations to standardize trade procedures. For example, the International Chamber of Commerce (ICC) has proposed the Uniform Customs and Practices for Documentary Credit Procedures. In [14,34,35], Lee and Bons describe the use of electronic trade scenarios to allow for open E-commerce. Business partners that have no prior trading relationship, retrieve trade scenarios from a publicly accessible library (typically maintained by an independent international organization) to do business. In [37], Malone et al. advocate the use of templates for modeling business processes and capturing domain knowledge.

The business partners involved in an interorganizational workflow process will typically customize the template according to their specific needs. All parties have to agree on the process they are going to use. Note that not only the process needs to be specified, also issues such as rates and timing issues need to be resolved. The moment there is consensus, the workflow process definition is distributed over all partners.

Sometimes it is inevitable that there are local variations of the workflow process. For example, the agencies extend the process shown in Figure 10 with extra tasks for strictly local purposes. Even within the same company, such variations exist. For example, the Customs department of a country has typically many offices located at the border, airports and harbors. From a legal point of view, these offices operate in the same way. However, in practice, an office located at an airport uses a slightly different procedure than for example an office located at a harbor. An architecture that does not allow for such variations is too restrictive. Therefore, we propose the *extended case transfer architecture* (ECTA). The ECTA allows all partners to extend the process in such a way that the local process inherits all the properties of the common process. Therefore, the definition of interorganizational processes in the ECTA is more complex than the definition of CT-IOWF.

Definition 4 (ECT-IOWF)

An *Interorganizational Workflow with Extended Case Transfers (ECT-IOWF)* is a tuple $ECT\text{-}IOWF = (B, WF, WF_1, WF_2, ..., WF_n, T, task\_map, spe_1, spe_2, ..., spe_n, gen_1, gen_2, ..., gen_n)$, where:

1. B is the set of business partners,
2. WF is a WF-net describing the common interorganizational workflow,

3. for each $k \in \{1,...,n\}$, WF_k is the local WF-net (i.e., the common WF-net extended with local features) of business partner *k*,
4. *T* is the set of tasks in the WF-net *WF*,
5. *task_map* is a function mapping business partners to sets of tasks, i.e., $task\_map \in B \rightarrow 2^T$. For each business partner $b \in B$, *task_map(b)* is the set of tasks *b* can execute,
6. $spe_1, spe_2, ..., spe_n$ are functions mapping states of the common workflow (*WF*) onto the local WF-net in case of a transfer,
7. $gen_1, gen_2, ..., gen_n$ are functions mapping states of the local WF-nets onto states of the common WF-net.

Suppose a case needs to be transferred from business partner *i* to business partner *j*. Let s_i be the state of the case, i.e., a state of WF_i. Using the function gen_i the state of the case is mapped onto a state of the common process, i.e., $gen_i(s_i)$. Then function spe_j is used to create the appropriate state of WF_j, i.e., the resulting state is $spe_j(gen_i(s_i))$. Compared to the CTA, ECTA uses advanced concepts which are difficult to implement. It is far from trivial to define appropriate specialization and generalization functions. Moreover, dynamically changing the state of a case during transfer may cause many problems. These problems are similar to the problems encountered when dealing with *adaptive workflows* [6,15], i.e., workflows where the process definition is changed on-the-fly.
The generalization and specialization functions in Definition 4, suggest that *inheritance* notions could be useful. The common workflow can be seen as a *superclass* and the local workflows can be seen as *subclasses* of this superclass. Fortunately, such inheritance notions are available. In [5,11] several notions of inheritance have been defined. These notions are tailored for workflow processes (i.e., workflows specified by WF-nets).
Consider two workflow processes *x* and *y*. When is *x* a subclass of *y*? *x* is a subclass of superclass *y* if *x* inherits certain features of *y*. Intuitively, one could say that *x* is a subclass of *y* if and only if *x* can do what *y* can do. Clearly, all tasks present in *y* should also be present in *x*. Moreover, *x* will typically add new tasks. Therefore, it is reasonable to demand that *x* can do what *y* can do with respect to the tasks present in *y*. In fact, the behavior with respect to the existing tasks should be identical. In [5,11] we have identified four different notions of inheritance: *protocol inheritance*, *projection inheritance*, *protocol/projection inheritance*, and *life-cycle inheritance*. Protocol/projection inheritance is the most restrictive form of inheritance. If *x* is a subclass of *y* with respect to protocol/projection inheritance, then *x* is a subclass of *y* with respect to protocol inheritance *and* projection inheritance. Life-cycle inheritance is the most liberal form of inheritance, i.e., protocol and/or projection inheritance implies life-cycle inheritance. The notion of *projection inheritance* is based on *abstraction*: *If it is not possible to distinguish x and y when arbitrary tasks of x are executed, but when only the effects of tasks that are also present in y are considered, then x is a subclass of y with respect to projection inheritance.* For distinguishing *x* and *y* under projection inheritance we only consider the tasks present in both nets (i.e., in *y*). All other tasks in *x* are renamed to τ. One can think of these tasks as silent, internal, or not observable. Since *branching bisimulation* [11] is used as an equivalence notion, we abstract from transitions with a τ label, i.e., for deciding whether *x* is a subclass of *y* only the tasks with a label different from τ are considered. The behavior with respect to these tasks is called the *observable behavior*. Added tasks (i.e., tasks present in *x* but not in *y*) can be executed but are not observable by the outside world, i.e., projection inheritance conforms to *hiding* or *abstracting* from tasks new in *x*. The notion of *protocol inheritance* is based on *encapsulation* (i.e., blocking of new tasks): *If it is not possible to distinguish x and y when only tasks of x that are also present in y are executed, then x is a subclass of y.* For distinguishing *x* and *y* under protocol inheritance all tasks present in *x* but not in *y* are blocked. The new tasks are simply disallowed to be executed.
Based on these inheritance notions, several *inheritance-preserving transformation rules* have been defined [5,11]. Each of the rules corresponds to a design construct which is often used in practice, namely choice, parallel composition, sequential composition, and iteration. In addition, we have found a complete set of rules to map any case in any state from a subclass to a superclass and vice versa. These transfer rules have been applied in the domain of adaptive workflow [6] and can also be applied to map a local workflow in the ECTA onto the common workflow and vice versa.. Therefore, as long as the designer sticks to the inheritance-preserving transformation rules, the generalization and specialization functions

can be calculated automatically! Moreover, the rules preserve to some extent syntactic and semantic correctness. Clearly, these inheritance notions, transformation rules, and transfer rules are crucial for the applicability of the ECTA. However, a detailed discussion on workflow inheritance is beyond the scope of this paper.

To summarize, the following steps are required to enact a new interorganizational workflow using the ECTA:

1. Select process template.
2. Modify the process template based on the needs of the market and characteristics of the business partners involved.
3. Specify inheritance constraints (i.e., define what kind of local extensions are allowed).
4. Distribute the common interorganizational workflow.
5. Each business partner is allowed to specialize the common workflow while respecting the inheritance constraints.
6. Enact the ECT-IOWF.

If no template is used, step 1 is skipped. In the CTA, steps 3 and 5 are skipped.

5.4. Technical aspects

Thus far, we mainly discussed the mechanisms used in the (E)CTA. Now, we focus on the technical aspects. Today's workflow management have problems dealing with interorganizational workflows, especially when the business partners involved use systems of different vendors. Despite the efforts of the WfMC, real standards are missing to truly support the (E)CTA proposed in this paper. Most of the existing standards abstract from the state of a case, i.e., the workflow process definition only identifies tasks and abstracts from the states in-between tasks. In most commercial workflow management system, there is no possibility to model states explicitly. This does not mean that there are no states, but they remain hidden inside the system. To transfer cases it is vital to access and create state data. Therefore, the workflow management system needs to be extended with a facility to *swap-in* and *swap-out* cases and their corresponding state data. Another issue is the routing of cases from one business partner to another. A *router* component keeps track of the location of a case and manages the transfers between business partners. A message trigger (i.e., an external event such as the receipt of an EDI message) for a specific case can be send to a business partner different from the current location of the case. Therefore, a message handler is needed to re-route triggers.

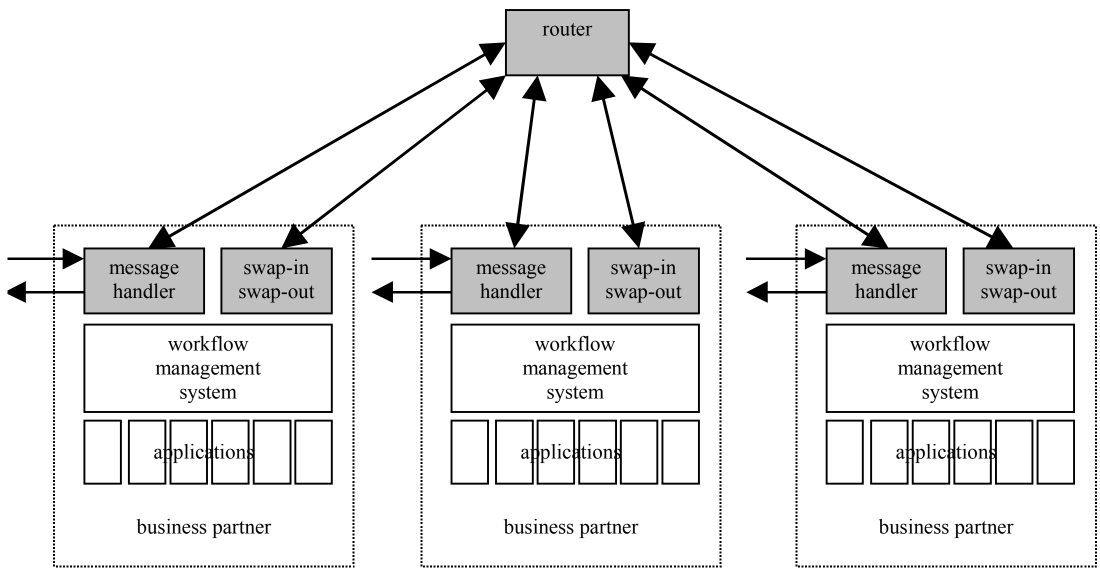

Fig. 12: (E)CTA: (Extended) Case Transfer Architecture.

Figure 12 shows the proposed architecture. The workflow management systems of the different business partners are connected via the central router and the swap-in/swap-out components. Since they do not directly communicate with each other, many interoperability problems are avoided. The message handler re-routes triggers, if the case is not present. The router is connected to the local swap-in/swap-out components and message handlers. Typically, the central router is located at the site of one of the business partners. For the CT-IOWF shown in Figure 10, the insurance company would be the natural location of the router. From a performance point of view, the (E)CTA is a very interesting architecture. Since the case resides at one site, no global control is needed to execute tasks. Therefore, as long as there are no transfers, the architecture offers the best performance possible. The only overhead is the transfer of cases. One might feel uncomfortable about the effect of additional network traffic (i.e., the transfer messages) on the network utilization. However, it turns out that this is not a problem at all. The state can be swapped-in and swapped-out very efficiently. Moreover, the state information is very compact. Consider a workflow with 100 tasks/places and 10 integer variables. To encode one case, only 100*1 + 10*32 = 420 bits (53 bytes) are needed. Compared to the case related data (e.g., electronic documents) the state information is negligible. Hence, the network can easily handle the extra traffic. Since the case resides at one place, the response time to access a case (e.g., to select a work-item in the in-basket) is not affected by the transfer time.

5.5. Sagitta 2000

We have experimented with this architecture in the ambitious Sagitta 2000 project (Dutch Tax/Customs department). The goal of this project was to develop a nationwide information system for the processing of all kinds of Customs declarations. The author was involved in the Sagitta project and the selection of the workflow management systems from 1995 until 1998. This section describes the state of the project at the beginning of 1998 and is based on [7]. The processing of a Customs declaration is a very complex process which is subject to change. Tasks that are needed to handle a declaration are typically related to the registration, checking and control of movements of communal goods. For some types of declarations more than 80 activities can be identified. Figure 13 shows a small part of the workflow process definition (in Dutch). Before starting the implementation of the Sagitta 2000 system the complete workflow process was modeled using a variant of Petri nets (cf. Figure 13 and [7]). Before Sagitta 2000, the handling of Customs declarations was only partly automated. A number of legacy systems supported the management of data related to the processing of declarations. The management of the processes was hardly supported at all. Paper documents form the pivot on which the processing of Customs declarations turns. As a result, the processes were hard to manage and service to the customer is poor. Moreover, the legacy systems were poorly integrated and formed a patchwork which reflected the history of Dutch Customs regulations. The goal of the Sagitta-2000 project was to build a flexible well-integrated information system which also supports and manages the process itself. One of starting points of the Sagitta-2000 project was the separation of information logistics and the implementation of Customs tasks. From the start, it was clear that it would be useful to use a workflow management system for the information logistics. By using a workflow management system as the basis for Sagitta-2000, it should be easy to accommodate the system to the continual changes of the regulations with respect to Customs declarations. Flexibility, maintainability and the ability to integrate applications were the keywords that served as a stimulus for using a workflow management system.

Sagitta-2000 will be a distributed information system, composed of a central system in Apeldoorn (a Dutch city in the eastern part of the Netherlands) and dozens of local systems (one for each Customs office). The architecture of Sagitta-2000 is as presented in Figure 12. The only difference is that the central location is more dominant. Messages which relate to Customs declarations are received by the message handler of the central system in Apeldoorn. The message handler translates these EDI-messages to an in-house format. At any moment a case, i.e., a Customs declaration, is being handled by one of the local platforms or by the central system located in the Apeldoorn. The router sends an incoming message which relates to a specific case (Customs declaration) to the proper location. Customs information about declarations is stored in a central database. About 50% of the cases do not require user interaction and are handled completely automatically by the central system. The other 50% require user interaction and need to be handled at a specific Customs office. Cases which require interaction with a Customs officer are partly handled by the central system and partly by one of the local systems, i.e., a case is transferred from the central system to one of the local systems the moment user interaction is required. The router takes care of these case transfers and the routing of messages. The central system which takes care of the processing of tasks for specific cases is split into two parts: (1) a flow controller (a workflow management system without any user interaction) and (2) a set of applications for the execution of Customs tasks. The flow controller is a system which takes care of the case logistics, i.e., it decides when to execute which task for a specific case using control information about the case. The functionality of the flow controller is comparable to the engine of a workflow management system. The flow controller initiates tasks by starting the proper applications. Note that only the applications can access the Customs information about declarations. The local platform has an architecture which is similar to the central system. Each of the local systems is also split into two parts: (1) a workflow management system and (2) a set of applications for the execution of Customs tasks. The workflow management system takes care of the case logistics by initiating required tasks. In addition, the workflow management system assigns interactive tasks to Customs officers. Note that the applications executed at the local platform which require user interaction cannot be executed by the central system. For performance reasons only, Customs information about declarations handled locally is temporarily stored in a local database. Performance issues are a constant point of attention for Sagitta-2000. The estimated number of declarations per year is more than 10.000.000. Moreover, there will be days on which more than 70.000 declarations have to be handled! The number of Customs officers using Sagitta-2000 will be more than 5000. These figures show that the Sagitta-2000 project is a very ambitious project.

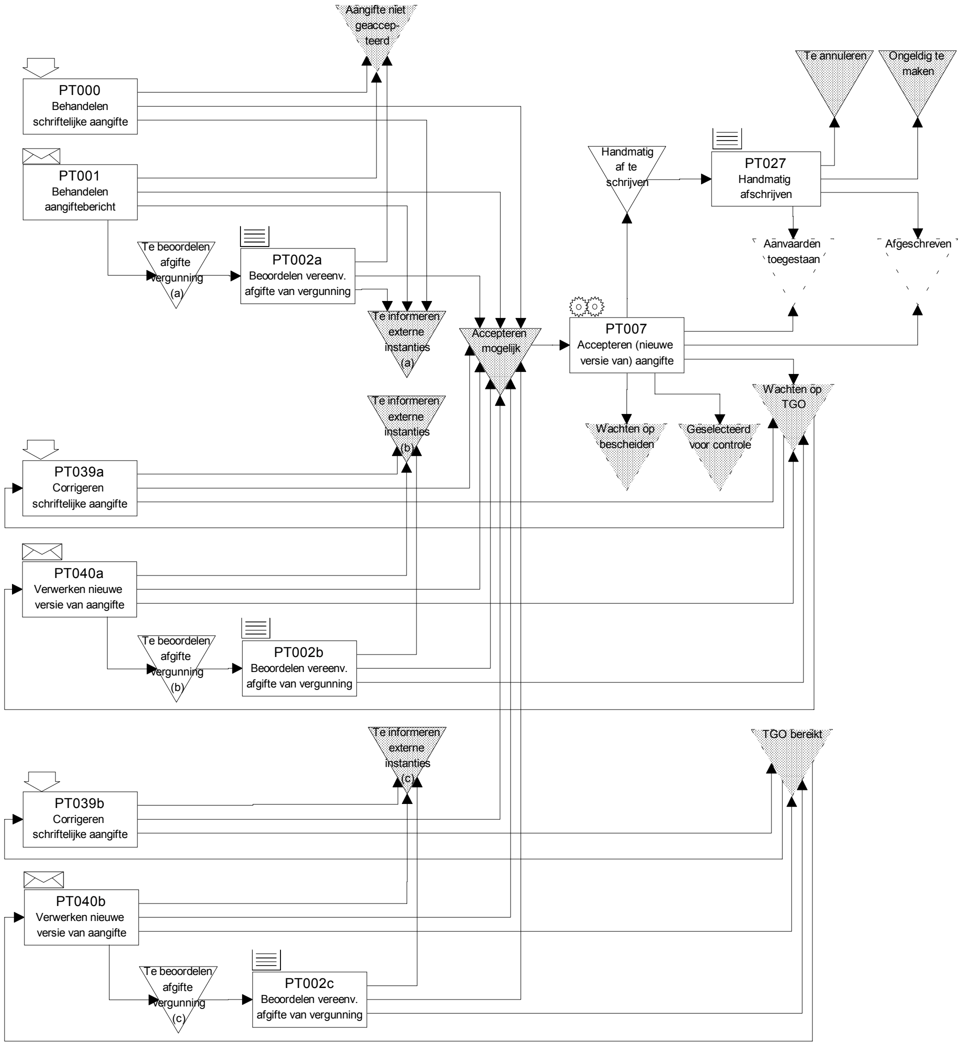

Fig. 13: A fragment of the Sagitta 2000 workflow process definition (in Dutch).

Parallel to the definition of the business processes, the technical infrastructure (hardware/software) has been selected. For the central system an IBM ES9000 mainframe will be used. The DBMS used for the central system is DB2 and the message handler, the router and the Customs applications are being implemented using Cobol, CICS and DB2. Since suitable workflow products are missing for the mainframe, the flow controller is also implemented using Cobol, CICS and DB2. For the local platform a client/server architecture is used. The server is a HP 9000 connected to dozens of client PCs using a Novell network. The server runs under UNIX and the clients are using Windows. The DBMS used for the local platform is Sybase. For the interactive applications the combination of Powerbuilder, C++ and Sybase is used. The Open Server for CICS is used to exchange Customs data between the central DB2 database and the local Sybase database. The exchange of cases will be handled by the IBD, a service offered by the Dutch PTT. After a thorough evaluation, the Petri-net-based workflow management system COSA (Software-Ley/COSA Solutions, [16]) was selected for the local hardware platform. It turned out to be relatively easy to extend COSA with a swap-in/swap-out mechanism. For the central workflow engine of Sagitta-2000 there were no suitable candidates. Therefore, the Dutch Customs Department decided to start building a proprietary workflow engine based on the Petri net formalism.

Sagitta 2000 is an example of an intraorganizational workflow where the CTA architecture turned out to be suitable. Each of the local offices can operate quite autonomous and the performance overhead caused by the transfers is minimal. See [7] for more detailed information on the Sagitta 2000 project.

5.6. Verification

An interorganizational workflow should be free of errors before it is put into production. Unfortunately, today's workflow management systems do not support advanced techniques to verify the correctness of workflow process definitions. These systems typically restrict themselves to a number of (trivial) syntactical checks. Therefore, serious errors such as deadlocks and livelocks may remain undetected. This means that an erroneous workflow may go into production, thus causing dramatic problems for the organization. An erroneous workflow may lead to extra work, legal problems, angry customers, managerial problems, and depressed employees. Therefore, it is important to verify the correctness of a workflow process definition before it becomes operational. In fact, for interorganizational workflows, the costs of putting an erroneous workflow process definition into production are enormous because of the efforts required to repair errors crossing organizational boundaries. In Section 4, we introduced the concept of soundness for WF-nets. In the CTA, the verification problem is very similar to the normal situation. There is just one workflow process definition that needs to be verified. For this purpose, we can use our verification tool Woflan [24]. Figure 14 shows the verification tool Woflan presenting some diagnostics for the CT-IOWF of Figure 10. Note that the CT-IOWF of Figure 10 has been implemented in the COSA workflow management system and is visible in the background. In the ECTA, verification is more involved because each business partner is allowed to modify the local workflow. Again, we refer to the inheritance-preserving transformation rules presented in [5,11]. These are a good starting point for ensuring correctness, i.e., the rules preserve to some extent syntactic and semantic correctness.

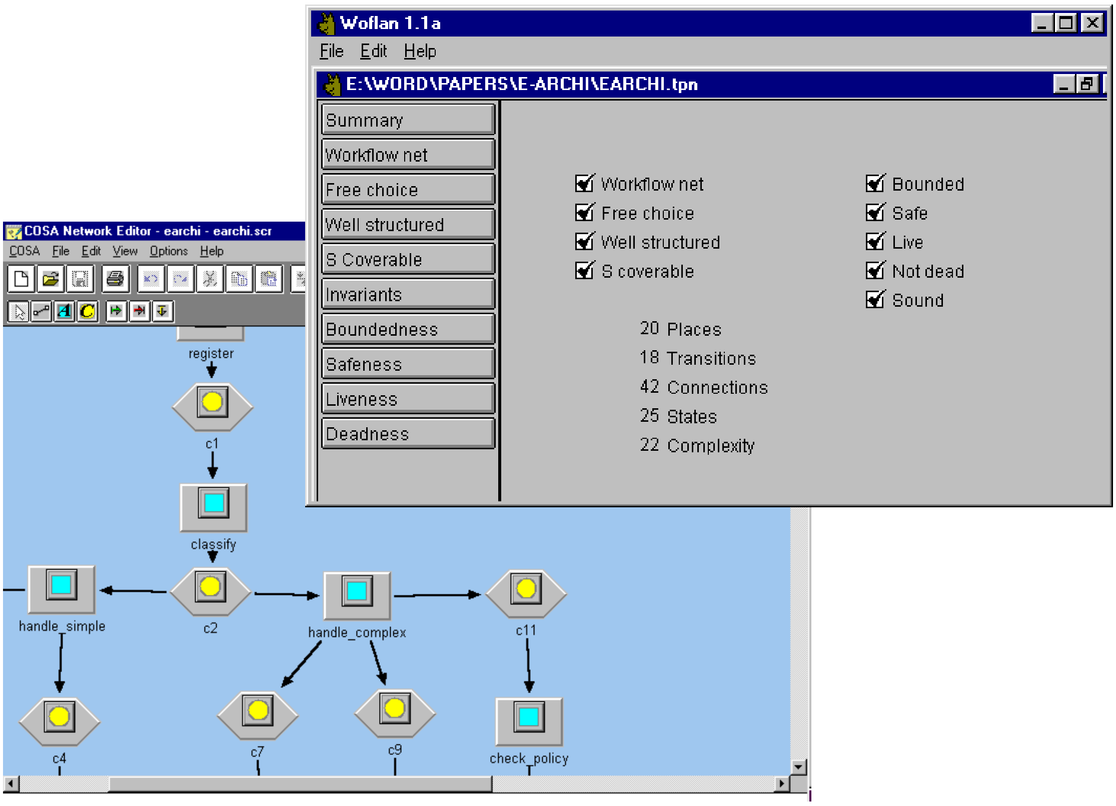

Fig. 14: The CT-IOWF shown in Fig. 10 modeled with COSA and verified for correctness with Woflan.

6. LOOSELY COUPLED

In this section, we consider interorganizational workflows that are essentially a set of loosely coupled workflow processes. These workflows are partitioned horizontally (i.e., the process is cut into disjoint pieces) rather than vertically (see Figure 6). Typically, there are *n* business partners which are involved in one 'global' workflow process. Each of the partners has its own 'local' workflow process. Each local workflow process is private, i.e., the corresponding business partner has full control over the local part of the workflow. However, these local workflow processes need to communicate because they depend on each other for the correct execution of cases. The global workflow process consists of local workflow processes and an interaction structure. There is just one way to interact: asynchronous communication by exchanging messages between local workflow processes.

6.1. Definition of LC-IOWF

To illustrate the concepts and techniques proposed in this paper, we model a workflow which involves four business partners: a customer, a producer and two suppliers. The customer orders a product by sending an order for product *a* to the producer. To produce the ordered product, the producer orders the products needed for production (*b* and *c*). Then the customer is informed that the order has been accepted. Supplier 1 produces products of type *b*, Supplier 2 produces products of type *c*. After both products have been delivered, they are assembled into a product of type *a* which is delivered to the customer. After delivery an invoice is sent which is then paid by the customer.

Figure 15 shows the corresponding loosely coupled interorganizational workflow which consists of four local workflows *Customer, Producer, Supplier_1* and *Supplier_2*. Each of the local workflows is described by a WF-net. However, the input and output places of the subnets which correspond to the producer and the two suppliers have been omitted. For example, the subnet which corresponds to *Supplier_1* can be converted into a true WF-net (see Definition 1) by adding a place *start_S1* which is an input place of *receive_order_b* and a place *end_S1* which is an output place of *send_del_b* (see also Fig. 18).

Definition 5 (LC-IOWF)

A *Loosely Coupled Interorganizational Workflow (LC-IOWF)* is a tuple $CL\text{-}IOWF = (B, WF_1, WF_2, ..., WF_n, P_M, send, receive)$, where:

1. B is a set of *n* business partners,
2. for each $k \in \{1,...,n\}$, WF_k is the local WF-net (i.e., the designated part of the whole workflow) of business partner *k*,
3. P_M is the set of communication places,
4. the nodes of the local WF-nets and P_M are disjoint,
5. Both *send* and *receive* are functions mapping places in P_M onto a subset of the set of transitions present in the local WF-nets, i.e., for any $m \in P_M$: *send(m)* is the set of transitions which produce a token for *m* and *receive(m)* is the set of transitions which consume a token from *m*.

Note that this definition allows for communication places which connect transitions within the same local workflow net. Although it does not make sense to do this, there is no compelling reason to forbid this kind of communication. Also note that each local workflow net has an input place and an output place (see Definition 1). Sometimes there is no need for these places, e.g., if one organization is a subcontractor of another organization, then the workflow of the subcontractor may be initiated by a message (Figure 15 shows such a situation). However, for semantical reasons we assume that there is one source place and one sink place. For a formal definition and formal semantics of a loosely coupled interorganizational workflow the reader is referred to [2,4].

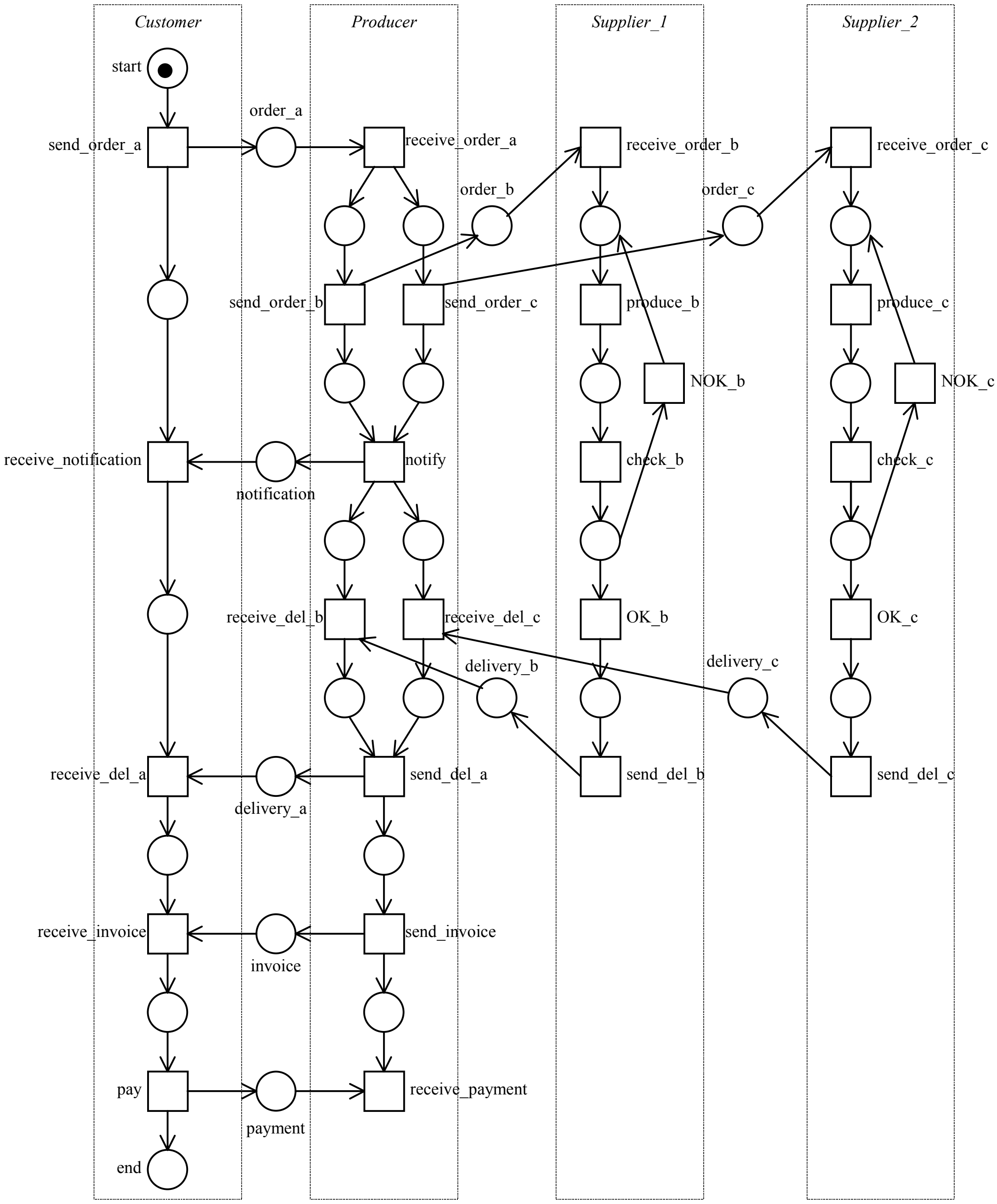

Fig. 15: An example of an LC-IOWF.

6.2. Message sequence charts

The *loosely coupled architecture* (LCA), i.e., the architecture based on LC-IOWF's, uses messages as a mechanism rather than moving cases around. Since every partner has its own private workflow, there is no need to agree upon one common process. However, this does not mean that any local workflow will do! The business partners have to agree on an *interaction protocol*. Other terms for protocol are *trade procedure* or *scenario*. To specify such protocols we use *Message Sequence Charts* (MSC) rather than Petri nets. Since we are only interested in the interface of a local workflow process and not the internal

behavior, it is more convenient to use message sequence charts. In most cases, the design of an interorganizational workflow starts with the specification of the communication structure, i.e., the protocol. Clearly, a description in terms of a Petri net is too detailed to start with.
Figure 16 models the interaction between the four business partners in terms of a message sequence chart. A message sequence chart specifies the messages that are exchanged and the ordering of events associated with sending and receiving messages. In this particular example there are five kinds of messages: orders, notifications, deliveries, invoices and payments.

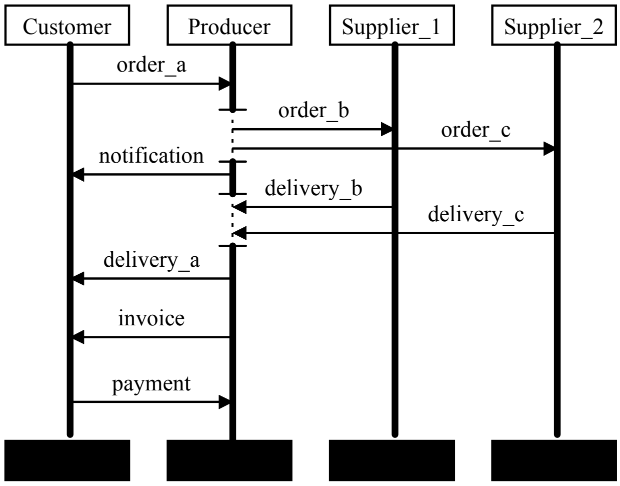

Fig. 16: The interaction protocol specified in terms on an MSC.

Message sequence charts are a widespread graphical language for the visualization of communications between systems/processes [22,28,38]. The representation of message sequence charts is intuitive and focuses on the messages between communication entities. Figure 16 shows a message sequence chart with four business partners also called *instances*. Instances communicate via messages. In Figure 16 there are 9 messages: *order_a, order_b, order_c, delivery_a, delivery_b, delivery_c, notification, invoice, payment*. Each message has a sender and a receiver. For example, the instance *Customer* is the sender of *order_a* and the instance *Producer* is the receiver of *order_a*. Within each instance events are ordered, e.g., the instance *Supplier_1* sends *delivery_b* only after the receipt of *order_b*. The ordering of events is specified by the time axis of an instance which is represented by a vertical line. In a message sequence chart, it is also possible to specify *coregions*. A coregion is represented by a dashed part of the time axis of an instance. Events in a coregion are assumed to be unordered in time. In Figure 16 there are two coregions: the sending of *order_b* and *order_c*, and the receipt of *delivery_b* and *delivery_c* are unordered in time.
In this paper, we consider a variant of message sequence charts as defined in [28]. The basic message sequence chart has been extended with coregions. However, for reasons of simplicity, we do not allow for other customary extensions such as process creation, process termination, timers, and refinement. To describe the semantics of the message sequence charts used in this paper, we use a notation adopted from [38].

Definition 6 (Message Sequence Chart)
A *message sequence chart* is a tuple $MSC=(I,M,from,to,\{\leq_i\}_{i\in I})$:
- I is a finite set of instances (business partners),
- M is a finite set of messages,
- *to* and *from* are functions from M to I,
- for each $i \in I$: $\leq_i$ is a partial order on $\{?m \mid m \in M \text{ and } to(m) = i\} \cup \{!m \mid m \in M \text{ and } from(m) = i\}$.

If m is a message (i.e., $m \in M$), then $!m$ corresponds to the event of sending message m and $?m$ corresponds to the event of receiving message m. For each instance i, $\leq_i$ specifies the ordering of events along the time axis of i. $\leq_i$ is a partial order instead of a total ordering because of the existence of coregions. Consider for example the instance i = *Customer* in Figure 16, *!order_a* $\leq_i$ *?notification, ?notification* $\leq_i$ *?deliver_a, ?deliver_a* $\leq_i$ *?invoice,* and *?invoice* $\leq_i$ *!payment*. Note that the events in the two coregions of the instance *Producer* are unordered, e.g., event *?order_a* is before *!order_b* and *!order_c* but there is no causal relation between *!order_b* and *!order_c* (i.e., not *!order_b* $\leq_i$ *!order_c* nor *!order_c* $\leq_i$ *!order_b*).

Definition 7 ($\leq^{MSC}$)
Let $MSC=(I,M,from,to,\{\leq_i\}_{i\in I})$ be a message sequence chart:

- $\leq^{inst} = \bigcup_{i\in I}$,
- $\leq^{io} = \{(!m,?m) \mid m \in M\}$, and
- $\leq^{MSC} = (\leq^{inst} \cup \leq^{io})^{+}$.

$\leq^{io}$ is a partial order which reflects the production before consumption principle. $\leq^{MSC}$ is the transitive closure of (1) the partial orders within the instances ($\leq^{inst}$) and (2) the partial order between the production and consumption of messages ($\leq^{io}$). Consider for example the message sequence chart shown in Figure 16: *!order_a* $\leq^{MSC}$ *?order_b,* *!order_b* $\leq^{MSC}$ *?notification, !notification* $\leq^{MSC}$ *?invoice, !delivery_c* $\leq^{MSC}$ *?payment,* and *!invoice* $\leq^{MSC}$ *!payment.*
A message sequence chart *MSC* is inconsistent iff $\leq^{MSC}$ does not define a partial order. In this case, the message sequence chart contains a deadlock due to cyclic dependencies. In the remainder we assume that the message sequence charts are consistent. The message sequence chart shown in Figure 16 is consistent. Note that most message sequence charts define a partial order which is not a total order. For example, there is no causal relation between the events *!notification* and *!delivery_c.*
In this subsection, we elaborated on the semantics of message sequence charts to show that they have formal semantics which allow for all kinds of analysis. Such analysis techniques are essential for the formal verification of interorganizational workflows. In practice, there is still a huge gap between the informal diagrams used to agree on a protocol and the actual implementation of the interorganizational workflow. As a result, many coordination efforts are needed to make the workflow run smoothly.
To summarize, the following steps are required to enact a new interorganizational workflow using the LCA:

1. Specify the protocol (e.g., a trade procedure in terms of message sequence charts).
2. For each business partner, design the local workflow such that it is consistent with the protocol.
3. Enact the LC-IOWF.

Specifying the protocol is an iterative process, i.e., it is subject to changes while the business partners are designing the local workflows. Therefore, the first two steps may take a long time and require many coordination efforts. To speed up these two steps, the use of high-quality templates is attractive. Templates similar to the ones described in Section 5.3 can be used as a starting point for configuring the interorganizational workflow. For the LCA, the workflow process definition needs to be partitioned into subprocesses (one for each business partner). The central repository containing templates could suggest characteristic partitions to facilitate the distribution of the workflow over the business partners involved.

6.3. *Technical aspects*

The LCA is more traditional than the (E)CTA. The business partners do not have to agree on one common workflow. Existing technology can be used to exchange the messages and to manage local workflows. Figure 17 shows the LCA. Each business partner has a *message handler* to exchange the messages with other partners. Note that the message handler needs to know how to send messages to other partners. Moreover, the message handler should be able to interpret incoming messages and route them to the appropriate spot in the workflow management system. In addition to messages between business partners, there are message triggers generated by external actors (not being a business partner). These are the normal message triggers and are handled directly by the workflow management system. Note that in the LCA there is no need to re-route these triggers to other business partners: these triggers are only relevant for the receiving business partner.

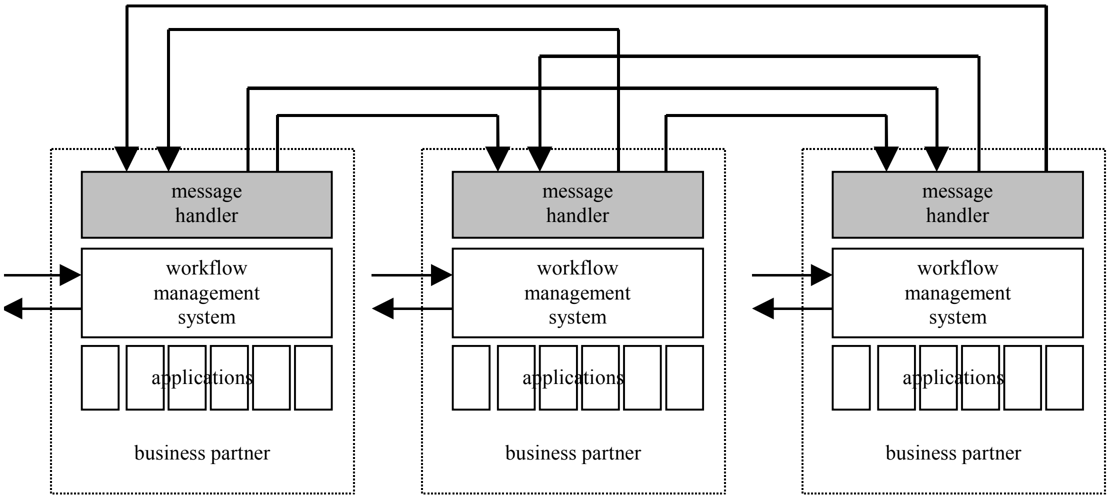

Fig. 17: The Loosely Coupled Architecture (LCA).

The architecture shown in Figure 17 assumes that each of the business partners uses a workflow management system. In [34,35], other architectures for LC-IOWF are explored. Instead of using workflow management systems and message handlers, special purpose components are used. The InterProcs system (cf. [34,35]) can be used for designing trade scenarios and automatically generating a run-time environment using the Internet to exchange messages between local Java applications. Another approach is the use of a central repository (see Section 5.3) where trade scenarios and software components are stored. The business partners involved in the LC-IOWF download and configure the components. Developments in the middleware marketplace (CORBA, Java applets, Enterprise JavaBeans, IIOP, etc.) indicate that in the near future it will be possible to provide such components. Unfortunately, the latter approach is only useful for highly standardized trade scenarios: the components need to be predefined by the organization maintaining the central repository.

6.4. Verification

Errors in the design of an LC-IOWF are difficult to trace and may have serious consequences. Moreover, the problems resulting from an incorrect LC-IOWF are difficult to repair because of the distributed control. We use Fig. 18 to illustrate potential errors. Compared to the interorganizational workflow shown in Figure 15 four alterations have been made:

1. *Alteration 1*: if the result of the check by *Supplier_2* is negative, then task *send_del_c* is skipped.
2. *Alteration 2*: task *NOK_b* in *Supplier_1* has an extra input place.
3. *Alteration 3*: *receive_del_b* and *receive_del_c* in *Producer* are executed sequentially instead of in parallel.
4. *Alteration 4*: *send_del_a* and *send_invoice* in Producer are executed in parallel.

These alterations are used to introduce two notions of correctness for LC-IOWF: *soundness* and *consistency*.

Recall that a WF-net is sound if and only if it satisfies the following requirement. For any case, the procedure will terminate eventually and the moment the procedure terminates there is a token in the output place and all the other places are empty. Moreover, there should be no dead tasks, i.e., it should be possible to execute an arbitrary task by following the appropriate route though the WF-net. A loosely coupled interorganizational workflow is *locally sound* if and only if each of the local WF-nets in isolation is sound. Note that the WF-net which corresponds to *Supplier_1* in Fig. 18 is not sound because *NOK_b* is dead. The other three WF-nets in isolation are sound.

Given an interorganizational workflow (i.e., a LC-IOWF), it is possible to construct an extended net by adding an input place connected to a transition which puts a token in each of the input places of the local WF-net and an additional output place connected to a transition which consumes a token from each of the output places of the local workflows. Note that the extended net is a WF-net (see [2,6]). An interorganizational workflow is *globally sound* if the extended net is sound. Clearly, alteration 2 invalidates global soundness. Alteration 1 also invalidates global soundness (if *NOK_c* fires, then there is a deadlock in *Producer*) although the local WF-net (*Supplier_2*) is sound.

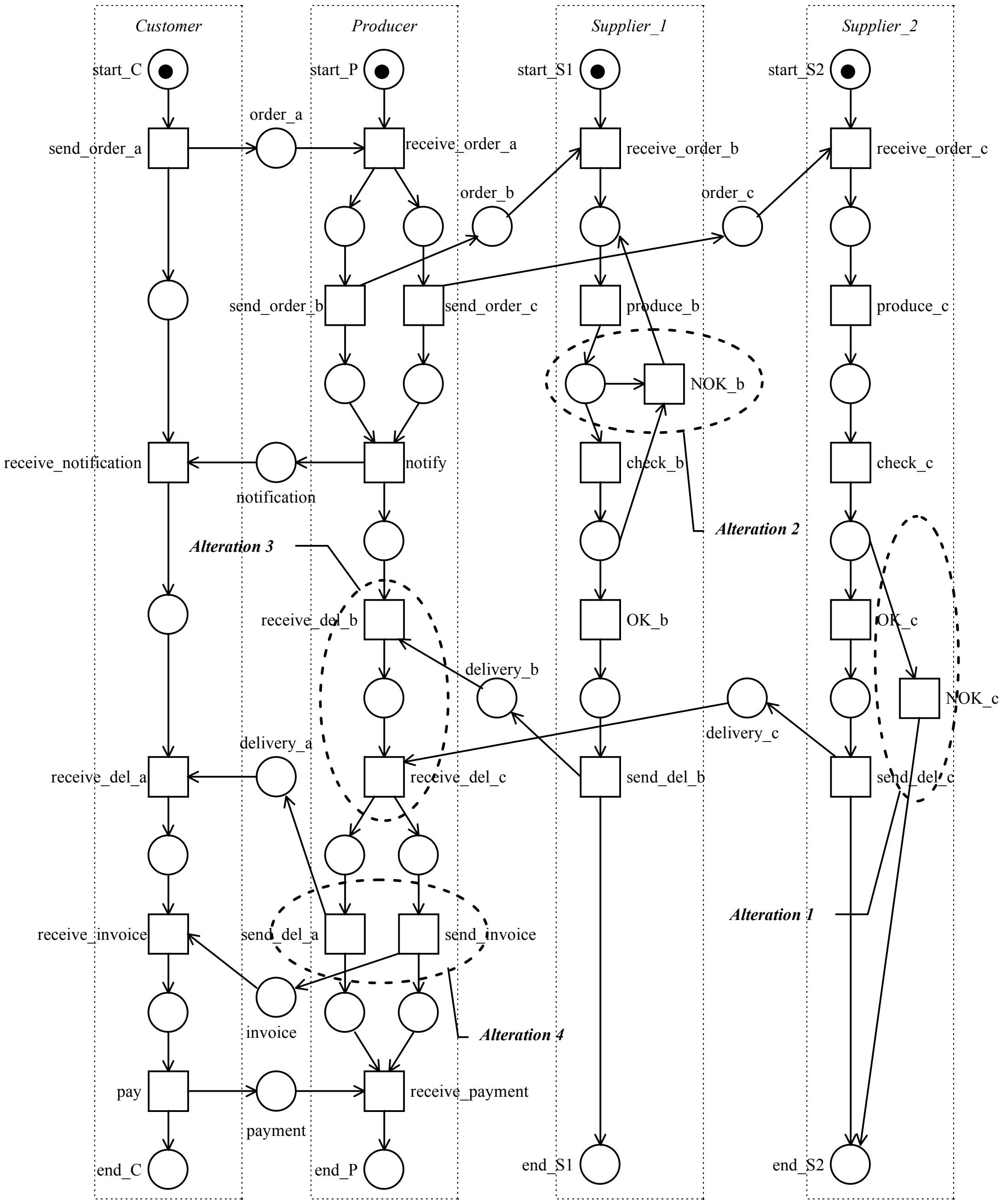

Fig. 18: An alternative interorganizational workflow.

It is also possible to have an interorganizational workflow which is globally sound but not locally and vice-versa. Therefore, we demand an interorganizational workflow to be both globally sound and locally sound. The interorganizational workflow shown in Figure 15 has this property. Moreover, alterations 3 and 4 in
Fig. 18 do not invalidate soundness.
In [3], a decision procedure is given to decide soundness (see Section 4). This procedure uses state-of-the-art Petri-net-based analysis techniques. In fact it uses the fact that soundness corresponds to two well-known properties: liveness and boundedness [1,3]. For an organizational workflow composed of n local workflows we need to prove liveness and boundedness for $n+1$ WF-nets (n local WF-nets and the extended net). We have developed a tool, named *Woflan* [1,24,48], to verify soundness. Woflan can interface with several workflow products (e.g., COSA, Staffware, METEOR, and Protos) and can be downloaded via the Word-Wide-Web (http://www.win.tue.nl/~woflan/).
Message sequence charts can be used to specify the interaction between loosely coupled workflow processes. These message sequence charts serve as a starting point for the design of complex interorganizational workflows. The interorganizational workflow should be designed in such a way that it is consistent with the message sequence charts, i.e., the message sequence charts can be seen as a partial specification of an interorganizational workflow. Therefore, it is interesting to be able to decide whether the implementation (interorganizational workflow) meets the specification (message sequence charts). In [2] we describe a technique to check the consistency between the interorganizational workflow modeled as an LC-IOWF and the message sequence chart(s) describing the protocol. Consistency can be defined as follows. Any trade scenario described by a message sequence chart should be supported by the LC-IOWF. Any trade scenario possible according to the LC-IOWF should correspond to a message sequence chart.

7. EVALUATION OF BOTH ARCHITECTURES

In this paper, we described two architectures. As indicated by Fig. 6, the essential difference is the way they partition the workflow. The (E)CTA architecture uses a vertical partitioning, i.e., the cases are distributed over several business partners but the process is not cut into pieces. The LCA architecture uses a horizontal partitioning, i.e., the process is cut into pieces and cases are not allocated to specific business partners (in fact several business partners may be working on the same case at the same moment). In this section, we compare both architectures. Table 1 summarizes the results.

criterion	(E)CTA	LCA
Coordination efforts at design-time	high	moderate
Coordination efforts at run-time	moderate	high
System performance (only the workflow aspect)	good	good
Transparency of the whole workflow	good	bad
Complexity of the whole workflow (relative)	moderate	high
Complexity of the local workflow (relative)	high	moderate
Degree of parallelism	moderate	high
Management information	good	limited
Flexibility (changing the process)	low	moderate
Flexibility (work load balancing)	high	low
Flexibility (changing allocation rules)	high	low

Table 1: Comparison of both architectures.

For the (E)CTA, the coordination efforts to design a process are quite high because the business partners have to agree on one common process. However, understanding the whole process reduces the coordination efforts at run-time. Transferring a case from one system to another system can be implemented very efficiently because the amount of data to be transferred is minimal. Moreover, since the case resides at a local system, the response time is limited. Therefore, if the case transfer mechanism is implemented properly, the overall performance with respect to the workflow aspect is quite good. Since there is one common process, the whole process is transparent to all business partners and it is possible to obtain management information reflecting the state/performance of the whole process. Moreover, because the business partners have to agree on one common process, the complexity of the overall process is likely to be moderate (there is no need to support the battle of the forms). However, from the viewpoint of one partner, the local process can appear to be more complex. The degree of parallelism is reduced by the (E)CTA, because just one business partner can work on the same case at the same time. Whether the reduced degree of parallelism effects the overall performance of the business process depends on several factors. Experience shows that the effect is minimal if there are many cases in the process because then cases are competing for resources rather than waiting to be synchronized. Changing the process is difficult in the CTA because all business partners have to agree on a new common process. The ECTA solves this problem partially because it allows for local variations. However, if change affects the interaction structure (i.e., the business-to-business protocol), all the partners involved in the change have to adapt their local processes. One of the most attractive features is that workload balancing can be implemented by changing the transfer policy, i.e., functional migration and migration for performance reasons go hand-in-hand. Note that capacity can be reallocated by simply moving cases from one partner to another.

In the LCA, the coordination efforts at design time are moderate because the business partners only have to agree on the protocol. However, there are more coordination efforts at run-time because there is no clear understanding of the whole workflow. This is a direct result of the lack of transparency. In fact, the whole workflow becomes more complex because of the decoupling (battle of the forms). The local workflow appears to be less complex because it only considers the tasks relevant for the business partner. The degree of parallelism is unrestricted by the LCA: multiple partners can work on the same case at the same time. The lack of transparency complicates the collection of good management information. Large parts of the overall workflow process remain hidden for an individual business partner. The partitioning of the process makes it relatively easy to change the local workflow (as long as the protocol/interface does not change). Reallocating work from one partner to another is very difficult. Therefore, there are hardly any possibilities to balance the workload over the business partners involved.

The results presented in Table 1 show that none of the two architectures is superior to the other. (E)CTA has some very attractive features but requires all business partners to agree on a process. Therefore, it is more suitable for intraorganizational E-commerce than business-to-business E-commerce. LCA is applicable for both intraorganizational E-commerce and business-to-business E-commerce. However, the lack of transparency, limited management information, and reduced flexibility are complicating factors. For both architectures the availability of repositories with templates of interorganizational workflows (e.g., trade scenarios [14,34,35]), is vital. Such repositories (typically maintained by international organizations for different sectors of industry) publicly offer workflow process models which can be customized for specific E-commerce applications.

8. RELATED WORK

Petri nets have been proposed for modeling workflow process definitions long before the term "workflow management" was coined and workflow management systems became readily available. Consider for example the work on Information Control Nets, a variant of the classical Petri nets, in the late seventies [19,20]. For the reader interested in the application of Petri nets to workflow management, we refer to two workshops on the topic [8,40] and an elaborate paper on workflow modeling using Petri nets [1].

Although there are some commercial workflow management systems based on Petri nets (e.g., COSA and Income), most workflow management systems use a vendor-specific diagramming technique. These diagramming techniques typically abstract from the explicit representation of states and their expressive power is comparable to free-choice Petri nets. Nevertheless, the flow-oriented nature of these diagramming techniques is close to Petri nets and therefore Petri nets constitute a good formal basis close to today's workflow management systems. Only a few other formal models have been proposed for specifying workflow processes, e.g., state charts [42,52] and task structures [26].

At several places in this paper we stressed the need of verification. Only a few papers in literature focus on the verification of workflow process definitions. In [26] some verification issues have been examined and the complexity of selected correctness issues has been identified. In [3] and [9] concrete verification procedures based on Petri nets have been proposed. In [43] a reduction technique has been proposed. This reduction technique uses a correctness criterion which corresponds to soundness and the class of workflow processes considered corresponds to the class of free-choice Petri nets.

In E-commerce environments exceptions are difficult to handle [10]. Consider for example the following scenario. Amazon.com offers the following service: Within 30 days of receipt of the order a customer may return purchased books and get a full refund. However, Amazon.com typically bought the books purchased by the customer from various other companies. This means that handling a returned book requires a lot of interorganizational coordination efforts (e.g., Where to store the book and what about the payments?). Canceling an order is typically very hard to handle if it needs to be propagated to other companies. There is a lot of literature on exception handling in the context of workflow management and extended transaction models [21,23,45]. Exceptions are considered to be unexpected undesirable events. Most of the results concentrate on failures of applications and networks. Deliberate change, human failures, and evolving environments are hardly taken into account. Exception handling mechanisms are primarily used to separate the failure semantics from the process logic and thus facilitate the design of readable, comprehensible workflow models. Tasks or sub-processes which fail, return an exception which is interpreted by an exception handler. The exception handler either takes action (e.g., compensate and restart or rollback) or propagates the exception to a higher level. Note that in an E-commerce setting it is typically not possible to rollback or compensate transactions. Further research is needed to see in which way exception handling should be incorporated.

Most of today's commercial workflow systems use a centralized enactment service. Therefore, many of the research prototypes such as MENTOR (University of Saarland at Saarbrucken), METEOR (University of Georgia), MOBILE (University of Erlangen), Panta Rhei (University of Klagenfurt), and WASA (University of Muenster) focus on distribution aspects. The MENTOR system [49] is based on state charts which are partitioned into fragments using the approach described in [42,52]. These fragements are distributed under the supervision of Tuxedo, a TP monitor. The METEOR system [44] is entirely based on CORBA to provide a platform independent and reliable environment. It also supports interoperability mechanisms like SWAP and JFLOW. Moreover, the METEOR3 model introduces the notion of *foreign* task vs. *native* tasks. A foreign task refers to a task whose realization (implementation) is unknown to workflow designer, whereas the implementation details are known to the workflow designer for a native task. Another important feature for E-commerce are *channels* (also called sink nodes) that are used to specify communication or synchronization between two independent workflows [31]. An interesting project focussing on the application of workflow technology to E-commerce is the WIDE Project (http://dis.sema.es/projects/WIDE/). To goal of this project is to enable interorganizational workflows across multiple platforms by linking geographically separated applications including IBM's MQSeries Workflow, SAP R/3, Opera, and Structware [10]. Each of the tools/projects mentioned in this section uses a horizontal partitioning of the workflow and none of the tools supports the separation of the business-to-business protocol from the actual workflow (i.e., having a separate model for specifying the global agreements which at all times should be consistent with the actual workflow).

Compared to existing work, the contribution of this paper is threefold. First of all, the paper provides an overview of the various forms of interoperability in the context of E-commerce. Second, the paper introduces an architecture based on vertical partitioning rather than horizontal partitioning and this architecture has been put to work in the Sagitta project. Finally, for loosely coupled workflow process (i.e., a horizontal partitioning), an approach to separate the business-to-business protocol (using message sequence charts) from the actual workflow (using a Petri-net-based model) is presented.

9. CONCLUSION

Speed and distribution in every aspect of most business and organization undertaking characterize the new millennium. Organizations are challenged to bring ideas and concepts to products and services in an ever-increasing pace. Companies distributed by space, time and capabilities come together to deliver products and solutions for which there is any need in the global marketplace. The trends for virtual corporations and e-commerce, and increasing global networking of economies are real and will accelerate. Information systems will play an increasingly critical role in providing competitive edge to organizations in the networked economy. So far, lion's share of the attention in Information Systems has gone to data. However, currently there is a shift from information and knowledge to processes. Today's workflow management systems are able to support workflows inside one organization but have problems dealing with workflows crossing organizational boundaries. Therefore, we explored two architectures for supporting these interorganizational workflows.

The *(extended) case transfer architecture* (CTA) uses a vertical decomposition of the workflow, i.e., cases (workflow instances) are partitioned over the business partners involved. The *loosely coupled architecture* (LCA) uses a horizontal decomposition where the process itself is partitioned, i.e., each business partner has a private workflow process which is connected to the workflow processes of some of the other partners. In this paper we compared both architectures and identified criteria to select a suitable architecture.

For both architectures flexibility and locality are of prime concern. The ECTA deploys advanced inheritance notions to allow for local variants (subclasses) of the common process. These local workflows may have any structure and change continuously as long as they remain a subclass of the process all business partners agreed on. The LCA uses message sequence charts to specify the protocol. Each partner may design and adapt any local workflow process, as long as it is consistent with protocol all business partners agreed on.

We also emphasized the importance of suitable verification techniques. Interorganizational workflows (CT-IOWF, ECT-IOWF, and LC-IOWF) tend to be very complex. Since the design is distributed over multiple organizations (there is no chief architect), many errors are only detected at runtime. These errors are typically very costly because the repair has to be coordinated between autonomous business partners. By using rigorous specification methods and highly selective verification techniques, these costs can be reduced.

Acknowledgements – The author of this paper would like to thank the reviewers for their constructive comments and the Sagitta-team, in particular Peter van der Toorn and Jaap Rigter, for contributing to this work.

REFERENCES

1. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21-66, 1998.
2. W.M.P. van der Aalst. Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets. *Systems Analysis - Modelling - Simulation*, 34(3):335--367, 1999.
3. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407-426. Springer-Verlag, Berlin, 1997.
4. W.M.P. van der Aalst. Modeling and Analyzing Interorganizational Workflows. In L. Lavagno and W. Reisig, editors, *Proceedings of the International Conference on Application of Concurrency to System Design (CSD'98)*, pages 1-15, IEEE Computer Society Press, 1998.
5. W.M.P. van der Aalst and T. Basten. Life-cycle Inheritance: A Petri-net-based approach. In P. Azéma and G. Balbo, editors, Application and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science, pages 62-81. Springer, Berlin, Germany, 1997.
6. W.M.P. van der Aalst, T. Basten, H.M.W. Verbeek, P.A.C. Verkoulen, and M. Voorhoeve. Adaptive Workflow: An Approach Based on Inheritance. In M. Ibrahim and B. Drabble, editors, *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 36-45, Stockholm, Sweden, August 1999.
7. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Modellen, Methoden en Systemen (in Dutch)*. Academic Service, Schoonhoven, 1997.
8. W.M.P. van der Aalst, G. De Michelis, and C.A. Ellis, editors. *Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98*). UNINOVA, Lisbon, June 1998.

9. N.R. Adam, V. Atluri, and W. Huang. Modeling and Analysis of Workflows using Petri Nets. *Journal of Intelligent Information Systems*, 10: 131-158, 1998.
10. G. Alonso, C. Hagen, and A. Lazcano. Processes in Electronic Commerce. In Proceedings of the ICDCS Workshop on Electronic Commerce and Web-Based Applications. Austin, Texas, USA, June 1999.
11. T. Basten. In Terms of Nets: System Design with Petri Nets and Process Algebra. PhD Thesis. Eindhoven University of Technology, Department of Computing Science, Eindhoven, the Netherlands, 1998.
12. T. Bauer and P. Dadam. A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration. In Proceedings of the Second IFCIS Conference on Cooperative Information Systems (CoopIS'97), Kiawah Island, South Carolina, USA, pp. 99-108, 1997.
13. R. Benjamin and R. Wigand. Electronic markets and virtual value chains on the information superhighway, *Sloan Management Review,* 62-72, 1995.
14. R.W.H. Bons, R.M. Lee, and R.W. Wagenaar. Designing trustworthy interorganizational trade procedures for open electronic commerce, *International Journal of Electronic Commerce,* 2(3), 61-83, 1998.
15. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. *Data and Knowledge Engineering*, 24(3):211-238, 1998.
16. COSA Solutions. COSA 2.0 User Manual. COSA Solutions, Pullheim, Germany, 1998.
17. J. Desel and J. Esparza. *Free choice Petri nets*, volume 40 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge, 1995.
18. A. Dutta. The physical infrastructure for electronic commerce in developing nations: historical trends and the impact of privatization. *International Journal of Electronic Commerce,* 2(1), 61-83, 1997.
19. C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems, pages 225-240, Boulder, Colorado, 1979. ACM Press.
20. C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1-16. Springer-Verlag, Berlin, 1993.
21. D. Georgakopoulos, M. Hornick, and A. Sheth, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, *Distributed and Parallel Databases*, 3:119-153, 1995.
22. J. Grabowski, P. Graubmann, and E. Rudolph. Towards a Petri net based sematics definition for Message Sequence Charts. In O. Faergemand and A. Sarma, editors, *SDL'93 - Using Objects, Proceedings of the sixth SDL Forum*, pages 179-190. North-Holland, 1993.
23. C. Hagen and G. Alonso. Flexible Exception Handling in the OPERA Process Support System. In proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS), Amsterdam, The Netherlands, 1998.
24. D. Hauschildt, H.M.W. Verbeek, and W.M.P. van der Aalst. WOFLAN: a Petri-net-based Workflow Analyzer. Computing Science Reports 97/12, Eindhoven University of Technology, Eindhoven, 1997.
25. K. Hayes and K. Lavery. *Workflow management software: the business opportunity*. Ovum, 1991.
26. A.H.M. ter Hofstede, M.E. Orlowska, and J. Rajapakse. Verification Problems in Conceptual Workflow Specifications. *Data and Knowledge Engineering*, 24(3):239-256, 1998.
27. ISO. *The Open-edi Reference Model*, IS 14662, ISO/IEC JTC1/SC30, International Standards Organization, 1996.
28. ITU-TS. ITU-TS Recomendation Z.120: Message Sequence Chart 1996 (MSC96). Technical report, ITU-TS, Geneva, 1996.
29. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation* International Thomson Computer Press, 1996.
30. R. Kalakota and A.B. Whinston. *Frontiers of Electronic Commerce*. Addison-Wesley, Reading, Massachusetts, 1996.

31. J. Kang, N. Froscher, A. Sheth, K. Kochut and J. Miller: A Multilevel Secure Workflow Management System. *Proceedings of the 11th Conference on Advanced Information Systems Engineering*, Heidelberg, Germany, June 1999.
32. T.M. Koulopoulos. *The Workflow Imperative*. Van Nostrand Reinhold, New York, 1995.
33. P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.
34. R.M. Lee and R.W.H Bons. Soft-Coded Trade Procedures for Open-edi. *Journal of Electronic Commerce*, 1(1): 27-49,1996.
35. R.M. Lee. Distributed Electronic Trade Scenarios: Representation, Design, Prototyping. *Journal of Electronic Commerce*, 3(1), 1999 (to appear).
36. T.W. Malone, R.I. Benjamin, and J. Yates. Electronic markets and electronic hierarchies: effects of information technology on market structure and corporate strategies. *Communications of the ACM,* 30(6), 484-497, 1987.
37. T. Malone, W. Crowston, J. Lee, B. Pentland, and et. al. Tools for inventing organizations: Toward a handbook for organizational processes. *Management Science*, 45(3): 425-443, 1999.
38. S. Mauw and M.A. Reniers. An algebraic semantics of Basic Message Sequence Charts. *The Computer Journal*, 37(4):269-277, 1994.
39. M. Merz, B. Liberman, K. Muller-Jones, and W. Lamersdorf. Interorganisational Workflow Management with Mobile Agents in COSM. In *Proceedings of PAAM96 Conference on the Practical Application of Agents and Multiagent Systems*, 1996.
40. G. De Michelis, C. Ellis, and G. Memmi, editors. *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, Zaragoza, Spain, June 1994.
41. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541-580, April 1989.
42. P. Muth, D. Wodtke, J. Weissenfels, A. Kotz Dittrich, and G. Weikum. From Centralized Workflow Specification to Distributed Workflow Execution. *Journal of Intelligent Information Systems*, 10(2), 1998.
43. W. Sadiq and M.E. Orlowska. Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99), volume 1626 of Lecture Notes in Computer Science, pages 195-209. Springer-Verlag, Berlin, 1999.
44. A. Sheth, K. Kochut, and J. Miller. Large Scale Distributed Information Systems (LSDIS) laboratory, METEOR project page. http://lsdis.cs.uga.edu/proj/meteor/meteor.html.
45. L.A. Streeter et al. How open data networks influence business performance and market structure. *Communications of the ACM*, 39(7):62-73, 1996.
46. D.M. Strong and S.M. Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems*, 13(2):206-233, 1995.
47. A. Tsalgatidou. Selection Criteria for Tools Supporting Business Process Transformation for Electronic Commerce. *Proceedings of EURO-MED NET'98*. Nicosia, Cyprus, pages 244-253, March 1998.
48. E. Verbeek and W.M.P. van der Aalst. Woflan Home Page. http://www.win.tue.nl/~woflan.
49. G. Weikum et al. MENTOR: Middleware for Enterprise-wide Workflow Management. http://www-dbs.cs.uni-sb.de/~mentor/Welcome.html.
50. WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.
51. The White House, *A Framework for Global Electronic Commerce*, http://www.ecommerce.gov/framewrk.htm, 1997.
52. D. Wodtke and G. Weikum. A Formal Foundation for Distributed Workflow Execution Based on State Charts. In proceedings of the 6th International Conference on Database Theory - ICDT '97, Delphi, Greece, 1997.
53. V. Zwass. Electronic commerce: structures and issues, *International Journal of Electronic Commerce,* 1(1), 3-23, 1996.