Web services and business process management

by F. Leymann
D. Roller
M.-T. Schmidt

Web services based on the service-oriented architecture framework provide a suitable technical foundation for making business processes accessible within enterprises and across enterprises. But to appropriately support dynamic business processes and their management, more is needed, namely, the ability to prescribe how Web services are used to implement activities within a business process, how business processes are represented as Web services, and also which business partners perform what parts of the actual business process. In this paper, the relationship between Web services and the management of business processes is worked out and presented in a tutorial-like manner.

Web services are unanimously supported by major suppliers of middleware technology (see for example References 1–5). Standards in this area, such as SOAP (Simple Object Access Protocol),[6] WSDL (Web Services Description Language),[7] and UDDI (Universal Description, Discovery, and Integration),[8] are being proposed or agreed on, and serve as a basis for implementing product features. Many vendors provide early implementations of technologies and proposed or drafted standards (see for example References 9, 10). The World Wide Web Consortium runs a working group defining a set of standards in this area. Known as the "XML (Extensible Markup Language) Protocol,"[11] it is based on input collected from vendors and early adopters of this technology.[12] One of the components of the XML Protocol will have to describe how Web services are integrated into business processes; proposals for such a standard can be found in References 13 and 14.

In this paper we focus on the relation between Web services and business processes, as well as elements needed for a suitable standard. In the next section we summarize the basics of the service-oriented architecture and sketch SOAP, WSDL, and UDDI as the relevant proposed standards. In the following section we describe a high-level scenario that links business processes and Web services. We review some background from the area of workflow technology[15] in the section "Flows between Web services" and demonstrate the applicability of this technology to the problem area. Next, in "Process topologies" we show how enterprises can do business with each other and provide joint services to their customers based on the previously introduced technology. Next we discuss the opportunities for re-engineering business processes. Then, in "Business process management" we put it all together and describe how business processes can be managed in a Web services environment. We briefly sketch IBM product support for business processes and Web services, and in the last section we summarize the contents of the paper and point to some open research questions.

©Copyright 2002 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

0018-8670/02/$5.00 © 2002 IBM

Service-oriented architecture and relevant standards

The service-oriented architecture (SOA) approach, as described in Reference 16, is the latest in a long series of attempts in software engineering that try to foster the reuse of software components.

Indeed, the first major step of this evolution was developing the concept of functions in which programs are broken down into smaller programs through functional decomposition. Central to the concept was the idea of the application programming interface (API) as the contract to which a software component commits. The second major step was developing the concept of object as building block, which combined data and functions into an encapsulated unit. It introduced the notions of classes, inheritance, and polymorphism and thus allowed the construction of class lattices. Specification of signatures was sufficient, because the semantics of the individual classes were "known" to the community. This changed dramatically with the concept of making services available on the Web (Web services). A Web service could be as simple as running a check on a credit number, and as complex as handling a mortgage application. It now required that the services be described and published in a way that anyone can locate and invoke them. This mandates, for example, the use of taxonomies and ontologies to capture syntax and semantics of the offered services.

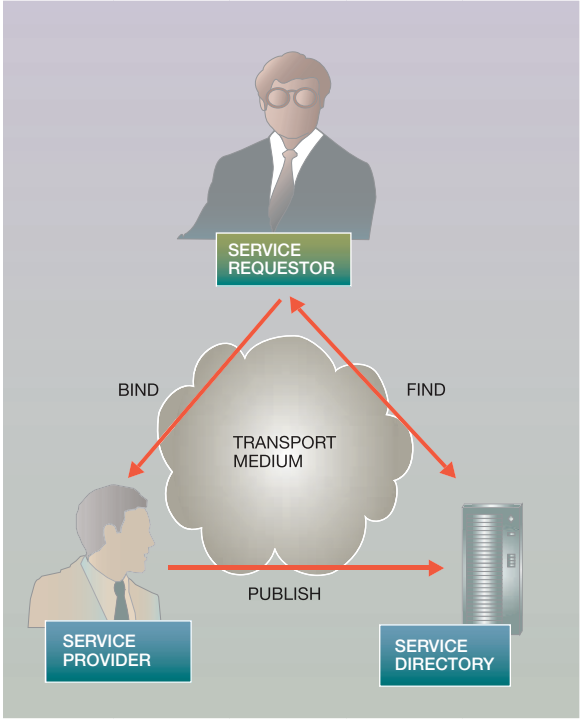
An architecture that supports Web services, known as a service-oriented architecture, covers the following aspects.

- The dynamic discovery of registered services. This includes searching for services that meet certain criteria, especially business criteria such as delivery time, price, etc.
- The organization of services, so that one can easily understand what a service offers
- The description of services, so that a service can be properly invoked. This includes formats and protocols for invoking the Web service.

Please note that in contrast to the service-oriented architecture a "service-based architecture," such as RosettaNet[17] or OBI[18] (Open Buying on the Internet), focuses solely on the formats and protocols between services. As such it represents just one of the pieces of the service-oriented architecture.

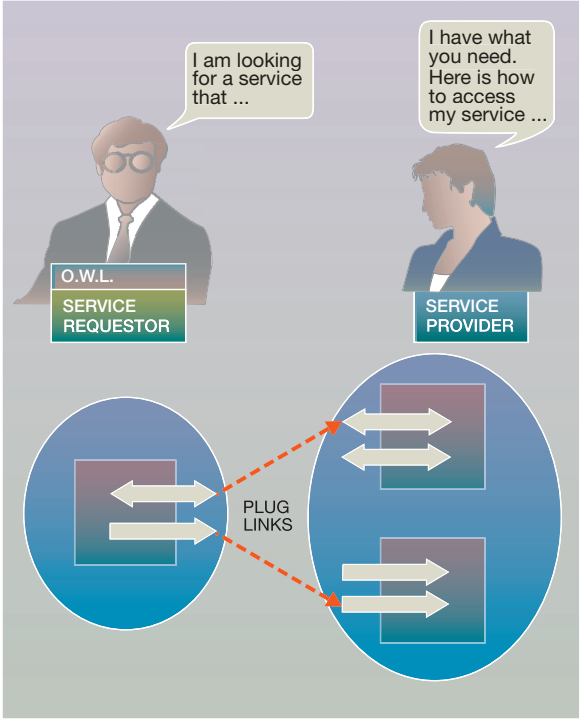
Figure 1 shows the individual components of the service-oriented architecture. The *service directory* is the place where all information about all available services is maintained. A *service provider* that wants to offer services publishes its services by putting appropriate entries into the service directory. A *service requestor* uses the service directory to find an appropriate service, that is, a service that matches its requirements. An example of such a requirement is the price a service requestor is willing to pay for a service. The service directory will thus include not only taxonomies that facilitate the search, but also information such as the price or the delivery time associated with a service. When a service requestor locates a suitable service, it binds to the service provider, using binding information maintained in the service directory. The binding information contains the specification of the protocol that the service requestor must use as well as the structure of the request messages and the resulting responses. The communication between the various agents occurs via an appropriate transport mechanism.

Figure 1 Service-oriented architecture: The players



The Simple Object Access Protocol (SOAP)[6] defines a mechanism for the communication with Web services over the Internet. It specifies the format of mes-

Figure 2 Looking for business partners



sages that are exchanged between the service requestor, the service provider, and the service directory. In particular, SOAP describes how HTTP (HyperText Transfer Protocol) can be used to realize a remote procedure call (RPC) mechanism over the Internet—a combination of the information in the HTTP header and a SOAP body, which allows one to exactly specify the endpoint of a SOAP request. An encoding schema for the SOAP body describes how to exchange requests and the corresponding responses.

Universal Description, Discovery, and Integration (UDDI)[8] defines the structure and the contents of the service directory. UDDI provides two distinct but related pieces of information. The first piece is the registration of service types, which are typically standardized by standards bodies, such as RosettaNet. The second piece of information is the actual business information, such as the name of the company offering the service, the type of service, taxonomies that classify the offered service, and references to those standard service types or to Web Services Description Language (WSDL) specifications.

WSDL[7] provides the capability to describe a Web service, without the need to have it formally standardized. A WSDL description of a Web service provides all information needed to actually invoke it. A *port type*, the *operations* that the port type supports, and the structure of the input and output *messages*, describe a particular service in an abstract way. In order to actually communicate with the service, this abstract specification of the service must be translated into concrete formats and protocols (for example into SOAP over HTTP) based on separate *binding* information. The separation of the binding and the description of the service provides for great flexibility for the service requestor by allowing the service requestor to select the most appropriate binding. The third piece of information defines the provider where the individual services are offered and the *ports* that implement the bindings by which the port types and operations can be reached.

Flexible partnership—an initial scenario

How might the Web services infrastructure be used to provide additional value, and what is still missing? The following high-level scenario will help us to determine that. More details about the concepts introduced in this section can be found in Reference 13.

Let us assume the CIO of a company (depicted by the individual with initials O.W.L. in the figures) has determined that by outsourcing some services the company's operations can be made more efficient. The CIO consults his favorite UDDI directory to find potential business partners that offer such services (see Figures 1 and 2) and, based on certain criteria, he decides to contract with a service provider for the required services.

Technically, this service provider is a business partner that implements a collection of port types covering the required services as its operations; these services are made available via corresponding ports. At his end, the CIO has to provide a port type with operations that work as "stubs" and that are capable of communicating with the ports on the business partner side. In fact, the CIO may decide to provide multiple port types that collectively cover these stubs, i.e., the CIO's company becomes itself a service provider. The operations at both ends that must communicate in order to perform the required functions are wired together via *plug links*. A plug link identifies pairs of operations that communicate with each other, and describes which operation initiates this

communication. Communication here means the exchange of messages, for example, sending a request message and receiving a response message.

Once two businesses establish a partnership, that is, the corresponding service partners are plug-linked, their cooperation can begin. This implies that messages can be exchanged through the plug links defined. But assume that the order in which the plug-linked operations are used is not specified. For example, suppose O.W.L. first uses B service at A (see Figure 3) and then uses D service at C. The service provider expects the sequence of actions in reverse order: first service D is invoked, then service B.

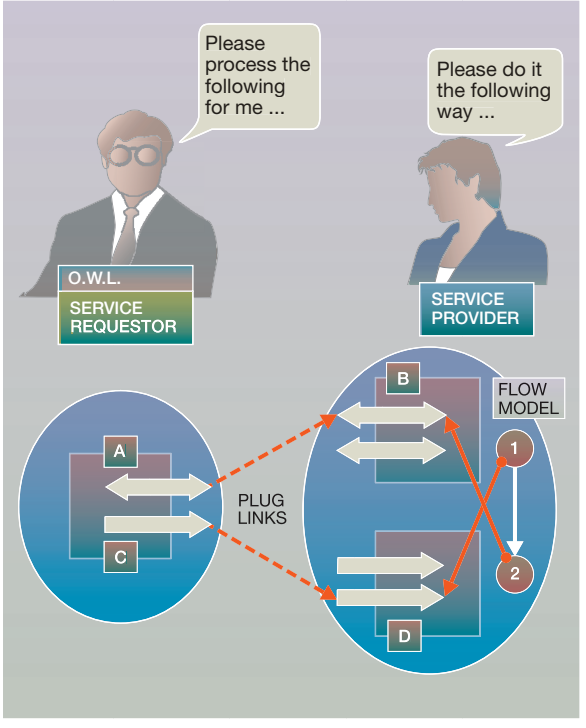
To constrain the order in which operations at a service provider are to be invoked, a *flow model* has to be defined. A flow model is a directed graph that connects nodes, known as *activities*, via *control links*. The control links determine the invocation order of activities by pointing from an activity to its successors. An activity is related to an operation of a port type of the service provider that is constrained by the flow model.

Our service provider has implemented a flow model with activities labeled 1 and 2 (Figure 3). A control link points from 1 to 2; that is, activity 1 must precede activity 2. Furthermore, operation B is specified as the implementation of activity 2, and operation D as the implementation of activity 1. As a consequence, operation D cannot be used before operation B, because the corresponding activity 2 is only ready once activity 1 is completed, i.e., operation B had been used. The next section provides details about flow models.

After a while, O.W.L. is no longer satisfied with the business partner chosen. He has new requirements that have to be met in order to achieve his goals. These requirements are transformed into a query called *locator*[13] (see Figure 4), which is run against a service directory. The locator returns a list of qualifying service providers. From this list O.W.L. chooses a new business partner that better serves his needs.

Finally, O.W.L. decides to provide added-value functions to outdo his competitors. For this purpose he creates a *global model*, i.e., a new Web service that aggregates the services of multiple service providers (see Figure 5). This new global Web service can be published in a UDDI directory, which will enable service requestors to search and find the new service.

Figure 3 Order really matters!



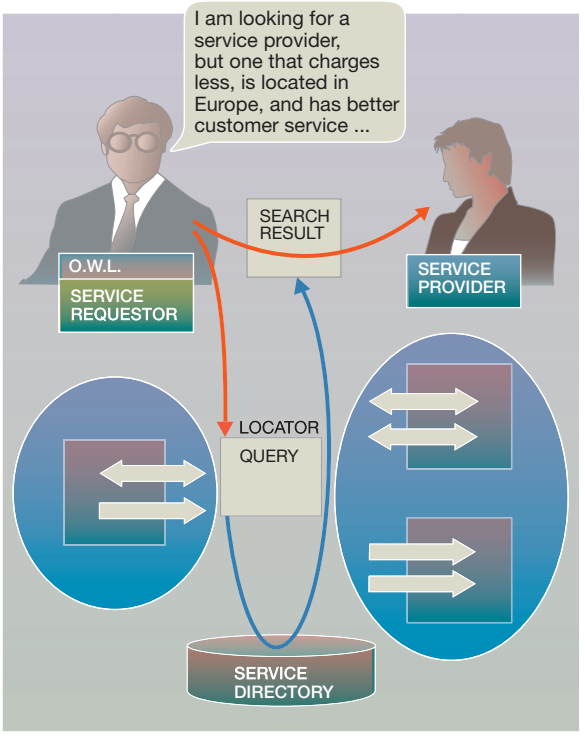
The internal details of the new Web service can be hidden from the user (the service requestor in Figure 5). Thus, a potential user of the new Web service only needs to know about the operations that can be invoked. Additional details about the new Web service can be revealed if needed—for example, if certain ordering constraints have to be obeyed.

Flows between Web services

Figure 6 illustrates the different dimensions of a flow model: the business activities and the sequence in which they are to be performed (the "what" dimension), the resources involved such as the organizational entity that is responsible for carrying out the activity (the "who" dimension), and the tools that are used to handle the activity (the "with" dimension). Since flows often represent business processes, a flow model is sometimes referred to as a *business process model*, or simply a process model.

The first dimension is the process model that defines the different steps or activities that need to be performed, such as Get Order or Check Customer. Each

Figure 4 Searching dynamically for the ideal partner



activity is shown in Figure 6 as an oval labeled with the name of the activity. The potential sequence, in which the different activities are being carried out, is expressed via control links, shown as solid arrows from one activity to the next. For example, the control link from the Get Order activity to the Check Customer activity indicates that the Check Customer activity needs to be carried out after the Get Order activity has been completed successfully. The control link from the Get Order activity to the Check Customer activity is always followed. Sometimes, a control link is taken only if a certain condition associated with the control link (also known as a *transition condition*) evaluates to true. This is illustrated by the two control links leaving the Check Customer activity. The control link to the Accept Order is taken when the transition condition "Good Customer?" evaluates to true. Otherwise the control link to the Reject Order activity is taken. It should be noted, that because an activity may have multiple outgoing control links, it is possible that more than one transition condition associated with an activity could evaluate to true, which causes multiple control links to be followed in parallel.

The different activities in the flow model consume information and also generate information. To make the information generated by an activity available to subsequent activities one uses *data links*. Data links are shown in Figure 6 as dashed arrows. For example, the data link between the Get Order activity and the Check Customer activity indicates that data are passed from the Get Order activity to the Check Customer activity. Credit card information, which has been collected during the Get Order activity, is passed to the Check Customer activity, which might use it to invoke a Web service that checks a credit card number for validity and for credit amount. The information passed between activities together with the information provided by the initiator of the business process is called the *context* of a flow.

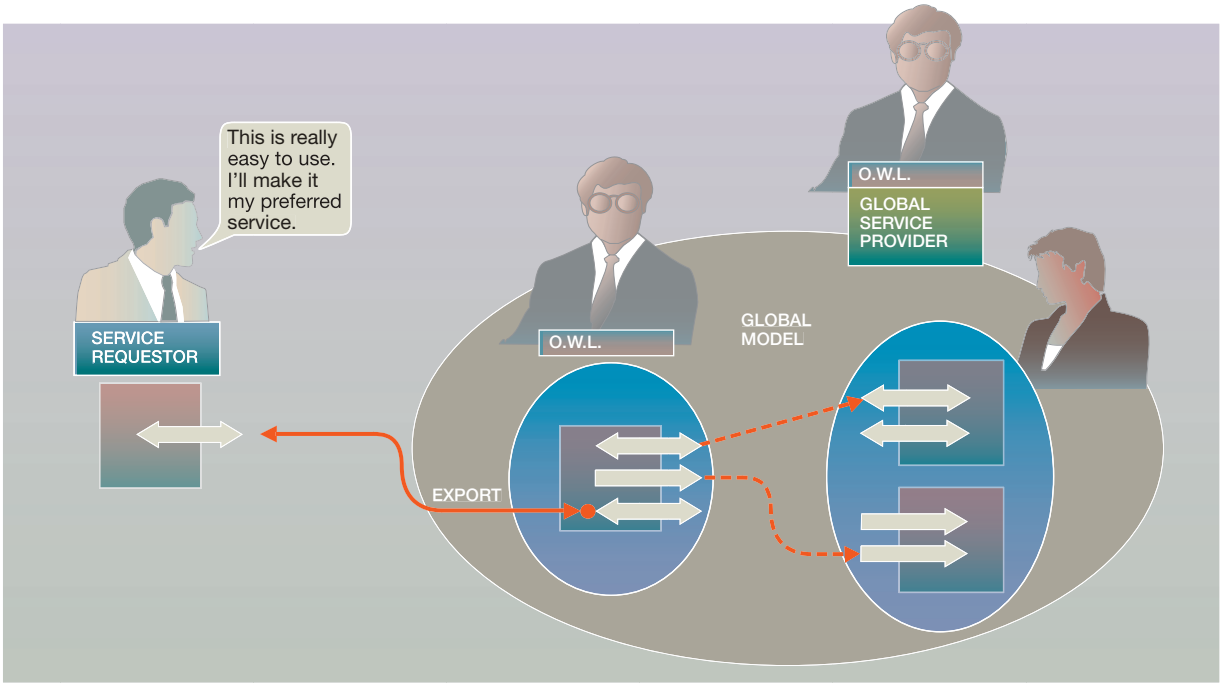
The second dimension is the resource used to monitor, to carry out, or simply to be responsible for a particular activity, usually a person. This is in general determined as the result of a query against a resource database. When the resource involves people, an organizational database is queried (*staff query*), and the process of finding the appropriate people is called *staff resolution*.[15]

The third dimension is the set of tools available to carry out the various activities. This could be a simple phone call being made by the assigned person, the writing of a letter, the invocation of an existing application, such as a CICS (Customer Information Control System) transaction, or the sending of an order to a supplier. The various implementations range from a simple desktop application to a sophisticated Web service offered by some service provider.

Our use of directed graphs to model flows is similar to the work within the Workflow Management Coalition[19] (WfMC) or the Web Services Conversation Language[20] (WSCL). The theoretical underpinning of this approach is rooted in Petri Nets[21,22] and the π-calculus.[23] Languages closer to the latter are XLANG[14] and BPML.[24]

The processing associated with a flow model involves the creation of a flow instance, possibly using initial values, the navigation through the flow model using both the data passed to the flow model and the data generated by individual activities, and the deletion of the flow instance either immediately after completion or after some user-specified time period. Processing of an activity includes selecting the activity, determining the required resources and tools, activation, and termination. Thus the execution of a flow

Figure 5 Creating a new service



instance can be represented as a set of points in the three-dimensional space with the dimensions what, who, and with (the so-called W^3 space).

The execution of a flow instance is controlled via a set of life-cycle functions.[13]

- Call creates and starts a flow instance from a flow model and optionally provides information that is made part of the flow instance context. Control is not returned to the requestor until the flow instance has completed.
- Spawn performs the same functions as the Call life-cycle command with the exception that control is returned to the requestor as soon as the request itself has been processed, either by creating and starting the appropriate flow instance or by returning an appropriate message that the flow instance cannot be created.
- Suspend suspends a flow instance either for a specified time period or until resumed by an appropriate Resume life-cycle function. A suspended flow instance does not involve node activation.
- Resume restarts the activation of a flow instance that has been previously suspended.
- Inquire provides the capability to query for flow instances that have certain properties, such as being in the state Running or being a loan process with a loan value greater than $100000. The result of this function is either a set of flow instances that

Figure 6 The flow model as a multidimensional entity

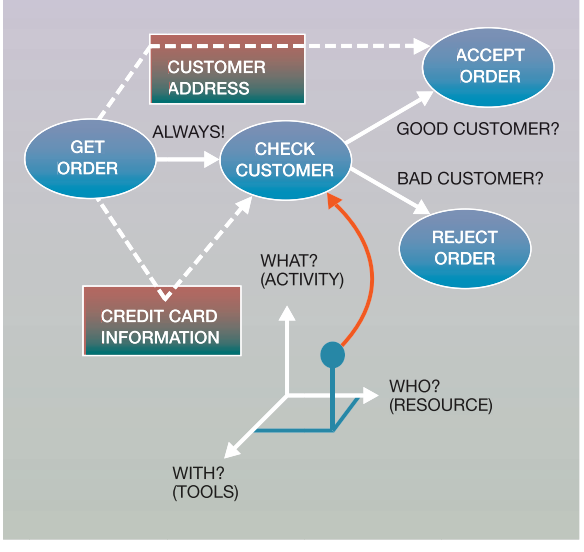
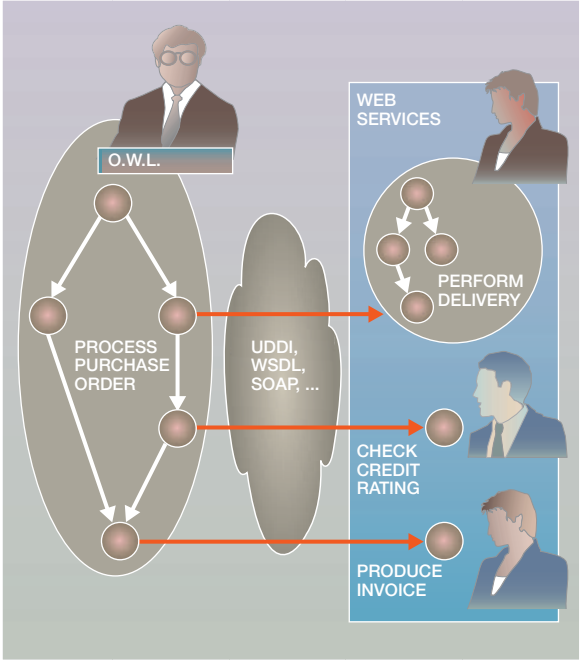


Figure 7 Flow models and Web services



meet the specified criteria, or in the case of a single flow instance, all information about the process, so that the state of process can be displayed.
- Terminate stops the processing of a flow instance, including any activity that may be running.

When a service provider publishes a flow model as a Web service (via WSDL), all life-cycle operations become available as operations of the port type that represents the flow model as a Web service.[10] For ease of use, the standard life-cycle commands can typically be renamed—for example, to be able to use a startOrder request instead of a Call OrderProcess life-cycle command. In addition, all activities that wait for some request from the outside, such as waiting for an order to come in, are also made available as operations of subject port type.[13] If stored in UDDI, service requestors can use this information to invoke the appropriate methods, including the life-cycle commands, associated with the Web service.

Consumers of Web services may post in a service directory those activities that use certain Web services. This enables service providers to query for users that need a particular service.

Figure 7 illustrates the two roles that flow models play in the context of Web services. First, activities of a process can be implemented via a Web service, as shown for three of the activities of the process run by O.W.L. Whether the Web service is actually a flow model or not is transparent to these activities. They simply invoke the Web service, which has been either specified statically, or determined dynamically through the use of the UDDI directory. Second, a flow model is made available as a Web service; see the Perform Delivery Web service. When the Web service is invoked via a Spawn or a Call life-cycle command, an appropriate flow instance is generated from the flow model. Depending on the type of request, control returns to the activity in progress as soon as the request has been accepted or when the flow instance has completed execution.

Process topologies

As explained in the previous sections, flow models can be used to define the behavior of a Web service implemented by O.W.L. or his business partners. They complement the definition of the external interface of the service, which describes operations requesting services from service providers or offering services to service requestors. The flow model describes the flow of control and information between requesting and performing operations of the Web service and can be used by business partners to figure out how they can interact with the given service—both as service requestors and as service providers.

A business process involving two or more business partners can be realized by composing Web services offered by the individual business partners while taking into account the constraints and requirements defined for each participating Web service. For example, let us assume that O.W.L. and his favorite business partner want to collaborate to achieve a particular business goal. Both would define services that they contribute to the composite business process and flow models that define the behavior of those services. To make the composite process work, requesting operations of one process (A, C, and T in the example of Figure 8) must be associated with matching "performing" operations (O, R, and E in the example) in the other process. In the interaction O.W.L. and the business partner switch between the roles of service provider and service requestor. As a result, there are two flows that exchange messages to achieve a common goal. The two interacting business processes are said to have a *peer-to-peer* structure. The example can be easily extended to involve

multiple business partners, e.g., by connecting nodes B and D in O.W.L.'s flow model to operations supported by a third partner.

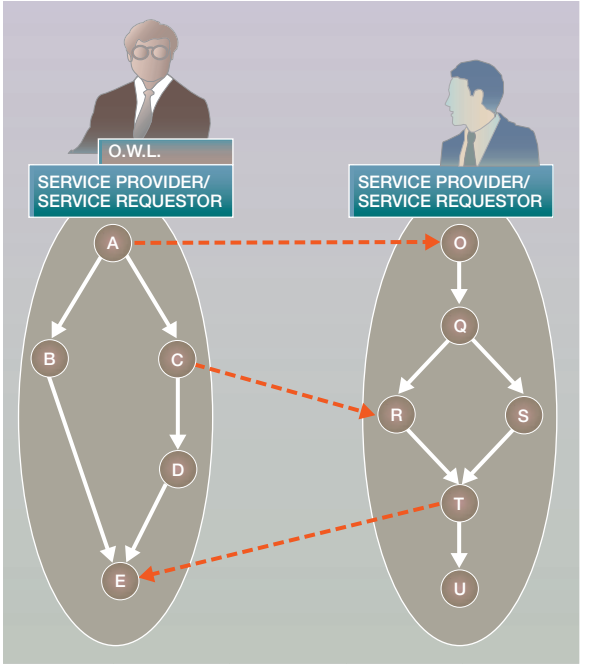
In a peer-to-peer structure, the composite business process is implicitly defined by the flow instances that drive sequencing within the individual processes and by the interaction between the public operations of processes on both sides. Obviously, the necessary synchronization between the services involved requires cooperation of the participating partners; this can be achieved via private agreements, or it can be based on public standards defining the necessary interactions and the roles of the parties involved (for example, the ebXML Business Process Specification Schema[25] provides a formalism for defining public standards for peer-to-peer collaborations).

The peer-to-peer structure is well suited for realization of dynamic collaborations between business partners. Potential participants define their contribution to a process and their requirements from other participants in the form of a Web service definition with flow model annotations. The actual "matchmaking" between requesting and performing operations of these services can be dynamically established, by taking into account the quality of service provided by the potential partners and other factors. In the example of Figure 8, O.W.L. might choose between different partners that provide matching operations for process steps A, C, and E, or he might even choose to use several different partners without having to make any changes to his own process.

In many situations, however, the interactions between business partners can be described by a single top-level (root) business process. This business process defines the steps necessary to achieve the overall business goal and maps these steps to services provided by the contributing partners. The resulting global structure of the business process involving the set of business partners is said to be *hierarchical* (see Figure 9).

The overall process in a hierarchical structure often uses the life-cycle operations provided by process flow services to integrate subprocesses into the top-level process (e.g., the flows of partners P2, P3, and P4). Depending on the nature of the subprocess it either spawns the service implementing the subprocess and waits for its results, or it uses the synchronous Call operation to execute the subprocess.

Figure 8 The peer-to-peer structure of two interacting business processes

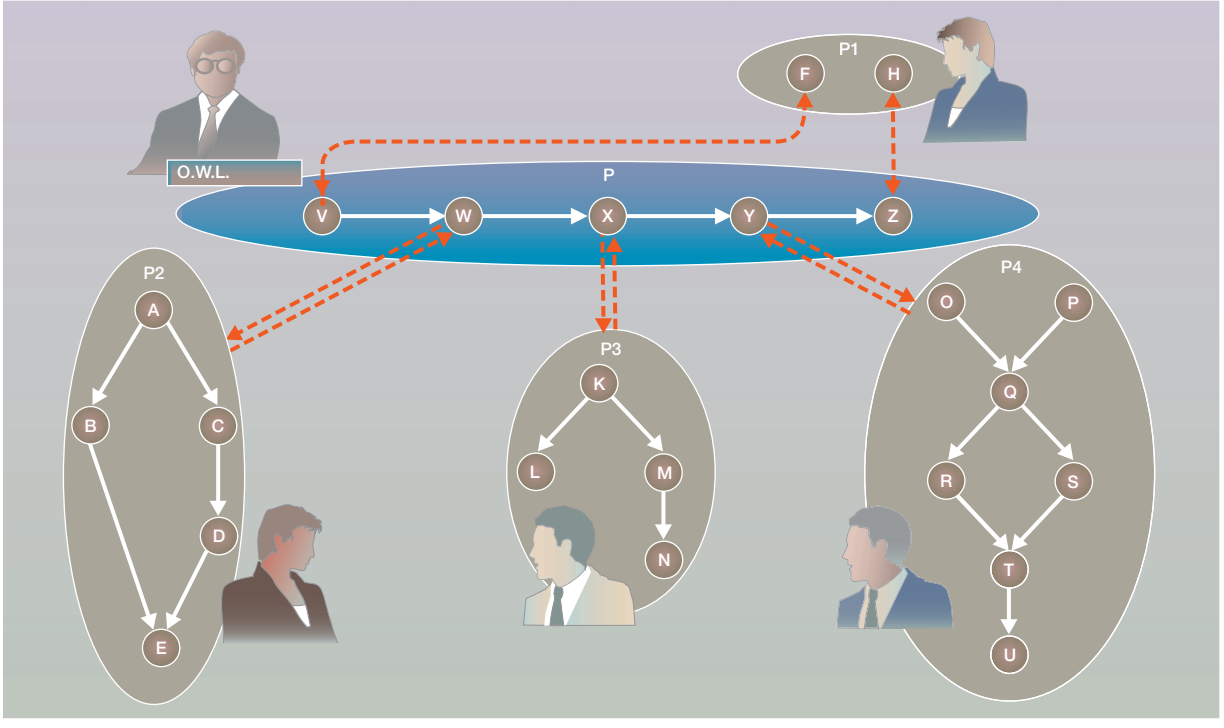


But activities of the overall process may also be implemented via "discrete" operations (i.e., services without any structure) provided by a business partner. In the example depicted in Figure 9, partner P1 implements support for V and Z of the top-level process as individual activities F and H.

The hierarchical structure is typically used in "traditional" business relationships where interactions between cooperating partners are agreed to in advance. The overall process is developed in a top-down fashion, starting with a definition of the top-level process. This process is decomposed into coarse-grained process steps that are performed by individual partners; the partners further decompose their respective tasks into subprocesses that they control.

Process management and monitoring of the overall process status is generally easier in hierarchical structures than in peer-to-peer interactions since the main process provides a single point of control and additional details can be obtained by "drilling down" into the process along the lines of the process hierarchy. However, peer-to-peer processes can also be instrumented to support the management and

Figure 9 The hierarchical structure of a business process involving multiple partners



monitoring of federated processes (see the section "Business process management" for more details).

It is important to note that in the hierarchical scenario the top-level process is not necessarily owned by one of the participating partners. It merely defines the rules for cooperation of the partners in a "virtual enterprise," thus choreographing the collaboration between the partners of this virtual company. The top-level process can be hosted by one of the partners or by an outside party, and in many cases the host can be dynamically determined depending on the specific application scenario. As a consequence, the process must be specified in a standardized way that can be executed in a number of process execution environments—this is one of the motivations behind IBM's proposal for the WSFL[13] standard that supports definition of flow models that can be implemented by a variety of process engines.

In many cases, a mixture of hierarchical and peer-to-peer structures is used to realize complex multipartner business processes. For example, in Figure 9, partner P4 might have established a peer-to-peer relationship (not shown) with yet another business partner in order to integrate the services that this partner supports within the hierarchical structure shown.

Flow models can be used to describe the *public* aspects of implementing a business service. They provide information that the service provider shares with potential users of the service by publishing it in a service directory. For example, the root process in a hierarchical structure realized by a virtual enterprise should be visible to all customers of the virtual enterprise. But in some cases the flow model that defines the behavior of a service can be hidden from potential users of the service because the structure is not relevant for the interactions between the service and its users. For example, the business processes owned by partners P2, P3, and P4 in Figure 9 can be represented as having no inner structure; the root process only needs to know how to invoke the services provided by these processes. This is important, because business processes are very often company assets that must not be revealed.

Sometimes, selective parts of a business process of a partner must be made public. For example, in Figure 9 partner P1 implements activities V and Z of the root process as F and H. In more complex situations, the support is viewed as a flow model that represents the public view of the service provided. Such a flow model in fact is in general a graph created by a projection mapping of the graph representing the business process owned by the service provider, that is, a "public view" on a "private flow." Figure 10 shows the public views of the business processes shown in Figure 8.

In a top-down development scenario, a given public process could be further refined into a more complex, private (i.e., "hidden") flow model that integrates externally visible operations with internal operations, implemented using intraenterprise services. In a bottom-up development scenario, specific aspects of a complex process would be projected into a subgraph that is exposed to the outside world.

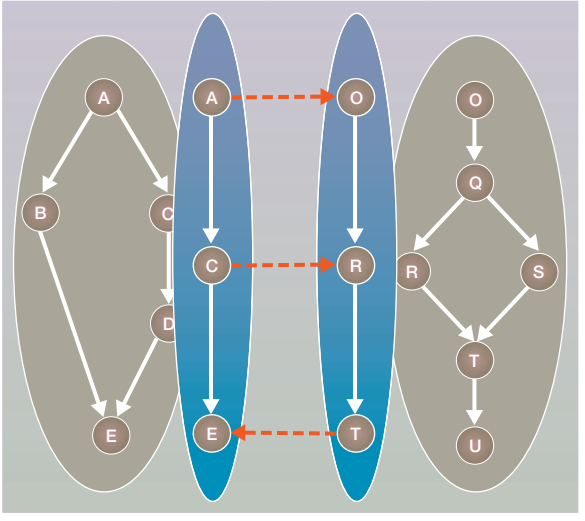
"Private" and "public" are relative terms in dynamically changing interactions between business partners. Private views of a process can become public when, say, an enterprise decides to outsource components of a business process. Public interactions can become private views of a process as the result of a merger involving two partners. As a consequence, it is very useful to have a flow model formalism that can be used both for modeling private and public aspects of a business process.

Business process re-engineering

In many industries, such as finance and insurance, the speed with which new products can be rolled out to customers provides for a competitive advantage. An insurance company, for example, may give a 10 percent discount for cars parked overnight in a closed garage, once it has been determined, by appropriate market research, that this would increase their market share. Thus it is important that the offering goes early to the market in order to maximize the competitive advantage (until the competition catches up).

In fact, those companies equate the business process (flow model) used to create a product with the product itself. For example, the business process used to create an insurance policy for closed-garage car-holders is equivalent to the insurance policy itself. That means, changing the closed-garage car-holder insurance policy means changing the underlying business process. In other words, creating a new insurance policy means designing a new business process. The faster the business process can be re-engineered, the faster a new type of insurance policy can be developed and offered, providing competitive advantage in rapidly changing environments. This rapid change of business processes is facilitated through the use of workflow management technology that separates the construction of the flow model from the construction of the individual activity implementations (i.e., two-level programming).[15]

Figure 10 Public views on private flows



Being able to change a flow model or create a new one quickly is providing one competitive advantage; carrying out business processes efficiently provides another. Carrying out business processes efficiently means to optimize the execution of business processes according to some measurable criterion, such as cost, process execution time, or customer satisfaction. This mandates that, during the design of the business process, the different parts of the flow model, such as activities or transition conditions, are instrumented to measure relevant metrics, such as the time it takes to carry out an activity. Simulation tools can be used to determine the costs associated with the individual activities or the total business process, as well as to calculate the minimum and maximum processing times well before the business processes are put into production.[15] If one determines, for example, that providing an activity in-house is more expensive when compared to what an equiv-

alent Web service costs, then the activity is a candidate for outsourcing. The Perform Delivery outsourced activity in Figure 7 illustrates such a case.

The instrumentation of business processes can later be used for monitoring when the business process enters the production phase. It can be used, for example, in a dashboard-type of tool to detect unusual conditions.

Business process management

Business process management (BPM) is all about transferring the results of business process re-engineering discussed above into production. BPM technology provides not only the tools and infrastructure to define, simulate, and analyze business process models, but also the tools to implement business processes in such a way that the execution of the resulting software artifacts can be managed from a business process perspective.

The BPM infrastructure provides the run-time environment for public and private process models as discussed in the previous sections. It allows users to monitor the execution of individual processes, to analyze the overall behavior of a set of business processes, to verify their successful performance, and to provide input for process optimization. It is important to note that public and private process models are only one half of a complete business process realization; the other half are services that implement the process steps and these services must be managed together with the process models.

The scope of a business process can be limited to a particular department in an enterprise, it may span multiple divisions within an enterprise or it may require interenterprise collaborations. In the simple case of a departmental process, the BPM infrastructure will probably be homogeneous (e.g., a specific process management system, local resources). In the general case, however, and especially in interenterprise processes, processes are executed in a federated process management infrastructure. Our CIO O.W.L. has to deal with a situation in which the business process he manages relies on one or more external service providers. This has impact on process modeling (e.g., selecting the optimum service provider as part of the business process simulation), on the deployment of the overall process into a federated environment, and on process monitoring (e.g., monitoring input that can be expected from the external partner). The BPM *solution builder* tools and the BPM *dashboard* facilitate these tasks.

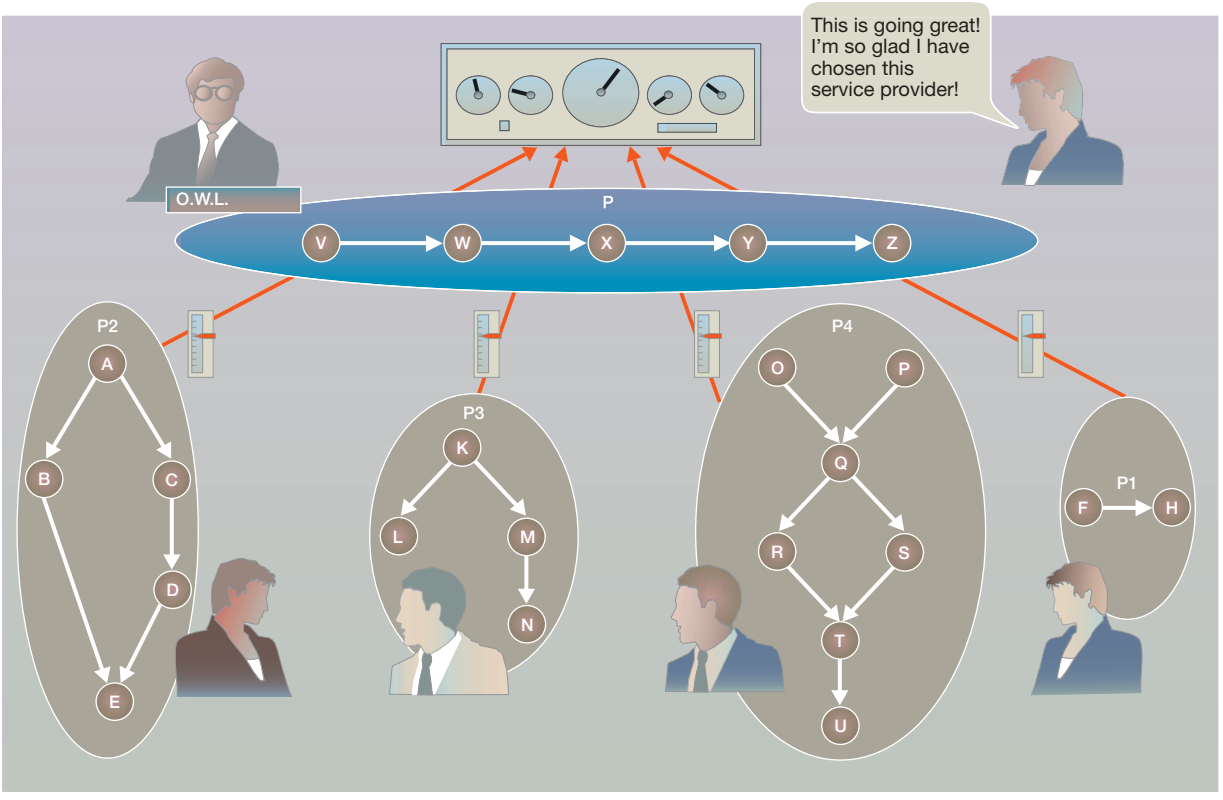
O.W.L.'s company uses BPM solution builder tools to define public and private processes, to instrument them for process monitoring, to identify the services that implement the process steps, and to implement services used internally to implement the overall solution. These tools also support the deployment of business processes and services by supporting their run-time environments (e.g., Web services gateway, workflow management system, message broker, application server).

O.W.L. starts the solution building process by defining the interactions between his company and the external partners involved. As previously mentioned, hierarchical or peer-to-peer structures can be used to specify the interactions at this level. In the hierarchical case, an owner for the top-level process is identified, whereas in the peer-to-peer case the interoperating partners drive the execution of the various components of the process. Binding of process steps to specific service providers and their services can be done at this time (based on the service level guarantees associated with their service). Alternatively, it can be deferred to process execution time, in which case the process model contains selection criteria for services that are resolved during process execution.

The "interaction contract" defined by the overall process model specifies to what extent individual services support process monitoring and auditing. Some external services may allow "drill-down" into the internals of the service realization, whereas others may only log events reporting the start and the end of their participation in the process. Depending on the capabilities of the process components, O.W.L. (and his partners) will instrument the overall process to enable process monitoring when the process is put into production. The solution builder tools also support deployment of the business process and its required services. In the federated scenario we are interested in, this includes the exchange of contracts between O.W.L. and his partners.

Once a process has been deployed, O.W.L. will use BPM *solution management* tools, e.g., the business dashboard, to manage the execution of the solution artifacts from a business process perspective. The solution elements will produce business events (as defined during process instrumentation) that can be used to report on the progress of individual pro-

Figure 11 Federated dashboard



cesses, and to aggregate process execution audit information for analysis of the overall behavior of processes in a particular business domain (see Figure 11). As explained above, solution elements are deployed into a variety of run-time environments and their individual progress reports are collected by a federated dashboard infrastructure.

The dashboard can provide information about process execution status and progress on various levels of detail, depending on the interest of the target audience and the quality/quantity of information provided by the solution elements. Some users may only be interested in a high-level summary of process status, whereas others may wish to drill down into a detailed analysis of specific solution components.

For a public process, all participants may have access to performance information of the top-level process, whereas details of process execution inside a particular process step might only be available to those responsible for implementing that process step.

Note that for Web service interactions, process monitoring is significantly simpler if the overall process is structured hierarchically rather than in a peer-to-peer fashion. In the hierarchical case, the top-level process is managed by a specific process participant who can provide all the information on the progress of this process that is required for process monitoring and auditing at this level. Obtaining information beyond that level requires cooperation of other service providers involved who would have to interact with a federated dashboard infrastructure to support real-time status queries and postmortem process analysis. The same is true on all process levels (including the top-level one) in the peer-to-peer case. It should also be noted that the dashboard service can actually be realized as another Web service that participates in business processes as an observer rather than an active participant. Similarly, Web services participating in business processes can expose monitoring operations that support status queries or propagate progress events as part of their interface.

Product support

This section provides a brief overview of the IBM technologies and products that support realization of business processes in general, and Web service collaborations in particular (for additional details see Reference 26).

The IBM WebSphere* J2EE** (Java 2 Platform, Enterprise Edition) environment provides the basic facilities for implementing business components and the tooling to make those services available as Web services. Using WebSphere tooling, Java** components such as EJBs (Enterprise JavaBeans**) can be rendered as WSDL-defined Web services that are accessible through the WebSphere SOAP gateway. And for any given WSDL-defined Web service, Java wrappers can be generated that make the service accessible to any WebSphere intraenterprise business component. This provides the basic machinery for publishing intraenterprise business components as Web services and for using external Web services as components in intraenterprise processes.

Many IBM products use this or related technologies to render their specific software artifacts as Web services or to use Web services to realize those artifacts (see Reference 10). We will use WebSphere Business Integrator[27] (WBI) to illustrate the collaboration of a number of IBM middleware components for implementing end-to-end business processes both at the intraenterprise and the interenterprise level.

The WBI public process gateway supports the definition and the realization of the public interface of a service offered by an enterprise. It facilitates the selection of business partners and the negotiation of interaction terms with those partners. It serves as an intelligent switchboard between the external side of the business services and the business partners that request services from the enterprise or perform services requested by the enterprise. The WebSphere Application Server provides the means to realize business services either based on existing applications (e.g., via J2EE connectors), or implementing new business logic (e.g., via EJBs).

WBI uses two additional components to bridge the gap between external requests and internal business operations: the Information Delivery Manager (IDM) and the Business Flow Manager (BFM). IDM facilitates information exchange between potentially incompatible business services; it supports message-based interactions between services via JMS (Java Message Service) and uses the IBM MQSeries* Integrator message broker for information routing and transformation between messaging end points. In addition it uses micro scripting for constructing adapters that perform the scripting necessary to map between the services offered by an application and the service interface required in the context of a business process. BFM provides technology for composition of business services into business processes. It uses flow composition to realize interactive or automated workflow processes that can be deployed into MQSeries Workflow; it uses state controllers to manage the life-cycle state of important business entities manipulated by a business process and to support state dependent process brokering between the workflow processes involved in executing particular tasks of that process.

The WBI Solution Studio provides the integrated set of tools for modeling end-to-end business processes, instrumentation of the process models for auditing and monitoring, and (in cooperation with the WBI Solution Manager) the means to deploy the various software artifacts into the appropriate execution environments. The WBI Solution Manager also provides facilities for tracing, exception logging, and auditing business processes and enables business-level process monitoring and analysis.

Conclusion

In this paper we have discussed the relationship between Web services and business processes. We have shown that Web services can be used as implementations of activities within a business process, and that business processes in turn can be externalized as Web services. The business processes that can be built based on this simple observation range from simple to complex. The scope of business process management has been sketched, covering the specification of business processes based on an associated programming model, as well as the monitoring of business processes especially based on a federated dashboard. The IBM technology and products supporting BPM, and mainly based on the WebSphere Business Integrator family, have been briefly described.

Dynamic e-business requires the study of possible interaction patterns between business partners to ensure proper collaboration. The concepts of the WSFL standard have been sketched. More advanced concepts, such as public views on private flows have been introduced. The topic leaves a number of questions

for further research. For example, how can we derive from a given private flow a public view that encompasses a certain set of activities; and how do we prove that a given private flow is an implementation of a certain public view?

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc.

Cited references

1. See http://www-4.ibm.com/software/solutions/webservices/resources.html.
2. See http://msdn.microsoft.com/library/techart/websvcs_platform.htm.
3. See http://technet.oracle.com/products/dynamic_services/.
4. See http://www.sun.com/software/sunone/wp-arch/.
5. See http://e-services.hp.com/.
6. See http://www.w3.org/TR/SOAP12.
7. See http://www.w3.org/TR/WSDL.html.
8. See http://www.uddi.org.
9. F. Skrzypczak and R. Junghuber, "Web Services Process Management Toolkit," IBM Corporation (2001); see http://www.alphaworks.ibm.com/tech/wspmt.
10. P. Lambros, M.-T. Schmidt, and C. Zentner, "Combine Business Process Management Technology and Business Services to Implement Complex Web Services," IBM Corporation (2001); see http://www-4.ibm.com/software/solutions/webservices/pdf/BPM.pdf.
11. See http://www.w3.org/TR/xmlp-am/.
12. See http://www.w3.org/2001/03/WSWS-popa/.
13. F. Leymann, "Web Services Flow Language," IBM Corporation (2001); see http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.
14. S. Thatte, "XLANG—Web Services for Business Process Design," Microsoft Corporation (2001); see http://www.gotdotnet.com/team/xml_wsspecs/xlang-c.
15. F. Leymann and D. Roller, *Production Workflow*, Prentice Hall, Inc., Upper Saddle River, NJ (2000).
16. S. Burbeck, "The Tao of e-Business Services," IBM Corporation (2000); see http://www-4.ibm.com/software/developer/library/ws-tao/index.html.
17. See http://www.rosettanet.org/.
18. See http://www.openbuy.org/.
19. See http://www.wfmc.org.
20. See http://www.uddi.org/pubs/wsclBPforUDDI_5_16_011.pdf.
21. E. Best and C. Fernandez, *Non-Sequential Processes: A Petri Net View*, Springer-Verlag, Berlin (1988).
22. W. Reisig, *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*, Springer-Verlag, Berlin (1998).
23. R. Milner, *Communicating and Mobile Systems: The π-Calculus*, Cambridge University Press, Cambridge, UK (1999).
24. See http://www.bpmi.org.
25. See http://www.ebxml.org/specs.
26. D. Ferguson, "IBM Web Services: Technical and Product Architecture Roadmap," IBM Corporation (2001); see http://www-4.ibm.com/software/solutions/webservices/pdf/roadmap.pdf.
27. WebSphere Business Integrator, IBM Corporation; see http://www.ibm.com/software/webservers/btobintegrator/index.html.

Accepted for publication November 3, 2001.

Frank Leymann *IBM Software Group, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (electronic mail: ley1@de.ibm.com).* Dr. Leymann is an IBM Distinguished Engineer and a member of the IBM Academy of Technology. He is the chief architect of IBM's workflow technology, and a member of the IBM Application Integration Middleware (AIM) Architecture Board that sets the overall direction of IBM's middleware. He has worked on database systems, database tools, and transaction processing. Dr. Leymann has published many papers in various journals and conference proceedings, filed a multitude of patents, and is the coauthor of textbooks on repositories and workflow systems. He served on program committees and organization committees for many international conferences and is editor of the journal of the database management system special interest group of the German computer society *Gesellschaft für Informatik* (GI).

Dieter Roller *IBM Software Group, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (electronic mail: rol@de.ibm.com).* Mr. Roller is an IBM Senior Technical Staff Member and a member of the IBM Academy of Technology. He held several technical and management positions in his IBM career. His current focus is on the architecture and design of MQSeries Workflow, where he is contributing to all facets of the development and enterprise-wide deployment of workflow-based applications, and is deeply involved in various customer projects. Mr. Roller has published papers in various journals and conference proceedings, mainly on workflow technology, he has filed many patents, and has given talks at conferences and professional society meetings. He is a coauthor of a textbook on workflow systems.

Marc-Thomas Schmidt *IBM Software Group, Hursley Park, Winchester Hampshire S021 2JN, United Kingdom (electronic mail: mts@uk.ibm.com).* Mr. Schmidt is an IBM Senior Technical Staff Member, the lead architect for IBM's WebSphere Business Integrator solution, and a member of the AIM Architecture Board. He has been working in the area of business process management for many years, focusing on workflow management, message brokering, integration of message-based and object-oriented technologies, and more recently applications of BPM technologies for implementing Web services. He has published several papers, filed a number of patents on these topics, and contributed to workflow standardization efforts in the Object Management Group and Workflow Management Coalition.