RC22728 (W0302-089) February 20, 2003
Computer Science

IBM Research Report

Web Service Support for Dynamic Business Process Outsourcing

Paul Grefen[1], Heiko Ludwig[2], Asit Dan[2], Samuil Angelov[1]

[1]University of Twente, The Netherlands

[2]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g. , payment of royalties). Copies may be requested from IBM T. J. Watson Research Center ,
P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home .

Web Services Support for Dynamic Business Process Outsourcing[§]

Paul Grefen[1], Heiko Ludwig[2], Asit Dan[2], Samuil Angelov[1]

[1]University of Twente, The Netherlands
[2]IBM T.J. Watson Research Center, USA

Abstract

Outsourcing of business processes is crucial for organizations to be effective, efficient and flexible. To meet fast-changing market conditions, dynamic outsourcing is required, in which business relationships are established and enacted on-the-fly in an adaptive, fine-grained way unrestricted by geographic distance. This requires automated means for both the establishment of outsourcing relationships and for the enactment of services performed in these relationships over electronic channels. Due to wide industry support and the underlying model of loose coupling of services, Web services increasingly become the mechanism of choice to connect organizations across organizational boundaries. This paper analyzes to which extent Web services support the dynamic process outsourcing paradigm. We discuss contract-based dynamic business process outsourcing to define requirements and then introduce the Web services framework. Based on this, we investigate the match between the two. We observe that the Web services framework requires further support for cross-organizational business processes and mechanisms for contracting, QoS management and process-based transaction support and suggest ways to fill those gaps.

1. Introduction

Nowadays, many organizations focus on their core business processes – they concentrate on doing what they are best in to keep (or build) a competitive advantage. Consequently, they have to buy business process services from partners in the market to perform the additional parts of the process required to reach their business goals. We call this the business process outsourcing paradigm. In this paradigm, the outsourcing organization is referred to as service consumer, the service implementing organization as service provider. The details of business process outsourcing are specified in a contract between both parties. The combination of service consumer and service provider can be seen as a virtual enterprise that presents itself to a third party (for example a customer) as a single entity.

Traditionally, these virtual enterprises have a more or less stable character over time – service outsourcing is static in the sense that it takes place between fixed pairs of organizations under conditions stated in long-term contracts. In dynamic e-commerce settings, however, players in a market and competitive situations change that fast, that a more dynamic approach is required to service outsourcing to create or retain a competitive position for a commercial organization. This means that in business process outsourcing, service consumers dynamically determine which service providers to use in the enactment of their business processes. We call this dy-

[§] *Submission to the ITM Journal Special issue on Universal Enterprise Integration.*

namic business process outsourcing. This paradigm implies dynamic selection, contracting, coupling and executing of business services.

In Figure 1, dynamic business process outsourcing is illustrated. On the left, we see a service consumer that wants to outsource a part of its business process (steps D and E). On the right, we see a number of service providers offering compatible services consisting of steps that can replace D and E (steps D+ and E+). Typically, outsourced subprocesses offer added value with respect to in-house subprocesses, for example higher efficiency or flexibility (hence the '+' in the names of the outsourced process steps). Process enactment at the consumer side is dynamically linked to process enactment at the provider side. In the sequel of this paper, we present an example scenario.

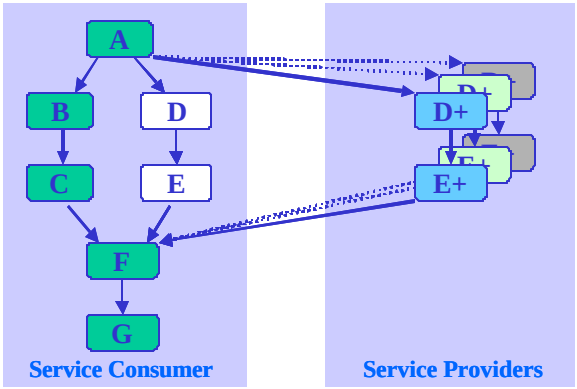

Figure 1: dynamic business process outsourcing

To enable efficient and fast dynamic business process outsourcing, electronic means are required for contracting, executing and monitoring business services. We take a look at the state of the art of these means below.

Traditionally, structured electronic collaboration between organizations has been based on Electronic Data Interchange (EDI). EDI provides a static connection through which predefined messages can be exchanged. The connection is static as the establishment of the connection is specific to a pair of collaborating organizations, making the implementation slow and costly. As the name suggests, EDI is focused on data exchange, not so much on process integration in service outsourcing.

Many developments in the area of cross-organizational workflow management target at cooperation between organizations specified at process definition time. An example is the WISE project [Alo99]. The WISE project (Workflow based Internet SErvices) at ETH Zürich aims at providing a software platform for process-based business-to-business electronic commerce, focusing on support for networks of small and medium enterprises. WISE relies on a central workflow engine to control cross-organizational processes (called virtual business processes). A virtual business process in the WISE approach consists of a number of black-box services linked in a workflow process [Alo99]. A service is offered by an involved organization and can be a business process controlled by a workflow management system local to that organization – but this is completely orthogonal to the virtual business process.

Currently, standards are emerging that allow a more dynamic connection between organizations, typically through the use of Internet and XML technology. Dynamism in cooperation is achieved through electronic intermediaries (or market places) through which service consumers and providers find each other. Well-known examples of approaches supporting cooperation are ebXML [eb01a, eb01b, eb01c] and RosettaNet [RoN02]. These are high-level frameworks that aim at complete solutions. Consequently, they introduce complex standards, e.g. choreography definitions. They are not focused on the integration of complex business processes.

Many B2B integration product vendors such as Oracle, IBM, Microsoft and Vitria are offering B2B gateways that implement coordination mechanisms to integrate above-mentioned B2B pro-

tocols with internal processes. However, they are not focused on the integration of complex business processes running at a remote provider.
In the research domain, there are developments towards the dynamic integration of business processes in a service outsourcing paradigm. An advanced approach has been developed in the CrossFlow project [Gre00]. The CrossFlow approach uses a proprietary contract specification language and a dedicated Java-based collaboration layer for support of dynamic business process outsourcing on top of standard workflow management software. These research approaches are typically of a specific nature in the sense that they rely on self-defined 'standards' and specific infrastructures.
A paradigm that is quickly gaining popularity is the Web service paradigm. This paradigm has a promise to become one of the primary standards for business application integration over the Web. Developments in Web services, however, typic ally focus on black box services that do not expose an internal process structure to their users. Also, as we will see in the sequel of this paper, a number of functionalities is missing that we require for dynamic business process outsourcing. Hence, 'pure' web services are not directly fit for business process integration as required for dynamic business process outsourcing.
In this paper, we try to bridge the business-oriented world of dynamic business process outsourcing and the technology-oriented word of Web service – in other words, we try to 'glue together' requirements pull and technology push in dynamic business process integration. The main issue that this paper addresses is: 'How can the Web services technology stack be used for dynamic business process outsourcing and what is still missing?'
The structure of this paper is as follows. In Section 2, we describe our model of dynamic business process outsourcing, which provides the requirements for this business paradigm. In Section 3, we describe the Web service technology, which we take as basic platform in this paper. In Section 4, we map the requirements of Section 2 onto the platform of Section 3. In doing so, we discuss shortcomings and suggest future developments. We end the paper with conclusions in Section 5.

2. Model of Dynamic Business Process Outsourcing

In this section, we describe our conceptual view of dynamic business process outsourcing to define the requirements that business process outsourcing poses on a software platform. We first explain the paradigm in general and present an example from the world of telecom and logistics operators. Next, we focus on the dynamic integration of business processes across organizational boundaries, required for the enactment of outsourced business processes. Finally, we show how business process services are dynamically contracted to form the formal business relationship required for the enactment of business processes. We end with a short summary of the requirements.

2.1. Dynamic business process outsourcing paradigm

In Section 1, we have introduced the concept of dynamic business process outsourcing. We use the following definition of this concept:

> *Dynamic business process outsourcing is constituted if a non-trivial part of a business process instance of an organization is executed by another organization that is determined dynamically on the basis of characteristics of the process instance and the value of context variables, the execution being governed by an explicit contract between the two organizations pertaining to that process instance only.*

The above definition leads to a number of observations:

1. We consider non-trivial business subprocesses in the outsourcing context. This is different from the simple business services without explicit internal process structure, as often addressed in Web service contexts.
2. The dynamically forged cooperation between two organizations requires automated means for the establishment and the enactment of the cooperation to obtain the required efficiency to deal with single process instances.
3. The enactment of a business process across the boundaries of the participating organizations requires dynamic business process integration. We address this issue in Section 2.3.
4. The contract defining the cooperation requires a context in which both organizations operate. Standardization and legal aspects related to the contract are determined by this context. We address this issue in Section 2.4.

In our definition of dynamic business process outsourcing, we use a 'purely dynamic' approach, in which all contact between collaborating organizations is dynamic, i.e., there is no pre-established bilateral relation between the organizations. Semi-dynamic situations can exist as well, for example by allowing service outsourcing to take place in the context of umbrella contracts covering multiple instances of process outsourcing. In the definition, contracts between parties are always completely explicit. In some situations, contracts can have a (partly) implicit character, e.g., when a contract is established implicitly by the invocation of a service. For reasons of clarity, we do not address these relaxations explicitly in this paper. Our analysis does, however, cover these aspects.

2.2. Example scenario

Service outsourcing can be found in many market segments. An example is an insurance company that focuses on the core insurance process and outsources secondary activities like customer call handling and insurance claim loss assessment to other companies. In this case, customer call handling may be outsourced statically to a fixed business partner and claim loss assessment dynamically to one of a set of possible business partners. Other examples can be found in the logistics domain, where companies outsource their logistic sub-processes to specialist organizations in this field.
Our example is based on a real-world logistics scenario in the telecom industry developed in the CrossFlow project [Dui00, Gre00]. At a telecom operator, packages of mobile (GSM) telephones and subscriptions are sold, the network connection is set up and administrative steps are taken, and the process is finalized by contacting the customer to check satisfaction – all activities that are core activities for the telecom operator. The physical logistics part of the process, i.e., actually delivering the telephone to the customer, is not a core activity of a typical telecom operator. This activity is dynamically outsourced to a specialized logistics provider. The logistics provider is selected in the course of the sales process, based on shipping requirements of the telecom operator and current offers of the logistics providers.

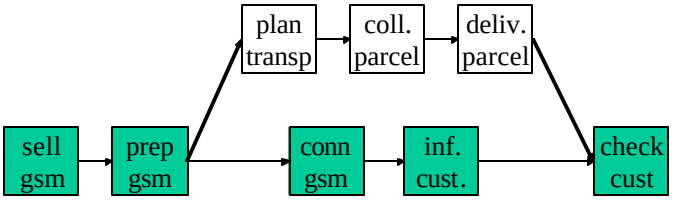

Figure 2: example business process

The example process at the telecom operator is specified in an abstract technique at the conceptual level in Figure 2. The shaded process steps represent the part of the process that is dealt with internally, the white steps the dynamically outsourced part of the process. The telecom op-

erator sells a GSM package, using telephone or web communication with its client. It then prepares the GSM for transport. After that, two parallel process branches are executed. In-house, the GSM is connected to the network and the customer is informed by mail about the connection, the personal identification code and the subscription. The process of physically delivering the GSM to the customer is outsourced to a logistics operator, who performs the delivery in three sequential steps: 'plan transport', 'collect parcel' and 'deliver parcel'. After completion of the two parallel branches, the telecom operator checks up on the customer by telephone.

2.3. Cross-organizational integration of business processes

As we have seen above, dynamic business process outsourcing requires the integration of business processes across the boundaries of organizations. We discuss the following aspects of this integration: control flow integration, data flow integration, levels of visibility of process details, transactional behavior of processes, and quality of service aspects of business processes.

2.3.1. Control flow integration

In integrating business processes, we first have to integrate the control flow, i.e., make sure that the processes are connected such that all activities in the cross-organizational overall process are executed in the right order.
Given our asymmetric model of service outsourcing with service consumer and service provider, we distinguish between four control flow interface classes:

Black box: with a black box interface, the service consumer observes the service process at the provider as a black box. This means that the consumer has no information about the way the service is executed. A black box service is modeled by a single outsourced activity in the service consumer process. This interface class is not interesting in the context of this paper – it is actually excluded by our definition of service as given in Section 2.1.

Glass box: with a glass box interface, the service consumer can observe the internal state of the outsourced process, but does not synchronize with it through control flow. This is the case in the example process as depicted in Figure 2: there are no control flow relations between in-house and outsourced activities except for start and end of the outsourced service.

Half-open box: with a half-open box interface, the state of the outsourced process is synchronized with that of the consumer's in-house process through one or more control flow relations, i.e., arrows going from consumer to provider, but not in the opposite direction. This means that the progress of execution at the provider is influenced by that of the consumer, and that the provider's autonomy is consequently reduced.

Open box: in the open box interface class, there can be arbitrary control flow relations between consumer and provider. The execution progress of both parties depends on one another and the execution autonomy of both parties is reduced. The process in Figure 3 is an example of this class: arrows go in both directions between in-house and outsourced activities.

Clearly, the chosen interface class is heavily dependent on the collaboration paradigm that is chosen in service outsourcing.

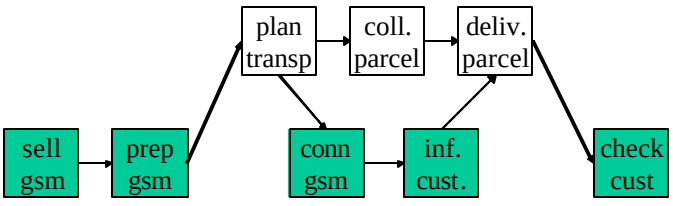

Figure 3: example business process with modified control flow

2.3.2. Data flow integration

Control flow integration, as discussed above, ensures that steps in a cross-organizational business process are executed in the right order. Apart from this control flow integration, data flow integration is required to make sure that data is transferred in the right way between organizations. Data flow integration has two faces: connection of data flows and matching of data formats of collaborating organizations.

Connection of data flows between organizations requires output interfaces of one organization to be connected to input interfaces of the other organization. In other words: a task of one organization that produces output data to be used in another task of the other organization should be connected with a data flow. Note that data flows typically exist in both directions between service consumer and provider.

Formats of data produced by one organization and consumed by the other organization should of course match. In business scenarios, these formats may be complex – they can be highly structured business documents. Both syntax and semantics of data formats must be agreed upon by service consumer and service provider. Often, this means that the format of one organization has to be mapped to that of the other [Gr02a] or that both organizations have to conform to a common standard (as traditionally the case in EDI-based communication).

2.3.3. Levels of visibility

In the specification of complex business processes, we typically find multiple aggregation levels: a coarse top-level description of a process is refined into more detailed process specifications. In dynamic business process outsourcing, the question is which aggregation level of a process to expose to external parties. Often, the most detailed level is not the most adequate level. On the one hand, competitive details of internal business organization should not be revealed to the external world. On the other hand, details not interesting to external parties should not be exposed for reasons of clarity and simplicity of interoperation.

In Figure 4, we see an example based on our telecom/logistics scenario. The service provider in this scenario has a two-level process specification: steps 'collect parcel' and 'deliver parcel' at the top level are refined into more detailed specification. The provider decides, however, to expose only the top-level process. Note that that a service provider might offer different levels under different conditions: a more detailed process view of a service – allowing more detailed process monitoring and control – might be offered for a higher price.

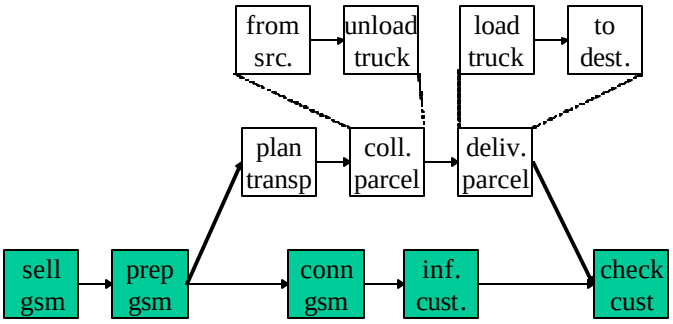

Figure 4: example business process with refined service

Apart from the above discussion with respect to aggregation levels, there may also be a difference between abstraction levels. An organization may abstract from local specifics with respects to standards used and details of technical platforms for reasons of interoperability.

2.3.4. Transactional behavior

In business processes, we distinguish steps that are to be executed in an isolated, atomic fashion. Isolation of a step in this context means that intermediate values of information manipulated by a step cannot be observed by its context. Atomicity of a step means that a step is executed in an all-or-nothing fashion, i.e., that is cannot be ended halfway. Typically, these isolated, atomic steps take a relatively short time to be executed. In our telecom logistics example, we can take the steps shown in Figure 2 (although clustering is possible).

The overall business process should not be executed in an isolated, atomic fashion (remember that a black-box control flow interface is not adequate for a business process service). Information produced during the execution of the process should be visible to the environment of the process. Controlled undo of parts of the execution of a process should be possible to recover from errors and redo the associated parts of the process – not throw away all the work spent in executing the process so far. Undo of a part of a process is often based on the execution of a compensating process, i.e., a process that undoes the effects of another process by executing compensating steps in the inverse order of the original process (as originally proposed in the Saga model [Gar87]).

The above characteristics of the execution of a business process require a two-level approach that caters for strictly transactional (isolated and atomic) steps and relaxed atomic (non-isolated and non-atomic) processes. A typical example of a two-level approach is the WIDE transaction model [Gre97]: global transactions with compensation-based rollback [Gre01] are used over nested local transactions with strict transactional characteristics. The WIDE model has been extended to a cross-organizational context in the CrossFlow project [Von00, Von03].

2.3.5. Quality of service

The decision on outsourcing a part of business process is critically dependent on whether or not a business partner can be trusted to provide an on-time reliable service. To ensure this quality of service, the service client defines jointly with the service provider a service level agreement (SLA) as part of the overall service contract. A SLA defines unambiguously the service levels agreed by the parties in terms of responsiveness, availability and other measurable metrics. The same service may be offered by a service provider at different service levels to different clients with different prices for using this service. Failure to meet this service level may result in incurring a penalty on the service provider as per the SLA.

In the telecom outsourcing example introduced in Section 2.2, the service level parameters for responsiveness include total time to perform the steps of the outsourced process. Note in this example, depending on specific business requirements, the service level metric can be defined in many different ways. First, to ensure customer satisfaction, the total delivery time for each process instance should not exceed a telecom business defined threshold, say two days. The service provider will be reluctant to set this threshold too low, since it becomes harder for the provider to meet this guarantee (due to many uncertainties, e.g., occasional breakdown of the delivery truck, unavailability of delivery person, etc.). So, for repeated execution of a business process (each instance representing a new customer), the service level may be better defined in terms of statistical guarantees, e.g., average delivery time, percentile of the delivery time exceeding the threshold, etc. A more comprehensive service level definition may include guaranteed delivery time as long as the volume does not exceed a predefined threshold. (Since larger delivery volume implies a larger pool of available delivery personnel on-demand.) Finally, the telecom business may provide gold and silver services to its customers, where gold service level is superior in every aspect including short delivery time. The telecom business may have established multiple SLAs with different service levels. Depending on this responsiveness need, it dynamically selects the partner to outsource this process instance.

Quality of service data can be input to agreed decision-making of a service provider, e.g., use a fast alternative path of the workflow if the first part as been progressing slowly. Klingemann et

al. propose a Markov chain-based model for collecting data on expected timing behavior of processes [Kli00, Gre00].

2.4. Dynamic contracting of business process services

Dynamic integration of business processes involves the formal cooperation of two autonomous organizations. Formal relationships between independent organizations are defined in contracts. We discuss contracting in the context of dynamic business process outsourcing in this subsection. First, we pay attention to the concept of (electronic) business service contract. Next, we focus on the life cycle of service contracts. Then, we pay attention to contracting functions that a service consumer respectively service provider should support. Finally, we move to service brokering: the process through which parties that want to establish a contract find each other.

2.4.1. Business service contracts

"A contract is a promise or a set of promises for the breach of which the law gives a remedy …" [Roh00]. This basic definition underlies American contract law and has a corresponding phrasing in many legislative domains. A promise establishes an obligation of one party and a right of another party that the obligation is met. What determines the establishment of a contract is the process in which it is created (voluntarily, etc.), not the way the content of the contract is represented, although for particular subject matters, some legal domains require written form.

The fundamentals of every contract are the provisions specifying the obligations of the parties for the exchange of values, i.e., the outsourced service and the corresponding financial reward. A number of other provisions specify additional rights and obligations of the parties with respect to the business and legal context of the value exchange. For example, contracts contain provisions about the jurisdiction that will handle the situation in case of dispute, the law that governs the contract, etc.

In the context of dynamic business process outsourcing, an electronic representation of contracts is required to enable a fast contracting process since contractual relationships are built on the fly [Gr02b]. Beyond the contract's function as a legal context, it serves as a specification of the process to be performed and the relationship between provider and customer. This specification is input to the configuration of the process management systems of the contracting parties. Dynamic business process outsourcing requires establishment of e-contracts with different parties, in different contexts. This diversity creates interoperability problems between the supporting technologies of service providers, service consumers and mediators. To ensure the interoperability between parties during the contracting process and the contract enactment, standards that govern the contract representation, contract structure, and the contracting processes are required. A number of standardization efforts on the e-contracting process or separate aspects of it like the communication protocol between parties, the messages structure, etc., exist and are constantly emerging. Well-known standardization efforts in this area are ebXML [eb01a] and RosettaNet [RoN02].

A contract can have both machine readable and/or human readable representation. The existence of a human readable representation of the contract is required if the contract creation and management involves the participation of human beings.

We divide the contract content into three general parts.

1. The first part describes the participating parties and mediators.
2. The second part provides the rights and obligations of the parties. This part contains the service (payment) description, its delivery process, legal and technical provisions, etc.
3. The third part gives definitions required for the contract enactment. These definitions can range from the business context of the contract to different terms and formulae used in the contract. The definitions aim at establishing an identical understanding about the contracts between all participating parties.

Electronic contracts can have different machine readable representations. Depending on the contract representation (format), different levels of automation can be achieved. For example, the contract can be represented as a collection of sections (clauses), sub-sections, etc. that provide some level of semantics. An elaboration of this approach is representing the contract (or part of it) as a collection of name-value tuples. This provides formalization of the contract content to a low level and allows high degree of automation to be achieved. Finally, special techniques for the contract definition part can be used in conjunction with special techniques for the "rights and obligations" part. Such techniques can be for example DAML and OIL [Con00] for the contract definition part and formal languages for rights and obligations like the ODP Enterprise Language [ITU01] respectively. A more elaborate discussion on the possible techniques can be found in Ludwig et al. [Lud02]. A composition of different formalization techniques and different formalization languages can be used for achieving high level of automation of the contracting process.

2.4.2. Life cycle

The e-contracting process can be divided into four phases, i.e., information, contracting, deployment, and enactment phases. Next, we briefly describe these phases.

- In the information phase, parties prepare for the e-contracting process and identify possible trading partners. Mediators like service brokers (see Section 2.3.3) or web registries (see Section 3.6) can support parties during the execution of the information phase.
- In the contracting phase, parties perform activities related to the contract negotiation and contract establishment. This phase includes the exchange of contract offers, validation and signing of a contract, and potentially its storing.
- In the deployment phase, preparations for the contract enactment are performed. This phase comprises deployment planning activities and the technical setup of the enactment system required for the specific service delivery.
- In the enactment phase, parties enact the agreed upon contract. The exchange of values is accompanied by monitoring of the contract and if necessary by exercise of control on the service delivery. At the end of the contract enactment phase, evaluation of the e-contracting process can take place.

In many cases, the contract will be re-negotiated. This leads to a re-initiation of the contracting phase followed again by the deployment and enactment phases. The e-contracting life cycle ends with the termination of the enactment phase.
There are other phase models for e-contracting. From a business modeling perspective, a pre contracting phase is considered separately from the contracting phase by Gisler et al. [Gis00]. Furthermore, the deployment phase is not distinguished as a separate e contracting phase. However, for the goals of the paper, we take a more architectural view of the e-contracting process, which brings forth a different phase division. A detailed description of the e-contracting process from a business perspective can be found in Angelov at al. [An02b].

2.4.3. Contracting functions

With the understanding of the phases of the contracting process, we now further analyze the role of the contracting functions in the outsourcing and service-providing party. A decomposition of the contracting functionality for both providers and customers, the flows between its components and external partners, as well as the information associated with these flows are shown in Figure 5.

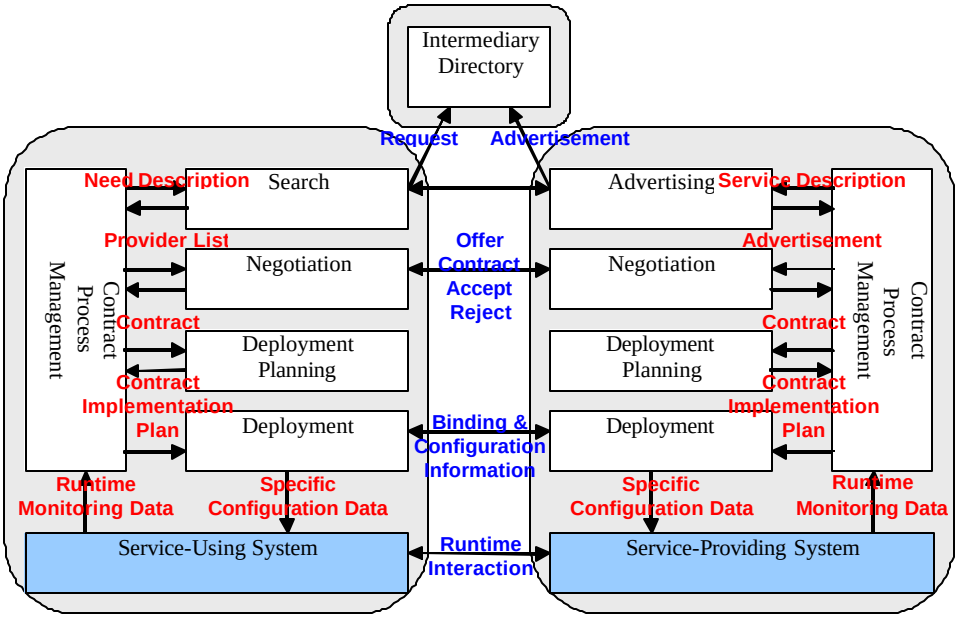

Figure 5: contracting functions

We decompose the contracting function of each party into five conceptual elements:

- The role of the advertising function of a service provider is to deal with the information phase of contract life cycle. It decides how to advertise a service and to whom. It also answers requests from potential customers. These advertisements are based on a description of the service to be advertised, which has been developed prior to the contracting process in the preparation phase and are input to this function. In the case of the parcel delivery, this would comprise a specification of a service and its decision-relevant attributes, etc. the delivery time. The contracting function segments the market for a particular type of service and creates advertisements that correspond to different types of customers. The advertising function sends the advertisement directly to potential customers and to intermediaries such as directories or online marketplaces.

- The search function of a service customer identifies provider organizations that can potentially address a customer organization's service requirements. It sends requests to intermediaries, e.g., directory services, which yields a list of service providers that meet the criteria specified in the request. In addition, a request can be sent to a service provider directly to receive an advertisement. The result of the search is a list of suitable providers for a particular service requirement.

- The negotiation function negotiates contracts with potential customers or providers; hence it corresponds to the negotiation phase of the contract life cycle. The provider's negotiation function receives the set of current advertisements as input; the customer's negotiation function a list of suitable providers and the service requirements. The negotiation function comprises two parts: (1) Interaction; the negotiation is conducted by exchanging offers and counter-offers with potential partners. If an offer is accepted a contract is established. The negotiation process can be initiated either by a potential customer reacting to an advertisement or by the provider responding to a call for bids. (2) Decision-making; offers are made or evaluated by a service provider based on the resource situation at the time the customer wants the service, the status of the customer, e.g., the customer is very loyal and does much business with the provider, and the general market situation. It may be that because of generally low demand for a process the provider is prepared to accept a lower offer and vice versa. Negotiations can be conducted with multiple potential partners at the same time. This func-

tion can be very complex if the parties are very flexible with respect to what can be negotiated, or relatively simple, if the advertisements, which form the basis of the negotiation, are very specific already, e.g., a specific services bundle at a fixed price.

- Once a contract has been agreed upon, or signed, an organization must plan which resources to assign to fulfill the obligations arising from the contract and also, from a customer's point of view, which resources are needed to consume services according to the contract. This is called the deployment planning function. In the example case, the provider decides which type of delivery workflow to instantiate and which functions the process management interface will offer. The result of this planning function is the contract implementation plan.
- The deployment function executes the contract implementation plan. It creates specific configuration information for the involved automated components of the fulfillment system and executes the respective provisioning processes. In the business process context, a provider instantiates the appropriate process type and makes management interfaces available to the provider.
- The contracting process management function manages the execution of the contracting functions. The contracting functions are not necessarily executed in a strict order. Advertisement is done in many cases far prior to negotiation. Also, deployment planning and the deployment itself can be separated if services are purchased in advance. In addition, the contracting process management maintains the state of the contracting processes and stores the corresponding documents.

Those functions are associated with their corresponding phase in the life cycle, with the exception of the process management. As the contracting phases can be invoked in multiple times, the contracting functions can be invoked in multiple times as well, e.g., in iterative negotiations, in nested contractual structures and in situations where contracts contain options for renegotiations. The contracting function deals with a number of documents - conceptually, not necessarily in the form of document files. Advertisements, requests, offers, and contracts are external documents that are exchanged with customers and intermediaries (Figure 5). The provider list, contract implementation plan and the specific configuration information are internal to the service provider.

2.4.4. Service brokering

In order to allow dynamic business process outsourcing, service consumers and service providers of specific types of functions must be able to find each other. In static business process outsourcing, this finding of partners can be based on traditional, non-electronic market mechanisms. Dynamic outsourcing in fast-changing ebusiness markets, however, requires electronic means for service consumers and providers to find and contact each other. These electronic means are offered in the form of service brokers. A service broker can be seen as an advanced yellow pages server for business process services.
Brokering can take place on the basis of service offers by service providers or service requests by service consumers. Both an offer and a request can be seen as an incomplete service contract. To support interoperability, these incomplete service contracts are based on contract templates [Hof01] that are standardized in a specific market segment. If a service offer is advertised in a broker, service consumers can react by completing the offer with their service requests and send it to the service provider as a contract offer. Sending can be direct or through the service broker. The other way around, if a service request is advertised in a broker, service providers can react by completing the request with details of the services they provide and send it to the service consumer – again directly or indirectly. After a contract between consumer and provider has been closed, the service broker is not involved any more – it has no function in the service enactment.

Brokers typically operate in specific market segments, e.g., a logistics market to broker various types of transport services or an insurance market to provide various insurance services. The fact that a broker operates in a specific market makes it easier to reach agreement on the semantics of the communication taking place through the broker. In brokering business process services, two aspects deserve special attention. Firstly, quality of service aspects of services are both important and complex – as we have discussed in Section 2.3.5. Hence, brokering on service level agreement elements is very relevant in this context. Secondly, brokering of business process services may be based on the structure (e.g. control flow) of a service process. This means that an advanced business process broker must be able to match process specifications.

2.5. Summary of requirements

In this section, we have taken a close look at dynamic business process outsourcing. The main requirements to the support of this business paradigm are the following:

- Dynamic binding of business processes, externally visible process model: the business process runtime architecture must support the late binding of activities and subprocesses to organizations; a business process model and an execution environment must support the notion of domain boundaries and must facilitate data and control flow across.
- Transaction support through control flow based compensation mechanisms: the transaction model associated with the business process model must support transactional behavior and flexibly integrate organization-internal from organization-external rollback and compensation.
- Support for cross-organizational process interaction: inter-organizational process interaction mechanisms are required to allow a service consumer to monitor and control a business service enacted on its behalf by a service provider.
- Brokering of business processes: the binding protocol of business processes must support negotiations (based among others on QoS aspects) and the use of intermediaries such as brokers.
- Electronic contracting of business processes: the relationship between outsourced process and customer process must be defined in an explicit contract specified in a formal language such that the establishment of the runtime relationship can be automated.

We will revisit these requirements in Section 4, after we have taken a look at a basic implementation platform in the next section.

3. Web Services - Description and Runtime Models

In this section, we describe the Web service approach to cross-organizational coupling of business information systems. We first describe the Web service paradigm. Next, we discuss the basic technology stack used for support of Web services and pay attention to service brokering. Then we go into advanced topics of Web services relevant to the topic of this paper: process and conversation description, transactional behavior, quality of service (QoS) management, service level agreements (SLAs), and policy specifications.

3.1. Web service paradigm

The Web service paradigm allows the dynamic composition of application functionality using the Web as a medium. The central concept in this paradigm is – as the name suggests – a Web service. A web service is an encapsulated piece of software functionality with a well-defined interface that is made available on the Web. The functionality can be quite divers, depending on

the application domain. A Web service can be very simple, e.g. the functionality to convert an amount of money from one currency to another, or very complex, e.g. the functionality to invoke complex scientific applications.
In the web service paradigm, three roles are distinguished: service provider, service broker, and service requester. A service provider implements a Web service and publishes its interface through a service broker. A service broker holds a registry for published web services for a number of providers. A service requester requires a specific service to implement part of its application functionality. For this purpose, it finds a compatible service through a service broker and invokes it at a service provider.
A service provider itself may use other service providers in the implementation of its services (i.e., auxiliary implementers as explained in Section 2.3.1). In this way, hierarchic composition of services is possible. In the composition of multiple services at a specific level, languages are available that allow specification of the structural relation between the services used.

3.2. Web services basic stack

The foundations of the Web services stack are the Web Services Description Language (WSDL) [Chr01] for the description of service interfaces and the suggested interaction protocol to invoke services comprising the Hypertext Transfer Protocol (HTTP) and the Simple Object Access Protocol (SOAP) [Box00] to encode messages exchanged with a Web service at runtime.
WSDL is a generic XML-based language to define interfaces of services in a distributed system. Abstract interfaces are described as port types comprising a set of operations. An operation is defined by its input and output messages – at least one must be defined. The message content is defined based on the XML Schema Language [Bir01, Tho01]. Message elements can be defined as simple Schema types or specific complex types in a separate Schema. Abstract interface definitions can be bound to a particular transfer protocol. In a binding, the specifics of the implementation of a Web services interface are defined, e.g., for each operation the encoding format and address information. The most common binding is to SOAP over HTTP. In this case, the address information may contain a URL and the mapping of WSDL operation and message descriptions to SOAP runtime encoding is predefined. Any other binding can be defined, e.g., to SOAP over HTTPS or even an encoding format other than SOAP, e.g., MIME.
The Simple Object Access Protocol (SOAP) is a protocol specification for exchanging messages in a distributed environment. The specification comprises: an envelope format for messages that captures context information of the message, e.g., information on what is in the message and how it should be processed; a message encoding format that can represent structured data; a model for dealing with messages that follow the request-response pattern. The representation is XML-based. SOAP can be used in conjunction with different transport protocols such as HTTP, HTTPS or even SMTP email. SOAP is the most frequently used message format in the context of Web services, often using HTTP as transport protocol.

3.3. WS policy framework

The Web Services Policy Framework (WS-Policy) provides a mechanism to specify policies of Web services [Bo02a, Bo02b]. A policy comprises a set of assertions that can carry the semantics of a requirement, a preference, or a capability. The assertions are part of a policy expression that states that all assertions, exactly one, or one or more holds. Expressions can be nested.
The basic syntax does not carry strong semantics. It is intended that the basic WS-Policy schema is extended for particular domain policies, e.g., for security. Those domain-specific policies can be combined in policy expressions. WS-Policy can be used to describe additional properties and usage policies of a Web service beyond its WSDL interface specification. In particular, it can express requirements on the client of a Web service.
The WS Policy Attachment specification describes how to relate a particular WS-Policy to a subject. If the policy relates to an arbitrary XML element, the specification proposes an XML

extension to be able to associate a policy with any XML element. This is certainly not practically feasible in the foreseeable time since current XML parsers do not support it. The alternative is to include a policy in an attachment statement that contains a reference to the subject of a policy. This can be done in any way since there is no canonical form to refer to a Web service or a BPEL process at this time.

3.4. Service brokering

Universal Description, Discovery and Integration (UDDI) [UDD02] is the name of a group of Web based registries that expose information about a business or other entity (i.e., organization). These registries are run by multiple operator sites and can be used by anyone to find information about published business services. The registries comply with an information structure consisting of a set of core data types and web services for querying registry information, both of which are being standardized via OASIS [OAS03].

At the top level of the registry are businessEntity entries, one for each business that publishes information about a set of its services. Each businessService entry describes detailed information about a published business service including name, text description of the service, bindingTemplates that provides technical information on using this service and a category bag that provides many categorizations of this business service. Each bindingTemplate in turn provides technical information on accessPoint such as URL for invoking a Web service. It also provides details on instance parameters for one or more tModels referenced from this binding template.

A tModel entry contains a set of keyed metadata technical information about the service. The actual metadata information contained in a specific tModel instance will depend on the types of information required by a service specification convention (e.g., wire protocols, interchange formats, interchange sequencing rules, etc.) and from UDDI perspective could be any specification. Industry consortia, such as RosettaNet, OAGI, may define tModel for industry specific service descriptions. The same tModel entry may be used across many bindingTemplates with distinct instance parameters. Finally, each publisherAssertion entry defines a business relationship asserted by both parties, and includes fromKey, toKey business entities and keyedReference to a tModel.

The latest version of UDDI (v3) [OAS03] specification now provides the ability to nest subqueries within a single query, reducing the number of round trips a client must make to a UDDI registry. It also provides a notification service via which a client can track the changes to the results of a specific query or set of entities in the registry.

3.5. Process description

The need to compose individual Web services into larger aggregates requires a model for the composition of Web services, potentially from different organizations, and a description language for the specification of compositions. Multiple proposals have been made, among them WSFL and XLANG. IBM has proposed the Web Services Flow Language (WSFL) as a graph-oriented web service composition language. WSFL allows a workflow-like specification language style – the language shares many properties with the specification language of IBM's MQSeries Workflow system. Microsoft has proposed XLANG as an extension to WSDL to allow the orchestration of Web services. In XLANG, WSDL definitions can be amended by control flow expressions that define the sequencing, repetition and other interdependencies of operations defined in ports of a WSDL file. The Business Process Execution Language for Web Services (BPEL4WS or BPEL for short) [Cur02] has been proposed by IBM and Microsoft to supersede XLANG and WSFL as a common approach to Web services composition. Although it is labeled as "work in progress" by the authors, it contains a rich set of constructs to describe business processes.

The objective of BPEL is to define business protocols that orchestrate the invocation of Web services among a set of parties, thus focusing on coordinating the message exchange between those collaborating parties. Business protocols defining the mutually visible message exchange are termed *abstract processes*. An internal implementation of an abstract process containing all details is called *executable business process*.
Both levels of description use the same expressive means. It is assumed that Web services being composed are described in WSDL documents. The BPEL document itself comprises the definition of "partners" (organizations) participating in the business process, fault and compensation handlers, definition of the shared data, "containers", "correlation sets" for identifying messages relating to a particular instance of a business process, and the process flow description itself.
Invocations of Web services that are part of the process are described as references to WSDL port types and operation and are assigned to partners. A rich set control flow constructs are available to express selection, repetition, and parallelism. Containers are shared data structures that comprise message definitions. Message definitions can be in-lined or being referred to as part of the WSDL definitions that underlie the BPEL document. Data flow is defined as assignment from one container to another, where the "assign" activity is regular activity. "Receive" and "reply" activities facilitate the initiation of a process upon receipt of a message and the return of the result message, corresponding to the in and out messages of a WSDL operation.
BPEL also supports the processing of exceptions. Fault handlers can be related to synchronous invocations of Web services. Furthermore, each activity can be associated with a compensation handler that undoes its effects. Also, a process can be associated with a compensation handler, to be triggered from outside the scope of the process.
In the current specification version 1.0, a number of issues are listed as to be dealt with in future versions. Among them is the scoping for containers and event handlers, atomicity scopes and compensation scopes as well as process management operations. In addition, the relationship of WS-Transaction is planned to be investigated.

3.6. Transactional behavior

Transactionality is an important aspect of loosely coupled business interactions between independent parties as facilitated by Web services. However, ACID transactions - as mostly used in a tightly coupled environment – are not applicable in a scenario of loose coupling where transactions may take long time, e.g., days, activity effects are visible early, parties participating do not trust each other, and communication may fail frequently. In this case, exceptions occurring in transactions must be exposed and dealt with using compensating activities that undo and heal inconsistencies on a level of business semantics. The added flexibility is achieved (and paid for) by the implementation of business specific compensation logic, as opposed to using the one-size-fits-all rollback and retry of ACID transactions.
There are two relevant standardization approaches that provide transaction mechanisms for a loosely coupled environment: The standards pair Web Services Coordination (WS-Coordination)/Web Services Transaction (WS-Transaction) proposed by Microsoft, IBM and BEA [Ca02a, Ca02b] and the Business Transaction Protocol (BTP), which has been agreed upon in the Organization for the Advancement of Structured Information Standards (OASIS). Both approaches define transaction models and the corresponding interactions for both ACID transactions and business transactions as discussed above.
WS-Coordination describes a model of establishing coordinators mediating distributed Web services. The core mechanism is a coordination context, which scopes the coordination. Applications use coordinators to process coordination activity so that applications do not need to understand coordination protocols. An application can have its own coordinator or share one with other applications. A coordinator exposes an activation interface for an application to create and interact with coordination contexts, to register an application for participation in the coordination, and interfaces for particular coordination protocols that are used by other coordinators. Those interfaces are defined in WSDL. Coordinators can be part of a coordination hierarchy,

interacting with other coordinators. While WS-Coordination establishes a coordination architecture, it does not define a coordination protocol itself.
The WS-Transaction standard defines coordination protocols based on WS-Coordination; one for standard distributed two-phase commit and a "business agreement protocol" for "business activity", which is the standard's term for loosely coupled activities requiring exception management by compensation, as discussed above. The standard describes the models of two-phase commits and the business agreement protocol and it defines the WSDL interfaces to the coordinators and the coordination contexts as extensions to the corresponding WS-Coordination items.
BTP has a similar scope. It offers an interaction model and interface definitions for conducting both short, atomic transactions and "cohesive business transactions" (cohesions), which correspond to long-running transactions in a loosely-coupled environment. Applications involved in a transaction are called application elements. Application elements that manage an atomic transaction are called coordinators, those that manage cohesions are called composers. Application elements being managed are called participants. Nesting of transactions yields sub-coordinators and sub-composers that have both roles, participant and coordinator or composer. The coordinator, participant, etc. functions are intended to be separate from the applications being coordinated. An application invoking another, joining it into the transaction, passes a context, which is then used by the called application's participant to register with the calling application's coordinator or composer. Transactional interaction such as prepare, commit, etc. is between coordinators, composers and participants. The atomic coordinator follows the model of the two-phase commit while, however, a participant does not guarantee isolation. The composer is given more flexibility in coordinating its participants in that it can cancel some participants and invoke others – or receive failures from some and continue anyway, depending on the business situation. In addition to this interaction model, the specification comprises the definition of the relevant messages, in the abstract and in an XML format. This format does not follow WSDL.
Both approaches define models for dealing with atomic as well as long-running transactions among loosely coupled services. In addition, each party defines a message format to implement the interactions defined in the respective models. However, both standards focus on the runtime aspects of transactions. They do not provide a means of expressing the transactional behavior of a service, e.g., the definition of the types of open transactions a service is prepared to be involved in.

3.7. Quality of Service and SLAs

The description of quality of service properties of simple Web services and composites has not yet been standardized in the course of the Web services standardization initiatives. However, a number of approaches aim at providing such a description language specifically for Web services or are applicable to the Web services stack but cover a wider scope.
The Web Service Level Agreement (WSLA) framework by IBM provides a language for describing service level agreements (SLAs) of services and an architecture for distributed monitoring of SLA-compliance [Lu02b, Kel02]. The focus of the WSLA framework is the performance aspect of services, e.g., response time and throughput. The WSLA language is based on XML. A WSLA document contains a definition of the service objects, the SLA parameters they expose, and a definition of the metrics that define how the SLA parameters must be computed. Based on those parameters of service objects, service level objectives (SLOs, performance goals) and action guarantees (promises to do something) can be defined. Service objects are descriptions of any element of a service that has SLA parameters. Depending of the type of service elements, it contains different description elements. In the case of WSDL operation in a SOAP binding, the description contains a pointer to the WSDL file and the names of binding, service, and operation. References to other service elements, e.g., BPEL process definitions would contain corresponding description elements. The metric definition comprises resource metrics that describe how counter, gauges and the like are retrieved from instrumentation or how one must probe a value.

Based on those resource metrics, composite metrics can be defined hierarchically to finally yield the high-level SLA parameters that are subject to SLOs. The composite metric definition uses functions that can be nested and applied to metrics and constants, e.g., sum(timeSeriesMetric). The WSLA document also defines which party of an SLA has which role. Measurements and evaluation of guarantees can be performed by the service provider, the customer, or independent third parties. The WSLA language is based on XML. The WSLA compliance monitoring framework comprises a measurement service, which collects resource metrics and aggregates to composite metrics, and a condition evaluator, which evaluates SLOs and trigger actions as defined in the action guarantees. Both components can be deployed multiple times by different parties to supervise different – or the same - aspects of an SLA. The WSLA authoring toolkit supports creating WSLA documents and templates.

HP Labs propose a business cockpit to supervise the quality of service of Web services and cross-organizational business processes [Say03]. The approach proposes a model for defining quality of service aspects of a Web service or process. Its core concepts are metrics, mapping templates and meters. Metrics are description of values, i.e. name and type. Mapping templates are descriptions how to compute a value. This description includes information in a suitable (programming) language, e.g., SQL or Java that must be executed when measuring the value. Meters define which mapping templates are used to compute a metric. Metrics can recursively be input to mappings.

While the WSLA language is aimed at defining agreements between organizations, and hence focuses on clarifying metric computation and objective definition as part of a the language, HP's proposal serves as input to the business cockpit and contains business-internal information - among other things information about how to visualize metrics. Both approaches share the concept that custom metrics can be defined based on input taken from a system's instrumentation.

4. Web Services for Dynamic Business Process Outsourcing

In this section, we explore how the requirements of dynamic business process outsourcing, as discussed in Section 2, can be mapped to the Web service paradigm, as discussed in Section 3. In doing so, we identify missing elements in the Web service framework and propose how to provide support for these elements. We follow the structure of the requirements in Section 2.5. We first pay attention to support of the distributed business process model as required for process outsourcing and transactional behavior of processes in this process model. Next, we turn to business process brokering and contract handling. After that, we discuss runtime process management and intervention by service consumers in service provider processes. We end this section with an illustration of the proposed approach that places the various elements in relation to each other.

4.1. Business process model

In this subsection, we study the support for distributed business processes in the Web service model. We first analyze limitations of the standard model, and then propose extensions to overcome these limitations. In the analysis below, we focus of control flow aspects (as discussed in Section 2.3.1). Data flow integration (as discussed in Section 2.3.2) is well supported by the explicit message interfaces of Web services and requires less attention in this context.

4.1.1. Service composition for dynamic business process outsourcing

In the Web services paradigm, a basic web service is viewed as a black box piece of functionality: it is described by its message interface and has no internal process structure that is visible to its environment. Building multi-step business processes with Web services relies on composing multiple Web services using composition languages like BPEL (as discussed in Section 3.3).

The default model underlying BPEL is that an abstract process is shared among the partners but the executable business process remains private to each partner. However, this does not suit the business process outsourcing model where a service provider explicitly wants to provide information about an internal process structure to external parties and enable them to monitor and control an internal process (as discussed in Section 2). Applied to our example from Figure 2, the three activities in the provider process must be visible to a service consumer, both from a specification and a runtime management point of view, to allow fine-grained monitoring and control of the outsourced subprocess. This is not supported by the default BPEL model.
If business process service providers do want to offer multi-activity business processes using the default BPEL model, they could offer the individual activities used in the process as Web services and leave the composition into a business process to the service consumer. Applied to our example, this would mean that the three provider activities would be offered as separate Web services. This has a number of drawbacks, however. Firstly, it may imply that a provider has to expose service elements that are in isolation meaningless to the outside world. An activity 'plan transport' by a provider has no meaning unless that same provider is actually going to perform the transport. Secondly, it moves the burden of composing a (possibly complex) business service process to the consumer, thereby decreasing the added value of the service provision. Thirdly, it allows a service consumer to compose processes that have an invalid structure from the provider's point of view. In the example, a consumer could compose a process with 'deliver parcel' preceding 'collect parcel'.
The conclusion here is that the requirements of the dynamic business process outsourcing scenario call for changes in the way the BPEL specification language is used and for extension to the underlying runtime model. In the case of business process outsourcing, the BPEL specification that is shared among the parties must comprise also the publicly visible activities of the service provider. No syntactical extensions are needed, though. The execution model of an activity having a process structure should be extended to accommodate service consumer monitoring and control.

4.1.2. An extension of the basic Web service framework

We introduce an extension to the basic Web services paradigm that caters for services with an internal process structure that can be observed externally. This internal process coincides with a business service process offered by a service provider. The state of execution of the internal process can be observed and specific control primitives can be applied on this state to allow external control over the execution.
We introduce the concept of a business process Web service (abbreviated to BP-WS) that does have an (internal) business process specification that can be accessed externally through a number of dedicated Web service interfaces (ports). A BP-WS has the following interfaces (in the order of typical use):

- The 'traditional' invoke and reply interfaces: these are not different from the situation where the BP-WS would have been a traditional, black box Web service. We cluster them in an activation (ACT) interface.
- An interface to obtain the business process model (SPEC): through this interface, a consumer can obtain a process specification of the business process service[1], e.g. in WSFL or BPEL. The SPEC interface can be considered a reflection interface, as it given information about the behavior of a BP-WS. Note that the specification can also be offered through a contract (see Section 4.4).
- An interface to monitor the state of execution (MON): through this interface, a consumer can obtain information regarding the state of execution of the business process service after it has

[1] *Note that this interface may be considered a reflection interface of a BP-WS, as it is used to obtain information on the behavior of the BP-WS.*

been invoked through the INV interface; information obtained is to be interpreted in the context of the process model obtained through the SPEC interface.

- An interface to control the execution of the internal process (CTRL): through this interface, a consumer can issue control primitives to influence the execution of a service executed on its behalf; typical control primitives are 'pause process', 'resume process' and 'abort process'; invocation of control primitives is typically based on information obtained through the MON interface.

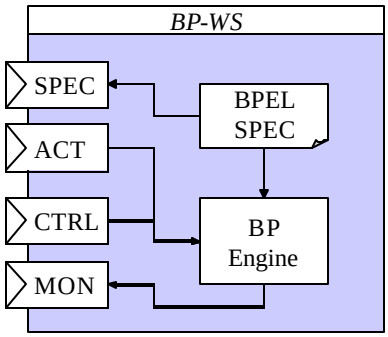

Figure 6: BP-WS architecture

We show a stylized architecture of a BP-WS in Figure 6. Here we see that a BP-WS contains a business process specification in BPEL. The specification can be obtained externally through the SPEC interface. The specification is interpreted by a business process engine. The state can be observed through the MON interface, its transformations influenced through the ACT and CTRL interfaces.

Business process Web services can be combined like any 'traditional' Web service to enable process composition. In the context of dynamic business process outsourcing, composition mainly means combining the process of service consumer with that of service provider. In other words, service composition is used to 'embed' an insourced business service process into the process of a consumer (such as the logistics service process into the telecom process of our example – see Figure 2).

The proposed extension can be realized on top of the basic Web services stack as discussed in Section 3.2, e.g., as a standard port type defining the relevant operations that is by default available for all BP-WS. Note that the proposal essentially maps the main interaction paradigm developed in the CrossFlow project [Gre00, Hof01] (which uses a dedicated infrastructure for support of the above primitives) to the Web service platform (which uses a general-purpose infrastructure).

4.2. Transactional behavior

The WS-Transaction standard (as discussed in Section 3.4) provides a number of basic elements required for a transactional infrastructure for web services. However, some elements are missing, that are essential to the transactional support required for (composite) BP-WSs. Those elements concern both the specification of transactional behavior and the run-time execution of this behavior.

4.2.1. Transactional process specification

As discussed in Section 4.1, we propose to use a standard Web service composition language (e.g. BPEL) for the specification of the internal process of a BP-WS. In Section 2.2.4, we have seen that we require specific transactional primitives for business processes. The specification of these transactional primitives should be possible in the BP-WS process specification lan-

guage, i.e., be supported by a complementary extension of BPEL. A consumer obtains the transactional process specification through the SPEC interface as discussed in Section 4.1.
As we have seen, an important class of transactional behavior for a BP-WS is relaxed atomicity supported by control flow based compensation. To enable this type of compensation, we can use the following primitives [Gre01]:

- Specification of an application-dependent compensating activities for BP-WS activities (where original and compensating behavior may be provided by different Web services); BPEL already includes possibilities for specifying compensating activities.
- Specification of safepoints (restart points) to specify partial compensation patterns in a BP-WS process; safepoints are an alternative for the use of spheres [Ley95] allowing parameterized partial rollback.
- Transactional capabilities specification primitives to specify for a service in what transactional protocols it can engage, e.g. standard two-phase commit, specific compensation protocols, etc. This is an extension to the current WS-Transaction standard [Ca02b], which only specifies a runtime interface.

A service consumer can query a provider's transactional specifications and capabilities through the dedicated SPEC interface. Transaction specifications can easily be realized as annotations in a language like BPEL.

4.2.2. Transaction support

To support execution of transactional BP-WS specifications, we propose the following:

- Parameterized transaction control interface [Von00], e.g. enabling parameterized partial compensation to a specified safepoint [Gre01]. This interface is part of the CTRL interface mentioned in Section 4.1. To use it, the service consumer typically also requires the CAP interface to determine the capabilities of a service provider and the MON interface to determine the state of a business process service in execution.
- Well-specified compensation protocols using control flow based (WF-like) service compositions, as opposed to nested-scope composition assumed in Business Agreement protocol of WS-Transaction. These protocols can be layered over WS-Coordination [Ca02a] and included in WS-Transaction (WS-T) [Ca02b] as process-oriented alternatives to the current Atomic Transaction and Business Activity protocols. The protocols are supported by dedicated coordination services. A coordination service for cross-organizational compensation is comparable to the CrossFlow XTM [Von00, Von03].

4.2.3. Semantics specification

Apart from the operational support of protocols, we require clearly described semantics of complex transactional behavior (e.g. compensation protocols) in terms of combination of intra- and inter-service process execution [Von00, Von03]. The semantics specify precisely and unambiguously *what* the effect of a transactional operation is, whereas protocol specifications describe *how* to realize this. The semantics must be completely independent from implementation platforms used to cater for process reliability in dynamic collaboration settings.

4.3. Runtime process management and customer interventions

In the sections above, we were discussing the need for an extended model of interfaces to business process web services comprising additional operations to manage the transactional operations of a process (SPEC, CTRL, etc.). With those extensions we get a comprehensive set of operations that define the interfaces between organizations engaging in business process out-

sourcing. However, an interface description does not encompass when and under which conditions an operation may be used.
Preconditions of operations are important in the cross-organizational process context. For example, it may be possible to roll back and compensate the parcel delivery process if the process is still in a particular state, e.g., the parcel is still in the country. For legal reasons, it might be impossible to roll back or suspend the process at this point, only earlier. Hence, we need to be able to associate preconditions to process and transaction monitoring and control operations.

4.4. Business process brokering

Dynamic outsourcing demands late binding of outsourced business process steps based on various partner selection criteria including existing SLAs with partners, current state of a business process instance (i.e., elapsed time so far, business process steps performed), SLAs guaranteed to a client(s) and other business objectives (e.g., cost minimization). For example, a higher priced service provider with smaller delivery time may be selected for performing an outsourced business process step in a specific process instance, either to meet a higher QoS guaranteed to its end-client or to improve the total elapsed time of a process instance that is behind its normal schedule.
UDDI registries can be used by the service providers to publish supported web services and various service offerings (i.e., tModels referring to SLA templates). With this as starting point, many forms of enhanced brokers may emerge, ranging from brokers that provide enhanced searching based on published services in UDDI and associated links to other business information, marketplaces that provide specially negotiated deals, to service intermediaries that actually takes responsibility for performing this service [Dan97, Dan98]. As part of this responsibility, the service intermediary needs to isolate (i.e., manage separately) the interactions with its service client and service provider, however through a delayed and/or weaker consistency. (A cancellation from a client is honored immediately even if the provider is not willing or able to do so.) In the scenario, where the business process owner forms a direct relationship with the end service provider, following the discovery process it may create an SLA to perform services over a longer time period. Note however, a business may create multiple SLAs with the same or different service providers at the same time, and following a late binding of process instance step actual provider is selected to satisfy business objectives.

4.5. Contract establishment and negotiation

The Web services model and the specification stack do not have the explicit concept of a contract and hence there is no support for contract establishment and negotiation. The binding process to a single Web service does generally not involve negotiation – at least not in an agreed-upon manner – and there is no defied proceeding for the coming into existence of a BPEL specification.
In some aspects, however, we can interpret elements of a contract and parts of a contract establishment process into existing approaches: The shared view of a process defined in BPEL can be regarded as an agreement of the involved parties on the process sequence and some transactional aspects. Descriptions in the WS-Policy format, which augments the interface description of a service with additional information and usage requirements, can be seen as an offer to use the service if the preconditions are being fulfilled.
The Web services stack needs a contract format. A contract provides a context for interface specifications, the process description, and the transactional and quality properties that the contract must put into context and associates those elements with parties. In addition, a business environment requires description of pricing and payment as well as a slot to define natural language parts of a contract that are not interpreted programmatically, which is not difficult.
Since contracts as described above may be very complex and contain many different aspects, there is a need for negotiation protocols that involve human negotiators, protocols that can be

processed automatically, and mixed forms. Common to all forms of negotiations are offers, accepts, rejects, etc. The content of offers depends on the particular negotiation protocol. Automation of negotiation of complex subject matters usually relies on templates [Lu02a]. Contract templates are contracts with a set of open "fields" that can represent either individual values to be filled in, e.g., a URL, or can encompass entire aspects of the contract, e.g., the QoS definition. A contract template can also carry constraints on valid values to fill the fields.
Since the validity of the contract may start later than the "signing" of the contract, we need a protocol that actually starts up the contract at a given point in time.
We have communication with respect to contract establishment (as shown in Figure 5[2]) take place through the ACT interface of a service provider. Contract establishment can be related two service invocation in two ways: a one-phase and a two-phase protocol. In the one-phase protocol, establishment of a contract implicitly invokes (activates) the contracted service. In the two-phase protocol, contract establishment and service invocation are performed in two discrete communication phases between service consumer and provider. The former is obviously simpler, the latter allows more flexibility (e.g. to allow contracting to take place decoupled in time from enactment or to allow umbrella contracts covering multiple service invocations).

4.6. Illustration of the approach

We illustrate the approach outlined above with two stylized example architecture diagrams, showing the service consumer context (Figure 7) and service provider context (Figure 8).
In Figure 7, we see a service consumer on the left. It queries a business process (BP) broker to find a matching business process service provider to implement part of its business process. In our example shown in Figure 2, the service consumer is the telecom operator. On the right, we see a BP-WS offering a service. In the example, the BP-WS is employed by a logistics provider, offering the three-step logistics process of Figure 2. The BP-WS exposes the four interfaces that we have defined in Section 4.1.2 to the consumer.

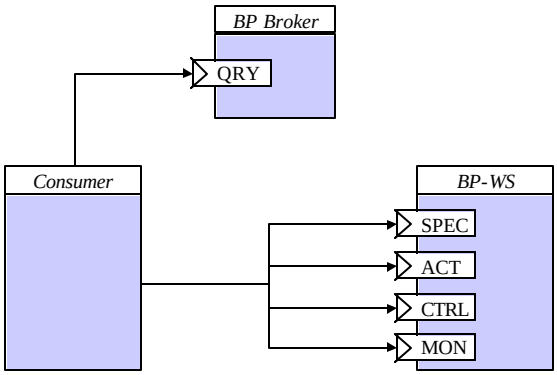

Figure 7: BP-WS global architecture, consumer context

In Figure 8, we see the provider context with the same BP-WS and BP Broker. The BP Broker uses the CTRL interface of the BP-WS to obtain the specification of the business process offered (in our example, the three-step logistics process). In this example case, the BP-WS makes use of two basic web services to implement the 'collect parcel' and 'deliver parcel' activities (which may be in-house Web services, or external Web services if the logistics provider relies on external transport companies). The BP-WS participates in the context of a process-oriented transaction, in this case an X-transaction managed by an extended WS-Transaction (WS-TX)

[2] *Note that this does not include the 'runtime interaction' shown in Figure 5 – this interaction is supported by the MON and CTRL interfaces of the BP-WS model.*

component. The WS-TX component makes use of the CTRL interface of the BP-WS to control its transactional behavior.

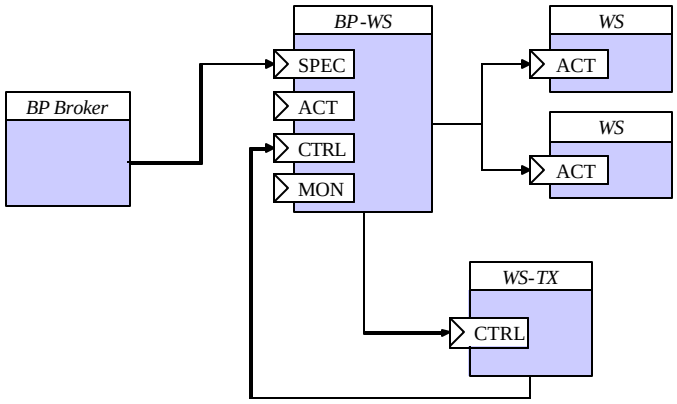

Figure 8: BP-WS global architecture, provider context

5. Conclusion

In this paper, we have made an analysis of the applicability of the Web services framework for the support of dynamic business process outsourcing (DBPO). For this purpose, we outline in section 2 a model of dynamic business process outsourcing addressing both the requirements in terms of a business process model as well as the support of a contracting process. Subsequently, we introduced the relevant aspects of Web services technology and, finally, discussed open issues of the application of Web services technology to the domain of DBPO and suggested directions how to extend the current Web services technology to suit the DBPO scenario.
We have come to the conclusion that the Web services framework offers a basic platform that is very suitable for DBPO support due to its support for loosely coupled services that matches well the DBPO case. However, a number of open issues need to be addressed:

- Business process and transaction model: The DBPO environment requires an execution and interaction model for the management of composite business processes. This particularly includes that a DBPO customer can access the internal structure of a process-structured Web service and monitor and control the progress. Furthermore, support for cross-organizational transaction management is needed, including safepoint support that facilitates flexible inter-organizational and intra-organizational rollback. In addition, we need to define preconditions - dependent on a process state - when monitoring and control operations can be used. Quality of service aspects are relevant in a cross-organizational setting as a metering parameter and must be captured in a specification and measured and managed by the runtime environment.
- Brokering, negotiations and contract: The binding to a Web service does not support the notion of contracts as part of the standard specification stack, which is essential for business process outsourcing. Specification formats for offers and contracts are required and a standard negotiation protocol must support this process.

With those extensions, the Web services platform would be well suited for DBPO and contribute to the development of flexible process support platforms for universal enterprise integration.
The current developments around Web services promise support for extended specification of Web services. Based on the WS-Policy framework, additional aspects of Web services such as their transactional capabilities, their QoS options, etc. could be expressed. Other aspects such as contracting are dealt with in entirely different environments such as the OASIS Legal XML - eContracting technical committee.

References

[Alo99] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, N. Weiler; *WISE: Business to Business E-Commerce*; Procs. 9th Int. Workshop on Research Issues on Data Engineering; Sydney, Australia, 1999; pp. 132-139.

[An02a] S. Angelov, P. Grefen; A *Conceptual Framework for B2B Electronic Contracting*; Procs. 3rd IFIP Working Conference on Infrastructures for Virtual Enterprises; Sesimbra, Portugal, 2002; pp. 143-150.

[An02b] S. Angelov, P. Grefen; *Support for B2B e-Contracting – The Process Perspective*; Procs. 5th International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services; Cancun, Mexico, 2002; pp. 87-96.

[Bir01] P. Biron, A. Malhotra (eds.); *XML Schema Part 2: Data Types*; W3C Recommendation; May 2001, http://www.w3.org/TR/xmlschema-2/.

[Box00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer; *Simple Object Access Protocol (SOAP) 1.1*; W3C Note May 2000; http://www.w3.org/TR/SOAP/.

[Bo02a] D. Box, F. Curbera, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, M. Nottingham, C. von Riegen, J. Shewchuk; *Web Service Policy Framework, Version 1.0*; 2002.

[Bo02b] D. Box, F. Curbera, M. Hondo, C. Kaler, H. Maruyama, A. Nadalin, D. Orchard, C. von Riegen, J. Shewchuk; *Web Service Policy Attachment, Version 1.0*; 2002.

[Ca02a] F. Cabrera, G. Copeland, T. Freund, J. Klein, D. Langworthy, D. Orchard, J. Shewchuk, T. Storey; *Web Services Coordination (WS-Coordination)*; August 2002.

[Ca02b] F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, S. Thatte; *Web Services Transaction (WS-Transaction)*; 2002.

[Chr01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana; *Web Services Description Language (WSDL) 1.1*; W3C Note March 2001, http://www.w3.org/TR/wsdl.

[Con00] D. Connolly, F. van Harmelen, I. Harrocks, D. McGuinness, P. F. Patel-Schneider, L. A. Stein; *Annotated DAML+OIL Ontology Markup*; World Wide Web Consortium, 2000.

[Cur02] F. Curbera, Y. Goland, J. Klein, F. Leyman, D. Roller, S. Thatte, S. Weerawarana; *Business Process Execution Language for Web Services (BPEL4WS) 1.0*; August 2002, http://www.ibm.com/developerworks/library/ws-bpel

[Dan97] A. Dan, F. N. Parr; *The Coyote Approach for Network Centric Business Service Applications: Conversational Service Transactions, a Monitor, and an Application Style*; HPTS Workshop, Asilomar, CA, USA, 1997.

[Dan98] A. Dan, D. Dias, T. Nguyen, M. Sachs, H. Shaikh, R. King, S. Duri; *The Coyote Project: Framework for Multi-Party e-Commerce*; Procs. 2nd European Conference on Research and Advanced Technology for Digital Libraries; Heraklion, Greece, 1998; pp. 873–889.

[Dui00] M. Duitshof; *Logistics Prototype Deployment Paper*; CrossFlow Project Deliverable D13; KPN Re-search, The Netherlands, 2000 (available via http://www.crossflow.org).

[eb01a] *ebXML Technical Architecture Specification v1.0.4*; February 16, 2001; http://www.ebxml.org.

[eb01b] *ebXML Business Process Specification Schema (BPSS 1.01)*; ebXML Business Process Project Team, May 2001; http://www.ebxml.org/specs/index.htm#technical_specifications; 2001.

[eb01c] *ebXML Collaboration-Protocol Profile and Agreement Specification v1.0*; http://www.ebxml.org; 2001.

[Gar87] H. Garcia-Molina, K. Salem; *Sagas*; Procs. 1987 ACM SIGMOD Int. Conf. on Management of Data; San Francisco, California, USA, 1987; pp. 249-259.

[Gis00] M. Gisler, K. Stanoevska-Slabeva, M. Greunz; *Legal Aspects of Electronic Contracts*; Procs. CAiSE'00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing; Stockholm, Sweden, 2000.

[Gre97] P. Grefen, J. Vonk, E. Boertjes, P. Apers; *Two-Layer Transaction Management for Workflow Management Applications*; Procs. 8th Int. Conf. on Database and Expert System Applications; Toulouse, France, 1997; pp. 430-439.

[Gre00] P. Grefen, K. Aberer, Y. Hoffner, H. Ludwig; *CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises*; International Journal of Computer Systems Science & Engineering, Vol. 15, No. 5, 2000; pp. 277-290.

[Gre01] P. Grefen, J. Vonk, P. Apers; *Global Transaction Support for Workflow Management Systems: from Formal Specification to Practical Implementation*; VLDB Journal, Vol. 10, No. 4; Springer, 2001; pp. 316-333.

[Gr02a] P. Grefen, H. Ludwig, S. Angelov; *A Framework for E-Services: A Three-Level Approach towards Process and Data Management*; CTIT Technical Report 02-07; University of Twente, 2002; also appeared as IBM Research Report RC22378; IBM Research Division, 2002; submitted for external publication.

[Gr02b] P. Grefen, S. Angelov; *On t-, m-, p- and e-Contracting*; Procs. CAiSE Workshop on Web Services, e-Business, and the Semantic Web; Toronto, Canada, 2002; pp. 68-77.

[Hof01] Y. Hoffner, S. Field, P. Grefen, H. Ludwig; *Contract Driven Creation and Operation of Virtual Enterprises*; Comp uter Networks - The International Journal of Computer and Telecommunications Networking; Vol. 37, No. 2; Elsevier, 2001; pp. 111-136.

[ITU01] International Telecommunication Union; *ITUT Recommendation X.911 – Information Technology – Open Distributed Processing – Reference Model – Enterprise Language*; 2001.

[Kel02] A. Keller, H. Ludwig; *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services*; IBM Research Report RC22456; Yorktown Heights, USA, 2002.

[Kli00] J. Klingemann; *Controlled Flexibility in Workflow Management*; Procs. 12th Conf. on Advanced Information Systems Engineering; Stockholm, Sweden, 2000; pp. 126-141.

[Ley95] F. Leymann; *Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems*; Procs. Datenbanksysteme in Büro, Technik und Wissenschaft; Dresden, Germany, 1995; pp. 51-70.

[Ley01] F. Leymann; *Web Services Flow Language (WSFL 1.0)*; IBM, May 2001; http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[Lu02a] H. Ludwig; *A Conceptual Framework for Electronic Contract Automation*; IBM Research R eport RC 22608; IBM Research Division, 2002.

[Lu02b] H. Ludwig, A. Keller, A. Dan, R. King; *A Service Level Agreement Language for Dynamic Electronic Services*; Procs. WECWIS 2002, Newport Beach, CA, 2002; pp. 25 - 32.

[OAS02] OASIS; *Business Transaction Protocol, Version 1.0*; June 2002.

[OAS03] OASIS UDDI Specifications TC; http://www.oasis-open.org/committees/uddi-spec/; Februrary 2003

[Roh00] C. D. Rohwer, A. M. Skrocki; *Contracts in a Nutshell (5th Edition)*; West Group, 2000.

[RoN02] *RosettaNet Implementation Framework: Core Specification (RNIF 02)*; RosettaNet, http://www.rosettanet.org; 2002.

[Say03] M. Sayal, A. Sahai, V. Machiraju, F. Casati; *Semantic Analysis of E-Business Operations*; Journal of Network and Systems Management, March 2003.

[Ste00] M.W.A. Steen, J. Derrick; *ODP Enterprise Viewpoint Specification*; Computer Standards and Interfaces, Vol. 22, 2000; pp. 165-189.

[Tha01] S. Thatte; *XLANG – Web Services for Business Process Design*; Microsoft, 2001; http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

[Tho01] H. Thompson, D. Beech, M. Maloney, N. Mendelsohn; *XML Schema Part 1: Structures*; W3C Recommendation, May 2001, http://www.w3.org/TR/xmlschema-1/.

[UDD02] *UDDI Technical White Paper*; http://www.uddi.org; 2002.

[Von00] J. Vonk, W. Derks, P. Grefen, M. Koetsier; *Cross-Organizational Transaction Support for Virtual Enterprises*; Procs. 5th Int. Conf. on Cooperative Information Systems; Eilat, Israel, 2000; pp. 323-334.

[Von03] J. Vonk, P. Grefen; *Cross-Organizational Transaction Support for E-Services in Virtual Enterprises*; To appear in Journal of Distributed and Parallel Databases; 2003.