Towards Patterns of Web Services Composition

B. Benatallah[1], M. Dumas[2], M-C. Fauvet[1,4], F.A. Rabhi[1]
[1] School of Computer Science
University of New South Wales, Sydney NSW 2052
boualem@cse.unsw.edu.au

[2] Cooperative Information Systems Research Centre
Queensland University of Technology, Brisbane QLD 4001

[3] School of Information Systems, Technology and Management
University of New South Wales, Sydney NSW 2052

[4] On leave from LSR-IMAG, University of Grenoble, France

UNSW-CSE-TR-0111

November 2001

Abstract

The ability to efficiently and effectively share services on the Web is a critical step towards the development of the on-line economy. Virtually every organisation needs to interact with manifold other organisations in order to request their services. Reciprocally, an organisation providing a service is often required to interact with a large and dynamic set of service requestors. The lack of high level abstractions and functionalities for Web service integration has triggered a considerable amount of research and development efforts. This has resulted in a number of products, standards, frameworks and prototypes addressing sometimes overlapping, sometimes complementary aspects of service integration.

In this report we summarise some of the challenges and recent developments in the area of Web service integration, and we abstract some of them in the form of software design patterns. Specifically we present patterns for both bilateral service-based interactions, multilateral service composition, and execution of composite services both in a centralised and in a fully distributed environment. The report also shows how these patterns map into a variety of implementation technologies including object-based approaches (e.g. CORBA and EJB), EAI and ERP suites, cross-enterprise workflows, EDI and XML-based B2B frameworks.

1 Introduction

The growth of the Internet has unleashed a wave of innovations that are reshaping the way organisations interact with their partners and customers. In particular, the concept of electronically-accessible service (also known as *e-service* or *Web service*) has gained a considerable momentum as a paradigm for supporting both Business-to-Consumer (B2C) interaction and Business-to-Business (B2B) collaboration.

In a nutshell, the term (Web) service as used in this report, denotes an abstraction of a set of computational and/or physical activities intended to fulfill a class of customer needs or business requirements. In other words, a service provides an interface to access functionalities offered by information systems, application programs, and business processes. Typical examples of services include booking an airline ticket through a HTML-based interface, or providing access to a database of wheather forecasts through a CORBA-based interface.

Following the recent explosion in the number of Web-accessible services, established enterprises are continuously discovering new opportunities to form alliances with other enterprises, in order to share their costs, skills and resources by offering *integrated services* (also called *composite services*). An example of an integrated service is an accounting management system that uses payroll, tax preparation, and cash management services as components. The component services may be outsourced to business partners, and these business partners can in turn outsource part of the activities involved by the delivery of a service to other businesses or organisations.

The ability to efficiently and effectively share services on the Web is a critical step towards the development of the on-line economy. Virtually every organisation needs to interact with manifold other organisations in order to request their services. Reciprocally, an organisation providing a service is often required to interact with a large and dynamic set of service requestors.

Unfortunately, the technology to organise, abstract, search, compose, evolve, analyse, monitor, and access Web services has not entirely kept pace with the rapid growth of available opportunities. Indeed, the development of integrated services is still largely ad-hoc, time-consuming and requiring a considerable effort of low-level programming. This approach is clearly tedious and hardly scalable because of the volatility and size of the Web. Worse, as service integration is done in an ad hoc manner, it is likely to rely on proprietary solutions, thereby rendering inter-service coordination a difficult task.

The lack of high level abstractions and functionalities for Web service integration has triggered a considerable amount of research and development efforts, both in the academia and in the industry. This has resulted in a number of products, standards, frameworks and prototypes addressing sometimes overlapping, sometimes complementary aspects of service integration. In particular, numerous XML-based standards for describing, advertising, retrieving and inter-connecting services have been defined, and some of these standards have been adopted, or are expected to be adopted by manifold commercial products, including catalog management suites and business process managers. Although the emergence of these standards is undoubtedly a significant step towards facilitating service integration, the need to provide tools and methodologies supporting the rapid service integration of dynamic services still subsists.

In this report we summarise some of the challenges and recent developments in the area of Web service integration, and we abstract some of them in the form of patterns. Specifically we present patterns for both bilateral service-based interactions, multilateral service composition, and execution of composite services both in a centralised and in a fully distributed environment. Due to the emerging nature of the topic addressed in this report, these patterns are still in an early stage of development, and could rather be called "proto-patterns".

The remainder of the report is organised as follows. In section 2 we provide an overview of related fields such as application integration, workflow management and Web service development. Section 3 describes patterns related to service request, provisioning, and outsourcing. Section 4 introduces patterns related to the specification of composite services involving multiple partners, while section 5 describes patterns of execution of composite services. Finally, section 6 provides some concluding remarks.

2 Review of enabling technologies

Service composition is an active area of research and development in different fields including component-based frameworks, cross-enterprise workflows, Electronic Data Interchange, XML-based B2B frameworks, and agent-based frameworks. While these approaches provide solutions for different application domains, our discussion will concentrate on service composition requirements. We note that these approaches are not orthogonal. Indeed, in many cases, they rely on each other. For instance, workflows may use mediators, agents may use XML, etc.

2.1 Component-based Frameworks

Component-based mediators for e-commerce applications (e.g., OFFER, EMP, and GEM) [Dog98, Dog98, Dog99, Whi97] typically rely on distributed object frameworks such as CORBA, DCOM, EJB, and other state-of-the art technologies such as Enterprise Application Integration (EAI) (e.g., IBM MQSeries) and Enterprise Resource Planning suites (e.g., SAP R/3), database gateways and transaction monitors [Bro00]. Briefly stated, components are software modules that can be independently developed and delivered. They are designed to interoperate with each other at runtime and can be developed using different technologies. Salient features of component-based systems include integration with legacy applications, interoprability across networks, and portability on different hardware and software platforms [BSZ98]. In general, components represent high-level services such as business objects.

Component-based mediators focus on the integration of a small number of tightly coupled services. In addition, This approach puts emphasis on the syntactical integration (e.g., wrapping heterogeneous systems).

Enterprise Application Integration (EAI) suites (e.g., IBM's MQSeries and TibcoSoftware's TIB/Active Enterprise Suite, TSI Software's Mercator product, and IBM SanFrancisco) provide standard data and application integration facilities (e.g., pre-built application adapters, data transformations, and messaging services among heterogeneous systems). Another solution to application integration is ERP systems. ERP systems promise a single, homogeneous solution for a number of back-office applications. Again ERP suites provide static integration of enterprise applications within the scope of the suite. In many situations it is required to integrate applications packaged using different autonomous ERP systems, integrate ERP applications with legacy systems, etc. More specifically in B2B e-commerce this integration is required both within and across enterprises. Some EAI suites provide messaging services among ERP systems. For instance TSI Software's Mercator product offers messaging services among the ERP systems SAP R/3 and PeopleSoft. The differences between component-based mediators, ERP systems, and EAI suites cannot easily be identified. However, they all introduce a common layer that spans the participant systems. This layer (middelware) a common data representations and application APIs to access the heterogeneous participant systems.

2.2 Cross-enterprise Workflows

Traditional workflow systems are based on the premise that the success of an enterprise requires the management of business processes in their entirety. Indeed, an increasing number of organisations have already automated their internal process management using workflows and enjoyed substantial benefits in doing so. Traditional workflow systems are not very effective for the needs of composite services and their complex partnerships, possibly among a large number of highly evolving services. Current research efforts in the workflow area promise to deliver a next generation workflow systems that has the ability to easily thread together cross-organisational business processes, supporting the integration of diverse users, applications, and systems [YP00].

The purpose of cross-organisational workflows is to automate business processes that interconnect and manage communication among disparate systems. Early projects in this direction (e.g., InterWorkflow, WISE, FlowJet) focus mostly on the integration of a known and small number of tightly coupled business processes [Geo99, Ga99, GT98, DR99]. New emerging service composition projects (e.g., CMI, EFlow, CrossFlow, Mentor, CPM, SELF-SERV, and ADEPT consider loosely coupled services [VLD00, CH01, BDSN02]. They consider some critical requirements of B2B e-commerce such as dynamic selection, adaptability, and external manageability of services.

2.3 Electronic Data Interchange - EDI

EDI is commonly defined as the interoganisational application-to-application transfer of business documents (e.g., purchase orders, invoices, shipping notices, billing and payment information) between computers. EDI documents are structured according to a standard and machine-processable format (e.g., ANSI X12 and UN/EDIFACT)[ADGY98]. Trading partners exchange business documents via a Value-Added Network or VAN, using an EDI standard as follows. The document must be created in the business application of the sender (e.g., an invoice document). The mapper software is used to describe the relationship between the information elements in the application and the EDI standards. The EDI translator software converts the document into an EDI message according to the agreed upon standard. The translator wraps the EDI message in an electronic envelope that has an identifier for the receiver. The actual transmission of the electronic envelope is achieved by the communication software which maintains the phone numbers of the trading partners and performs dialing and exchanges. The VAN reads the identifier on the envelope and places it in the mailbox of the receiver. At the receiver side, the reverse process occurs.

EDI depends on a moderately sophisticated technology infrastructure. This infrastructure is based on proprietary and expensive networks (added value networks), mainframes, and ad-hoc development, where direct communication between two organisations can be used to exchange electronic documents. EDI has been mostly used for the automatic transfer and processing of documents in industries such as goods transportation, food manufacturing, and automobile production, in which organisations trade on high volumes. EDI although adopted by several organisations as an electronic commerce approach, it remains not affordable by the majority of the business community. EDI is limited in its ability to enable the integration of a large number of dynamic services. It is mainly used to communicate standardised business documents.

EDI has been extended recently in many directions. In particular, the combination of EDI and Internet removes the cost of using proprietary networks. Among existing efforts in this direction, we mention the Open Buying on the Internet (OBI)[Bus01]. OBI is standard that laverages EDI to define an Internet-based procurement framework. More precisely, OBI is intended for high-volume, low-dollar amount transactions, which account for 80% of the purchasing activities in most organisations. These are transactions for Maintenance, Repair, and Operations (MRO) materials, office supplies, laboratory supplies, and other indirect materials (i.e., those that are not used in a production process) [Bus01]. OBI relies on the ANSI X12 EDI standard to describe the content of order documents (i.e., OBI order requests and OBI orders). Order documents are encapsulated in OBI objects (or messages). OBI objects encapsulate also other non-EDI messages such as digital signatures of buying and selling organisations. It uses the SSL (Secure Sockets Layer) over HTTP for secure communications between partners. It also uses digital signatures and digital certificates for ensuring the authenticity and integrity of messages.

2.4 XML-based B2B Frameworks

XML is emerging as the standard for exchanging data over the Internet. It provides a simple and extensible tag-based language that can be use d to represent all forms of digital information (i.e, structured database records, unstructured documents and everything in between). XML promise several features for Interoperability of both data and services across applications and platforms. Encoding business information in XML will eliminate the need for one-to-one information translation. XML will provide a common format to publish business information. By publishing publicly business information in XML, businesses would allow unpredictable interactions with each other. XML-based frameworks, protocols, and standards (e.g., SOAP, BizTalk, RosettaNet, cXML, eCO, and XML-based Web Service Description Language - WSDL, Universal Description, Discovery, and Integration - UDDI) aim at providing a common building blocks (e.g., vertical ontologies, message representation, service advertisement and discovery) for all classes of Internet applications [Bus01].

One way to support B2B interoperability is to describe the semantics and structure of data and operations of services using XML and domain ontologies. Briefly stated, an ontology defines terms that can be used to describe entities (e.g., service properties, operations) of a specific domain (e.g., healthcare, finance, travel) and relationships among terms [Bou99]. Several industries (e.g., RossettaNet for Information Technology products) developed their common ontologies. In this approach, an organisation creates and publishes the XML documents that describe its offers, requirements, assumptions, and terms for doing business. Partners can interact with each other after inspecting, understanding, and using each other's definitions. The vision of this approach is to allow the use of services without prior agreement, and without help of external mediators. The establishment of a new relationship with existing partners does not require any additional work for this partner. This feature is essential to allow the dynamic formation of trading communities. The business process of the trading community is specified by the shared document definitions. The partners in the trading community are interconnected in terms of largely agreed upon documents. The business logic implementation at a partner side is invisible to other trading partners. Of course a complete integration solution along the line of using XML requires standardised meta-data tags for industry sectors (ontologies), mappings between different ontology descriptions, and means for processing XML documents and invoking the appropriate services (e.g., workflows, applications, and legacy systems) to handle requests.

It should be noted that several e-commerce platforms that rely on XML-based standards and protocols have emerged recently including IBM's WebSphere, WebMethods, Sun Open Network Environment – Sun ONE, and BEA Collaborate. The focus of these platforms is mainly on describing, advertising, and discovering services. Very little work has been done on service composition. For example, WebMethods and Sun ONE provide workflow-based support to static composition of services.

2.5 Discussion

Without loss of generality, interoperability in B2B e-commerce applications concerns three layers, namely, communication, content, and business process layers. Component-based frameworks focus more on interoperability at the communication layer in the context of tightly coupled services. They are mainly appropriate to integrate intra-enterprise services. EDI focuses more on point-point interoperability at communication and content layers in the context of loosely coupled services. It is appropriate to integrate inter-enterprise services with long-term and static trading relationships. XML-based frameworks vary in their support of interoperability at the different layers. In general, they focus on interoperability at communication and content layers in the context of loosely coupled Internet-based services. Few efforts such as RossettaNet and ebXML provide support for interoperability at the business process layer. Although, workflow-based systems focus on interoperability at the business process layer in the context of tightly coupled services, emerging efforts in this area aim at supporting all interoperability layers in the context of inter-enterprise services.

3 Elementary service-based interactions

In this section, we review some issues that arise when an organisation wishes to consume and provide services from/to other organisations. We first discuss the context under which these issues appear, and we then introduce two architectural patterns addressing specific aspects.

3.1 Context

In the setting of B2B e-services, the interaction between a service provider and a service consumer entails an interaction between the information systems of the underlying organisations. Being developed by separate teams, for different purposes, and perhaps at different times, these information systems are certainly heterogeneous both from the managerial and from the technological viewpoints. For similar reasons, given two services S1 and S2 provided by two distinct organisations, one can expect that everything from the business rules up to the document formats of S1, differ from those of S2. Hence, an organisation requiring an external e-service needs to make sure that its information system is capable of interoperating with that of the prospective providers, and more importantly, that this connection is loose enough so that alternative providers can be accommodated in the future. On the other hand, any e-service provider needs to make sure that its information system has a clearly defined interface to this e-service, and that the information systems of the consumers are properly interacting with this interface.

At a lower level, this issue of information systems interaction becomes that of application and workflow interoperation. Indeed, the consumption of a B2B e-service is by definition initiated by a business application (possibly acting within a workflow), and similarly, the processing of this request is performed by another application or workflow. These applications or workflows, which are located in different organisations, interact through messages containing business documents. Here, we are interested in the case where this interaction is carried out through an open and volatile network such as the Internet. Key questions that arise in this context are:

ﬄ How does a service consumer and a provider come to know each other?

ﬄ How does the provider can describe the interface to its service? How does the service consumer gets to understand this interface?

ﬄ How does the provider and the consumer adapt their systems to interact with each other?

ﬄ How is the security and consistency of the business exchanges ensured?

In the sequel, we present two patterns dealing with these issues.

3.2 The External Interactions Gateway Pattern

Problem Description An organisation needs to interact with others in order to consume and provide services. Each of the services provided by the organisation, as well as each external service consumed by the organisation, has its own interaction requirements (e.g., document formats, data model, domain ontologies, message sequencing), which can change over time following changes in business rules. The number of services consumed and provided is large and volatile. Some of the applications which provide and consume services within the organisation are implemented by legacy systems which cannot be modified in order to accommodate new interaction requirements. Even those applications that can undergo modifications, require a considerable amount of programming effort to adapt to new requirements.

Forces The specific issues that arise in this situation are:

ﬄ Assuming that the data model and format in which a business document is generated differs from that in which it is interpreted, how and when is the conversion between formats operated? How and by whom is this conversion implemented?

ﬄ Assuming that the applications use different interaction protocols (e.g., different message names, semantics and sequencing), how are these protocols aligned? How can the proper interaction between the applications be enforced?

ﬄ Given that the applications belong to different organisations, and that they are likely to exchange critical business information, how is the confidentiality, integrity and non-repudiation of these exchanges ensured?

Example. A French company "Traduit-Tout" provides translation services in several languages: English-French, French-English, Spanish-French and French-Spanish. The English-French and Spanish-French translations are entirely handled by a business process within the company. The French-English and French-Spanish translations are first treated internally, and once a draft of the translation is produced, it is sent for proofreading to partner companies in UK and Spain respectively. These partners are statically selected, but from times to times, a given partner may be replaced by another one.

The "Traduit-Tout" company therefore provides 4 services (the 4 kinds of translations), and consumes 2 services (the proofreading services from its partners in UK and Spain). Although statically selected, the partner companies may be replaced by others at certain points in time. Also, the partner companies change their business rules from times to times, and this may result in changes to the interface of the services that they provide (e.g., the list of accepted document formats is extended or the list of accepted messages and their inter-relationships is modified). Similarly, from times to times the company changes its own service interfaces, whether to enhance or to simplify them, or to cope with internal policy changes.

Solution The basic idea of the pattern is to separate a service offer from its underlying implementation. On the provider side, this means separating the implementation of a service (e.g., a standalone program, a software component, or a workflow), from the interface to this service. On the requestor side, a distinction is introduced between an abstract and concrete service requests. An abstract service request is a need for a capability at a particular point in time (e.g., the need to translate a document from French to English by next week), while a concrete service request is an invocation to a specific service offered by a given provider (e.g., a request for the F-E service offered by "Traduit-Tout").

The mapping from service interfaces to service implementations on the one hand, and from abstract requests to concrete requests on the other, is carried out by a dedicated software entity, subsequently refered to as the External Interactions Gateway (EIG).

Figure 1 summarises this approach in the context of the "Traduit-Tout" company. Any request for a service emanating from an application or workflow within "Traduit-Tout", goes through the EIG of this company. Symmetrically, every request for any of the four services provided by "Traduit-Tout" transits through the EIG.

The internal architecture of the EIG is depicted in figure 2. The idea of this architecture is to view the security, the document format heterogeneity,

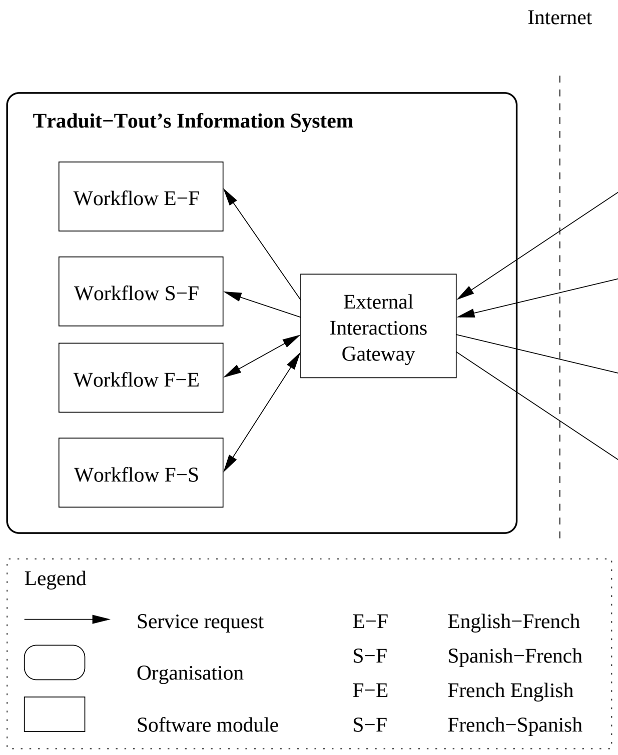

Figure 1: The EIG of the "Traduit-Tout" company.

and the the conversational protocol heterogeneity, as orthogonal aspects, and to handle these aspects in three separate layers, namely the security manager, the document format manager, and the conversational protocol manager.

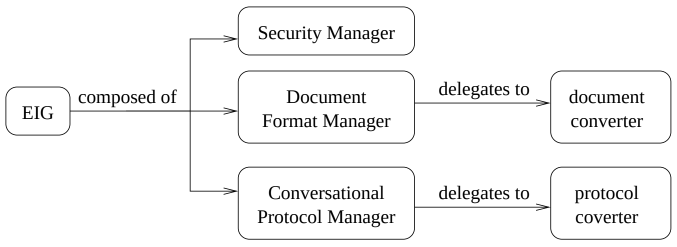

Figure 2: Internal architecture of an EIG.

The document format manager handles conversions of inbound and outbound messages. For example, if an internal application provides an interface based on xCBL [Com00], and this application needs to interact with an external service which expects documents in cXML [cXM], the conversion between these two formats is handled by the document format manager.

The conversational protocol manager acts both as a conversation controller and a conversation translator. As a controller, the conversational protocol manager processes inbound and outbound messages related to business conversations. For each message that it processes, the controller identifies the current stage of the conversation to which this message relates, and checks that the message's type is appropriate. If so, the controller forwards it either to the appropriate implementation of a service (if it is an inbound message), or to the appropriate business partner (if it is an outbound message). As a protocol converter on the other hand, the conversational protocol manager translates sequences of messages in a given protocol into equivalent sequences in a different, though compatible protocol. The need for this type of conversions arises when independent tasks or messages are carried out in different orders by different external services. For example, a given service offer may require a login procedure to be performed prior to any access to the service's functionalities, while another service offer with similar capabilities may provide access to some functionalities of the service without password verification, and only require a full login procedure for accessing other functionalities. This is the case of the auctioning services offered by Yahoo[1] and eBay[2]. In eBay, it is possible to consult detailed data about past and ongoing auctions without going through a login procedure; login is only required for placing a bid. On the other hand, in Yahoo, a login procedure is required before accessing the data about the auctions.

Last but not least, the security manager handles security issues. In particular, this module maintains a set of public and private keys, encrypts and decrypts inbound and outbound messages, and stores these messages in a log so as to ensure traceability and non-repudiation.

Internally, the document format manager is architectured as an aggregation of modules with identical interfaces, each specialised in a given type of conversion. For example, a document format manager may be composed of two converters, namely "xCBL to cXML" and "cXML to xCBL". Depending on the type of conversion required, the document manager decides which converter it should use. The number of converters attached to the document manager is variable, thereby catering for extensibility.

In theory, the conversational protocol manager can also be architecture as a composition of a set of controllers and converters. However, to the best of our knowledge, this idea has not yet been explored in details.

Implementation aspects In the following, we present techniques for handling document format and conversational protocol heterogeneity: two of the three aspects addressed by the EIG. The issue of security is not discussed here as it can be handled using established cryptography techniques, secure communication protocols (e.g., SSL), and message logging.

The issue of handling document format heterogeneity at the syntactical level is more or less well addressed by existing technologies such as XML and RDF parsers and generators. It is important to note however, that these technologies do not address the issue of semantic integration, which is still an open area of research.

In the context of XML-based business document standards for exam-

[1]http://www.yahoo.com

[2]http://www.ebay.com.

ple, the XSLT language provides a means for expressing transformations from documents abiding to a given standard, into documents abiding to another standard. The specification of these transformations in XSLT is however quite cumbersome, especially when the granularity and ordering of the document elements in the source standard differ from those in the target standard. Several approaches can be envisaged to cope with this difficulty. For instance, Microsoft's BizTalk 2000 (see section 2.4) uses XSLT as the underlying transformations language, but provides a graphical tool on top of it (namely BizMapper). However, whilst BizMapper hides the details of the XML syntax, it does not reduce the complexity of the mappings that need to be specified. An alternative approach based on separation of concerns is proposed by [OF01]. The authors suggest to introduce a distinction between the syntax and the data model of a standard. The syntax of a document standard is specified as an XML DTD or an XML schema. The data model is specified in the RDF Schema Language. The transformation of a document XD in a given XML standard S, into a document XD' in another standard S' is carried out in three steps, each of which involves a set of XSLT rules:

ﬄ Abstraction: translate XD into an RDF document RD abiding to the data model of S.

ﬄ Conversion: Translate RD into another RDF document RD' abiding to the data model of S'.

ﬄ Refinement: Translate RD' into an XML document XD' abiding to the syntax of S'.

[OF01] shows in the context of four existing XML business document standards, that the transformations involved by these three individual steps are simpler to build and maintain, than a direct transformation from XD into XD'. In particular, the development of a translator from xCBL to cXML using this approach is sketched. The authors also point out that the transformations involved by the abstraction and the refinement steps are reusable.

In contrast to document format heterogeneity, the issue of handling conversational protocol heterogeneity is still widely open. Indeed, while B2B standards defining conversational protocols have recently emerged (e.g., RosettaNet's PIPs [Ros]), the issue of mapping a conversation in a given protocol into an "equivalent" conversation in another protocol is an open problem.

[KLKD01] advocates the use of the Web-Service Conversation Language (WSCL) for specifying conversational protocols. The authors show that these specifications can be used to automatically build conversation controllers, which is one of the two components of the conversational protocol manager.

[SCDS01] describes an approach to extend existing workflow technology in order to handle both document format and conversational protocol heterogeneity in B2B interactions. Specifically, given a structured description of a B2B protocol standard (e.g., a description of a RosettaNet PIP in XMI), a process template is generated which encodes the sequencing of activities that is required in order to handle a conversation in that standard. This process template can then be refined by a developer into a full workflow implementing a given business process. At runtime, this workflow interacts with external service providers through a conversations manager (the equivalent of the EIG), which handles the conversion of internal workflow variables into external documents. The conversations manager also processes inbound documents and it either forwards them to the appropriate running workflow instances, or it initiates new workflow instances for treating them.

Known implementations and related patterns. Several architectures for the EIG have been reported under different names: the Proxy Gateway of [GAHL00], the Semantic Integration Engine of [Bus01] and the Trade Partners Conversations Manager of [SCDS01].

The EIG specialises the Gateway Pattern [BMR$^+$96] by explicitely addressing the issues of document format and conversational protocol heterogeneity. The EIG can also be seen as a combination of the Facade pattern with the Proxy pattern [GHJV95]. On the one hand, the EIG acts as a proxy which handles calls to remote servers on behalf on an application. On the other hand, the EIG offers a unified entry point to a set of services offered by an organisation. Notice that the scope of the EIG is far more specialised than those of the above two patterns. In particular, the EIG addresses the issues of security management, conversational protocol heterogeneity, and document format heterogeneity.

3.3 The Contract-Based Outsourcing Pattern

Problem Description Outsourcing is the process whereby a provider of a service delegates all or part of the provisioning of an instance of a service to other providers. Outsourcing involves the following participants: a requestor, a delegator, and one or several delegatees. From the viewpoint of the delegator, outsourcing can be seen as one or several outbound service requests, caused by an inbound service request. From a global point of view, outsourcing can be seen as a simple form of inter-organisational service composition. Indeed, the delegator and the delegatees can be seen as forming an alliance for provisioning a global service.

One of the main problems that arises during outsourcing, is that the delegator is responsible vis-a-vis of the original service requestor for the provisioning of the overall service. If one of the delegatees fails during the delivery of its service, the delegator must undertake some kind of compensation action, or at least, it must notify this failure to the original requestor. For this reason, it is essential that the rights and responsibilities of each party are well defined before the outsourcing takes place. Since the participating organisations are independent from each other and have different objectives, the definition of these rights and responsibilities should be formalised as one or several contracts, possibly obtained through bilateral or multilateral negotiations.

This pattern deals with the issue of negotiating and enacting contracts for service provisioning, focusing on the case where part of the service is outsourced. The pattern is however general enough to cope with the case where no outsourcing takes place.

Forces and example Given that contracts need to be negotiated and enacted on-the-fly, the following questions arise: (i) how can a contract be established (i.e., negotiated) without or with minimal human intervention? (ii) how can the fulfillment of a contract be automatically enforced?

For example, the translation office “Traduit-Tout” (see previous pattern) receives orders with various requirements. Each of these orders has a set of terms attached to it, which specify parameters such as the deadline for delivering the instance of the service triggered by this order, the pricing, the modalities for billing and payment, several kinds of penalties and guarantees, etc. The terms of small orders are set by pre-established contracts, while the terms of bigger orders are generally negotiated on a per case basis.

In order to ensure the fulfillment of the obligations set by the contracts that it passes, “Traduit-Tout” requires its partners (i.e., the proofreading agencies) to abide by some terms. Two cases arise here: either these terms are set by pre-established contracts, or the terms are negotiated on a one-by-one basis. The former case arises during the processing of small orders, while the latter case is typical of important orders. In fact, when negotiating an important order with a client, “Traduit-Tout” may need to negotiate at the same time with the proofreading agencies, in order to make sure that it can meet the terms that the client is requesting. Given that important orders often have particularly tight deadline, it is highly desirable that the negotiation process is carried in the shortest possible time.

Solution The relationship between the requestor, the delegator, and the delegatee(s) are formalised through bilateral contracts. In other words, there is one contract linking the requestor with the delegator, and one contract linking the delegator with each delegatee.

A contract is esentially a planned set of actions and interactions that need to be undertaken during the delivery of a service. Contracts are specified in an executable language (whether rule-based or procedural) and their fulfillment is monitored by a contract enactment module. The contract enactment module analyses the sequences of messages exchanged between the business partners during the execution of a service, and detects any deviations from the planned course of actions. The contract enactment module can be hosted by the service requestor and providers themselves, or by a neutral third party.

In order to avoid building contracts from scratch, contracts for a given service are abstracted into contract templates. In a nutshell, a contract template is a function which generates a contract given a set of contract parameter values, which capture the variable part of a class of contracts.

The relationship between these parameters can be fixed or negotiable. For example, the "Traduit-Tout" company may establish that the size of a document and the price of its translation are fixed by a formula (e.g., 10 cents per word), or it may leave this relationship open to negotiation.

Contract templates are included in the advertisement of a service offer. These advertisements are stored either in a local catalog hosted by a service provider, or in a common catalog (also known as a service marketplace). An individual or an organisation requiring a given service, sends a query to one or several catalogs and selects a set of service offers which satisfy its requirements. If the contract template(s) of the selected offer(s) allow so, the requestor undertakes one or several negotiations with the providers referenced in the selected offer(s). Eventually, a contract emanates from this process, and this contract is fed into a contract enactment module.

The negotiation between the potential consumer and the provider can be manual, semi-automated, or fully automated. In general, the degree to which a negotiation can be automated depends on the nature of the negotiable parameters. If the number of parameters is fixed and their values are numerical, a high degree of automation can be attained. If on the other hand new parameters are allowed to be introduced during the course of a negotiation, or if the domain of the parameters are not known in advance, then the negotiation must involve human actors. Still, even if human actors carry out the actual negotiation, their interactions and their decision-making can be facilitated by software tools.

As a concrete example, when "Traduit-Tout" needs to outsource a service such as proofreading a document, it queries a number of service catalog(s) and retrieves the offers, together with their contract templates, which have the potential to satisfy its requirements (e.g., in terms of pricing and delivery times). After selecting one of these offers, the accompanying contract is instantiated by providing a set of parameters such as the size of the document to be proofread, the degree of specialisation of the document, and the time frame. In the case of special requirements (e.g., tight deadlines), "Traduit-Tout" may need to negotiate with one or several proofreading agencies instead of just selecting and instantiating a given contract template. For example, in order to negotiate a special contract with a client for translating a large document in a very short time frame, the "Traduit-Tout" company may need to know if any of its proofreading partners would be willing to proofread the resulting translation in an extraordinarily short delay. This example points out the fact that the negotiation between the original requestor and the delegator may take place at the same time as the negotiation between the delegator and the delegatee.

Implementation aspects In the general case, the effective representation and the automated enforcement of business contracts for service outsourcing remains an open question. Currently, e-business enabling products such as Microsoft's BizTalk and Extricity's Alliance (see section 2), do not address in any way pricing and payment, legal issues, agreements regarding arbitration, or recovery from partner failures.

Known implementations Partial implementations of this pattern have been developed in the CrossFlow, MEMO, and ADEPT projects (see section 2).

In CrossFlow [The01, GAHL00], contracts are statically specified by the service providers and advertised in a service marketplace. A potential service requestor (whether a terminal requestor or an organisation wishing to delegate part of its responsibilities) selects offers from the marketplace and instantiates their accompanying contract templates (there is no support for negotiation). Once the contract is instantiated, it is enacted by a set of cooperating modules deployed in the participating organisations. Contracts are not dynamically negotiated in CrossFlow.

The MEMO project [QS00] advocates the idea that the service marketplace, called the Electronic Commerce Broker Service (ECBS) by the authors, should provide support for structured and mediated negotiations between humans acting on behalf of the consumer and the provider organisations. Specifically, negotiations are conducted through the exchange of standardised messages whose semantics is based on speech-act theory. These messages transit through the ECBS, thereby allowing their storage for future references. Alternatively, if the trust and confidentiality offered by the ECBS is not a requirement, the negotiation can be conducted without any intermediation. Still, the structure, sequencing and semantics of the messages exchanged during the negotiation is fixed.

The ADEPT platform [JNF+00, JFN+00] goes a step further towards the automation of negotiations for service provisioning, by proposing an agent-based approach to this problem. Specifically, agents in ADEPT negotiate on behalf of their organisations using a one-to-many negotiation framework based on multi-attribute utility theory. An agent acting on behalf of a consumer organisation simultaneously negotiates with several other agents representing potential service providers. Each agent tries to find a deal which maximises its own utility function, which encodes the preferences and business constraints of the organisation that it represents. For space reasons, we do not enter into details about this negotiation framework. The reader is referred to [SFJ97].

4 Service composition

In this section, we discuss about how a composite service can be built, whether statically or dynamically. We start with static composition, in which the "plan" of the composite service is known a priori (i.e. alliances are fixed). We then move into dynamic service composition, which relies on service discovery techniques.

4.1 Context

The fast and dynamic integration of business processes is an essential requirement for organizations to adapt their business practices to the dynamic nature of the Web. Business partners may need to form permanent (long term) or temporary (short term) relationships. In the former type of relationship, components are known in advance and alliances are statically defined. The latter form of partnership does not assume an *a priori* trading relationship among partners. An e-service would in this case need to dynamically discover partners to team up with to execute the required transactions. Thus, this type of dynamic integration (also called on-the-fly integration) requires support for automated partner discover and fast e-service integration.

4.2 Service Composition Pattern

Problem We distinguish between *elementary* and *composite* services. Elementary services are pre-existing services, whose instances execution is entirely under the responsibility of the previously discussed pattern. As an example, we consider an organisation who wants to offer an on-line travel planner called "Travel Solutions". This planner aims to provide users build their own itinerary in a given city. To do so, the organisation requires to integrate the following independant services:

ffl Flight booking: to search for a flight and when the more suitable flight is found to proceed the booking and the paiement. This service can be assigned to an individual provider (e.g. an airline company).

ffl Accommodation booking: to search for different styles of accomodation (hotels, hostels, bed & breakfast, camping, etc). This service is assigned to a community of providers. This community federates entities such as public central booking and private booking sites. When an execution request is addressed to the community, its representative forwards it to one of its members.

ffl Tourist attractions searching: this service gives information about the mains tourists attractions (schedules, venues, etc.).

ffl Bicycle hire and car rental booking: the idea is to give the user the choice to ride a bike or to drive a car. This choice is based upon the distance from the booked accomodation to the major tourist attractions. Both of these services are assigned to an individual provider.

ffl Event attendance planner which is decomposed into two others: events searching and tickets purchasing (if ticket pre-purchasing is required for the selected events).

To offer a value-added composite service, organisations face the problem of identifying the characteristics of the services that need to be composed and the nature of their interactions.

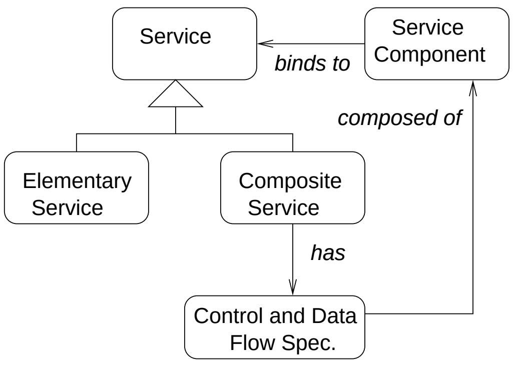

Figure 3: Basic elements of service composition

Forces. In summary, the following are important characteristics of static composition:

ffl How do you describe interactions amongst services without referring to any implementation or execution model ? A high-level approach that allows fast integration and easy maintenance is needed.

ffl How do you support arbitrary nesting of composite services ?

ffl How do you maintain a high level specification of a composite service while ensuring its executability ?

Solution. The solution defines composite services recursively as an aggregation between other composite services and elementary services, which are referred to as component services (see figure 4.2). This aggregation specification must include descriptions about these two aspects:

ffl control-flow: establishes the order in which the component services should be invoked, the timing constraints, the signals that may interrupt or cancel their execution, etc.

ffl data-flow: captures the flow of data between component services

Implementation Aspects A natural way for describing the control-flow of composite services, is to adapt to this purpose an existing process-modeling language, and especially one of those which have proven to be suitable for workflow specification. In a nutshell, a workflow consist of a set of activities with explicitly specified control and data flow between activities. An activity may invoke a transaction or some specific application (in our context a service). There are numerous workflow specification languages based upon different paradigms. In fact, each commercial Workflow Management System (WfMS) implements its own specification language, with little effort being done to provide some degree of uniformity between products. In this respect, the Workflow Management Coalition (WfMC) [Coa96] has defined a set of glossaries and notations that encompass many of the concepts and constructs provided by existing workflow specification languages. Unfortunately, these efforts have had a very limited impact. To add to the lack of uniformity, most of the existing workflow specification languages, including the one defined by the WfMC, lack a formal semantics, making it difficult to compare their capabilities and expressiveness in order to make an objective choice between them [ABtHK00].

The use of formal notations for workflow specification has been considered in e.g. [Aal98] and [MWW$^+$98]. [Aal98] discusses several advantages of using Petri-nets for describing the control-flow perspective of workflows, among which their expressive power. However, many designers find the Petri-net formalism difficult to grasp. Moreover, Petri-nets do not provide any means for structuring a specification into recursive compositions. As a tradeoff between expressiveness on the one hand, and ease of use and modularity on the other, [MWW$^+$98] advocates the use of statecharts instead [HN96]. The main argument is that statecharts are based upon finite automata and Event-Condition-Action (ECA) rules, two paradigms which are easy to comprehend. Finally, the statechart formalism has been integrated into the Unified Modeling Language (UML) [RJB99], as the foundation of many intra and inter-object process modeling constructs.

A statechart is made up of states and transitions. Transitions are labeled by ECA rules. The occurrence of an event fires a transition if (i) the machine is in the source state of the transition, (ii) the type of the event occurrence matches the event description attached to the transition, and (iii) the condition of the transition holds. When a transition fires, its action part is executed and its target state is entered. An event occurrence can be the reception of a signal, or a change in the system's clock (i.e. timeouts). The event, condition, and action parts of a transition are all optional. A transition without an event is said to be triggerless. The outgoing transitions of a state are exclusive, that is, for a given event only one of them may fire. Hence, if there are two or more transitions with the same source state and the same event specification, their conditions are disjoint. For example, the statechart depicted in Figure 4 specifies the composite service "Travel Solutions" described earlier.

As stated out before, the objective of cross-organisational workflows (see Section 2.2) is to automate business processes that interconnect and manage communication among disparate systems. In this approach, the description of the composite service can be defined collaboratively among partners. However, the enactment of a composite may be either be centralised or distributed across the participant partners.

Component-based frameworks (see Section 2.1) provide for the connection and coordination of data and operations among services. The description of a composite service is worked out and agreed to offline. After that, the global description of a composite service is generally spread through the implementation code of every component. Thus the composition of services in this approach is mainly ad-hoc.

Document-based approaches such EDI (see Section 2.3) and XML-based

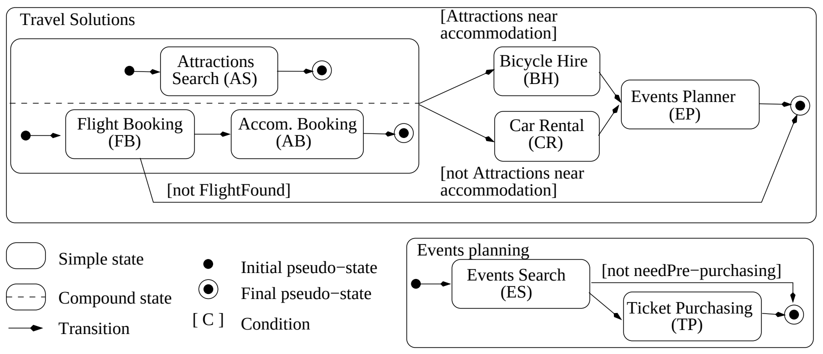

Figure 4: Example of a control flow specification using statecharts

frameworks (see Section 2.4) the interactions among the components of a composite service is specified by the shared document definitions. The components are interconnected in terms of agreed upon documents. The business logic implementation at a partner side is invisible to other trading partners. Interactions between components (partner services) may be carried out according to a specific B2B standard (e.g., EDI, OBI, RossettaNet, cXML) or bilateral agreements. B2B standards define formats and semantics of messages (e.g., request for quote, purchase order), bindings to communication protocols (e.g., HTTP, FTP), business process conversations (e.g., sequencing), security mechanisms (e.g., encryption, non-repudiation), etc.

Known implementations In the following, we briefly illustrate the composition of services in the context of the following e-service platforms: The Collaboration Management Infrastructure (CMI) [SGCB00], the eFLOW project [CIJ$^+$00], the Web Base of Internet Accessible e-Services (WebBIS) [BMB$^+$00], and SELF-SERV [FDBP01, BDSN02]. CMI is a platform for modeling and managing inter-enterprise business processes. A service is modeled by state machine that specifies that possible states of a service and their transitions. Transitions are caused by service operation (also called service activity) invocations or internal service transitions.

EFlow is a platform for the specification, enactment, and management of composite e-services. A composite service is modeled by a graph, which defines the order of execution among the nodes in the process and may include service, decision, and event nodes. Service nodes represent the invocation of a basic or composite service, decision nodes specify the alternatives and rules controlling the execution flow, while event nodes enable service processes to send and receive several types of events. WebBIS is a platform for modeling, managing, and evolving e-services. WebBIS adopts an ECA-rule (Event Condition Action) approach for defining composite e-services. ECA rules are used to specify interactions between a composite service and its components. Encoding the business logic of services as ECA rules is especially attractive to support the customization and increase in the flexibility of composite services. Indeed, rules can be added, modified, or removed to reflect changes in both operational (e.g., server load) and market environments (e.g., user requirements).

In SELF-SERV, a subset of statecharts has been adopted to express the control-flow perspective of composite services. In this approach, states can be simple or compound: a simple state corresponds to the execution of a service, whether elementary or composite. Accordingly, each simple state is labeled by a description of a service offer, and the set of parameters that are to be passed to this service upon instantiation. When a basic state is entered, the service that labels it is invoked. The state is normally exited through one of its triggerless transitions, when the execution of the service is completed. If the state has outgoing transitions labeled with events, an occurrence of one of these events provokes the state to be exited, even if the corresponding service execution is ongoing (i.e. this execution is cancelled). In SELF-SERV, the data-exchange perspective is implicitly handled by variables: parameters of services (inputs and outputs) and events (consumed and produced by services).

4.3 Service Discovery Pattern

Problem The problem relates to Web-based service integration in large, autonomous, heterogeneous, and dynamic environments. For B2B e-commerce to scale to the Internet, there is a need for automated integration with all relevant partners, established a priori or on demand. For example, a conference trip business process may need to be composed from several tasks including hotel booking and flight reservation. In an ad-hoc composition approach, a human must select the services that can execute the tasks hotel booking and flight reservation. After that, the services may be invoked independently and the results are manually combined. It is also possible to create a custom-code to compose the services.

On-the-fly B2B integration and interoperation is better supported by establishing online marketplaces for e-services. These marketplaces should provide capabilities for brokering and dynamic collaboration with participants of the marketplaces. Instead of statically binding e-services to each other, marketplaces should dynamically discover new e-services with the right set of features and bind them at run time. Selecting a partner should consider the available e-services, characteristics, organizational policies and resources that are needed to accomplish the integrated e-service.

Forces In summary, the following are important characteristics of dynamic composition:

ffl What type of information is needed to identify service components at run-time ?

ffl Given a high level specification of the composite service (see previous pattern), how to you integrate these component services in the provision of this composite service ?

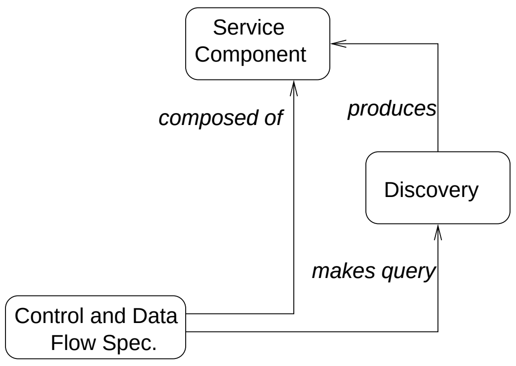

Figure 5: Basic elements of service discovery

Solution The pattern for dynamic composition consists of the following elements:

ﬃ An automated service discovery facility: given that component services are described using software-interpretable information (i.e, using meta-data or ontology languages), this facility provides means (e.g., service discovery engine) to locate component services based on constraints over their meta-data.

ﬃ A composite service specification allows component services to be automatically discovered and integrated.

Implementation Aspects A number of industry efforts to define standards that provide common building blocks for e-service discovery and integration emerged recently including UDDI (Universal Description, Discovery, and Integration), WSDL (Web Services Description Language), and ebXML [Bus01]. UDDI provides an XML-based registry for advertising businesses and services. The advertisement and discovery of services and businesses in UDDI exploit keywords categorisation. The registry organises advertisements as white and yellow pages. An advertisement of a business includes name, key information, categorisation, and offered services. An advertisement of a service includes name, key information, categorisation, and multiple bindings (i.e, technical information that is relevant to access the service). WSDL is an XML-based languages for describing the content and capabilities (i.e, messages and operations) of e-services. Services can be defined using abstract terms. Bindings between abstract descriptions and concrete implementations (e.g, specific data formats and protocols) can also be defined. ebXML has similar features with UDDI. However, it focuses on business processes from a workflow perspective. Currently, these standards focus on service discovery and advertisement. They do not provide service composition capabilities. They essentially aim at providing common building blocks among all e-service platforms.

It should be noted that several agent-inspired efforts that aim at providing supports for discovering and composing e-services are underway. In the recent years, several markup languages (e.g., DAML+OIL, DAML-L) have been developed with the purpose of realising the Semantic Web Vision including markup for Web services discovery and composition [Web].

Known implementations In the following, we briefly illustrate the dynamic discovery and composition of services in the context of the following e-service platforms: The *Collaboration Management Infrastructure* (CMI) [SGCB00], the *EFlow project* [CIJ+00], and the *Web Base of Internet Accessible Services* (WebBIS) [BMB+00].

CMI's service definition model features the concept of a *placeholder activity* to cater for dynamic composition of services. A placeholder activity is an abstract activity replaced at runtime with a concrete activity type, that must have the same input and output data of those defined as part of the placeholder. A selection policy is specified to indicate the activity that should be executed in place of the placeholder. For example, a generic reference service can be configured to use available activity implementations by providing a selection policy that plugs in the implementation for each activity interface. If multiple providers offer implementations for an activity interface, the selection policy may use a broker to choose the implementation that offers the best quality of service.

In EFlow, the definition of a service node contains a *search recipe*. A search recipe is a query represented in a query language. When a service node is invoked, a search recipe is executed in order to select a reference to a specific service. After that, operations of the selected service can be invoked.

WebBIS proposes a concept called push-community as a solution to the problem of integrating dynamic e-services. A push-community describes the capabilities of a desired service without referring to any specific service. In this sense a community defines a request for service. In order to be accessible through a push-community, services can register with it to (fully or partially) offer the desired operations. This involves the definition of the *mappings* between operations defined in the community and those defined in the actual services. An actual service can register with one or several push-communities of interest. It can also leave these communities at any time. In this approach, the task of composing a complex service is gracefully distributed by enabling each provider to contribute to one or more communities without knowledge of the details of other participating services. The means by which a community chooses a member to execute an operation is specified via a *selection policy*. A selection policy is specified using ECA rules. A selection policy can be based on 1-N negotiation protocol (e.g., an auction), or any ranking algorithm involving parameters such as customer's profile, service's reliability, etc.

5 Composite Service Execution

In this section, we discuss the execution of a composite service assuming that its control and data semantics are already defined.

5.1 Context

Let us consider again the example of the "Travel Solutions" composite service, whose control flow semantics have been described in figure 4.. We assume that this service is hosted and executed by a given provider. When it is invoked by a user, its execution involves the activation of all its component services which are potentially hosted on remote providers.

For instance, a "Travel Solutions" service request triggers the execution of both services "Flight Booking" and "Attractions Search". When "Flight Booking" completes its execution, the service "Accomodation Booking" should be executed next. When both "Accomodation Booking" and "Attractions Search" services are completed, a check is made on whether the selected accomodation is either close or far from the main tourist attractions. The service "Bicycle Hire Booking" is triggered in the former case and the service "Car Rental Booking" is triggered in the latter case.

The whole execution of the "Travel Solutions" service is carried out in a distributed manner. The issue addressed here is to determine a suitable execution model for a composite service. There are two possibilities (leading to two different patterns):

ﬄ the components of a composite service are coordinated by a central scheduler executed by a single provider which hosts the composite service.

ﬄ the entities participating in a composite service coordinate the execution through peer-to-peer communication. The execution of the composite service is carried out independently from the provider by which it is hosted.

5.2 Service Execution with Central Authority Pattern

Problem Description. Assuming that the composite service control semantics are already determined (see section 4.2), it is possible to derive an execution order of the component services from any of the well known notations (e.g. workflows, Petri nets and statecharts).

In this context, the following questions arise:

ﬄ When should a component service be invoked?

ﬄ What should be done after a component service is invoked?

ﬄ When should its execution be completed?

ﬄ What should be done after its execution is completed?

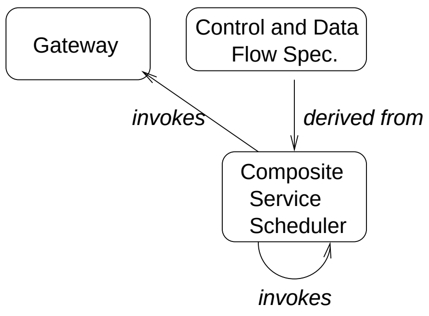

Figure 6: Basic elements of service execution with central authority

Forces. The key question which arises in this context is mainly: how to ensure that a service execution (whether composite or native) is carried out faithfully to the service specification?

Solution. Assuming a composite service S, there is one dedicated provider for this service. This provider should host a Composite Service Scheduler (scheduler in short) which could be partially or totally derived from the semantics of the service (see figure 5.2). This scheduler is responsible for:

ffl Initiating the execution of the components of S according to the control-flow associated with S. To do so, S's scheduler invokes each of S's components (or their gateway if they are native services) in the order and under the conditions specified in the control-flow.

ffl While the service S is available, the scheduler receives and processes service requests.

ffl The scheduler is also responsible for handling and processing data according to the data semantics of the composite service.

In other words, the scheduler is typically a software module which determines when should one of its components be initiated, and what needs to be done when an external event received while the service is available. The coordination model behind this approach is depicted in figure 5.2, which shows the scheduler associated to the service "Travel Solutions" linked to several other services (the service's components). As it is shown for "Events Planner" service (linked to both "Events Search" and "Ticket Purchasing" services) the above process may be carried out recursively thereby leading to a tree-structure.

Implementation Aspects An executable semantics of statecharts has been described in [HN96]. The same method could be applied to partially or totally generate a scheduler responsible for carrying out the overall execution of a composite services which control-flow is described as a statechart. It could also be applied to some extent when the control flow is expressed via a Petri net. In the latter case, the composite service recursive definition

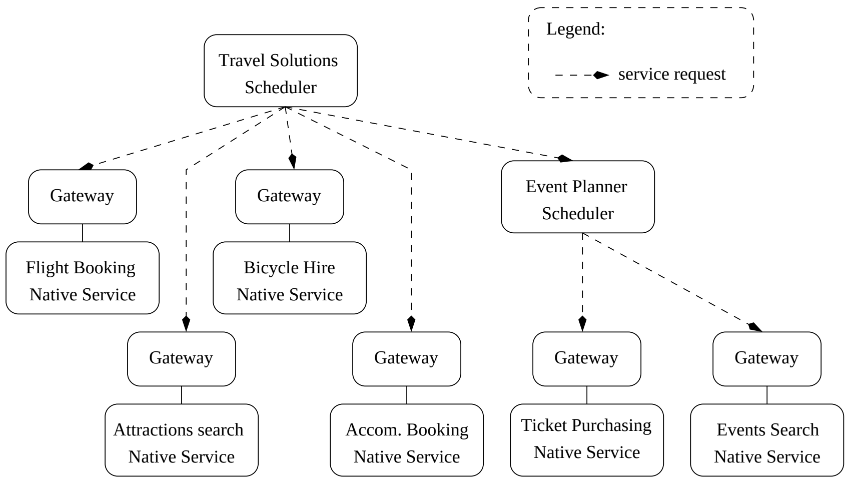
Legend:
service request
Travel Solutions Scheduler
Gateway
Flight Booking Native Service
Gateway
Bicycle Hire Native Service
Event Planner Scheduler
Gateway
Attractions search Native Service
Gateway
Accom. Booking Native Service
Gateway
Ticket Purchasing Native Service
Gateway
Events Search Native Service

Figure 7: Example of a centralised execution of the "Travel Solutions" service

is not supported. Workflows are generally based upon a centralised engine which is responsible for scheduling the execution of the tasks composing the workflow.

On the other hand, distribution and communication can be handled by most of the component-based frameworks (see section 2). Finally, XML can be used as the format for exchanging data (services' inputs and outputs).

Known Implementations. ADEPT [JNF$^+$00] is a multi-agent system designed to support inter-organisational workflows. In ADEPT, a workflow can be recursively decomposed into sub-workflows, leading to a tree structure. Each sub-workflow in ADEPT, is assigned to an autonomous agent. When the agent responsible for a workflow, wishes to invoke a sub-workflow, it has to negotiate with the agent(s) that provide(s) it.

EFlow [CIJ$^+$00] is a platform for specifying, enacting, and monitoring composite services. The execution model is based on a centralised process engine, which is responsible for scheduling, dispatching, and controlling the execution of the composite services. But, Eflow does not support recursive definition of composite services (E.g. there is only one composition level). The same remarks apply to the Collaboration Management Interface (CMI) [SGCB00].

5.3 Peer-To-Peer Execution Pattern

This pattern specialises the pattern "service execution with central authority" by addressing the issue of efficiency.

Problem. In the pattern for "service execution with central authority" execution of a composite service is dependent on a central scheduler. In such situation, the scheduler associated to a composite service communicates with

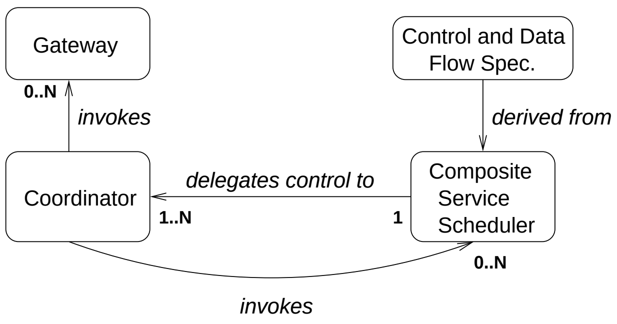

Figure 8: Basic elements of peer-to-peer service execution

each of the service's components: a first communication is established to trigger the execution of a component and another one to notify the scheduler that the component has been completed.

Forces. To avoid this potential bottleneck the question we want to address here is: how to limit as much as possible the number of messages exchanged during the execution of a composite service ?

Solution. We introduce a peer-to-peer execution model, as a solution to the recurrent problem of inter-service coordination. This means that the responsibility of coordinating the execution a composite service is distributed across the providers which host the components of the composite service.

The execution of a composite service is not only dependent on a central scheduler as in the previous pattern but rather on software components (called coordinators) hosted by each of the providers participating in a service composition (see figure 5.3). These software components interact in a peer-to-peer fashion in order to ensure that each instance of a composite service is executed in accordance with its control-flow and its data-flow specifications.

Therefore, the provider of a component service ST should host a coordinator responsible for:

ffl Initiating the execution of the service ST whenever all the preconditions are met.

ffl Notifying the completion of this execution to the coordinators of the services which potentially need to be executed next.

ffl While service ST is active, receive notifications of external events, determine if ST should be exited because of these event occurrences, and if so, interrupt the service execution if it is ongoing, and notify the interruption to the coordinators of the services which potentially need to be executed next.

All communications with native services are carried out through the gateways of these services.

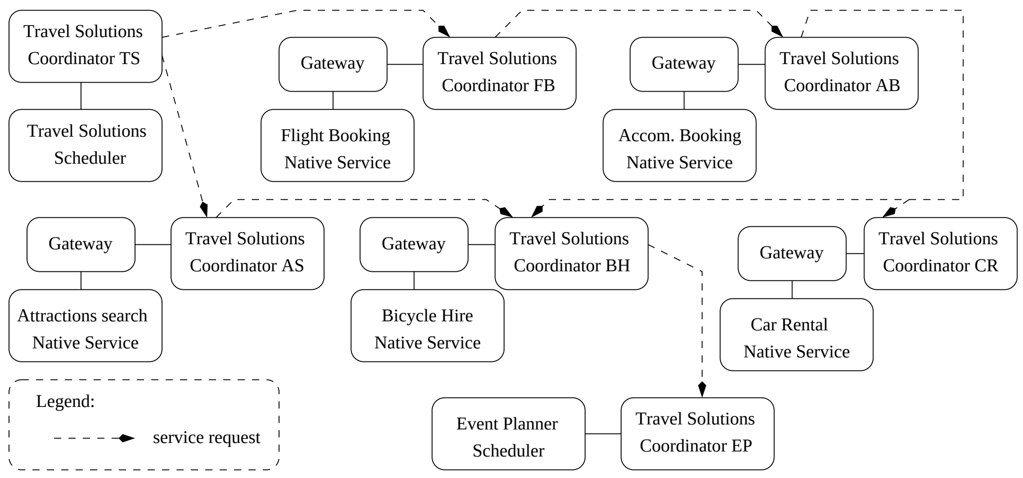
Travel Solutions Coordinator TS
Travel Solutions Scheduler
Gateway
Flight Booking Native Service
Travel Solutions Coordinator FB
Gateway
Accom. Booking Native Service
Travel Solutions Coordinator AB
Gateway
Attractions search Native Service
Travel Solutions Coordinator AS
Gateway
Bicycle Hire Native Service
Travel Solutions Coordinator BH
Gateway
Car Rental Native Service
Travel Solutions Coordinator CR
Legend:
service request
Event Planner Scheduler
Travel Solutions Coordinator EP

Figure 9: Example of a distributed execution of the "Travel Solutions" service

Example. An example illustrating the entities participating in the "Travel Solutions" composite service execution is depicted in figure 5.3.

When the coordinator of the "Travel Solutions" service receives an execution request, it sends a message to the coordinators of the services FB and AS. Upon receiving these messages, these coordinators invoke their associated services. When the service that books a flight completes its execution its coordinator sends a message to that of the service AB. This latter invokes the service that books an accommodation, waits for its completion, and sends a message to the coordinators of the services CR and BH. In the meanwhile, the coordinator of AS sends its completion message to the coordinators of CR and BH too. These completion messages contain the data that must be exchanged between these services, as per the data exchange perspective of the "Travel Solutions" specification. Using these data, the coordinators of BH and CR evaluate the condition "attractions near accommodation" appearing in the labels of their incoming transitions, and accordingly, they decide which service has to be executed next. Assuming that the attractions are far from the accommodation, it is the service CR that has to be executed. Once this service completes its execution, its coordinator sends a message to the coordinator of the service EP. Assuming that the notation adopted to express the control flow support recursive definition, the coordinator of the service EP sends an execution request to the coordinator of the composite service responsible for searching events. The execution of the service EP is carried out in the same way. Upon completion, its coordinator sends a message to the coordinator of the "Travel Solutions" service, thereby concluding the overall execution.

Implementation Aspects. Building the coordinators from a specification of a composite service as either a statechart or a Petri nets involves answering the following questions:

ﬃ What are the preconditions for triggering the execution of a service?

ﬃ When the service execution is completed, what are the services that may potentially need to be executed next?

The behaviour of a coordinator could therefore be captured using two sets: one set of preconditions such that the service execution is triggered when one of these preconditions is met, and another set of postprocessing actions that indicate which coordinators need to be notified about the fact that a service is being completed.

All communication and data exchange issues could be handled in the same way than for the previous pattern (E.g. component-based and XML-based frameworks).

Known Implementations. In Selfserv [BDFP01, FBDP01, FDBP01] the execution model for composite services, in which the providers of the services participating in a composition, collaborate in a peer-to-peer fashion in order to ensure that the control-flow dependencies expressed by the schema of the composite service are respected. Specifically, the responsibility of coordinating the providers participating in a composite service execution, is distributed across several lightweight software components hosted by the providers themselves (as depicted in Figure 5.3).

CPM [CH01] (Collaborative Process Manager) supports the execution of inter-organisational business processes through peer-to-peer collaboration between a set of workflow engines, each representing a player in the overall process. An engine representing a player P, schedules, dispatches and controls, the tasks of the sub-process that P is responsible for.

In Mentor [MWW$^+$98], although the scope of this latter proposal is that of intra-organisational workflows, the problem addressed is that of distributing the execution of workflows expressed as state and activity charts. The idea is to partition the overall workflow specification into several sub-workflows, each encompassing all the activities that are to be executed by a given entity within an organisation (assuming that this information is statically known). Each of these sub-workflows is itself specified as a statechart. The authors present some optimization techniques that reduce the number and the size of the messages exchanged by the sub-workflows, leading to a "weak synchronisation" model.

6 Conclusion

This report has discussed a number of patterns for the definition and implementation of composite services. These patterns suggest a methodology for building a new service that tackles each of these important issues separately:

ﬃ Identify native services and make them elementary services through a gateway interface (External Interactions Gateway Pattern) ffl Specify the control and data flow semantics of the new service based on these elementary services or other composite services, called component services (Service Composition Pattern).

ffl The new service can rely on component services identified at run-time (Service Discovery Pattern).

ffl The binding of component services to service providers must clearly define and handle the commitments and responsibilities of all parties involved (Contract-Based Outsourcing Pattern).

ffl The execution of the service must take into account performance and efficiency issues (Service Execution Patterns).

This methodology is a step towards providing a high level abstraction in the design, construction and maintenance of composite services. A number of software tools are expected to support this methodology from verification of requirements to automatic code generation.

References

[Aal98] W.M.P. van der Aalst. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[ABtHK00] W.M.P. van der Aalst, A.P. Barros, A.H.M. ter Hofstede, and B. Kiepuszewski. Advanced workflow patterns. In *Proc. of the 5th IFCIS Int. Conference on Cooperative Information Systems*, Eilat, Israel, September 2000. Springer Verlag.

[ADGY98] N. Adam, O. Dogramaci, A. Gangopadhyay, and Y. Yesha. *Electronic Commerce: Technical, Business, and Legal Issues.* Prentice Hall (ISBN: 0-13-949082-5), Inc., 1998.

[BDFP01] B. Benatallah, M. Dumas, M.-C. Fauvet, and H.-Y. Paik. Self-coordinated and self-traced composite services with dynamic provider selection. Technical report, The University of New South Wales, School of Computer Science & Engineering, 2001. Available at http://www.cse.unsw.edu.au/~mcfauvet/selfserv.ps.gz.

[BDSN02] B. Benatallah, M. Dumas, Q.Z. Sheng, and A.H.H. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *Proc. of the International IEEE Conference on Data Engineering*, San Jose, USA, February 2002. to appear.

[BMB$^+$00] B. Benatallah, B. Medjahed, A. Bouguettaya, A. Elmagarmid, and J. Beard. WebBIS: a system for building and managing

Web-based virtual enterprises. In *Proc. of the 1st workshop on Technologies for E-Services, in cooperation with VLDB2000*, Cairo, Egypt, September 2000.

[BMR+96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System Of Patterns*. John Wiley & Sons, West Sussex, UK, 1996.

[Bou99] A. Bouguettaya, editor. *Introduction to the Special Issue on Ontologies and Databases*, Distributed and Parallel Databases Journal. Kluwer Publishers, 1999. 7(1).

[Bro00] M. Brodie. The B2B E-commerce Revolution: Convergence, Chaos, and Holistic Computing. In in *Information System Engineering: State of the Art and Research Themes, S. Brinkkemper, E. Lindencrona, and Solvberg (eds.)*, London, June 2000.

[BSZ98] M. Bichler, A. Segev, and J. L. Zhao. Component-based E-Commerce: Assessment of Current Practices and Future Directions . *ACM SIGMOD Record*, 27(4), December 1998.

[Bus01] C. Bussler. B2B protocol standards and their role in semantic B2B integration engines. *IEEE Data Engineering Bulletin*, March 2001.

[CH01] Q. Chen and M. Hsu. Inter-enterprise collaborative business process management. In *Proc. of the Int. Conference on Data Engineering (ICDE)*, Heidelberg, Germany, April 2001. IEEE Press.

[CIJ+00] F. Casati, S. Ilnicki, L.-J. Jin, V. Krishnamoorthy, and M.-C. Shan. Adaptive and dynamic service composition in eFlow. In *Proc. of the Int. Conference on Advanced Information Systems Engineering (CAiSE)*, Stockholm, Sweden, June 2000. Springer Verlag.

[Coa96] WorkFlow Management Coalition. Terminology and glossary. Technical Report WFMS-TC-1011, Workflow Management Coalition, Brussels - Belgium, 1996.

[Com00] Commerce One Inc. XML Common Business Library (xCBL). http://www.xcbl.org, December 2000.

[cXM] cXML.org. Commerce XML resources. http://www.cxml.org.

[Dog98] A. Dogac, editor. *ACM SGMOD Record: Special Issue on Electronic Commerce*, ACM SIGMOD RECORD. ACM, December 1998. 27(4).

[Dog99] A. Dogac, editor. *Special Issue of Distributed and Parrallel Databases on Electronic Commerce*, Distributed and Parallel Databases Journal. Kluwer Publishers, 1999. 7(2).

[DR99] P. Dadam and M. Reichert, editors. *Proceedings of the Informatik'99 Workshop on Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications*, Paderborn, Germany, October 1999.

[FBDP01] M.-C. Fauvet, B. Benatallah, M. Dumas, and H. Paik. Self-coordinated and self-traced dynamic composite services. In *Actes des 17e Journées Bases de Données Avancées*, Agadir, Maroc, novembre 2001. 21 pages.

[FDBP01] M.-C. Fauvet, M. Dumas, B. Benatallah, and H. Paik. Peer-to-peer traced execution of composite services. In *Proceedings of the International Workshop on Technologies for E-Services (TES 2001). In cooperation with VLDB 2001.*, Roma, Italy, 2001.

[Ga99] D. Georgakopoulos and al. Managing Process and Service Fusion in Virtual Enterprises. *Information Systems*, 24(6):429–456, 1999.

[GAHL00] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises. *International Journal of Computer Systems Science & Engineering*, 15(5):277–290, 2000.

[Geo99] D. Georgakopoulos, editor. *Information Technology for Virtual Enterprises*, Proc. of the 9th Int. Workshop on Research Issues on Data Engineering. IEEE Computer Society, March 1999.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Readings MA, USA, 1995.

[GT98] A. Geppert and D. Tombros. Event-based Distributed Workflow Execution with EVE. In *Proc. of Middleware '98 Workshop*, Sept. 1998.

[HN96] D. Harel and A. Naamad. The *statemate* semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, October 1996.

[JFN$^+$00] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, B. Odgers, and J. L. Alty. Process management system using ADEPT: A real-world case study. *Journal of Applied Artificial Intelligence*, 14(5):421–465, 2000.

[JNF$^+$00] N.R. Jennings, T.J. Norman, P. Faratin, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *Journal of Applied Artificial Intelligence*, 14(2):145–189, 2000.

[KLKD01] H. Kuno, M. Lemon, A. Karp, and Beringer D. Conversations + interfaces =3d business logic. In *Proc. of the 2nd Workshop on Technologies for E-Services (TES)*, Roma, Italy, September 2001.

[MWW$^+$98] P. Muth, D. Wodtke, J. Weissenfels, A.K. Dittric h, and G. Weikum. From centralized workflow specification to distributed workflow execution. *Journal of Intelligent Information Systems*, 10(2), March 1998.

[OF01] B. Omelayenko and D. Fensel. A two-layered integration approach for product information in B2B E-commerce. In *Proc. of the International Conference on Electronic Commerce and Web Technologies (EC-Web)*, Munich, Germany, September 2001. Springer Verlag.

[QS00] C. Quix and M. Schoop. Facilitating Business-to-Business Electronic Commerce for Small and Medium-sized Enterprises. In *Proc. of the International Conference of the Institute for Operations Research and Management Sciences (INFORMS)*, Salt Lake City, USA, May 2000.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[Ros] RosettaNet. Home page. http://www.rosettanet.org.

[SCDS01] M. Sayal, F. Casati, U. Dayal, and M.C. Shan. Integrating workflow management systems with Business-to -Business interaction standards. Technical Report HPL-2001-167, HP Labs, July 2001.

[SFJ97] C. Sierra, P. Faratin, and N. Jennings. A service-oriented negotiation model between autonomous ag ents. In *Proc. of the 8th European Workshop on Modeling Autonom ous Agents in a Multi-Agent World (MAAMAW)*, Ronneby, Sweden, 1997. Springer Verlag.

[SGCB00] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In *Proc. of the Int. Conference on Advanced Information Systems Engineering (CAiSE)*, Stockholm, Sweden, June 2000. Springer Verlag.

[The01] The CrossFlow Project. Home Page. http://www.crossflow.org, 1998–2001.

[VLD00] *Proceedings of the First Workshop on Technologies for E-Services (In cooperation with VLDB2000)*, September 2000.

[Web] Semantic Web. Workshop proceedings. http://semanticweb2001.aifb.uni-karlsruhe.de.

[Whi97] A. Whinston, editor. Electronic Commerce: A Shift in Paradigm, IEEE Internet Computing. IEEE, November 1997. Special Issue on Electronic Commerce 1(6).

[YP00] J. Yang and M. Papazoglou. Interoperation support for electronic business. CACM, 43(6):39–47, June 2000.