Architecture Support for Flexible Chain Management

R. Seguel, R. Eshuis, and P. Grefen

Information Systems Group, School of Industrial Engineering,
Eindhoven University of Technology, The Netherlands.
{r.e.seguel,h.eshuis,p.w.p.j.grefen}@tue.nl

Abstract. In competitive markets, organizations collaborate in business chains using dynamic service outsourcing to deliver complex products and services. To enable the flexible formation of business chains, organizations can use protocol adaptation to ensure that their business protocols are compatible. In this paper, we present three different software architectures that enable the flexible formation and enactment of different chain structures, in which the protocol adaptation component is a key enabler. We show the feasibility of the approach by extending software architecture definitions from the literature for each flexible chain formation case.

Keywords: Business Chains, Dynamic Service Outsourcing, Interacting Services, Protocol Adaptor, Service Adaptation, Software Architecture

1 Introduction

Nowadays, the production of complex products and services in competitive markets involves a number of autonomous organizations [8] that collaborate in business chains. By locating the customer order decoupling point (CODP) in the business chain we identify three chain structures: demand chain, supply chain and hybrid demand/supply chain [13]. The CODP indicates how deeply the customer order penetrates into a business chain. Due to the shorter life-cycles and increasing complexity of products and services [6], the organizations in a business chain collaborate in a just-in-time fashion, using dynamic service outsourcing. In dynamic service outsourcing, an organization outsources a part of its business process, for instance order fulfillment, to a partner that is selected at the last possible moment from the marketplace [6]. This way, the organizations collaborate by invoking functionalities from each other according to their business protocols in their public process view [2, 4]. Each public process view abstracts an underlying private business processes that is executed by the organization.

In current dynamic service outsourcing approaches [7,8], the tacit assumption is made that interacting protocols are compatible: each sent message is received and processed by the other party, and thus no deadlocks occur. However, collaborating organizations have their own protocol that specifies their own way of working and they may easily have incompatible protocols. Since organizations

collaborate in a just-in-time fashion, incompatible business protocols hinder the flexible formation of business chains. To configure business chains in a flexible way the organizations can use protocol adaptation to ensure that their business protocols are compatible [13].

The goal of this paper is to present a supporting software architecture for each flexible chain formation case that we described in [13]. The software architectures are an extension and integration of two software architectures from the literature [7,8] that support the formation and enactment of supply and demand chain networks. The presented software architectures include business protocol adaptation as a key component to support the flexible configuration and enactment of business chains. The protocol adaptation component constructs an adaptor to resolve (if possible) incompatibilities between interacting services during chain formation, using any existing adaptation method [1, 10–12, 14–16].

Although there is some work on cross-organizational workflow collaboration using process views [2, 8, 9] they do not include adaptation in their architecture definitions. In this paper, we focus on adaptation of behavioral mismatches rather than interface mismatches. An interface mismatch is due to differences in the formats and specifications of messages exchanged that can be resolved using schema mapping and transformation tools [11, 15]. We will extend our approach adding interface adaptation in future work. However, we do not expect that this actually impacts the presented software architectures.

The contributions of this paper are three software architectures to support the flexible formation of business chain [13], in which the protocol adaptation component is a key enabler.

The remainder of this paper is organized as follows. Section 2 describes the three flexible chain formation cases. Section 3 presents the architecture support for flexible formation of demand chains. Section 4 details the architecture to enable the flexible configuration of supply chains. Section 5 describes the architecture for flexible hybrid demand/supply chain formation. Section 6 discusses a third-party adaptation factory and Section 7 presents the conclusions and future work.

2 Flexible Business Chain Management

There are three scenarios for flexible formation of business chains [13]. Each scenario defines the responsibility of a partner to construct a protocol adaptor to resolve (if possible) incompatibilities between their business protocols. We show in Figure 1 a business chain to illustrate the adaptation responsibility for each business chain formation scenario. The letters outside the services in Figure 1 represent the place where the protocol adaptor is constructed.

This way, in a *flexible demand chain formation* scenario [13], the service consumer defines its business protocol according to the customer business requirements. Since the CODP is moved to the last provider in the demand chain, the responsibility of constructing an adaptor is always of the provider; see ⓑ and ⓓ in Figure 1. In a *flexible supply chain formation* scenario [13], the service

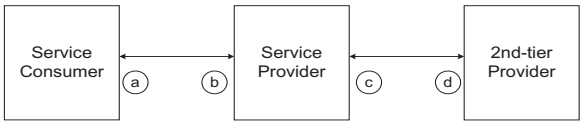

Fig. 1: Business Chain to Identify Adaptation Cases

provider sets its business protocol in conformance with market standards like SCOR [3] or eTOM [5]. For example, the service provider can be a big retail vendor like Dell. The service consumer searches the standard service provider in the marketplace to define its business protocol. In the supply chain the CODP is moved to the service consumer, and thus the responsibility of building an adaptor is always of the service consumer; see ⓐ and ⓒ in Figure 1. In a *flexible hybrid demand/supply chain formation* scenario [13], the service consumer and service provider form a demand chain while the service (1st-tier) provider and the 2nd-tier provider form a supply chain. Then, the CODP is at the service provider. This way, the responsibility of building an adaptor is always of the service (1st-tier) provider: it has to build an adaptor to resolve mismatches with the service consumer and 2nd-tier provider; see ⓑ and ⓒ in Figure 1.

To describe the flexible formation of a business chain, we explain the hybrid demand/supply chain case since it includes the other two business chain cases. We illustrate this case in Figure 2, in which the companies W and X operate in demand chain mode and companies X and Y operate in supply chain mode. Then, the company X constructs a protocol adaptor to resolve the mismatches with companies W and Y, using one of the adaptation methods presented in [1, 10–12, 14–16]. In this example we use the method presented in [14]. Note that the communication of messages is shown in the figure by the arrows crossing the organization borders, indicating send (source) and receive (target) actions. The company X constructs the protocol adaptors at the public process view and it deploys each adaptor in front of its business protocols. Then, the company X offers the protocol adaptors and its business protocol in the marketplace. Next, the companies W and Y select X, and then they deploy the business protocols in the architecture components.

3 Architecture for Flexible Demand Chain Formation

The CrossWork architecture [8] was developed to support the dynamic formation and enactment of a Network of Automotive Excellence. In the CrossWork business scenario, the service consumer is an Original Equipment Manufacturer (OEM) organization like BMW or MAN. The OEM defines a certain goal or solution objective that is sent to the service provider. Then, the provider forms the chain by selecting the 2nd-tier providers from the marketplace to meet the goal. The provider composes a global business protocol with the local protocols of the 2nd-tier providers to coordinate them. Then, the provider enacts the global process that enacts the local protocols in the 2nd-tier providers to later send the

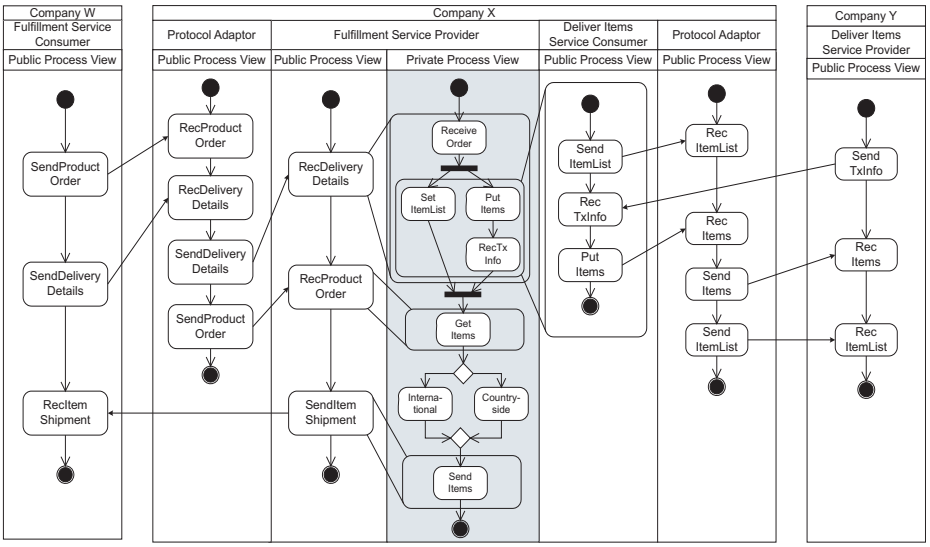

Fig. 2: Adaptation Case for Flexible Hybrid Demand/Supply Chain Formation

result back to the OEM. We extend the CrossWork architecture by adding the architecture components to support protocol adaptation. In Figure 3, we illustrate the CrossWork architecture (white boxes) plus the extension (grey boxes). The extended CrossWork architecture enables the flexible formation of a demand chain at design-time and at run-time.

At design-time, the extended CrossWork architecture is triggered by the service consumer that defines the solution objective in the goal support module according to the customer business requirements. The consumer defines its business protocol according to the goal. Then, the consumer selects the service provider from the marketplace that meets the goal. The consumer sends the goal definition and its business protocol to the service provider. Then, the service provider checks the compatibility between its business protocol and the consumer protocol. If the protocols are incompatible, the adaptor factory module constructs a protocol adaptor to resolve mismatches. The adaptor factory implements the adaptation method for tightly and loosely coupled interacting services [1, 10–12, 14–16]. Then, the provider deploys the business protocol in the protocol enactment module and the protocol adaptor in the adaptor enactment module. Similarly, the service consumer deploys its business protocol in the protocol enactment module.

Next, the service provider decomposes the goal into a required set of protocols (component services) in the goal support module. Then, the team formation module finds the 2nd-tier providers in the marketplace according to the set of protocols. Next, the composition module composes the set of protocols into a global business protocol. Note that the 2nd-tier providers are not only selected by protocol compatibility. It means that while there are parts of the global

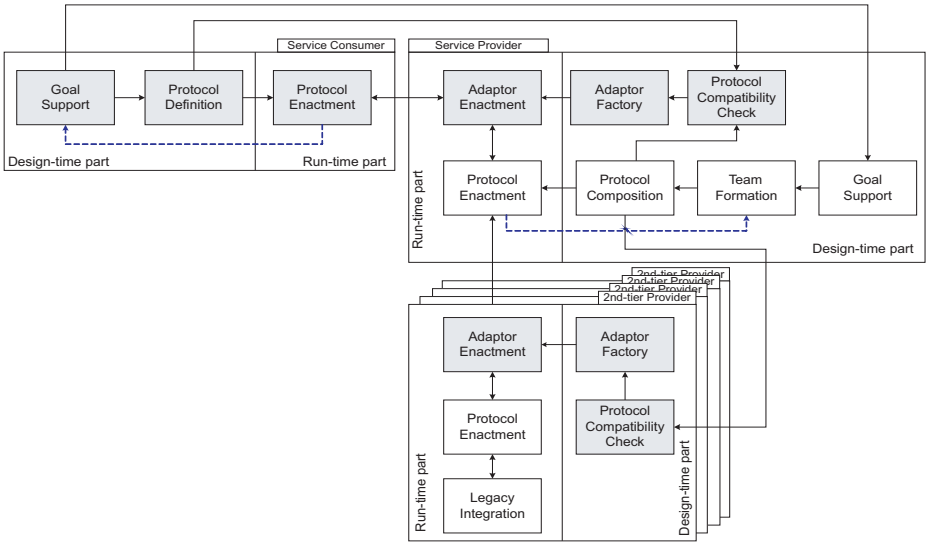

Fig. 3: Architecture Extension of CrossWork [8] for Flexible Demand Chain Management

protocol that are compatible with the 2nd-tier providers, there are other parts that are incompatible with the 2nd-tier provider protocols. This way, the 2nd-tier providers check the compatibility between their business protocols and the set of protocols that compose the global protocol. If they are incompatible, the adaptor factory at the 2nd-tier providers generates a protocol adaptor. Next, each 2nd-tier provider deploys the business protocol in the protocol enactment module and the adaptor in the adaptor enactment module. Finally, the service provider deploys the global protocol in the protocol enactment module. This way, the protocols can be later executed by the service consumer and provider, enacting the demand chain with the 2nd-tier providers.

At run-time, the extended CrossWork architecture enables the formation of the demand chain when the consumer business protocol is being enacted. This is illustrated with the 'reverse' dotted arrow from the protocol enactment module to the goal support module in Figure 3. This way, the consumer stops the business protocol execution to configure the demand chain (at design-time) to later continue the enactment of the business protocols. Similarly, the service provider can configure the demand chain with 2nd-tier providers when its protocol is being executed. Note that if the service provider acts as integrator only [6], then the adaptation is only needed at the 2nd-tier provider. The service provider uses the protocol sent by the consumer as global business protocol. Then, the service provider is protocol compatible with the service consumer since they interact to only synchronize the control flow of the global business protocol. The global business protocol interacts with the 2nd-tier providers, which can need adaptation. In the basic CrossWork architecture, if the global protocol cannot be

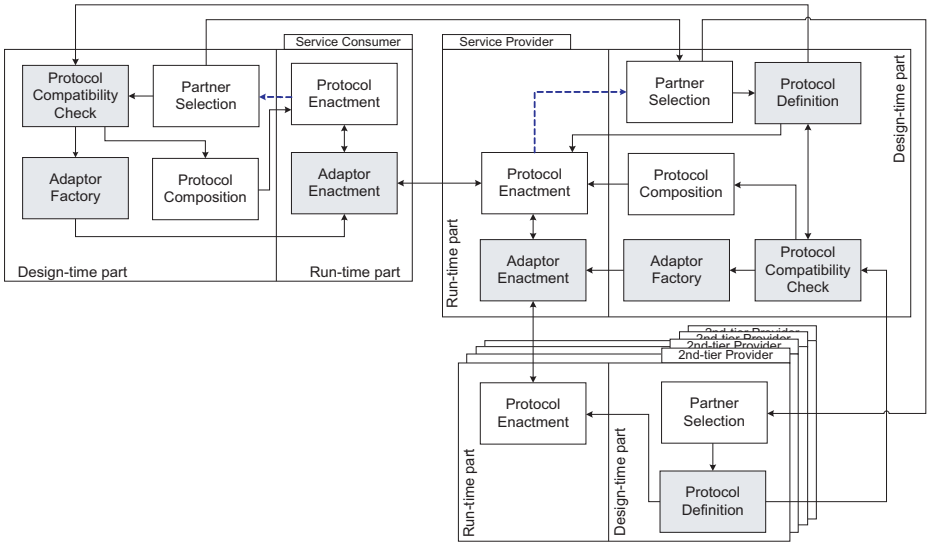

Fig. 4: Architecture Extension of CrossFlow [7] for Flexible Supply Chain Management

composed then the system backs up to the team formation or even to the goal support module to correct it [8]. However, in the extended architecture, these backtrack steps are needed only if the adaptor factory module at the 2nd-tier provider indicates the mismatch cannot be resolved and no adaptor can be constructed [12, 14]. Note that technically the business protocols and the adaptor can be enacted using the same enactment engine or two different enactment engines.

4 Architecture for Flexible Supply Chain Formation

The CrossFlow architecture [7] was developed to support the configuration of a supply chain, using dynamic service outsourcing. In the CrossFlow business scenario, the service consumer outsources a non-core part of its business protocol to a service provider. The service consumer takes the decision of outsourcing during the execution of its business protocol, and thus it selects the provider at the last possible moment. This way, the basic CrossFlow architecture is mainly focused on run-time supply chain configuration. Note that the architecture was developed in the pre web services era, but it conceptually supports the collaboration of two interacting services which share the business protocols at the public process view. We extend the CrossFlow architecture to support the flexible configuration of a supply chain at design-time and at run-time. We illustrate the basic architecture (white boxes) plus the extension (grey boxes) in Figure 4.

At design-time, the extended CrossFlow architecture is triggered by the consumer that selects the service provider from the marketplace to outsource its

protocol. Then, the provider sends its standard business protocol to the consumer that checks the compatibility with its protocol. If the business protocols are incompatible, then the service consumer constructs a protocol adaptor. Next, the service consumer couples the outsourced protocol part with its business protocol in the composition module for later deployment in the enactment module. Then, the consumer deploys the coupled business protocol in the protocol enactment module and the adaptor in the adaptor enactment module. Finally, the service provider deploys its business protocol in the enactment module after it sent the protocol definition. Similarly, the service provider can outsource part of its protocol by selecting 2nd-tier providers from the marketplace. Then, the 2nd-tier providers send their standard business protocol to the provider that checks the compatibility with its protocol. If the protocols are incompatible, then the provider adaptor factory module generates the protocol adaptor. Then, the provider couples the outsourced protocol part with its business protocol in the composition module. Next, the provider deploys the coupled protocol and the adaptor in the enactment modules while the 2nd-tier providers deploy their protocols too. This way, the protocols can be later executed by the service consumer and provider, enacting the supply chain with the 2nd-tier providers.

At run-time, the extended CrossFlow architecture supports the configuration of the supply chain when the consumer business protocol is being enacted. This is illustrated with the 'reverse' dotted arrow from the protocol enactment module to the partner selection module in Figure 4. Therefore, the consumer stops the business protocol execution to configure the supply chain (at design-time) to later continue the enactment of the business protocols. Then, the service provider can also configure a supply chain with 2nd-tier providers when its protocol is being enacted. Note that if the service provider acts as integrator only [6], then the adaptation is only needed at the service consumer. The service provider preselects the 2nd-tier providers before it is selected by the consumer. Once the provider is selected, it sends the standard protocol to the consumer. The service provider is protocol compatible with the 2nd-tier providers since they interact to only synchronize the control flow of the business protocol. On the other hand, the consumer protocol interacts with the standard business protocol, which can need adaptation. Like the extended CrossWork architecture, the CrossFlow architecture technically support the enactment of the business protocols and the adaptor using the same enactment engine or two different enactment engines.

5 Architecture for Flexible Hybrid Demand/Supply Chain Formation

To support the flexible configuration of a hybrid demand/supply chain, we define the architecture as an extension of the CrossWork (demand chain) and CrossFlow (supply chain) architectures. We illustrate the architecture for flexible formation of a hybrid demand/supply chain in Figure 5. It supports the configuration of the hybrid chain at design-time and at run-time.

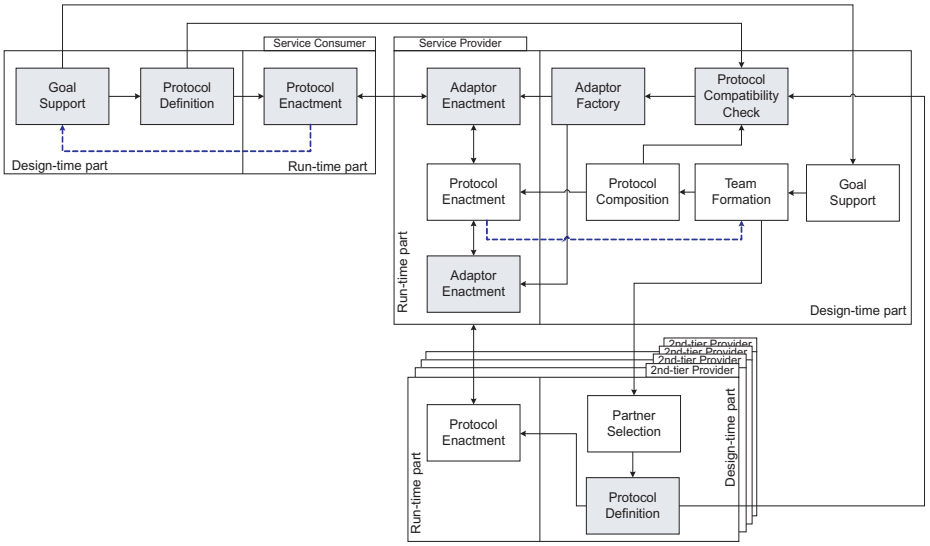

Fig. 5: Architecture for Flexible Hybrid Demand/Supply Chain Management

At design-time, the architecture is triggered by the service consumer that configures a demand chain with the provider. The consumer defines the solution objective in the goal support module according to the customer business requirements. Then, the consumer defines its business protocol according to the goal and selects the service provider from the marketplace that meets the goal. Then, the consumer sends the goal definition and its business protocol to the service provider. Next, the service provider checks the compatibility between its business protocol and the consumer protocol. If the protocols are incompatible, the adaptor factory module constructs a protocol adaptor to resolve mismatches. Then, the provider deploys the business protocol in the protocol enactment module and the protocol adaptor in the adaptor enactment module while the service consumer deploys its business protocol in the protocol enactment module.

At this stage, the service provider configures a supply chain with 2nd-tier providers. Next, the service provider decomposes the goal into a required set of protocols (component services) in the goal support module. Then, the team formation module finds the 2nd-tier providers in the marketplace according to the set of protocols. Next, the composition module composes the set of protocols into a global business protocol. Then, the 2nd-tier providers send their standard business protocol to the provider that checks the compatibility with its protocol. If the protocols are incompatible, then the provider adaptor factory module generates the corresponding protocol adaptors. Finally, the service provider deploys the global protocol and the adaptors in the enactment modules while each 2nd-tier provider deploys its protocol in the enactment module too. This way, the protocols can be later executed by the services by enacting the demand chain

between the consumer and provider and the supply chain between the provider and 2nd-tier providers.

At run-time, the extended architecture supports the configuration of the hybrid demand/supply chain when the consumer and provider protocols are being enacted. This is illustrated with the 'reverse' dotted arrow in Figure 4. The consumer stops the business protocol execution to configure the demand chain (at design-time) to later continue the enactment of the business protocols. Similarly, the service provider stops the protocol execution to configure the supply chain (at design-time) with the 2nd-tier providers to later continue with the protocol enactment.

Note that if the service provider acts as integrator only [6], then the adaptation is needed for compatibility with either the 2nd-tier providers or the service consumer. In both cases the adaptor is constructed and deployed by the service provider. In the first case, like in the extended CrossWork case, the service provider uses the protocol sent by the consumer as global business protocol. Then, the provider and consumer protocols are compatible since they interact to only synchronize the control flow. Thus, the global business protocol parts interact with the 2nd-tier providers, which can need adaptation. In the second case, like in the extended CrossFlow case, the service provider pre-selects the 2nd-tier providers before it is selected by the consumer. Then, the provider has to generate an adaptor if the consumer and provider protocols are incompatible. The service provider and the 2nd-tier providers are compatible since they interact to only synchronize the control flow of the global business protocol parts.

6 Third-party Adaptor Factory

The architectures that we defined for flexible formation of business chains support the participation of a trusted third-party that provides adaptation as a service (AaaS). The third-party is an adaptor factory that constructs and enacts protocol adaptors to support the flexible configuration of business chains. This way, the service consumer, service provider or 2nd-tier providers can buy the adaptation service from the trusted third-party, and thus they do not need to add the adaptor factory module in their architectures themselves. The participation of the third-party does not cause conceptual changes to the architectures defined previously, but technically it needs to add the technology to ensure quality of services and security, which are outside the scope of this paper.

7 Conclusions and Future Work

We have presented three software architectures that considers protocol adaptation component as a key enabler of flexible chain formation. Any existing protocol adaptation approach can be used to realize the actual adaptation. The presented architectures enables the flexible formation of business chains between organizations that collaborate in a just-in-time fashion to meet a solution objective, which is very important in competitive markets.

There are several directions for future work. We plan to extend our approach adding interface adaptation to the adaptor factory. We are currently extending our approach to define a reference architecture in which the presented three software architectures can be described. Moreover, we will extend our approach to deal with adaptation of running business chains that deadlock due to the propagation of changes on the partner business protocols.

References

1. A. Brogi and R. Popescu. Automated generation of BPEL adapters. In *Proc. ICSOC'06*, pages 27–39. Springer, 2006.
2. D. Chiu, S. Cheung, S. Till, K. Karlapalem, Q. Li, and E. Kafeza. Workflow view driven cross-organizational interoperability in a web service environment. *Information Technology and Management*, 5(3-4):221–250, 2004.
3. Supply Chain Council. SCOR: Supply-Chain Operations Reference model; version 9.0, 2008. http://www.supply-chain.org.
4. R. Eshuis and P. Grefen. Constructing customized process views. *Data and Knowledge Engineering*, 64(2):419–438, 2008.
5. TM Forum. eTOM: enhanced Telecom Operations Map; release 8.0, 2008. http://www.tmforum.org/.
6. P. Grefen. *Mastering E-Business*. Routledge, 2010.
7. P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises. *Computer Systems Science & Engineering*, 1(5):277–290, 2000.
8. P. Grefen, R. Eshuis, N. Mehandjiev, G. Kouvas, and G. Weichhart. Internet-based support for process-oriented instant virtual enterprises. *IEEE Internet Computing*, 13(6):65–73, 2009.
9. D-R. Liu and M. Shen. Business-to-business workflow interoperation based on process-views. *Decision Support Systems*, 38(3):399–419, 2004.
10. R. Mateescu, P. Poizat, and G. Salaün. Adaptation of service protocols using process algebra and on-the-fly reduction techniques. In *Proc. ICSOC'08*, pages 84–99. Springer, 2008.
11. H.R. Motahari Nezhad, G.Y. Xu, and B. Benatallah. Protocol-aware matching of web service interfaces for adapter development. In *Proc. WWW'10*, pages 731–740. ACM, 2010.
12. R. Seguel, R. Eshuis, and P. Grefen. Constructing minimal protocol adaptors for service composition. In *Proc. WEWST '09*, pages 29–38. ACM, 2009.
13. R. Seguel, R. Eshuis, and P. Grefen. Business protocol adaptation for flexible chain management. In Robert Meersman, Tharam Dillon, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2010*, volume 6426 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2010.
14. R. Seguel, R. Eshuis, and P. Grefen. Minimal protocol adaptors for loosely coupled services. In *Proc. ICWS'10*, pages 417–424. IEEE, 2010.
15. Z. Shan, A. Kumar, and P. Grefen. Towards integrated service adaptation - a new approach combining message and control flow adaptation. In *Proc. ICWS '10*, pages 385–392. IEEE, 2010.
16. D.M. Yellin and R.E. Strom. Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19:292–333, 1997.