Modeling *E*-service Orchestration through Petri Nets

Massimo Mecella[1], Francesco Parisi Presicce[2], and Barbara Pernici[3]

[1] Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica
mecella@dis.uniroma1.it
[2] Università di Roma "La Sapienza", Dipartimento di Informatica
parisi@dsi.uniroma1.it
[3] Politecnico di Milano, Dipartimento di Elettronica e Informazione
barbara.pernici@polimi.it

Abstract. B2B interaction requires new forms of coordination between participating organizations. Indeed, the main requirement is that the autonomy of each participating partner is preserved during the interaction, guaranteeing at the same time that the overall goals of the common process are reached. Mechanisms for regulating distributed workflow evolution when the workflow is composed of the invocation of *E*-services of different organizations are needed. The *E*-service orchestration model proposed in this paper provides a mechanism for supporting control of process evolution in terms both of control and data flows, and for distributing and assigning process responsibilities.

1 Introduction

Current network technologies allow the development of new interaction business paradigms, such as virtual enterprises: different companies pool together their services to offer more complex, added-value products and services; network technologies and Internet make services readily accessible and thus they allow to compose virtual enterprises in very flexible ways. Although such a new business paradigm has initially emerged in the business context, indeed it has been spreading in many other contexts, e.g., for the definition of what is referred to as *E*-government [1].

Systems supporting such models are commonly referred to as Cooperative Information Systems (CIS's) [2]; various approaches are proposed for the design and development of CIS's: schema and data integration techniques, agent-based methodologies and systems, business process coordination and service-based systems (e.g., [3]). In the latter case, cooperation among different organizations is obtained by sharing and integrating services across networks; such services, commonly referred to as *E*-services and Web-Services, are exported by different organizations as semantically well defined functionalities that allow users and applications to access and perform tasks offered by back-end business applications. By using a service-based approach, the cooperative system consists of different

A. Buchmann et al. (Eds.): TES 2002, LNCS 2444, pp. 38–47, 2002.

distributed applications which integrate the *E*-services offered by different organizations. Such an integration raises some interesting points regarding service composability, correctness, synchronization and coordination, as pointed out in [4].

In [5,6], an *E*-service framework, named PARIDE (Process-based frAmework for oRchestratIon of Dynamic E-services) has been presented, specifically addressing the issues of defining a common conceptual component model (and the related description language) for *E*-services, and defining the notions of compatibility and dynamic substitution of *E*-services, based on the concept of cooperative process. The composition of *E*-services requires the definition of rules for assembling *E*-services as needed and for the *orchestration* (also referred to as coordination or choreography) of different *E*-services. The orchestration of *E*-services should be specified through an *orchestration schema*, to be expressed in an appropriate orchestration language, which specifies interactions among them. Such an orchestration schema should act as a "script" (to be interpreted by appropriate orchestration engines) that coordinates the interactions among *E*-services. Orchestration is quite different from classical workflow management, as pointed out also in [4]: *E*-services need to be linked to the current enactment on the basis of dynamic assignments, and the task of the overall control can not be statically assigned to a single organization/system, but needs to be dynamically distributed among organizations. Indeed the aim of this paper is to introduce a model, based on Petri Nets, for describing the orchestration of *E*-services, and the related design of distributed orchestration engines, in the context of the PARIDE framework.

The remainder of this paper is as follows. In Section 2, the proposed framework for *E*-services is described, with specific focus on the distributed orchestration engines; Section 3 describes the proposed model for *E*-service orchestration based on Petri Nets. Section 4 presents an explanatory example in an *E*-government scenario. Section 5 presents related relevant research work and Section 6 concludes the paper by remarking which elements need to be further investigated.

2 The PARIDE Framework

In Figure 1 the framework for *E*-services is depicted; organizations willing to cooperate deploy software components on their *cooperative gateways* (possibly wrapping their internal legacy systems); such components realize E-*service schemas*, that are the abstract specifications of the offered services according to a conceptual component model. The coordination of different *E*-services is carried out by the *orchestration engines*, deployed on the cooperative gateways, which interpret *orchestration schemas*, possibly substituting compatible *E*-services at run-time. *E*-service schemas, orchestration schemas and component instance data are stored in a *repository*, which is accessed both by the orchestration engines and by the cooperative gateways.

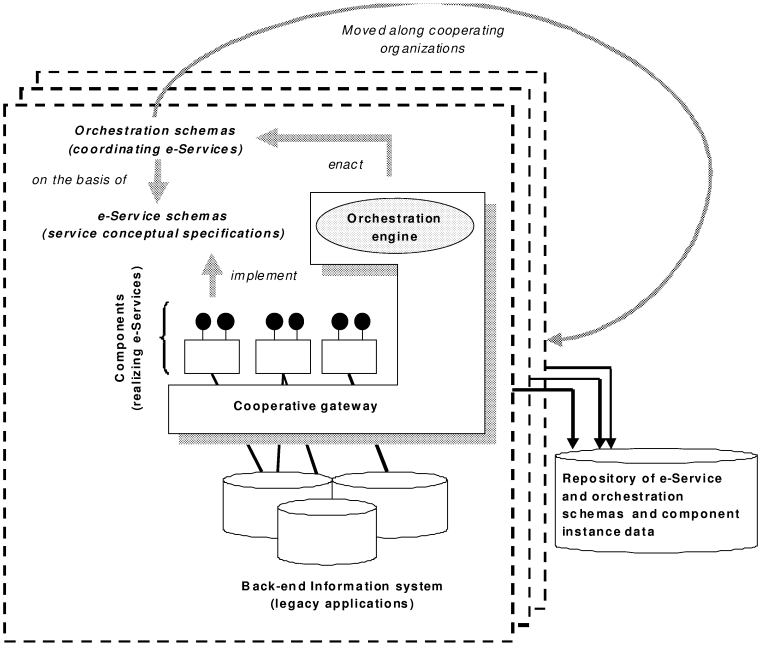


Fig. 1. The PARIDE framework

Cooperative gateways represent "where" and "how" different components are deployed. Each cooperating organization, through its cooperative gateway, offers to the others a specific element, i.e., the orchestration engine, which is able to interpret orchestration schemas and to effectively coordinate *E*-services.

Orchestration schemas are "scripts" which specifies how to create *E*-services and their correct synchronization. An orchestration schema $\mathbb{OS}$ refers to a set $(\mathbb{S}_1, \ldots, \mathbb{S}_n)$, $i = 1 \ldots n$, $n \geq 1$ of *E*-service schemas. An orchestration instance is a specific enactment of an orchestration schema; an orchestration instance $\mathcal{OI}$ of an orchestration schema $\mathbb{OS}$ uses a set of *E*-services $(\mathcal{S}_{11}, \ldots, \mathcal{S}_{m_1 1}, \ldots, \mathcal{S}_{1n}, \ldots, \mathcal{S}_{m_n n})$, where $\mathcal{S}_{ji}$, $j = 1 \ldots m_j$, $m_j \geq 1$, and $i = 1 \ldots n$, $n \geq 1$, is the j-th instance of the i-th *E*-service $\mathbb{S}_i$ which is referred to in $\mathbb{OS}$.

In our proposal, orchestration schemas are enacted by the orchestration engines, deployed by the different cooperating organizations on their cooperative gateways; a given orchestration schema, during its enactment, needs to be controlled and monitored by an organization, but such a task can not be assigned to a single organization, conversely it should be moved all along the enactment. Therefore an orchestration schema is moved from an orchestration engine to another one as the task of the overall coordination is assigned to some other organization; in a given time instant, only one and exactly one orchestration en-

gine is enacting the schema, i.e., is coordinating the different *E*-services involved in the given enactment.

An *E*-service is an event-driven component: an *E*-service reacts to messages, does some internal actions (which are not visible outside) and sends some messages. The sending and receiving of messages are the events driving the *E*-service evolution. Messages are not only simple primitive signals, but they carry parameters (i.e., structured data). In such a way, the description of an *E*-service comprises only the external interfaces (input/output data and offered services) and the conversations (i.e., sequences of operations to be invoked) for interacting with it [6].

3 *E*-service and Orchestration Nets

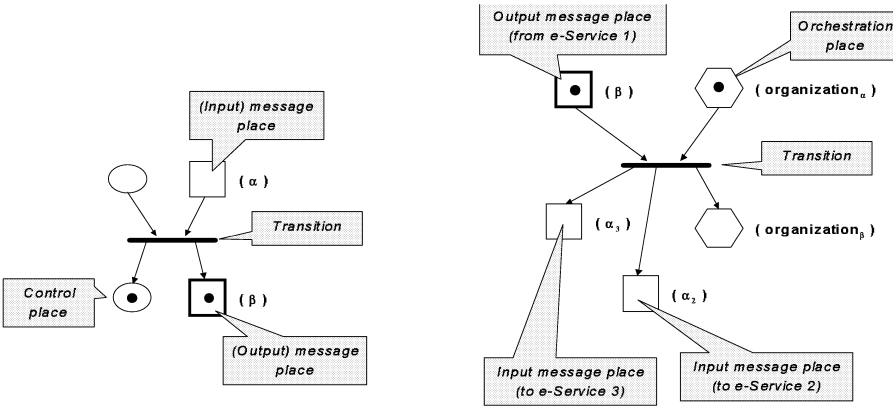
In order to precisely design the orchestration of different *E*-services, models and techniques need to be provided, specifically addressing the definition of both the control and data flows. In this paper we propose the adoption of a Petri Net-based model, thus providing to designers of the overall cooperative applications appropriate tools for correctly assembling different *E*-services; as an example, deadlock freeness of the overall process and reachability of the final configuration of the involved *E*-services can be verified by analyzing the configuration graph of the net. The proposed model allows both to specify the external behaviour of the different *E*-services (in terms of possible sequences of messages they can be involved in) and the overall orchestration, as routing of such messages and the passing of the overall controlling and monitoring task among different organizations/engines.

Hereafter, we assume that the reader knows the basics both of Petri Nets and of Coloured Petri Nets; otherwise, the reader may refer to [7,8].

An *E*-service is specified both in its static interfaces and in its behaviour. Specifically, an *E*-service communicates through messages, including both the ones the *E*-service receives and the messages it produces. Upon receiving the message α, the *E*-service does some work (and it takes some time) and then it sends the message β as output; at this point, the *E*-service is ready to accept new input messages. This basic scenario can be represented as in Figure 2(a), in which a portion of an E-*service Net* is shown: when a token is available on the upper "circle" place and another one is available on the upper "square" place, the transition can fire, thus moving a token on the lower "circle" place and another one on the lower "square" place. A "circle" place is the graphical notation for a *control place*, representing a state of the *E*-service with respect to possible message exchanges it can be involved in; conversely a "square" place is the graphical notation for a *message place*, representing an input message (square with a thin border) and an output message (square with a thick border). A token available on a message place represents the reception/production of a message.

An E-*service Net (*E-*S Net) is a net where:*

- ❒ *places are of three different types, specifically control places, referred to as CP, input message places, referred to as IMP, and output message*



(a) *e*-Service Net (b) Orchestration Net

Fig. 2. Basic elements of the Petri Net-based model

places, referred to as OMP; let us define $MP = IMP \cup OMP$ *and* $P = CP \cup MP$*;*

- ❒ *tokens placed on control (input/output message) places are referred to as control (input/output message, respectively) tokens;*
- ❒ *each place* $\in MP$ *is labeled with a message, i.e., a function* `mess`: $MP \rightarrow \mathcal{E}$ *is defined (*$\mathcal{E}$ *is the set of messages specified for the* E*-service, according to the formalization provided in [6]).*

Examples of *E*-service Nets are provided in the example of Figure 3, in which each grey area represent a different *E*-service.

The orchestration (i.e., coordination) of different *E*-services consists of correctly routing messages among *E*-services in the right sequence; this may require to manipulate messages (i.e., to compose new messages from more basic ones or to extract basic messages from complex ones). In turn such an orchestration task is not statically assigned to a single organization, but it is passed from an organization to another one as the process goes on. Therefore the orchestration schema allows to define how different messages are exchanged among *E*-services (i.e., *E*-service Nets) and how the enactment task is moved along organizations/orchestration engines.

The orchestration of different *E*-services can be represented as an *Orchestration Net*, which is a specific net connecting at least two *E*-service Nets, and specifying the routing of messages and the act of passing the task of the orchestration from an organization to another one. In Figure 2(b) a portion of orchestration is shown: when the output message β is received from an *E*-service and the orchestration task is assigned to $organization_\alpha$, the message is manipulated in order to produce two input messages α_2 and α_3 to send to two different *E*-services, and the task of the overall control (i.e., of the orchestration) is moved

to a different $organization_\beta$. An "hexagonal" place is the graphical notation for an *orchestration place*.

An Orchestration Net (O Net) is a net where:

- *places are of three different types, specifically orchestration places, referred to as* OP*, input message places, referred to as* IMP*, and output message places, referred to as* OMP*; input/output message places belong to different* E*-service Nets;*
- *each place* $\in OP$ *is labeled with an organization; the availability of a token in an orchestration place means that the task of the orchestration of the overall process is currently assigned to the organization labeling the place;*
- *for each transition* t*, there exist* $om \in OMP$*,* $im \in IMP$ *and* $p, q \in OP$ *such that* (om, t)*,* (p, t)*,* (t, im) *and* $(t, q) \in$ *set of edges of the net;* p *can be equal to* q*, whereas* om *and* im *necessarily belong to different* E*-service Nets.*

The orchestration engine of the organization, which, at a given time, is executing the task of controlling and monitoring, is in charge of manipulating the messages and correctly routing them; then, if it is specified in such a way, the engine moves the orchestration schema (i.e., the representation of the net, in the form of a XML document) to the engine of the next organization which is assigned the orchestration task.

4 Explanatory Example

In the following, we consider an example stemming from the Italian *E*-government scenario. The considered process is a simplified version of the one for "providing aid to disabled persons" [9]. A citizen who has some disabilities (e.g., due to an accident) can apply for the provisioning of aid, in the form of a disability pension. In order to start the process, the citizen needs *(i)* a certificate, provided by the City Council he lives in, assessing its residence (permanent address), and *(ii)* to fill in an application form. The residence certificate is needed in order to identify and certify which local administrations will manage the process. Such documents need to be presented to the Local Health Authority, which, after negotiating an appointment with the citizen, examines him and prepares a report. Such a report needs to be sent to the Prefecture, which has to take the effective decision based on some business rules. Meanwhile, the citizen communicates to the Prefecture his intention to apply for a disability pension, and receives an acknowledgment of this communication. Finally, when the decision is made by the Prefecture, it stores the final record of the process; in case the disability pension has been assigned to the citizen, it prepares all the documents (e.g., the pension book needed to draw money in a bank or in a post office) and delivers them to the citizen. At this point the citizen can receive the pension each month.

Currently the process lasts about eighteen months, and most of the time is spent in sending documents among administrations; they are often sent through

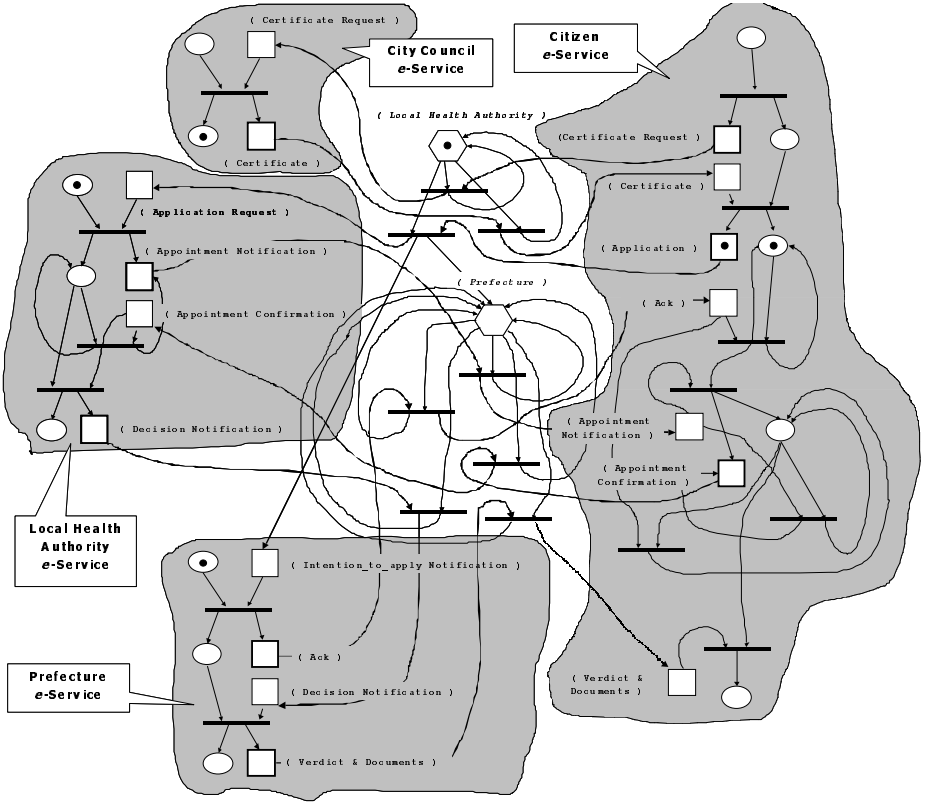


Fig. 3. The orchestration schema (*E*-service and Orchestration Nets) of the running example

ordinary mail, in some cases the citizen acts as a messenger among administrations. A radical solution would be to re-engineer the process, as for example to eliminate the activities of the Prefecture, and to let all the process be managed by the Local Health Authority; in such a case, the citizen would interact only with a single administration. Unfortunately, some issues hamper radical changes in the process, such as the fact that Local Health Authorities have neither resources and skills for managing the final document production activities, nor the legal responsibility, which in turn is assigned by law to Prefectures; moreover, modifying such laws is quite complex. Therefore the only viable solution is to support the current process through computer and network technologies, in order to speed as much as possible the interactions among the subjects involved. This in turn requires that each administration develop some *E*-services to be available on the Italian Nationwide Public Administration Network [9], and an appropriate *E*-service for the citizen, such as an application accessible through the Web and simple e-mail protocols. Such an application might be deployed on

a portal, managed by the Government, or be delivered to citizens as a special supplement in newspapers, etc. However, independently of its deployment, the citizen application is an *E*-service like the others involved in the process.

A cooperative application supporting this process would require the orchestration of the different *E*-services (i.e., the Citizen_*E*-service, the LHA_*E*-service, the PRE_*E*-service and the CC_*E*-service); the task of such an orchestration, either due to technological constraints or to organizational ones, needs to be moved along the process, e.g., at the beginning is assigned to the Local Health Authority and passed to the Prefecture after it receives the notification from the citizen (through the Citizen_*E*-service). In Figure 3 the orchestration schema (i.e., the overall net) is shown. The four involved *E*-service Nets are highlighted, and in the central part of the picture the Orchestration Net is visible.

5 Related Work

In the workflow community, much attention has been paid to adaptive and extensible systems and to the separation of concern between interface and implementation of a process and/or activity. The *e*Flow system [10] is a process management system that supports adaptive and dynamic service composition, by separating the concepts of process schema, service node and service process instance, all of them described through an XML-based description language. In [11], different activities of a multi-enterprise process are decoupled into activity interfaces and activity implementations. In [12], an approach is proposed in which B2B protocols expose the public processes whereas WfMS's implement the private processes of an enterprise. All these proposals do not address the specific issues of *E*-service coordination, which is a task with specific peculiarities, different from the ones of workflow integration, as pointed out in [4].

Some languages have been proposed for specifying *E*-service conversations, that is possible sequences of exchanged messages [13,6]; moreover, WSFL [14] is a XML-based language for the composition of Web Services. All these elements are building blocks onto which to develop an orchestration theory, but any complete proposals, as of authors' knowledge, has not been presented in the literature.

Petri Nets have been proposed for modeling workflow components (e.g., [15]) and for demonstrate specific properties in inter-organizational scenarios (e.g., [16]).

In [17] the issue of service composition is addressed in the context of Web Components, as a way for creating composite Web Services by re-using, specializing and extending existing ones. The aims of the Service Composition Specification Language (SCSL) and the Service Composition Planning Language (SCPL) are similar to the Petri Net-based model proposed in this paper, even if they differ on specific features; moreover, the Service Composition Execution Graph (SCEG) can be considered as a kind of orchestration schema.

6 Conclusions and Future Work

The model proposed in the paper allows the definition of *E*-services as conceptual components, corresponding to running services, which can be composed and used during the execution of different processes. *E*-service composition at the conceptual level needs to be defined both in terms of data flows and control flows between the *E*-services. Such an orchestration of *E*-services is a novel and difficult task, which can not be addressed with classical workflow-based technologies and methodologies. Substitution of *E*-services with compatible ones and delegation to different organizations during the enactment require the definition of an orchestration model and the development of suitable orchestration engines, distributed among the different cooperating organizations. In this paper, a framework for *E*-services has been presented, with specific focus on the orchestration engines, and a model for specifying orchestration schemas has been proposed, based on specific classes of Petri Nets.

The model proposed in the paper is being implemented and will be experimented within the *VISPO (Virtual-district Internet-based Service PlatfOrm)* Project. In the project, several services are provided for participants in a virtual district, and the orchestration model proposed in the paper has the goal of providing a reference model for orchestration of processes used by organizations participating in the virtual district. The VISPO platform has the goal of allowing participation in the activities of the district by invoking a series of available *E*-services according to the rules defined in the orchestration schema. The orchestration engines will allow controlling the correct evolution of the process, and alerting the current process responsible if exceptions are raised.

During the implementation and experimentation phases, possible representation in appropriate languages (e.g., XML) need to be carried out. Moreover the design of the orchestration engines need to be defined, in order to address specific issues such as deadlocks, possible timeouts, etc. On the basis of analysis techniques to be based on the Petri Nets, development environments for composing *E*-services and designing their orchestration will be also provided.

Acknowledgments. The work of Massimo Mecella and Barbara Pernici is supported by MIUR, "Fondo Strategico 2000" Project *VISPO*.

References

1. A.K. Elmagarmid and W.J. McIver Jr (eds.), "The Ongoing March Towards Digital Government (Special Issue)", *IEEE Computer*, vol. 34, no. 2, 2001.
2. G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, M.P. Papazoglou, K. Pohl, J. Schmidt, C. Woo, and E. Yu, "Cooperative Information Systems: A Manifesto", in *Cooperative Information Systems: Trends & Directions*, M.P. Papazoglou and G. Schlageter, Eds. Accademic-Press, 1997.
3. U. Dayal, M. Hsu, and R. Ladin, "Business Process Coordination: State of the Art, Trends and Open Issues", in *Proceedings of VLDB 2001*, Roma, Italy, 2001.

4. J. Yang, W.J. Heuvel, and M.P. Papazoglou, "Tackling the Challenges of Service Composition in *E*-marketplaces", in *Proceedings of RIDE-2EC 2002*, San Jose, CA, USA, 2002.
5. M. Mecella, B. Pernici, M. Rossi, and A. Testi, "A Repository of Workflow Components for Cooperative *E*-applications", in *Proceedings of the 1st IFIP TC8 Working Conference on* E-*commerce/*E-*business*, Salzburg, Austria, 2001.
6. M. Mecella, B. Pernici, and P. Craca, "Compatibility of *E*-services in a Cooperative Multi-Platform Environment", in *Proceedings of VLDB-TES 2001*, Rome, Italy, 2001.
7. W. Reisig and G. Rozenberg, Eds., *Lectures on Petri Nets I: Basic Models*, Springer Verlag, LNCS 1491, 1998.
8. K. Jenses, *Coloured Petri Nets. Volume 1*, Springer Verlag, EATCS Monographs on Theoretical Computer Science, 1992.
9. C. Batini and M. Mecella, "Enabling Italian *E*-government Through a Cooperative Architecture", *IEEE Computer*, vol. 34, no. 2, 2001.
10. F. Casati and M.C. Shan, "Dynamic and Adaptive Composition of *E*-services", *Information Systems*, vol. 6, no. 3, 2001.
11. H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker, "Modeling and Composing Service-based and Reference Process-based Multi-enterprise Processes", in *Proceedings of CAiSE 2000*, Stockholm, Sweden, 2000.
12. C. Bussler, "The Role of B2B Protocols in Inter-Enterprise Process Execution", in *Proceedings of VLDB-TES 2001*, Rome, Italy, 2001.
13. H. Kuno, M. Lemon, A. Karp, and D. Beringer, "Conversations + Interfaces = Business Logic", in *Proceedings of VLDB-TES 2001*, Rome, Italy, 2001.
14. F. Leymann, "Web Service Flow Language (WSFL 1.0)", IBM Document, May 2001. Available on-line (link checked October 1st, 2001): `http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf`.
15. W.M.P. van der Aalst and M. Weske, "The P2P approach to Interorganizational Workflows", in *Proceedings of CAiSE'01*, Interlaken, Switzerland, 2001.
16. V. Atluri and W.K. Huang, "A Petri Net Based Safety Analysis of Workflow Authorization Models", *Journal of Computer Security*, vol. 8, no. 2/3, 2000.
17. J. Yang and M.P. Papazoglou, "Web Components: A Substrate for Web Service Reuse and Composition", in *Proceedings of CAiSE'02*, Toronto, Canada, 2002.