Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce

W.M.P. van der Aalst[†] w.m.p.v.d.aalst@tm.tue.nl
Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, +31 40 2474295

Abstract

The World-Wide-Web is fast becoming a crucial medium for electronic commerce. Many companies are now involved in on-line retailing of goods and services to consumers through the Web. In some industries, business partnerships are being fostered in order to broaden the scope of their markets. One example is the telecommunications industry, where changes in business structure, spirited by deregulation, have resulted in alliances amongst different communications providers, including new players like the utilities and entertainment providers. In this environment, business processes may involve multiple co-operating entities and, supporting such interorganizational business processes can be achieved through the use of workflow management systems. In this paper, we present an approach for designing interorganizational workflows that supports co-operation of business partners, while preserving the autonomy of the partner organizations.

Keywords: electronic commerce, workflow management, trade procedures, interorganizational workflows

Received March 2000; Revised September 2001

[†] Part of this paper was written while visiting the LSDIS laboratory (Amit Sheth) of the Department of Computer Science of the University of Georgia in Athens (USA). The author would like to thank Kemafor Anyanwu for working on a previous version of this paper (Aalst and Anyanwu, 1999).

1. Introduction

E-commerce refers to the enabling of purchasing and selling of goods and services through a communications network (Benjamin and Wigand, 1995; Dutta, 1997; Kalakota and Whinston, 1996; Malone, Benjamin, and Yates, 1987; Zwass, 1996). The ability to conduct business activities involved in the marketing, finance, manufacturing, selling, and negotiation, electronically, is what E-commerce is all about. One major objective of adopting E-commerce strategies is to reduce costs and improve the efficiency of business processes, by replacing paper business with electronic alternatives.

E-commerce, in its earliest incarnation known as *Electronic Data Interchange* (EDI), has been traditionally used by larger corporations to share and exchange information between business partners and suppliers using private networks. EDI enables the exchange of business data from one computer to another computer. It eliminates the need to re-key information from documents or messages by supporting the creation of electronic versions of documents or messages using public standard formats, which can then be transmitted, received, and interpreted by other systems. Typical applications were (and still are) supply-chain management processes like order placement and processing. However with the explosive growth of the Internet in the last couple of years, electronic commerce is now able to offer solutions for a much broader range of business processes than EDI previously addressed. Also, the extensive availability of the Internet has enabled smaller companies, hindered previously by the large financial investment required for these private networks, to conduct business electronically. Technologies like bar coding, automatic teller machines, email, fax, video-conferencing, workflow, and the World-Wide-Web have continued to impact the success of E-commerce. Although the term E-commerce frequently refers to on-line retailing involving businesses and consumers, experts predict that as E-commerce continues to grow, business-to-business E-commerce will continue to enjoy the lion share of the revenue (Sheth, Aalst, and Arpinar, 1999).

Business-to-business E-commerce has seen tremendous growth due to the globalization of the worldwide economy, which in turn is enabled in large part by the omnipresence of the Internet. Many corporations are extending their markets by mergers and strategic alliances with other companies throughout the world. One industry where this has become very prevalent is the telecommunications industry. All over the world, this formerly regulated monopoly is undergoing a perestroika. In the United States, the 1984 divestiture of AT&T and the Telecommunications Act of 1996, have opened up previous monopolies on local and long distance telephone services, to include new players like cable companies electric utilities, broadcasters, long distance companies, and competitive access providers. Mergers amongst these companies have been on the rise with the aim being to gain the size necessary to offer a single-source global service to customers. Examples of such alliances include AT&T (long distance carrier) with BBN Planet (Internet service provider), Interchange service (online network), McCaw Cellular Communications (wireless service provider), and Teleport Communications Group (local service provider). This has allowed AT&T to expand their market to include Internet and on-line services, wireless, and local services in addition to their existing portfolio of long distance services. The bottom line is that many key players want to offer a full spectrum of services. These offerings include Internet access, local toll calling, interstate toll calling, wireless calling, satellite services, and delivery of entertainment services. Robert Allen (chairman of AT&T) states in a press release dated February 8, 1996, the date the Telecommunications Act of 1996 was passed (Dodd, 1997):

> *We will offer business and consumers bundles of services that will combine local and long distance, wireless, on-line services, even television. As much or as little as the customer wants.*

The result of all this is that provision of the telecommunications services may now involve several different entities. Business processes of each of the business partner now become coupled in some way, creating *interorganizational workflow processes* (Aalst, 1999). Workflow systems enable the automated management and coordination of tasks, people, and resources involved in performing a business process, in a way that streamlines and improves the efficiency of the business process. They provide tools for modeling, enactment, administration, and monitoring of business processes. They, therefore, could be very useful in managing complex workflow processes such as those that involve multiple organizations (interorganizational workflows). In particular the design of such workflow processes is often very complex and presents many challenges (Grefen, Aberer, Hoffner, and Ludwig, 2001).

There are currently two common approaches to designing interorganizational workflows. In the first approach, the complete workflow is designed explicitly, with entire details of all the sub-processes for the different organizations included in a single workflow design. This approach has two major disadvantages: First, it makes workflow design very complex. Second, it usually assumes the use of a single workflow management system, possibly with a

centralized architecture, for the management of the entire workflow. We argue that this approach is not well suited to the way in which business partners cooperate in the real world. In the second approach, each partner views other partner's sub-processes as black boxes, independently designed and managed. We believe that this approach is adequate where the relationship between sub-processes is somewhat hierarchical or chained. However, for more interactive processes where control moves back and forth between workflow processes, the black-box view is too restrictive and unnatural and, as will be illustrated in Section 3, can lead to deadlocks.

Business partners should be allowed to operate autonomously, with the ability to manage their own business processes, while at the same time participate towards the completion of an interorganizational business process. On the one hand, the local processes should be decoupled as much as possible. On the other hand, business partners should have some degree of understanding about the nature of the interaction between the processes of the different business partners.

The approach we discuss in this paper supports these two requirements. It suggests a skeletal design of the overall business process as the first step, identifying all key tasks and the control data dependencies between them. In subsequent steps, the different partners are assigned responsibility for completing certain parts of the workflow process and a workflow definition is created for each of the partners that includes only those parts of original workflow definition that they are responsible for. Partners may then locally make modifications to the created workflow definition to accurately reflect their own business process and capabilities. The modifications made locally by individual business partners are managed using the concept of *workflow inheritance* discussed in (Aalst and Basten, 1997; Aalst and Basten, 2001; Basten, 1998; Basten and Aalst, 2001). Workflow inheritance supports the creation of a new workflow by modifying parts of an existing workflow definition. This is analogous to the notion of inheritance in object oriented languages where a new class (subclass) can be created by extending an existing class in some manner. Specifically, we will discuss a set of rules that support modification through refinement and the addition of iteration, sequential, and parallel composition constructs. These rules ensure that the resulting workflow definition is sound and that it preserves the overall behavior of the original process.

The approach presented in this paper is based on *Petri nets*. They are a well-founded formalism for modeling the behavior of a dynamic system. They have a high expressive power and use a graphical notation, which enhances ease of use and understanding. In particular, we use a class of Petri nets, *WF-nets*, introduced in Aalst (1998), which demonstrates how the Petri-net formalism can be used to analyze the structure of workflow process definitions.

The rest of the paper is organized as follows. Section 2 discusses workflow modeling and reviews the Petri-net formalism and how it can be applied to workflow processes. The problems related to interorganizational are illustrated in Section 3. Section 4 presents our approach to tackle these problems. The approach is demonstrated in Section 5 using an example from the telecommunications domain. Section 6 discusses the issues that arise with the creation of the different workflows for the different partners and the modification of such workflows locally by the partners. They highlight the theoretical basis for an approach for the handling of these issues, specifically the inheritance-preserving rules. Section 7 relates the approach presented in this paper to other work in this domain. Section 8 concludes the paper.

2. Workflow modeling

The results presented in this paper are not specific for the Petri-net formalism. In fact, we have considered the approach in the context of the workflow management system METEOR (Kang, Froscher, Sheth, Kochut, and Miller, 1999; Sheth, Aalst, and Arpinar, 1999), which is not based on Petri nets. The Petri net basis is used for the unambiguous vendor/tool-independent representation of workflow processes and easy reference to state-of-the-art theoretical results. In this section, we review the basic terms and notations. We also provide some background information with respect to the modeling verification of workflows.

The *Workflow Management Coalition* (WfMC) defines a workflow management system as follows (Lawrence, 1997): *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*. 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 another department within the same organization (internal customer) generates a case. 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 definitions 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 may have 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*. In this paper we focus on the process perspective and abstract from other perspectives such as the organization, information, and application perspectives (Jablonski and Bussler, 1996).

Because processes are the dominant factor in workflow management, it is important to use an established framework for modeling and analyzing workflow processes. In this paper, we use a framework based on Petri nets to illustrate the concepts. Petri nets are a well-founded process modeling technique. Carl Adam Petri invented the classical Petri net in the sixties. Since then Petri nets have been used to model and analyze all kinds of processes with applications ranging from protocols, hardware and embedded systems to flexible manufacturing systems, user interaction and business processes. In the seventies, variants of Petri nets were already used to model office procedures, cf. the information control nets in Ellis (1979). There are several reasons for using Petri nets for workflow modeling: their formal semantics, graphical nature, expressiveness, analysis techniques, and tools provide a framework for modeling and analyzing workflow processes (Aalst, 1998; Ellis and Nutt, 1993).

A *Petri net* is a network composed of squares and circles (Desel and Esparza, 1995; Reisig and Rozenberg, 1998). 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. 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. 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. The set of reachable states (markings) consists of all states reachable by firing a sequence of (enabled) transitions.

Figure 1 shows a Petri net before (left) and after (right) the firing of a transition. In the Petri net before the firing, only place *p1* contains a token and transition *A* is the only enabled transition. Therefore, *A* will fire and consume one token and produce two tokens resulting in the state shown in Figure 1. In the resulting state, three transitions are enabled (*B*, *C*, and *D*). Either *B* and *C* fire (in parallel or in any order) or *D* fires. Finally, *E* will fire resulting in the state with just one token in *p6*. For the initial state shown in Figure 1 (left), there are 6 reachable states: *p1*, *p2+p3*, *p2+p5*, *p3+p4*, *p4+p5*, and *p6*.

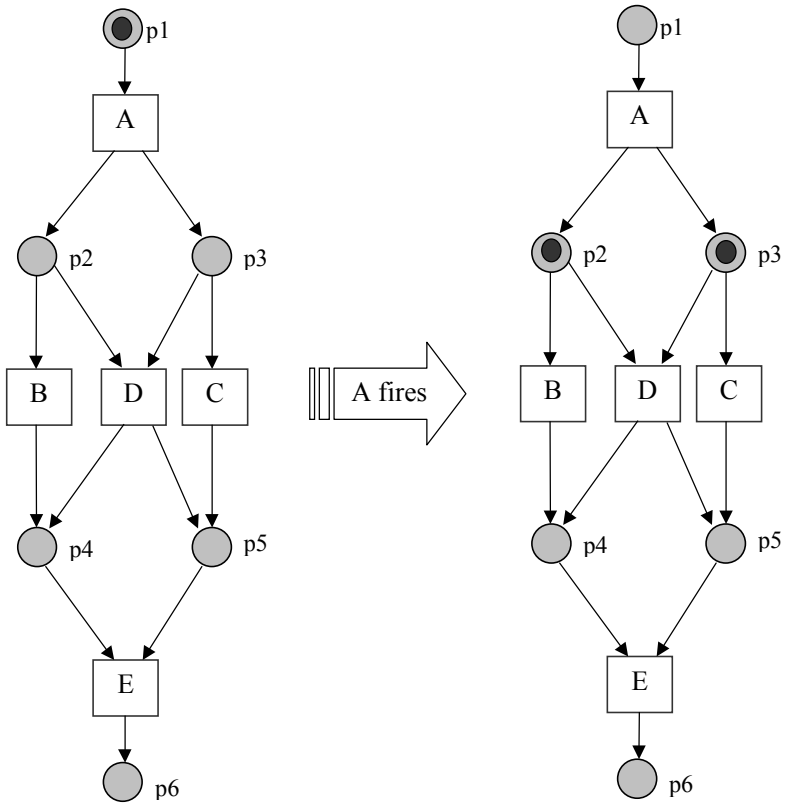

Figure 1: The result of firing the enabled transition A.

Clearly, transitions correspond to tasks and places correspond to conditions when modeling a workflow in terms of a Petri net. For the modeling of interorganizational workflows, it is convenient to define operators such as the union and intersection of Petri nets. Therefore, we formalize the definition of a Petri net and introduce some convenient operators (Reisig and Rozenberg, 1998).

Definition 1 (Petri net) A Petri net N is a tuple (P,T,F) where P is the set of *places*, T is the set of *transitions* and $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs, called the *flow relation*.

Definition 2 ($\subseteq,\cap,\cup$) Let $N_1 = (P,T,F)$ and $N_2 = (P,T,F)$ be two Petri nets.

- $N_1 \subseteq N_2$ if and only if $P_1 \subseteq P_2$, $T_1 \subseteq T_2$, and $F_1 \subseteq F_2$,
- $N_1 \cap N_2 = (P_1 \cap P_2, T_1 \cap T_2, F_1 \cap F_2)$,
- $N_1 \cup N_2 = (P_1 \cup P_2, T_1 \cup T_2, F_1 \cup F_2)$.

Definition 3 ($\bullet$) Let N = (P,T,F) be a Petri net and $x \in P \cup T$ a node (i.e., place or transition) of N. $\bullet x = \{y \in P \cup T | (y,x) \in F\}$ is the input set and $x\bullet = \{y \in P \cup T | (x,y) \in F\}$ is the output set of x. For $X \subseteq P \cup T$, $\bullet X = \bigcup_{x \in X} \bullet x$ and $X\bullet = \bigcup_{x \in X} x\bullet$.

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 (Aalst, 1998).

Definition 4 (WF-net) A Petri net N=(P,T,F) is a WF-net if the following three requirements are satisfied:

1. There is one source place i ($\bullet i = \varnothing$, i.e., a place with no incoming arcs).
2. There is one sink place o ($o\bullet = \varnothing$, i.e., a place with no outgoing arcs).
3. Every node (i.e., a place or a transition) is on a path from the source place i to the sink place o.

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. The Petri net shown in Figure 1 is clearly a WF-net: *p1* is the source place (i) and *p6* is the sink place (o).

Figure 2 shows three WF-nets. Note that special symbols indicate the source and sink places. Each of the nets satisfies the requirements stated in Definition 4. However, this does not mean that the corresponding workflows are correct; there can still be dynamic errors like livelocks, deadlocks, and other anomalies. In fact, the first two WF-nets shown in Figure 2 correspond to incorrect workflow process definitions. The first WF-net (A) has a potential deadlock. If after executing task *register*, task *handle* is executed, then the process will deadlock because one of the input conditions of task *check1* will never be fulfilled, i.e., the case gets stuck in the state with a token in both *p2* and *p5*. Note that if *send* is executed instead of *handle* the process terminates correctly. The second WF-net (B) will not deadlock but potentially leave a dangling reference. If task *handle* is executed, then the token produced by *check1* is not consumed. For most workflow management systems this situation is undesirable because completed cases leave garbage (taking up memory) or tasks are executed after completion of the case.

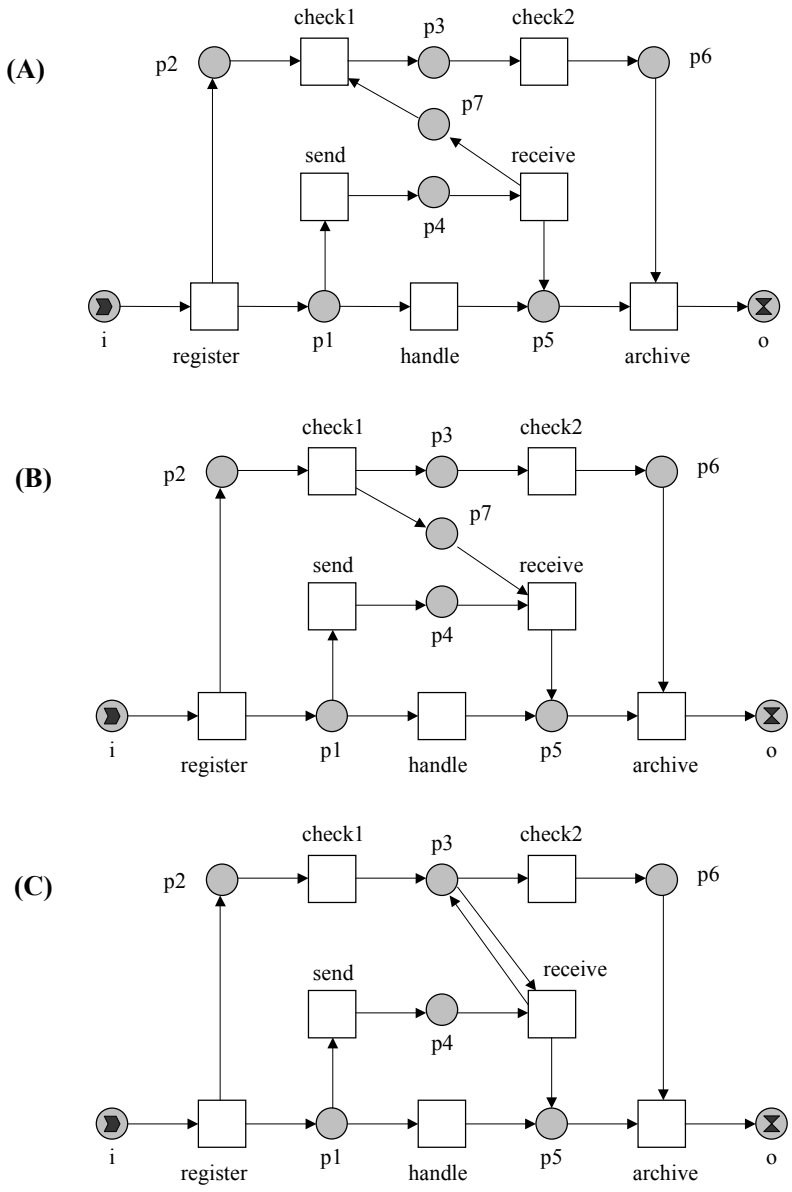

Figure 2: Three WF-nets.

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 (Aalst, 1998; Aalst, Desel, and Oberweis, 2000).

Definition 5 (Soundness) A workflow net N=(P,T,F) is *sound* if the following four requirements are satisfied:

1. 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 sink place of the WF-net).
2. There are no dangling references, i.e., the moment the case terminates (i.e., a token appears in o), there are no tokens left behind in the workflow net.
3. 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.

4. Conditions are safe, i.e., a condition cannot be fulfilled multiple times. In other words, the maximum number of tokens in a place corresponding to a case is one.[1]

The third WF-net (C) shown in Figure 2 is sound. The first WF-net (A), has a deadlock and does not satisfy the first requirement. The second WF-net can leave a dangling token in place *p7* and, therefore, does not satisfy the second requirement. 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 (Verbeek, Basten, and Aalst, 2001). 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 (Aalst, 1998). Although soundness can be decided in polynomial time for certain subclasses (e.g., by using the Rank theorem for free choice nets (Desel and Esparza, 1995), 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 factor. Woflan can cope with workflow process definitions having more than 200,000 states.

There is one more standard Petri net concept that will be used in the remainder.

Definition 6 (Implicit place) Let N=(P,T,F) be a Petri net and s be the initial state. Place $p \in P$ is implicit if and only if for each reachable state and each transition $t \in p\bullet$, if t is enabled in the net without p, then t is also enabled in N.

An implicit place is a place which does not restrict the enabling of transitions, i.e., every firing sequence enabled in the net without the implicit place is also enabled in the net with the implicit place. One could say that the implicit place does not change the actual behavior. When partitioning a workflow over multiple domains (i.e., business partners), these implicit places turn out to be very useful. Note that the notion of implicit place depends on the initial state. Therefore, it is difficult to check whether a place is implicit. Fortunately, most implicit places are also structurally implicit, i.e., polynomial-time linear-algebraic techniques can be used to identify/add implicit places. See Berthelot (1986,1987) for more information.

The application of Petri nets to the modeling and analysis of workflows within one organization has been reported in Aalst (1998), Aalst, Desel, and Oberweis (2000), Adam, Atluri, and Huang (1998), Ellis (1979), Ellis and Nutt (1993), Jablonski and Bussler (1996), and Verbeek, Basten, and Aalst (2001). The application of Petri nets to interorganizational workflows has been discussed in Aalst (1998, 1999, 2000a, 2000b), Aalst and Anyanwu (1999), Aalst and Weske (2001), Bons, Lee, and Wagenaar (1998), Kindler, Martens, and Reisig (2000), Lee (1999), and Lee and Bons (1996). See Section 7 (related work) for more details.

3. The challenge

Interorganizational workflows cross organizational borders resulting in conflicting requirements. On the one hand, the overall workflow should be managed and coordinated to avoid stagnation and errors. On the other hand, local autonomy is needed to enable each of the business partner to handle their part of the workflow as effective and efficient as possible. To illustrate the trade-off consider the public workflow shown in Figure 3. This workflow is partitioned over two domains: tasks *A*, *B*, *C*, and *D* are mapped onto domain *L* (left) and tasks *E*, *F*, *G*, and *H* are mapped onto domain *R* (right). Each of the two domains is handled by one of the business partners. Both business partners agreed on the public workflow and the division of tasks, i.e., the public workflow can be considered as a contract identifying the main steps of the overall process. One could say that Figure 3 specifies the interorganizational protocol.

[1] Note that the safeness requirement is not present in the earlier definition of soundness presented in Aalst (1998). Since places correspond to conditions, it is a reasonable assumption which simplifies the formulation of the inheritance-preserving transformation rules.

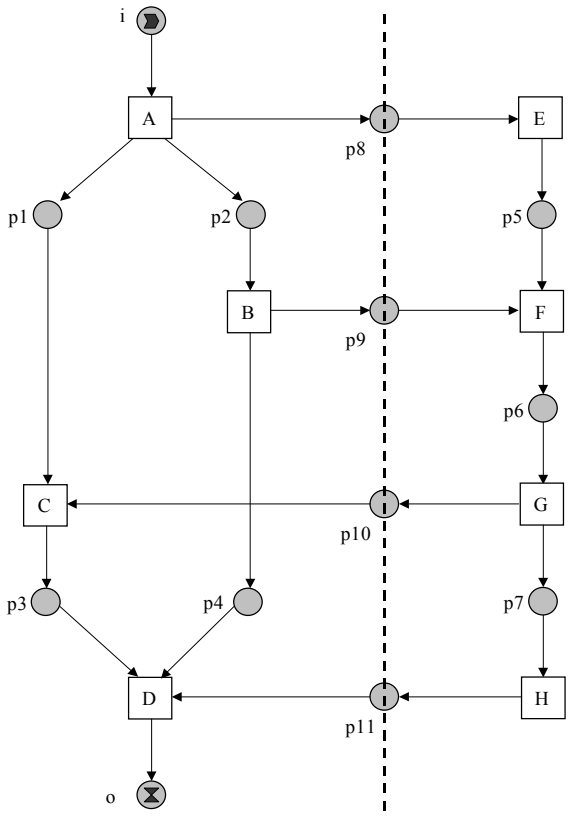

Figure 3: The public workflow used to illustrate problems resulting from local change.

After agreeing on the public workflow, each of the business partners involved starts to implement its part of the workflow. This local workflow is modified to accommodate local needs. In domain *L,* a causal relation is added between task *C* and task *B* to make sure that *C* is executed before *B*. In domain *R* the order in which *F* and *G* are executed is reversed. Figure 4 shows the two modified private workflows. Place *p12* has been added to force the execution of *C* before *B*.

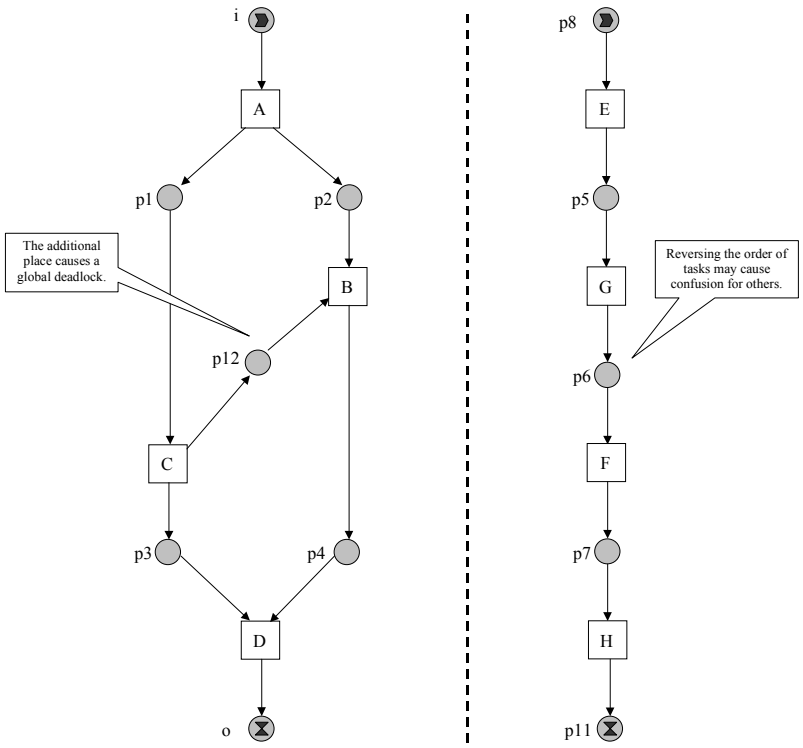

Figure 4: Both private workflows have been modified.

From a local point of view, this change does not cause any problems. For domain *L* it is perfectly acceptable to reduce the degree of parallelism. However, this change leads to a global deadlock. On the one hand *C* should be executed before B (place *p12*) and on the other hand *B* should be executed before *C* (causal relation via *p9*, *p6*, and *p10*). The circular dependency involving tasks *C*, *B*, *F*, and *G* causes the overall workflow process to deadlock in the state marking places *p1*, *p2*, and *p5*.

Figure 4 also shows the modification of the private workflow of domain *R*: Task *F* and task *G* are reversed. This change alleviates the problem caused by the addition of place *p12*. If both private workflows are modified as indicated in Figure 4 there would be no deadlock. However, the modification of the private workflow of domain *R* is also not acceptable. Based on the public workflow both partners agreed upon, the execution of task *C* may depend on the results of task *F*. By reversing the order of task *F* and task *G* there can be problems (e.g., missing data) because task *C* may be executed before task *F*. These examples show that local changes may cause global errors, i.e., modifications which are harmless from a local perspective may cause deadlocks, livelocks, missing data, etc.

4. The approach

In general, modeling workflow processes that span multiple organizations can be complex. Many workflow modeling tools require that the complete details of the entire process be fixed at this design stage. This approach is often the bottleneck in workflow design, because it requires each participating business partner to understand the nature of their partners' local processes, which can make things even more complicated. This is neither necessary nor desirable. Partners should be able to agree on their business process at a level that abstracts them from irrelevant and confusing details. In addition, most organizations may be unwilling to reveal business rules such as lending policies, in order to protect their competitive advantages. Therefore, organizations should be able to participate towards the completion of a business process, while at the same time be at liberty to construct their private processes in any way that places them at best advantage.

The approach we discuss here for designing interorganizational workflows, is a four-step process that involves the creation of a public process, partitioning the public process amongst the partners, and allowing for modification by the individual partners of their parts of the process to create private processes.

The steps are:

(1) *Design public workflow process*.

In the first step, the partners agree on the overall structure of the common business process. The key tasks are identified as well as the interfaces between them. The interfaces are specified as control and data dependencies. Process repositories may also be used at this stage to aid the design process. The workflow process model resulting from this stage will not contain organization specific information like task implementation, resource assignment, etc. We refer to the workflow process represented by this model as the *public* process. Figure 5 shows an example public process represented as a WF-net. The WF-net shows four tasks *A*, *B*, *C,* and *D*, and the data exchanged between them.

(2) Partition the public workflow process definition amongst business partners

In this step, the partners are assigned to be responsible for completing parts of the process. The WF-net is then partitioned along organizational lines, i.e., the public workflow is mapped onto several domains. The partitioning creates a set of WF-net fragments for each business partner. Each set contains fragments of the public WF-net that the partner is responsible for. Figure 6 shows the WF-net partitioned between two organizations, L (left) and R (right), such that L is responsible for the fragment set containing tasks *A* and *D*, and organization R is responsible for the fragment set containing tasks *B* and *C*.

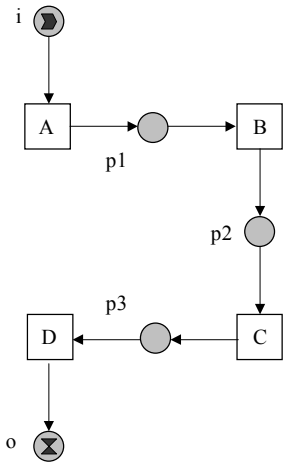

Figure 5: The public workflow process.

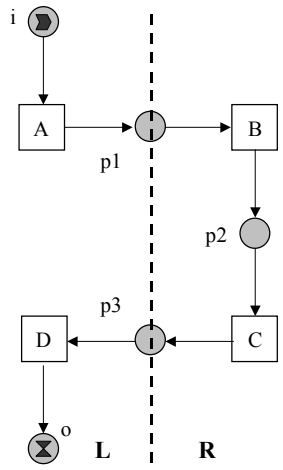

Figure 6: Partitioning the workflow definition amongst organizations L and R.

(3) For each business partner create a private workflow.

The partitioning stage creates sets that are not necessarily WF-nets, rather fragments of a WF-net. A fragment may have multiple input or output places, or no input or output places at all. Moreover, a fragment may contain disjoint subsets (i.e., is unconnected). In other words, the requirements stated in Definition 4 may be violated for one or more fragments.

This step automatically creates a WF-net from each fragment set for each of the partners to manage locally in their organization. Creating a WF-net from fragments involves connecting the fragments by introducing arcs and implicit places. The result is a sound WF-net whose behavior preserves the interface of the original public process. We will refer to each of the local workflows as *private* processes. In Figure 7, two WF-nets are created. For organization R, place *p1* becomes its input place and place *p3* becomes its output place. For organization L, and implicit place *p4* is added between *A* and *D* and arcs are used to connect them. An implicit place has the property that its addition does not alter the dynamic behavior of the WF-net. For example for organization L, when task *A* completes, it places a token in place *p4* and transition *D* can only fire after *A* is completed. This behavior is equivalent to the behavior of the original public process. The dark lines show the added arcs and places.

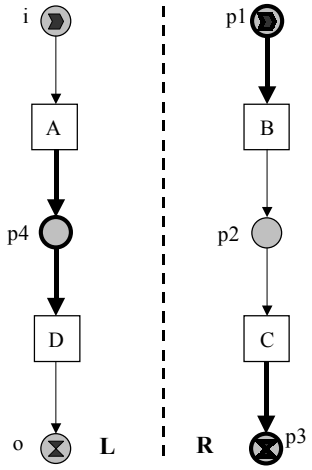

Figure 7: Private WF-nets for organizations L and R.

(4) *Modify private processes using inheritance-preserving transformation rules*.

Here, each business partner will modify their private process created in the previous step to accurately reflect their own business process and incorporate details of how the tasks will be implemented at their organization. Some of the details that will be added at this stage are task implementation, assignment of roles to the tasks, data objects to be used and any mapping of global data objects to these local data objects, business rules, deadline specification etc. These extensions or modifications to the minimal workflow must also preserve the behavior of the global business process.

To guarantee that local extensions and modifications do not disturb the behavior of the public process, we use a notion of inheritance of dynamic behavior. For this purpose, we use the inheritance relations defined in Aalst and Basten (1997), Aalst and Basten (2001), Basten (1998), and Basten and Aalst (2001). Basically, inheritance relations relate many variants of a single process definition using subclass-superclass relationships. When one WF-net extends the behavior of another WF-net, it is said to be a subclass of the WF-net which it extends. Workflow B extending workflow A means that abstracting or hiding the modifications made to B, yields a workflow whose dynamic behavior is equivalent to the behavior to workflow A.

In Aalst and Basten (1997) and Basten (1998), four notions of inheritance are introduced. We have found one notion of inheritance, *projection inheritance*, particularly relevant in handling modifications made locally by the partners to the workflow created in step three.

Roughly, projection inheritance says that, for two workflow process definitions A and B, where B contains all transitions in A and some additional ones, if it is not possible to distinguish between the behavior of A and B, when the effects of the transitions that are in B but not in A are hidden (ignored), then B is a subclass of A under projection inheritance. Section 6.2.2 discusses some transformation rules that correspond to refinement, parallel composition, sequential composition, and iteration. These rules ensure that modifications made to the WF-net result in a sound WF-net and that these modifications create subclass WF-nets that preserve projection inheritance.

Figure 8 shows an example of what a modified workflow might look like for organization L. The rules **PP**, **PJ**, **PJ3,** and **PJT** are the projection inheritance-preserving rules that are used to add iteration, sequential composition, parallel composition, and refinement respectively. Task A is refined by tasks *A*, *A2,* and *A3* using rule PJT. Tasks *E, F,* and *G* are inserted between tasks A3 and D using rule PJ (sequential composition rule). With rule PJ3 (parallel composition rule), a parallel branch containing tasks *H* and *I* is added between tasks F and D. Task *J* is added with an iteration construct using rule PP (iteration rule). Observing the resulting WF-net shows that when the effects of all the newly introduced tasks and branches are ignored the behavior of this WF-net is equivalent to the behavior of the original WF-net, i.e., *D* can only execute after *A* completes. (The shaded tasks are the newly introduced tasks).

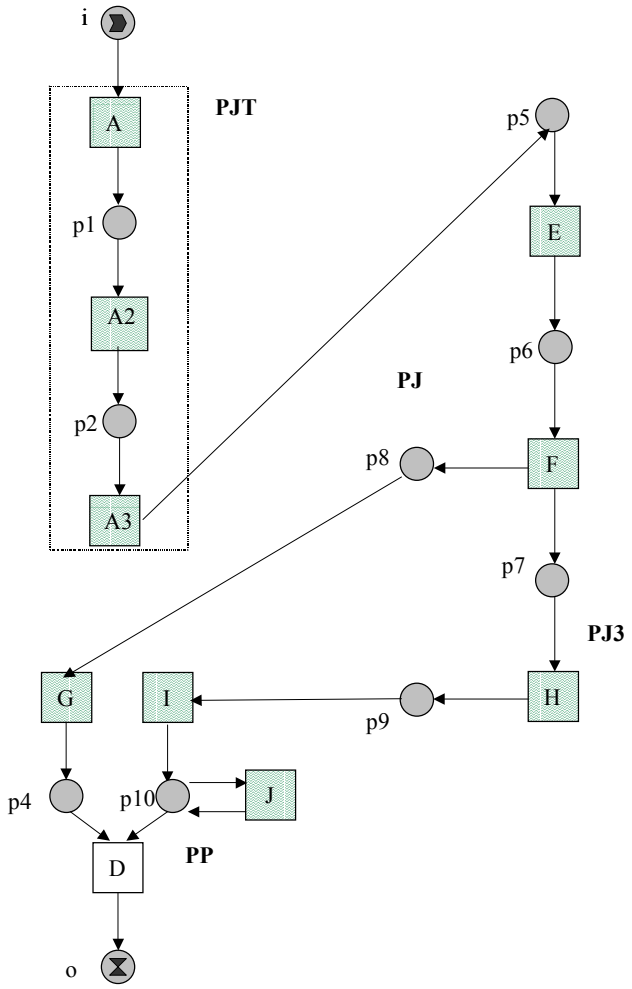

Figure 8: Modified workflow for organization L.

5. A telecom example

To further illustrate the value of our approach, we will apply it to a real-world business process from the telecommunications domain (Cole, 1999; Dodd, 1997).

Since its deregulation, the telecommunications industry has been undergoing a huge and rapid metamorphosis. The market has become fiercely competitive with major service providers now able to compete in several market segments at the same time. In addition, advancements in technology such as ATM, wireless, broadband and multimedia services have also made a contribution to the changing shape of the industry. The result of all this is that

new players like foreign carriers, utilities companies, cable TV operators, publishing, and computing and online services industries are now able to enter the market at various levels and offer selected services. Also, customers are now requiring newer and higher bandwidth services.

In order to survive in this environment, many service providers have adopted "service bundling" as their marketing strategy. This enables them to offer a wide range of services, local, long-distance, wireless, and Internet services, from one organization, thereby providing a single point of contact and one bill for all services for the customer. To achieve this, acquisitions amongst Interexchange Carriers (IECs – long distance phone service providers), Incumbent Local Exchange Carriers (ILECs – local phone service providers), Competitive Local Exchange Carriers (CLECs – non Bell local phone service providers) as well as other groups like Internet Service Providers and Cable TV companies, have occurred in order to improve competitive advantage by broadening the scope of services offered. For example WorldCom, a successful IEC providing long distance services acquired Metropolitan Fiber Systems (MFS), a CLEC providing local phone service, and more recently UUNet which operates the largest Internet backbone in the US. This has allowed WorldCom to offer bundled local, long distance, and Internet access services to small businesses. Another option that is adopted by many of the smaller providers that cannot afford such acquisitions, is to contract with one or more service providers to supply capacity, features or access, since many of these providers are unable to single-handedly satisfy the demands of today's customers with just their own resources either because of capacity, geographical or regulatory constraints. In both cases, business processes can span multiple providers.

In this very competitive environment, service excellence has become a compelling business objective. To improve the efficiency and cost effectiveness of their service operations, many organizations are automating provider-to-provider process flow for service management processes such as, order handling, service provisioning, fault tracking, billing, etc. The example we discuss here focuses on service provisioning involving two service providers. It is of-course the case that many real-world examples are much more complicated and may involve several more entities. However, we have chosen a somewhat simpler example, which is sufficient to demonstrate the practical relevance of our approach, without being too complex to be described in a paper. The example is a modified version of the scenario used in Adams and Willetts (1996).

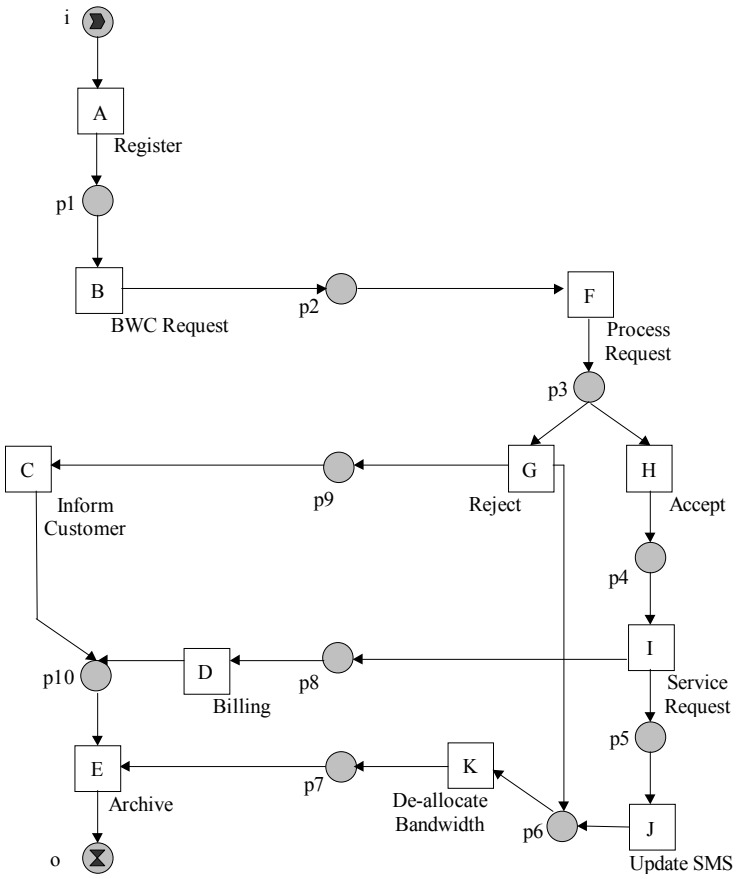

Figure 9: Bandwidth change request process.

In our example, a service provider "SuperCom" that offers fixed private line services has decided to add an on-demand bandwidth service to its portfolio, in response to a growing market trend. This service will allow customers to adjust their bandwidth requirements from a minimum of 9.6Kbs to a maximum of 2Mbs on demand. To do this, SuperCom has contracted this service to another provider NetMedia which has a modern network using Synchronous Digital Hierarchy (SDH) as a backbone, as well as modern Time Division Multiplexers (TDM) on the customer's sites to deliver the service. The customer subscribes for this service through SuperCom who is the primary service provider for customers. When a customer requests bandwidth change, the request is actually made to SuperCom. SuperCom routes it to NetMedia who can either accept or reject to service the request, but in the later case the rejection response must come within one minute. This requirement is specified in the Service Level Agreement (SLA) between SuperCom and NetMedia. Both SuperCom and NetMedia have identified a set of primary tasks that are involved in providing this service and specified the causal relations between these tasks. In Figure 9, the resulting public process is represented as a WF-net (Step1).

After a request is registered by task *Register*, another task *BWC Request* (Bandwidth Change Request) initiates the bandwidth change request. The request is processed by task *Process Request*, after which the request is either rejected and processed by task *Reject* or accepted processed by *Accept*. If the request is rejected, the customer is informed by task *Inform Customer*. If on the other hand the request is accepted, then it is serviced by task *Service Request* followed by task *Update SMS* which is used to update all service management systems (e.g., Performance Monitoring, Fault Tracking Systems). Task *De-allocate Bandwidth* frees the allocated bandwidth to make it available for use during any subsequent request and, (potentially) at the same time, *Billing* generates necessary billing information. The process ends by archiving information related to the request in task *Archive*.

The partitioning of the public process between SuperCom and NetMedia results is shown in Figure 10 (Step 2). The resulting fragment sets for SuperCom and NetMedia do not form WF-nets. For example, the partition for SuperCom is not connected, while the partition for NetMedia contains multiple output places.

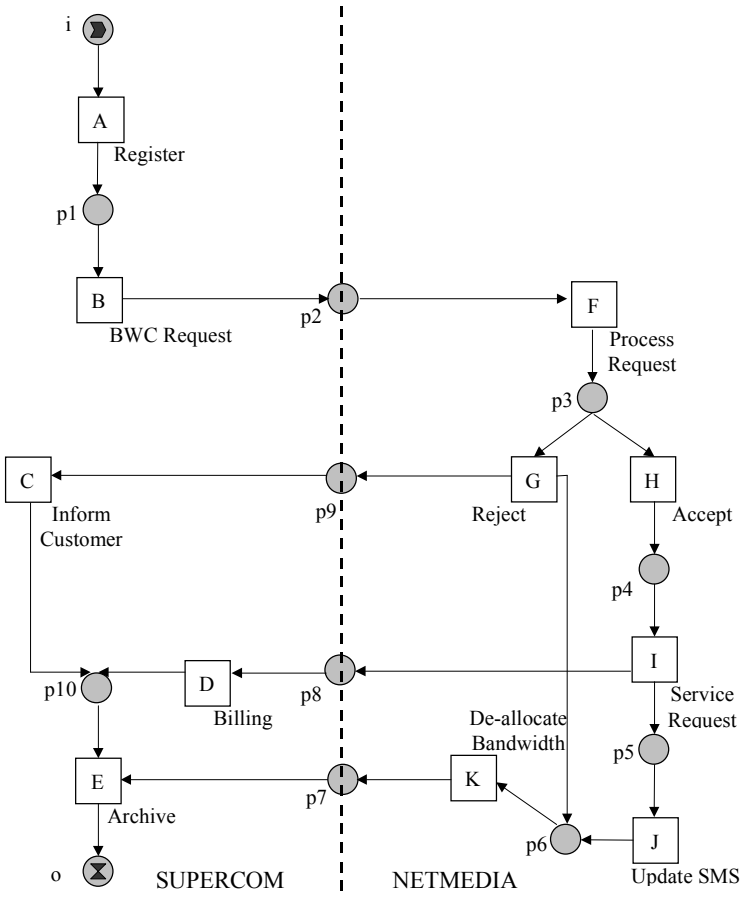

Figure 10: Workflow definition partitioned between SuperCom and NetMedia.

The next step is to create two WF-nets from the two partitions, one for SuperCom and one for NetMedia, as shown in Figure 11. For NetMedia, place *p2* is assigned to be the input place of the private workflow and place *p7* is assigned to be the output place. For SuperCom, an implicit place *p11* is added. Also arcs from *p11* to tasks *Billing* and *Inform Customer,* and another arc from *BWC Request* to *p11* are introduced. The dark lines in the Figure 11 mark these changes.

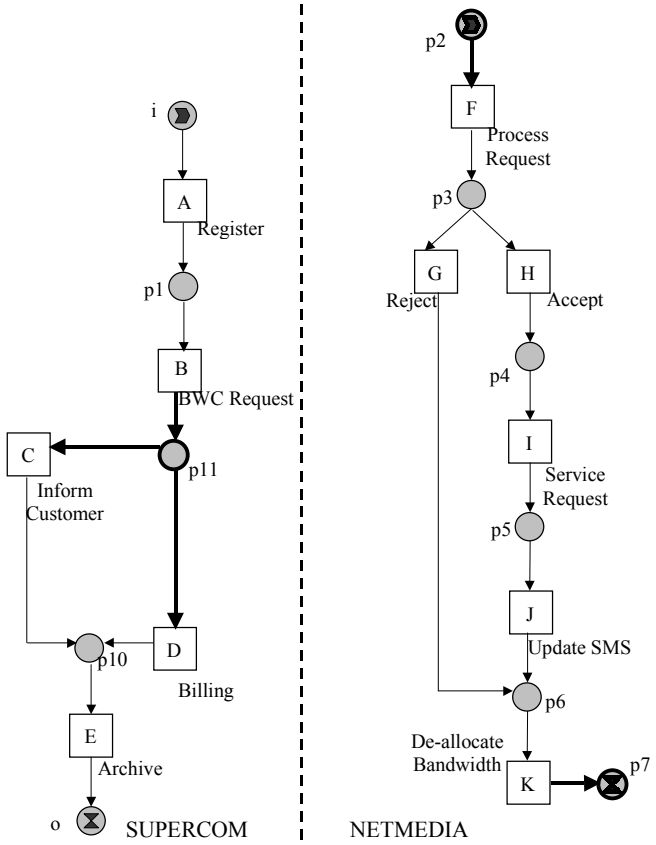

Figure 11: Private WF-nets for SuperCom and NetMedia.

In Figure 12 we see that both SuperCom and NetMedia modified their private processes created in step 3. (The shaded tasks are the newly introduced tasks.) Modifications made by SuperCom include the addition of two tasks, *ECB* (Evaluate Customer Billing Status) and *ECP* (Evaluate Customer Privileges), before the request is sent to NetMedia. Task *ECP* ensures that the customer has subscribed for such a service, and task *ECB* ensures that the customer is within billing limits specified in their contract. ECB is inserted using PJ (sequential composition rule) followed by the addition of ECP using PJ3 (parallel composition rule). Also, another task *Handle Cust* is added to handle any customer inquires that occur during this time. This task is inserted using inheritance-preserving transformation rule PP (iteration rule).

For NetMedia, a task *Evaluate BW* (Evaluate BandWidth) is inserted to determine if there is enough available bandwidth to service the request at that time and if so, the request is serviced, otherwise, the request is denied. This task is added using PJ (sequential composition rule). A parallel branch is added using PJ3 (parallel composition rule). This branch contains task *Monitor*: a task that monitors requests and usage patterns.

In real world situations, it is conceivable that partners may join or leave the consortium (i.e., the group of cooperating business partners) at any time. In this case, the public process will be repartitioned amongst the members of the group and then each member may again make any necessary modifications locally to the WF-net from step 3.

Some products like Extricity's Alliance also have a similar notion of public and private processes. However, the only possible type of local modification supported by Extricity is refinement, i.e., of the four inheritance-preserving transformation rules only PJT (refinement rule) is supported.

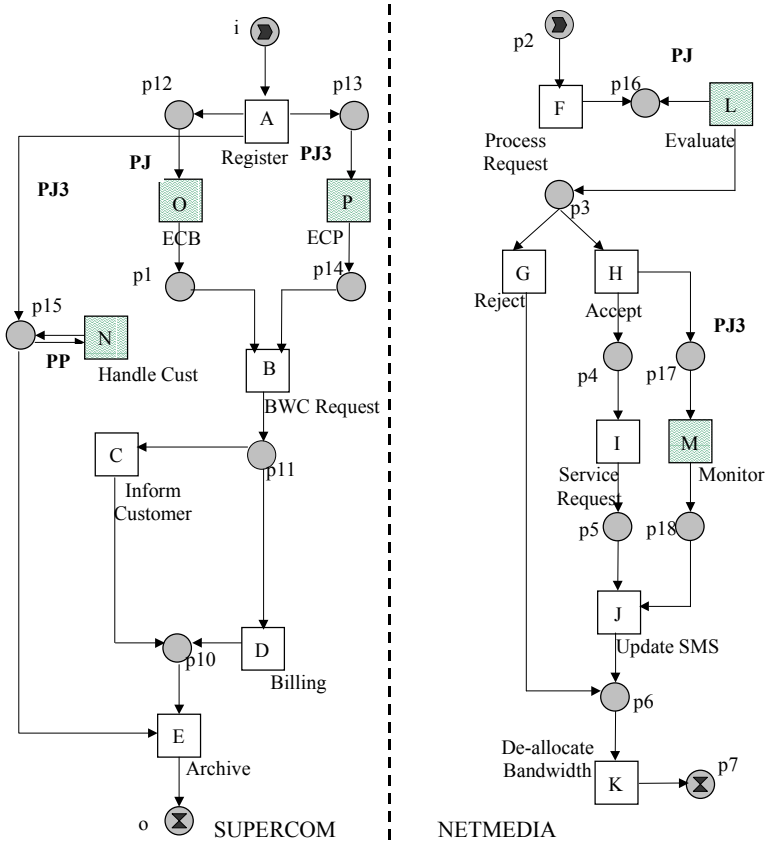

Figure 12: Modified workflows for SuperCom and NetMedia.

6. Problems and solutions

The approach described in Section 4 is characterized by two potential problems. The first problem is to make sure that after partitioning each private (i.e., local) workflow satisfies the requirements generally imposed by a workflow management system (cf. Definitions 4 and 5). The second problem is to clarify which modifications of the private workflow are allowed without jeopardizing the correct operation of the overall workflow.

6.1 Problem 1: Creating a private workflow

Not every partitioning of the public workflow yields meaningful private workflows. A partitioning of the public workflow may lead to private workflows, which consist of unrelated tasks without clear initiation and termination points. The partitioning should result in a local workflow for each domain, which satisfies the requirements stated earlier, i.e., the local workflow should correspond to a sound WF-net (cf. Definitions 4 and 5). To tackle the problem, we first introduce some notations for the public workflow and its partitioning.

Definition 7 (N^{pub}) $N^{pub} = (P^{pub}, T^{pub}, F^{pub})$ is a WF-net which represents the *public* (i.e., common) *workflow process definition*.

The public workflow is partitioned over a set of domains using a function which maps tasks onto domains.

Definition 8 (D,map) D is a set of *domains*, i.e., the business partners involved in the public workflow. The function $map \in T^{pub} \rightarrow D$ maps each task in the public workflow onto one of the domains.

Note that Definition 7 corresponds to Step 1 of the approach and that Definition 8 corresponds to Step 2. In the remainder of this section we focus on two issues: (1) Is the partitioning specified by function map possible? and (2)

How to create a private sound WF-net (Step 3)? To answer these questions, we consider the requirements stated in Definitions 4 and 5. Observing these requirements carefully shows that there are basically two things that may complicate the construction of a private sound WF-net: (1) the private workflow has no clear starting and/or termination point, and (2) the tasks in a private workflow are not related (i.e., unconnected). Since a private workflow is either initiated via the public input place i or via a message exchanged between domains, we identify the communication channels between domains.

Definition 9 (P^{exch}) $P^{exch} = \{p \in P^{pub} | \exists\ t_1,t_2 \in T^{pub}: (t_1 \in \bullet p) \wedge (t_2 \in p\bullet) \wedge (map(t_1) \neq map(t_2))\}$ is the set of *exchange places*, i.e., *communication channels* between domains.

A place p corresponds to a communication channel if the execution of a task in one domain can trigger a task in another domain via p. A private workflow is initiated by the presence of a token in i or by a token sent via one of the exchange places. Similarly, the termination of a private workflow is indicated by producing a token either for the global sink place o or one of the places in P^{exch}.

Definition 11 (inp, outp) Function $inp \in D \rightarrow P^{exch} \cup \{i\}$ maps every domain onto a *local input place* and function $outp \in D \rightarrow P^{pub} \cup \{o\}$ maps every domain onto a *local output place*.

Functions inp and outp clearly identify the starting point and termination point of each local workflow. All tasks mapped onto a domain d should be executed in-between both points. Moreover, every task should be on a directed path from inp(d) to outp(d) as indicated in Definition 4. Generally this will not be the case because causal relations between tasks may be established via tasks in other domains. Fortunately, these causal relations can be added using the concept of implicit places (see Definition 6). Recall that a place is implicit if it does not restrict the execution of transitions. If there is a causal relation, then it is possible to add an implicit place which makes this causal relation explicit. In the private workflow of a domain d, every task should be causally related to the input place inp(d) and the output place outp(d). If this is not the case, there is a task which can be executed before the private workflow is activated or after it has been completed. Therefore, it is possible to add implicit places to the public workflow such that each of the private workflows becomes connected.

Definition 10 (N^{epub}) $N^{epub} = (P^{epub},T^{epub},F^{epub})$ is a WF-net which represents the *extended public workflow process definition*.

N^{epub} extends N^{pub} with implicit places such that none of the private workflows contains tasks not on a path from the local input place to the local output place. Based on the definition of the functions map, inp, and outp, it is possible to partition N^{epub} into private workflows which can be used as a starting point for Step 4 of the approach.

Definition 12 (N_d^{priv}) For each domain $d \in D$, $N_d^{priv} = (P_d^{priv},T_d^{priv},F_d^{priv})$ is a Petri net which represents the local part of the extended public workflow process definition and is constructed as follows: $P_d^{priv} = \{inp(d),outp(d)\} \cup (\bullet T_d^{priv} \cap T_d^{priv} \bullet)$, $T_d^{priv} = \{t \in T^{pub} \mid map(t)=d\}$, and $F_d^{priv} = F^{epub} \cap ((P_d^{priv} \times T_d^{priv}) \cup (T_d^{priv} \times P_d^{priv}))$.

N_d^{priv} should be a sound WF-net which can be used a template for designing the private workflow for domain d. N_d^{priv} contains all the transitions corresponding to tasks mapped onto d, the input place inp(d), the output place outp(d), and all places connecting transitions corresponding to tasks mapped onto d. Note that the communication channels other than inp(d) and outp(d) are not present in N_d^{priv}. Not every choice of inp, outp, map is acceptable. We already mentioned some requirements. Now we give a complete list of the requirements which should be satisfied in Step 2 and Step 3 of the approach presented in the previous section.

Requirements Let N^{pub}, D, and map be as defined before. In the remainder, we require that there exist an N^{epub}, inp, and outp such that the following requirements are satisfied:

1. $N^{pub} \subseteq N^{epub}$ and $T^{epub} = T^{pub}$.
2. For all $p \in P^{epub} \setminus P^{pub}$, p is implicit in $(P^{pub} \cup \{p\},T^{pub},F^{epub} \cap (((P^{pub} \cup \{p\}) \times T^{pub}) \cup (T^{pub} \times (P^{pub} \cup \{p\}))))$ for the initial state with one token in input place i.
3. N^{epub} is a sound WF-net.
4. For all $p \in P^{exch}$, $|\bullet p| = |p\bullet| = 1$.
5. For each $d \in D$, N_d^{priv} is a sound WF-net.

The first requirement states that the extended public workflow should only add places. The second requirement states that these places should be implicit. The third requirement has been added to make sure that after termination nothing is left in any of the implicit places. Communication channels should be unidirectional (fourth requirement), i.e., there should be one input transition in one domain and one output transition in another domain. This means that tokens in $p \in P^{exch}$ correspond to messages from a specific task in one domain to a specific task in another domain. Note that this requirement indicates that there can be no choices at the interface. Any decision is made inside one domain. This does not limit the application since multiple logical communication channels (i.e. places in P^{exch}) can be mapped onto one tangible channel. The last requirement states that each of the resulting private workflows should be a sound WF-net, i.e., N_d^{priv} satisfies all the conditions mentioned in Definition 4 (WF-net) and Definition 5 (soundness). Requirement 5 is not as restrictive as it seems. Every partitioning of the public workflow will do as long as it is possible to identify a local input and output place and every task is executed in-between the states represented by both places.

6.2 Problem 2: Modification of the private workflow

In Section 3, it was shown that modifications in one domain can lead to deadlocks and other anomalies between domains. In this section we provide more details about the inheritance-based approach presented in this paper. Before we discuss the inheritance concept, we revisit the problem statement given in Section 3.

6.2.1 The problem

In Step 4 of the approach presented in this paper, the private workflow is modified to accommodate local needs. Recall that the public workflow comprises tasks which are of interest to all business partners involved. Tasks which are only of local interest are added without informing the other business partners. Tasks related to quality control, internal bookkeeping, and storage management are typical examples of tasks which are only of local interest. N_d^{priv} is the private workflow before modification, i.e., the local part of the extended public workflow process definition. Step 3 results in a N_d^{priv} for each domain $d \in D$. The goal of Step 4 is to modify N_d^{priv} into N_d^{epriv}: the modified private workflow. Since the modifications typically entail extensions, we name N_d^{epriv} the *extended private workflow* of domain d. Local extensions of the workflow may cause serious problems at the global level. Reordering tasks and adding or removing causal relations may cause deadlocks, livelocks, confusion, and other anomalies. Consider for example the public workflow shown in Figure 3. This workflow is partitioned over two domains: tasks *A*, *B*, *C*, and *D* are mapped onto domain *L* (left) and tasks *E*, *F*, *G*, and *H* are mapped onto domain *R* (right). The input place of the private workflow of domain *R* is *p8* and *p11* is the output place of this domain. There is no need to add implicit places in either domain. Therefore, the first three steps leading to N_L^{priv} and N_R^{priv} are straightforward given the public workflow shown in Figure 3.

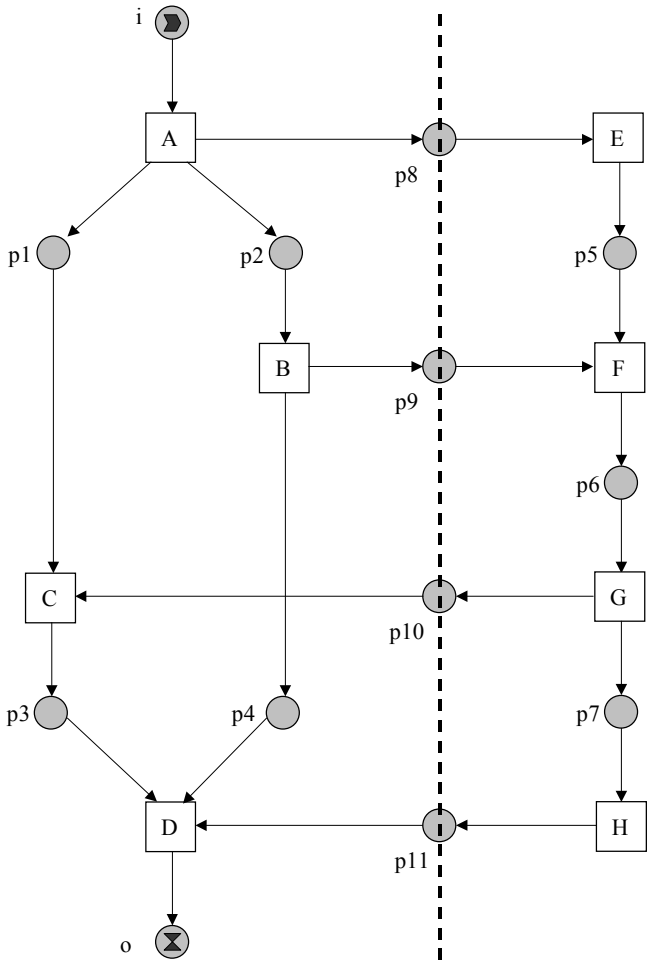

Figure 13: The public workflow used to illustrate problems resulting from local change.

N_L^{priv} and N_R^{priv} are modified to accommodate local needs. In domain *L* a causal relation is added between task *C* and task *B* to make sure that *C* is executed before *B*. In domain *R* the order in which *F* and *G* are executed is reversed. Figure 4 shows the two modified private workflows N_L^{epriv} and N_R^{epriv}. Place *p12* has been added to force the execution of *C* before *B*. From a local point of view, this change does not cause any problems. For domain *L* it is perfectly acceptable to reduce the degree of parallelism. However, this change leads to a global deadlock. On the one hand *C* should be executed before B (place *p12*) and on the other hand *B* should be executed before *C* (causal relation via *p9*, *p6*, and *p10*). The circular dependency involving tasks *C*, *B*, *F*, and *G* causes the overall workflow process to deadlock in the state marking places *p1*, *p2*, and *p5*. Figure 4 also shows the modification of the private workflow of domain *R*: task *F* and task *G* are reversed. This change alleviates the problem caused by the addition of place *p12*. If both private workflows are modified as indicated in Figure 4 there would be no deadlock. However, the modification of the private workflow of domain *R* is also not acceptable. Based to the public workflow both partners agreed upon, the execution of task *C* may depend on the results of task *F*. By reversing the order of task *F* and task *G* there can be problems (e.g., missing data) because task *C* may be executed before task *F*. These examples show that local changes may cause global errors, i.e., modifications which are harmless from a local perspective may cause deadlocks, livelocks, missing data, etc.

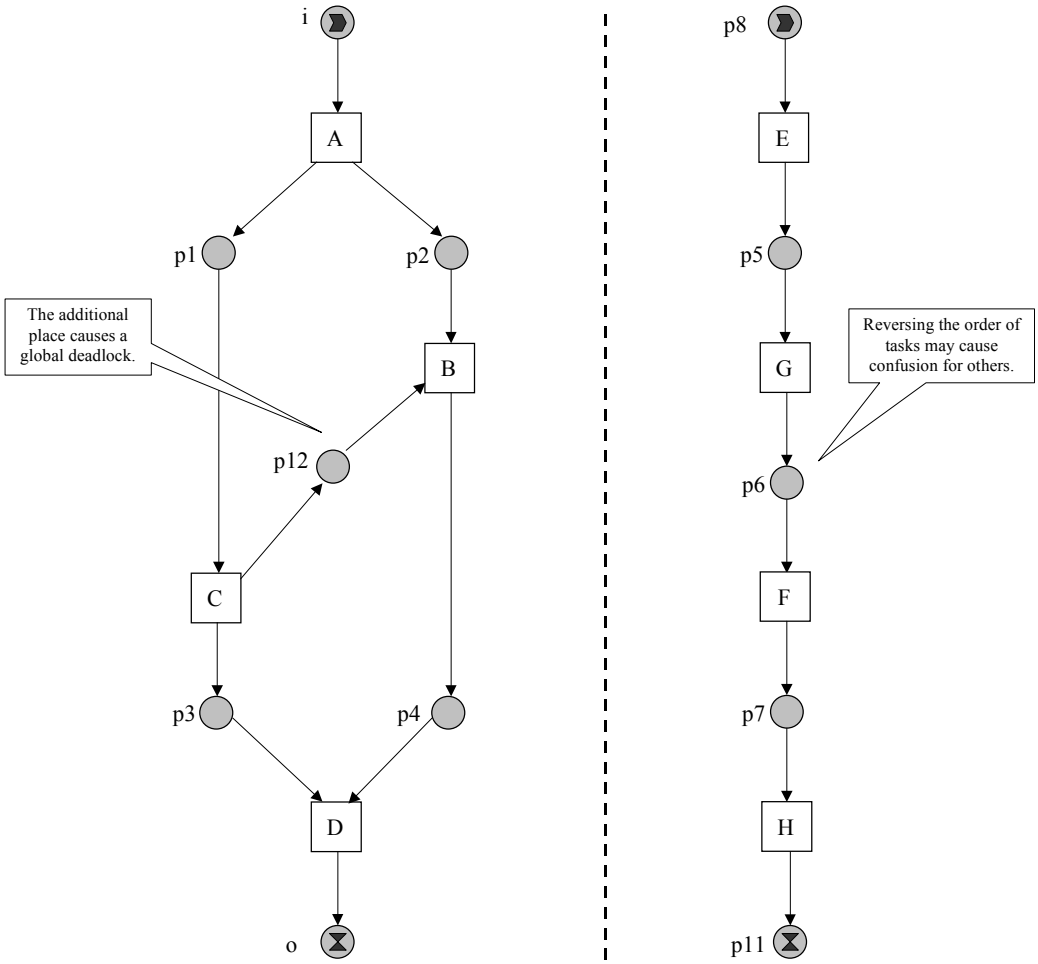

Figure 14: Both private workflows have been modified.

In this section, we focus on the problems that may be caused by local changes. For this purpose we use the following two correctness criteria:

1. The overall workflow (i.e., the combination of all extended private workflows) should be sound.
2. The ordering of tasks in the public workflow should be respected in each of the private workflows and the overall workflow.

The first criterion is apparent (see Definition 5) and can be checked using our verification tool Woflan. The second criterion is less clear. For this criterion we use the notion of *projection inheritance* (Aalst and Basten, 1997; Aalst and Basten, 2001; Basten, 1998; Basten and Aalst, 2001). Projection inheritance is a new and powerful concept that will be introduced in the remainder. The private workflows and the overall workflow should inherit certain properties of the public workflow. Projection inheritance uses *abstraction* as the primary mechanism. In the remainder of this section we focus on the use of this inheritance notion.

6.2.2 Projection inheritance

Inheritance is one of the cornerstones of object-oriented programming and object-oriented design. The basic idea of inheritance is to provide mechanisms which allow for constructing *subclasses* that inherit certain properties of a given *superclass*. In our case a *class* corresponds to a *workflow process definition* (i.e., a routing diagram) and *objects* (i.e., instances of the class) correspond to *cases*. In most object-oriented methods a class is characterized by a set of *attributes* and a set of *methods*. Attributes are used to describe properties of an object (i.e., an instance of the class). Methods symbolize operations on objects (e.g., create, destroy, and change attribute). The structure of a class is specified by the attributes and methods of that class. Note that the structure only refers to the static aspects of the interface. The dynamic behavior of a class is either hidden inside the methods or modeled explicitly (in UML the

life-cycle of a class is modeled in terms of state charts). Although the dynamic behavior is an intrinsic part of the class description (either explicit or implicit), inheritance of dynamic behavior is not well-understood. (See Basten and Aalst (2001) for an elaborate discussion on this topic and pointers to related work.) Given the widespread use of inheritance concepts/mechanisms for the static aspects, this remarkable. Every object-oriented programming language supports inheritance with respect to the static structure of a class (i.e., the interface consisting of attributes and methods). Since workflow management aims at supporting business processes, these results are not very useful in this context. To our knowledge, the work presented in (Aalst and Basten, 1997; Aalst and Basten, 2001; Basten, 1998; Basten and Aalst, 2001) is the only work which deals with inheritance of dynamic behavior in a comprehensive manner. This work is based on a particular class of Petri nets: the so-called sound workflow nets defined earlier. This class of Petri nets corresponds to workflow processes without deadlocks, livelocks, and other anomalies. Other inheritance-based approaches abstract from the causal relations between tasks/methods, i.e., the control or routing structure is not taken into account. Some of the workflow management systems available claim to be object-oriented and thus provide some support for inheritance. For example, the workflow management system InConcert (InConcert) allows for building workflow class hierarchies. Unfortunately, inheritance is restricted to the attributes and the structure of the process is not taken into account. Many workflow management systems have been implemented using object-oriented programming languages. However, these systems do not offer object-oriented mechanisms such as inheritance to the workflow designer or the designer has to program code to benefit from the object-oriented features provided by the host language. Nevertheless, we think that inheritance is a very useful concept for workflow management. Therefore, we advocate the use of the inheritance notions presented in (Basten and Aalst, 2001) and illustrate the usefulness by tackling the problem of being able to change private workflows without jeopardizing the correctness of the overall workflow.

First we define an inheritance notion 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 Basten and Aalst (2001) we identify 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* protocol inheritance. Life-cycle inheritance is the most liberal from of inheritance, i.e., protocol and/or projection inheritance implies life-cycle inheritance. In this paper, we only consider projection inheritance. This 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.*

For distinguishing x and y 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* (Basten, 1998; Glabbeek and Weijland, 1996) 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. Note that the subclass typically contains more tasks than the superclass. A formal definition of projection inheritance is beyond the scope of this paper. (The definition builds on branching bisimulation equivalence and an abstraction operator which renames a given set of tasks to τ.) The interested reader is referred to Basten and Aalst (2001).

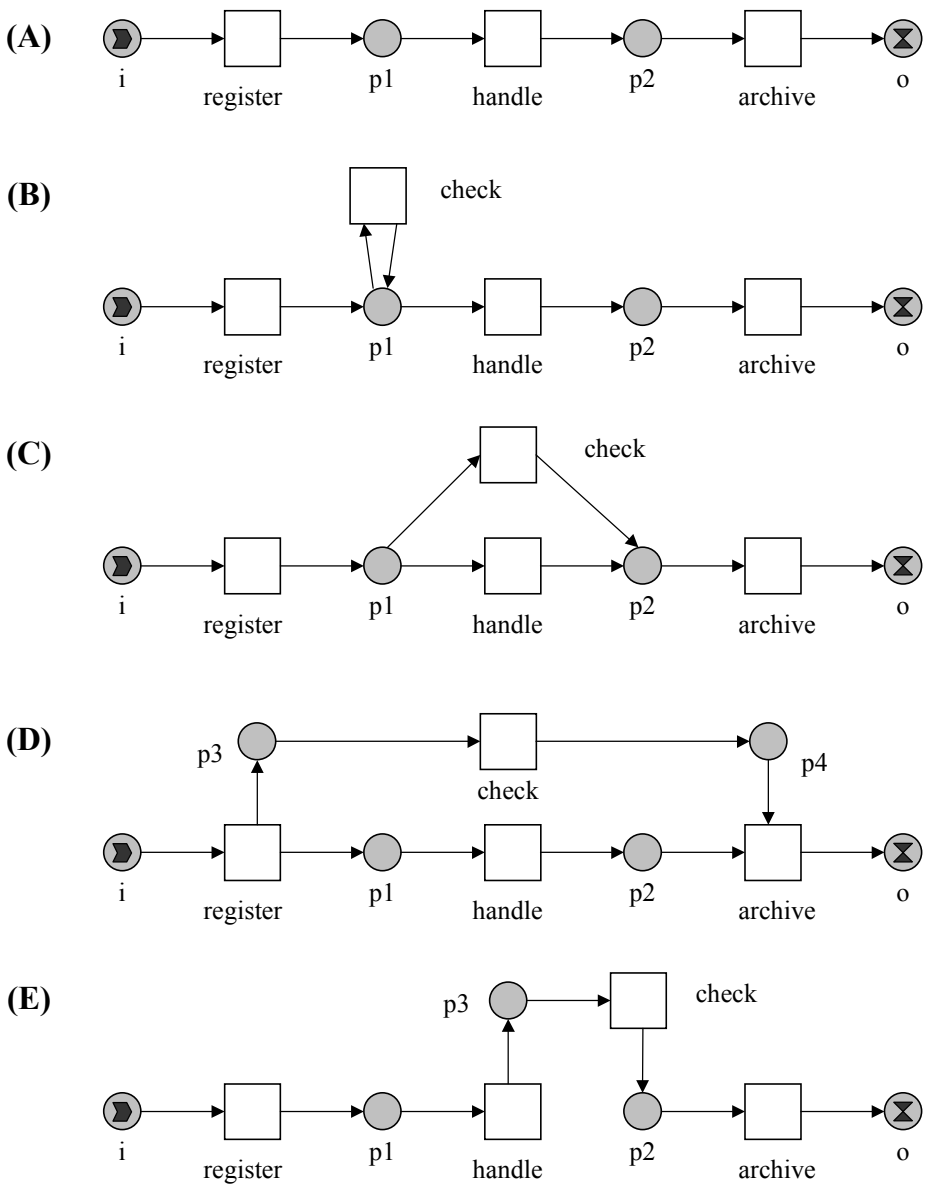

Figure 15: Five routing diagrams describing variants of a simple workflow process.

Figure 15 shows five WF-nets. Workflow process (A) consists of three sequential tasks: *register*, *handle*, and *archive*. Each of the other workflow processes extends this process with one additional task: *check*. In workflow process (B) task *check* can be executed arbitrarily many times between *register* and *handle*. Workflow process (B) is a subclass of workflow process (A) with respect to projection inheritance; if task *check* is abstracted from, then the two processes behave equivalently (i.e., are branching bisimilar, cf. Glabbeek and Weijland (1996)). Workflow process (C) is not a subclass with respect to projection inheritance; hiding task *check* introduces the possibility to skip task *handle* and thus change the actual behavior. Workflow process (D) is a subclass of workflow process (A) with respect to projection inheritance; hiding this task results in two equivalent processes. Workflow process (E) is a subclass of workflow process (A) with respect to projection inheritance; the detour via task *check* can be hidden thus yielding an observable behavior identical to (A). We also mentioned three other notions of inheritance. All workflow processes are a subclass of (A) with respect to life-cycle inheritance. Only workflow process (B) is a subclass with respect to protocol/projection inheritance and both (B) and (C) are subclasses of (A) with respect to protocol inheritance.

For a formal definition of the inheritance preserving transformation rules used to create workflow subclasses we refer the reader to Basten and Aalst (2001). We will use the WF-nets shown in Figure 15 illustrate the three rules. Rule PP (iteration rule) introduces new tasks which only postpone behavior. Workflow process (B) shown in Figure 15 can be constructed from (A) by applying this rule; task *check* only postpones the execution of *handle*. Rule PJ (sequential composition rule) inserts new tasks in-between existing tasks. Workflow process (A) shown in Figure 15 can be extended to workflow process (E) using this rule. The extension can be a single task but also a complex subflow containing many tasks and all kinds of causality relations. Rule PJ3 (parallel composition rule) adds parallel behavior. Workflow process (A) shown in Figure 15 can be extended to workflow process (D) using this rule. The rules correspond to design constructs that are often used in practice, namely iteration, sequential composition, and parallel composition. Another rule that is used in the example in Section 4 is PJR, which corresponds to refinement. In Figure 8, task A is refined by A, A2 and A3. If the designer sticks to these rules, inheritance is guaranteed!

For the readers familiar with the work presented in Aalst and Basten (1997) and Basten (1998) it is useful to point out two differences between the formulation of the inheritance preserving rules in this paper and previous work. First of all, we assume WF-nets to be safe. Therefore, we can remove the free-choice requirements. A similar approach is used in Aalst and Basten (2001) to simplify the rules. Second, we omitted the labeling function. Therefore, formally, we cannot have two transitions referring to the same task, i.e., a task can only appear once in a workflow process definition. However, all the results in this paper also hold for labeled WF-nets. The labeling function has been removed to simplify the presentation.

6.2.3 Application to interorganizational workflows

Projection inheritance is a suitable notion to avoid the problems illustrated by Figure 4. Abstraction, which is the key mechanism of projection inheritance, can be used to verify whether the environment (i.e., the other business partners) can detect any changes in the private workflow. In the remainder of this section, we will show that if the designer of a local workflow sticks to the inheritance preserving transformation rules PP, PJ, PJ3, and PJR, then the overall workflow will be sound and the ordering of tasks in the public workflow will be respected in each of the private workflows and the overall workflow.
First, we define the extended private workflow.

Definition 13 (N_d^{epriv}) For each $d \in D$, $N_d^{eprv} = (P_d^{epriv}, T_d^{epriv}, F_d^{epriv})$ is a WF-net constructed by applying a sequence of inheritance preserving transformation rules PP, PJ, PJ3, and PJR on N_d^{prv}.

Clearly, N_d^{eprv} is a subclass of N_d^{prv}. Also note that N_d^{eprv} is sound because of N_d^{prv} is sound and PP, PJ, PJ3, and PJR respect soundness. The overall workflow is composed of the private workflows and the places connecting different domains.

Definition 14 (N^{total}) $N^{total} = (P^{total}, T^{total}, F^{total})$ is the union of all private workflows including communication, i.e., $N^{total} = (\cup_{d \in D} N_d^{epriv}) \cup (P^{exch}, T^{pub}, F^{pub} \cap ((P^{exch} \times T^{pub}) \cup (T^{pub} \times P^{exch})))$.

N^{total} is the workflow which is actually enacted by the cooperating business partners. Since PP, PJ, PJ3, and PJR respect soundness and the behavior via the places connecting different domains did not change as a result of local changes, the overall workflow is sound.

Theorem 1 N^{total} is a sound WF-net.

A formal proof of this theorem is omitted but can be found in Aalst (2000a). However, intuitively it is clear that the inheritance preserving transformation rules PP, PJ, PJ3, and PJR can only postpone behavior. Therefore, it is clear that the overall workflow is sound.

Theorem 2 N^{total} is a subclass of N^{pub} under projection inheritance.

The behaviors of N^{pub} and N^{epub} are identical because the implicit places do not enable nor disable the execution of tasks. In fact, N^{pub} and N^{epub} are branching bisimilar. The inheritance preserving transformations applied to N_d^{prv} can also be applied to N^{epub} and yield N^{total} directly without composing the extended private workflows. Therefore, it is easy to see that N^{total} is a subclass of N^{pub} under projection inheritance. For a formal proof we refer to Aalst (2000a).
Note that Theorem 1 and Theorem 2 correspond to the two correctness criteria mentioned at the beginning of this section. Although the transformation rules are quite general, the designer does not have to worry about creating errors as long as (s)he sticks to these rules. Consider for example Figure 16 which shows two extended private workflows cooperating in the public workflow process shown in Figure 3. In the left process (domain *L*), rule PJ has

been used to insert task *J* in-between *B* and *D* and rule PJR has been used to refine task *C* into a small subprocess consisting of four tasks. In the right process (domain *R*), rule PP (iteration rule) has been used to add task *I* which can be executed arbitrarily many times and rule PJ3 (parallel composition rule) has been used to add a parallel subprocess. Since these extensions correspond to the four inheritance preserving transformation rules, the overall workflow process is guaranteed to be sound and is also a subclass of the public workflow process shown in Figure 3.

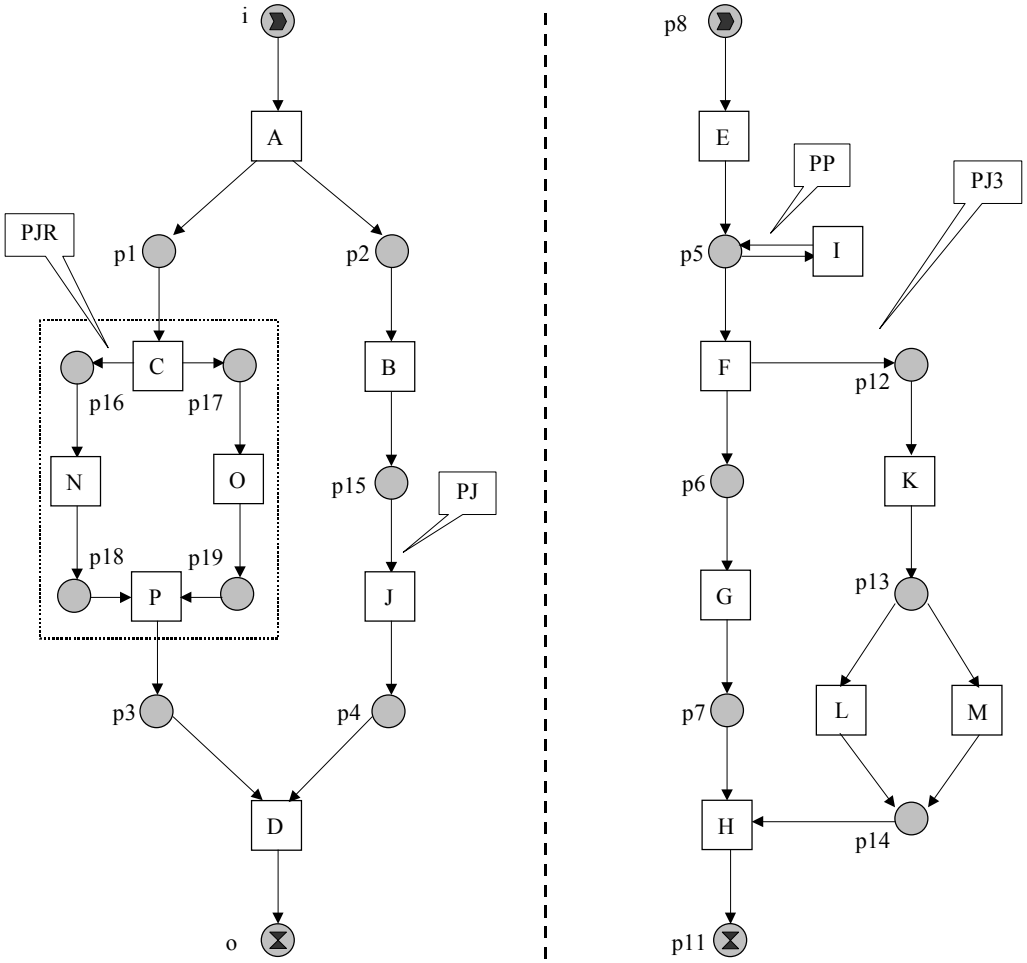

Figure 16: Both private workflows have been extended using the inheritance preserving transformation rules.

To illustrate the elegance of the approach we also define the view on the overall workflow as perceived by one of the business partners.

Definition 15 (N_d^{view}) For d∈ D, $N_d^{view} = (P_d^{view}, T_d^{view}, F_d^{view})$ is local view of the common workflow process where $N_d^{view} = N_d^{epriv} \cup (P^{pub}, T^{pub}, F^{pub} \setminus ((P_d^{priv} \times T_d^{priv}) \cup (T_d^{priv} \times P_d^{priv})))$.

N_d^{view} is the combination of the extended private workflow of domain d and the public workflow. The arcs of type $((P_d^{priv} \times T_d^{priv}) \cup (T_d^{priv} \times P_d^{priv}))$ are removed from the public workflow before joining both workflows because the extended private workflow may have replaced arcs by subnets. (Note that the rules PJ and PJR actually remove arcs.) N_d^{view} is called the *view process* of d and represents the common workflow process as it is seen by one business partner, i.e., a detailed description of the local workflow and an aggregate view of the other local workflows. Figure 17 shows N_R^{view}: the view of the overall workflow as perceived by the business partner managing domain R. The right-hand-side of the process corresponds to the extended private workflow and the left-hand-side of the process corresponds to the public workflow.

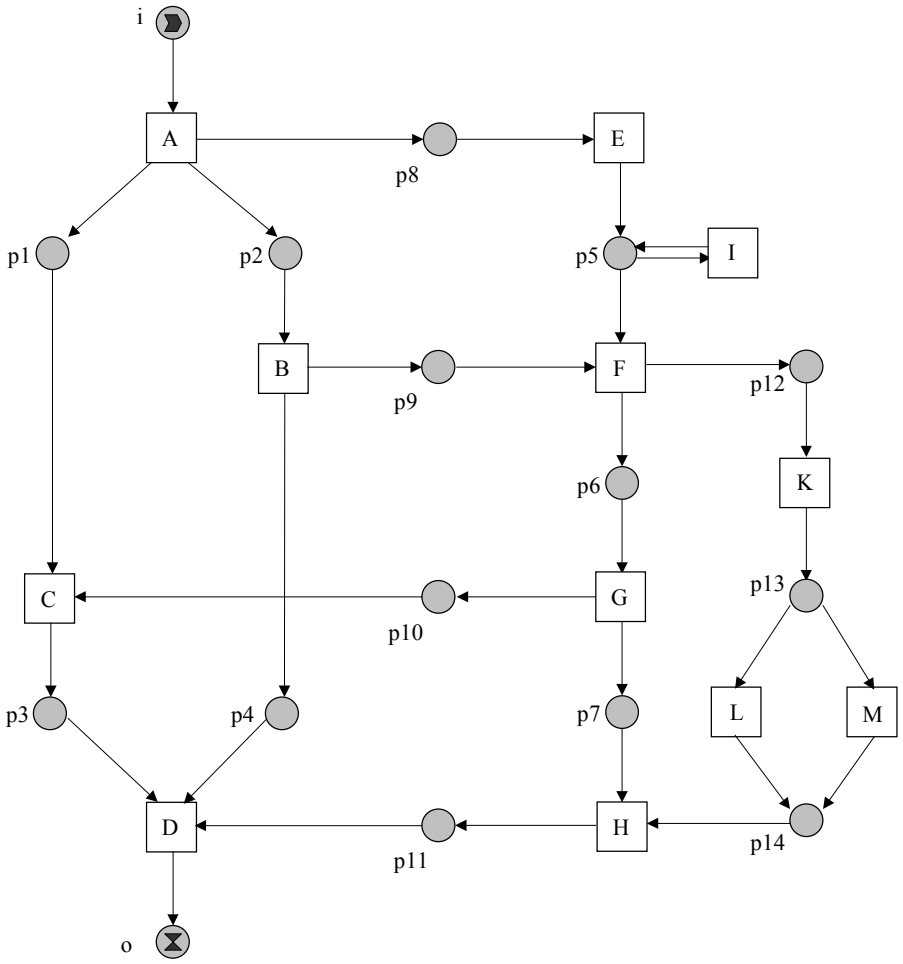

Figure 17: The overall workflow process as it is perceived from domain *R*.

The inheritance preserving transformation rules guarantee that the view process is sound.

Theorem 3 For $d \in D$, N_d^{view} is a sound WF-net.

The proof is similar to the proof of Theorem 1. The view provided by N_d^{view} is consistent and has a level of detail which is in-between N^{total} (all details) and N^{pub} (no details). The level of detail can be expressed in terms of projection inheritance. N_d^{view} is more detailed than N^{pub} because it adds tasks which are only of local interest.

Theorem 4 For $d \in D$, N_d^{view} is a subclass of N^{pub} under projection inheritance.

This theorem expresses that the view process and the public process agree on the common tasks, i.e., the tasks identified in Step 1 of the approach. N^{total} is more detailed than N_d^{view} because N^{total} is a subclass of N_d^{view} under projection inheritance.

Theorem 5 For $d \in D$, N^{total} is a subclass of N_d^{view} under projection inheritance.

The proofs of these two theorems are similar to the proof of Theorem 2: all rules used to extend the private workflow can be used to extend the public workflow. For a formal proof of these theorems we refer to Aalst (2000a). One can think of N^{pub} as the Greatest Common Divisor (GCD) and N^{total} as the least common multiple (LCM) of all N_d^{view}, $d \in D$. The public workflow is what all partners agree on (i.e., the GCD) and the overall workflow contains all details (i.e., the LCM). In Aalst and Basten (2001) it is shown that management information about the distribution of cases can be mapped onto a superclass as long as the inheritance preserving transformation rules are used to create subclasses. This means that the state of the overall workflow can be mapped onto the view process of a business partner or on the public workflow. This allows for the possibility to provide an aggregated

view of the common workflow process at different levels of detail. For a formal definition of GCD/LCM and a formal proof of the results presented in this paper we refer to a technical report (Aalst, 2000a).

The results presented in this section show that projection inheritance is a good basis for designing interorganizational workflows where the business partners only agree on a very basic common workflow process. The inheritance preserving transformation rules form the crux of the approach presented in this paper.

7. Related work

The work presented in this paper builds on well-established results in the domain of Petri nets (Desel and Esparza, 1995; Reisig and Rozenberg 1998), workflow management (Jablonski and Bussler, 1996; Koulopoulos, 1995; Lawrence, 1997), and inheritance (Basten and Aalst, 2001). The application of Petri nets to the modeling and analysis of workflows within one organization has been reported in Aalst (1998), Aalst, Desel, and Oberweis (2000), Adam, Atluri, and Huang (1998), Ellis (1979), Ellis and Nutt (1993), Jablonski and Bussler (1996), and Verbeek, Basten, and Aalst (2001). The application of Petri nets to interorganizational workflows has been discussed in Aalst (1998, 1999, 2000a, 2000b), Aalst and Anyanwu (1999), Aalst and Weske (2001), Bons, Lee, and Wagenaar (1998), Kindler, Martens, and Reisig (2000), Lee (1999), and Lee and Bons (1996). The work on Documentary Petri Nets (DPN) uses Petri nets to model and enact trade procedures (Bons, Lee, and Wagenaar, 1998; Lee, 1999; Lee and Bons, 1996). The approach described in Kindler, Martens, and Reisig (2000) also uses WF-nets and the soundness criterion to tackle problems related to the correctness of interorganizational workflows. This paper is an extended version of Aalst and Anyanwu (1999). For other applications of the method presented in this paper, we refer to Aalst and Weske (2001). Another approach based on the relation between message sequence charts (Grabowski, Graubmann, and Rudolph, 1993; Rudolph, Grabowski, and Graubmann, 1996) and WF-nets is presented in Aalst (1999). The latter approach proposes a technique to verify whether an interorganizational workflow is consistent with a specification represented in terms of message sequence charts.

There are also many other approaches that address the problem of designing and enacting interorganizational workflows. However, these methods do not focus on verification and correctness issues. For example, the work conducted in projects such as CrossFlow (Grefen, Aberer, Hoffner, and Ludwig, 2001), WISE (Lazcano, Alonso, Schuldt, and Schuler, 2001), OSM (Merz, Liberman, and Lamersdorf, 1997), and COSMOS (Merz, Liberman, and Lamersdorf, 1999) is highly relevant for the enactment of interorganizational workflows. However, these projects do not consider the correctness issues tackled in this paper. Consider for example the Common Open Service Market (COSM) infrastructure proposed in Merz, Liberman, and Lamersdorf (1999). This infrastructure proposes mobile agents. The control-flow within each agent is managed by a Petri-net-based workflow engine. Unfortunately, this work does not address the design problems mentioned in Section 3.

8. Conclusion

With the trend for business-to-business E-commerce on the rise, support for interorganizational workflows has become increasingly important. Designing such workflows in such a way that business partners preserve their autonomy is more desirable than workflow designs that tightly couple the processes of business partners. Our approach addresses these issues and we have used Petri nets as a formal basis. The approach consists of four steps and starts with the design of a *public workflow*. This public workflow serves as a contract between the parties involved. The public workflow is divided over a number of *domains* (each domain corresponds to one business partner involved) and then a *private workflow* is created for each domain. Finally, each private workflow is modified to reflect the actual process to be executed within one domain.

The approach relies heavily on a notion of inheritance named *projection inheritance*. If each private workflow is modified while preserving this inheritance notion, the resulting overall workflow is guaranteed to be free of errors and consistent with the original public workflow.

In this paper, the approach has been applied to a telecom example: the process to request bandwidth changes between two service providers. The approach has also been applied to an electronic bookstore, cf. Aalst and Weske

(2001). In the latter case there are four domains: the bookstore, the customer, the publisher, and the shipper. Clearly the approach can be applied to any situation where workflows of different organizations are intertwined and there is a need for both autonomy and carefully linked business processes.

We are currently exploring how this approach can be supported by current-generation workflow management systems. We have considered the application of the approach in the context of the workflow management system METEOR (Aalst and Anyanwu, 1999). The METEOR system is a comprehensive enterprise integration environment with support for large-scale multi-enterprise workflows that require support for distributed and heterogeneous environments and integration of legacy applications and systems. METEOR has two features that are relevant for our approach (Kang, Froscher, Sheth, Kochut, and Miller, 1999). First of all, METEOR supports so-called *foreign tasks*. These are tasks that are subcontracted to other workflow enactment services. Second, METEOR supports a mechanism that automatically partitions the workflow into parts that run on different systems and are synchronized though *external transitions* (cf. channels). The partitioning mechanism corresponds to steps 2 and 3 of the approach proposed in this paper. Step 4 is partially supported by Woflan (Verbeek, Basten, and Aalst, 2001). Woflan is a workflow analysis tool that allows for the verification and comparison of workflow processes. Woflan can verify soundness and check projection inheritance. Checking projection inheritance can be done quite efficiently (polynomial in the state space of both processes). Moreover, there is a link between METEOR and Woflan. Specifications made with METEOR can be verified and compared using Woflan. These tool developments are only the first steps towards real support for the approach described in this paper. However, they demonstrate the applicability of the approach and serve as a first proof-of-concept.

Acknowledgements

This paper builds on joint work with (former) members of the SMIS group, in particular Twan Basten, who developed much of the theory on inheritance used in this paper, and Eric Verbeek, who developed Woflan. Part of the research reported in this paper was conducted while visiting the LSDIS group (Amit Sheth) in Athens. In particular, I would like to thank Kemafor Anyanwu for her work on a previous version of this paper (Aalst and Anyanwu, 1999); the example presented in Section 5 was primarily developed by her.

References

Aalst, W. van der (1998). The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21-66.

Aalst, W. van der (1999). Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets. *Systems Analysis - Modelling - Simulation*, 34(3):335-367.

Aalst, W. van der (2000a). Inheritance of Interorganizational Workflows: How to Agree to Disagree Without Loosing Control? BETA Working Paper Series, WP 46, Eindhoven University of Technology, Eindhoven.

Aalst, W. van der (2000b). Process-oriented Architectures for Electronic Commerce and Interorganizational Workflow. *Information Systems*, 24(8):639-671.

Aalst, W. van der and Anyanwu, K. (1999). Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. In *Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC'99)*, pages 141-157, Nashville, Tennessee.

Aalst, W. van der and Basten, T. (1997). Life-cycle Inheritance: A Petri-net-based Approach. In Azéma, P. and Balbo, G., editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 62-81. Springer-Verlag, Berlin.

Aalst, W. van der and Basten, T. (2001). Inheritance of Workflows: An approach to tackling problems related to change. *Theoretical Computer Science* (to appear).

Aalst, W. van der, Desel, J., and Oberweis, A., editors (2000). *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin.

Aalst, W. van der and Weske, M. (2001). The P2P approach to Interorganizational Workflows. In *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *Lecture Notes in Computer Science*, pages 140-156. Springer-Verlag, Berlin.

Adam, N., Atluri, V., and Huang, W. (1998). Modeling and Analysis of Workflows using Petri Nets. *Journal of Intelligent Information Systems*, 10(2):131-158.

Adams, E.K., and Willetts K.J. (1996). *The Lean Communications Provider: Surviving the Shakeout through Service Management Excellence*, McGraw-Hill.

Basten, T. (1998). *In Terms of Nets: System Design with Petri Nets and Process Algebra*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Basten, T. and Aalst, W. van der (2001). Inheritance of Behavior. *Journal of Logic and Algebraic Programming*, 47(2):47-145.

Benjamin, R. and Wigand, R. (1995). Electronic markets and virtual value chains on the information superhighway. *Sloan Management Review*, pages 62-72.

Berthelot, G. (1986). Checking Properties of Nets Using Transformations. In Rozenberg, G., editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 19-40. Springer-Verlag, Berlin.

Berthelot, G. (1987). Transformations and Decompositions of Nets. In Brauer, W., Reisig, W., and Rozenberg, G., editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 360-376. Springer-Verlag, Berlin.

Bons, R., Lee, R., and Wagenaar, R. (1998). Designing trustworthy interorganizational trade procedures for open electronic commerce. *International Journal of Electronic Commerce*, 2(3):61-83.

Cole, M. (1999). *Telecommunications*, Prentice Hall.

Desel, J. and Esparza, J. (1995). *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK.

Dodd, A.Z. (1997). *The Essential Guide to Telecommunications*, Prentice Hall PTR.

Dutta, A. (1997) 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.

Ellis, C. (1979). 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. ACM Press.

Ellis, C. and Nutt, G. (1993). Modelling and Enactment of Workflow Systems. In Marsan, M. A., editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1-16. Springer-Verlag, Berlin.

Glabbeek, R. van and Weijland, W. (1996). Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555-600.

Grabowski, J., Graubmann, P., and Rudolph, E. (1993). Towards a Petri net Based Semantics Definition for Message Sequence Charts. In Faergemand, O. and Sarma, A., editors, *SDL'93 - Using Objects, Proceedings of the sixth SDL Forum*, pages 179-190. North-Holland.

Grefen, P., Aberer, K., Hoffner, Y., and Ludwig, H. (2001). CrossFlow: Cross-organizational Workflow Management in Dynamic Virtual Enterprises. *International Journal of Computer Systems, Science, and Engineering*, 15(5):277-290.

Jablonski, S. and Bussler, C. (1996). *Workflow Management: Modeling Concepts, Architecture, and Implementation,* International Thomson Computer Press.

Kalakota, R. and Whinston, A. (1996). *Frontiers of Electronic Commerce*. Addison-Wesley, Reading, Massachusetts.

Kang, M., Froscher, J., Sheth, A., Kochut, K., and Miller, J. (1999). A Multilevel Secure Workflow Management System. In *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Kindler, E., Martens, A., and Reisig, W. (2000). Inter-Operability of Workflow Applications: Local Criteria for Global Soundness. In Aalst, W., Desel, J., and Oberweis, A., editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 235-253. Springer-Verlag, Berlin.

Koulopoulos, T. (1995). *The Workflow Imperative*. Van Nostrand Reinhold, New York.

Lazcano, A., Alonso, G., Schuldt, H., and Schuler, C. (2001). The WISE Approach to Electronic Commerce. *International Journal of Computer Systems, Science, and Engineering*, 15(5):345-357.

Lawrence, P. editor (1997). *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York.

Lee, R. (1999). Distributed Electronic Trade Scenarios: Representation, Design, Prototyping. *International Journal of Electronic Commerce*, 3(2):105-120.

Lee, R. and Bons, R. (1996). Soft-Coded Trade Procedures for Open-edi. *International Journal of Electronic Commerce*, 1(1):27-49.

Malone, T., Benjamin, R., and Yates, J. (1987). Electronic Markets and Electronic Hierarchies: Effects of Information Technology on Market Structure and Corporate Strategies . *Communications of the ACM*, 30(6):484-497.

Merz, M., Liberman, B., and Lamersdorf, W. (1997). Using Mobile Agents to Support Interorganizational Workflow-Management. *International Journal on Applied Artificial Intelligence*, 11(6):551-572.

Merz, M., Liberman, B., and Lamersdorf, W. (1999). Crossing Organisational Boundaries with Mobile Agents in Electronic Service Markets. *Integrated Computer-Aided Engineering*, 6(2):91-104.

Reisig, W. and Rozenberg, G., editors (1998). *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin.

Rudolph, E., Grabowski, J., and Graubmann, P. (1996). Tutorial on Message Sequence Charts. *Computer Networks and ISDN Systems*, 28(12):1629-1641.

Sheth, A., Aalst, W. van der, and Arpinar, I. (1999). Processes Driving the Networked Economy: ProcessPortals, ProcessVortex, and Dynamically Trading Processes. *IEEE Concurrency*, 7(3):18-31.

Verbeek, H., Basten, T., and Aalst, W. van der (2001). Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246-279.

Zwass, V. (1996). Electronic commerce: structures and issues. *International Journal of Electronic Commerce*, 1(1):3-23.

Bio sketch

Wil M.P. van der Aalst is a full professor of Information Systems and head of the Department of Information and Technology of the Faculty of Technology Management of Eindhoven University of Technology. He is also a part-time full professor at the Computing Science department of the same university and has been working as a part-time consultant for Bakkenist for several years. His research interests are information systems, simulation, Petri nets, process models, workflow management systems, verification techniques, enterprise resource planning systems, computer supported cooperative work, and interorganizational business processes.