Overview of Some Patterns for Architecting and Managing Composite Web Services

B. BENATALLAH
School of Computer Science and Engineering, UNSW, Sydney NSW 2052, Australia
M. DUMAS
Centre for IT Innovation, QUT, Brisbane QLD 4001, Australia
M.-C. FAUVET
School of Computer Science and Engineering, UNSW, Sydney NSW 2052, Australia
F.A. RABHI
School of Information Systems, UNSW, Sydney NSW 2052, Australia
and
QUAN Z. SHENG
School of Computer Science and Engineering, UNSW, Sydney NSW 2052, Australia

The composition of Web services has gained a considerable momentum as a paradigm for enabling Business-to-Business (B2B) Collaborations. Numerous technologies supporting this new paradigm are rapidly emerging, thereby creating a need for methodologies that bring these technologies together. The identification and documentation of relevant patterns, both at the analysis and design levels, is an important step in this direction.

Categories and Subject Descriptors: D.2.10 [**Software Engineering**]: Design; H.4.3 [**Information Systems**]: Information Systems Applications—*Communications Applications*

General Terms: Design, Languages

Additional Key Words and Phrases: Web Services, B2B E-Commerce, Design Patterns

1. INTRODUCTION

Web Services are loosely coupled Internet-accessible software entities delivering functionalities provided by business applications and processes. Examples of Web services include Internet banking, search engines, auctioning sites, supply chain management, etc. The emergence of technologies and standards supporting the development of Web services, such as J2EE, .Net, SOAP, UDDI, and ebXML (see [WebServices.Org]) has unleashed a wave of opportunities for enterprises to form alliances by composing their services in order to provide "one-stop shops" for their customers [Casati and Shan 2001]. However, there is still a need for principles,

B. Benatallah can be contacted by email at boualem@cse.unsw.edu.au
M.-C. Fauvet is on leave from LSR-IMAG, University of Grenoble, France.

methodologies, and design techniques to architecture and manage composite Web services based on these emerging technologies.

This paper takes the viewpoint that an important step in this direction is the emergence of relevant patterns at various levels of granularity (i.e. analysis, architecture and design patterns, as well as idioms). Accordingly, the paper identifies a collection of prospective patterns addressing various activities in the life cycle of a composite Web services (Section 2). The paper then overviews some of these patterns (Sections 3, 4, 5, 6), and discusses ongoing and future work (Section 7).

2. PATTERNS FOR ARCHITECTING AND MANAGING COMPOSITE SERVICES

The life cycle of activities related to composite services is illustrated in Figure 1. Briefly stated, these activities are:

—*Wrapping native services*: ensuring that a native/proprietary service (e.g . legacy application) can be invoked by other Web services regardless of its underlying data model, message format and interaction protocol.

—*Service advertisement/discovery*: generating service descriptions and publishing these descriptions in registries for subsequent discovery.

—*Setting outsourcing agreements*: negotiating, establishing, and enforcing contractual obligations between partner services.

—*Assembling composite services*: identifying services to realise a given composition, specifying their interactions at a high level of abstraction, and deriving external descriptions and service level agreements for the resulting composite services.

—*Executing composite services*: enacting composite service specifications w.r.t execution models satisfying certain practical constraints (e.g. efficiency, availability).

—*Monitoring composite service executions*: supervising composite service executions (e.g., logging service invocations, state changes, and message exchanges) in order to detect contract violations, measure performance, and predict exceptions.

—*Evolving services*: adapting composite services to accommodate organisational changes, to take advantage of new technological opportunities, or to take into account feedback from monitoring.

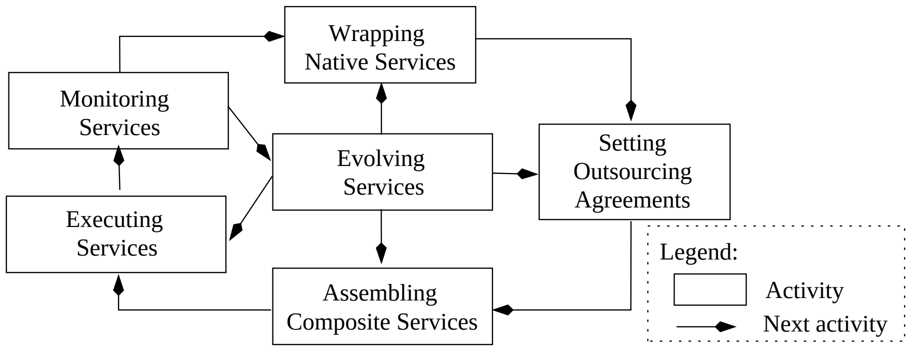

Fig. 1. Composite Service Life Cycle

Several technologies have been developed to support the various activities of the service composition life cycle. The problem is that (mostly for historical reasons) these technologies tend to provide overlapping features. In addition, they require intimate knowledge of several low-level features associated with the supporting infrastructure (e.g. operating systems, networks, and programming languages). As the infrastructure is modified or upgraded, software maintenance becomes an issue.

We believe that the analysis and design activities related to service composition should be first conducted at a high level of abstraction, primarily addressing the business requirements, and design issues at the architectural level. In accordance with standard software engineering practice, lower level design issues and the choice of adequate technology should be treated separately (at the implementation stage). Accordingly, we propose seven analysis/design patterns corresponding to the above activities: *Service Wrapper*, *Composite Service Specification*, *Service Discovery*, *Service Negotiation*, *Composite Service Execution*, *Service Monitoring* and *Service Evolution*. Due to the emerging nature of the topic, these patterns are in early stages of development and validation, and should rather be called *proto-patterns*.

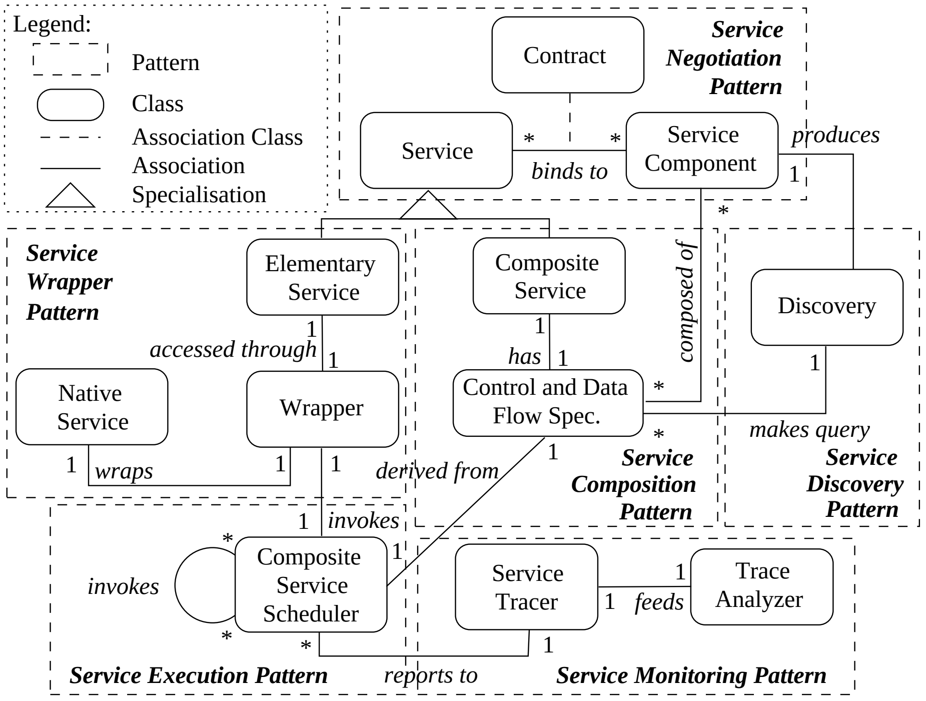

Fig. 2. Patterns for Composing Services

The structural elements brought in by these patterns and the dependencies between them are summarised by the UML [Booch et al. 1999] class diagram depicted in Figure 2[1]. Each pattern comes with a range of implementation strategies that ground it to the underlying technology while explaining the trade-offs involved and

[1]For layout reasons, the Service Evolution Pattern is not shown in the diagram.

the possible optimisations. In this paper, we overview the first four patterns only. A detailed description of all the patterns are given in [Benatallah et al. 2002, first].

3. SERVICE WRAPPER PATTERN

Description. Interoperation with both internal and external Web services is central because composite Web services are built on top of heterogeneous and otherwise independent Web services. What is needed is to abstract and to hide the heterogeneity of *proprietary* Web services built on top of (e.g.) CORBA, J2EE, .Net.

Solution. The purpose of the Service Wrapper Pattern (SWP) is to separate a pre-exiting service specification (e.g., service's interface) from its implementation (e.g., a standalone program, an ERP application, or a workflow). A service wrapper may be composed of a:

—*communication manager*: Web services may use different communication protocols (e.g., HTML/HTTP, SOAP/HTTP, Java RMI, CORBA IIOP, and DCOM). The communication manager should support the translation of messages between heterogeneous communication protocols, for example through systematic translation to and from a common protocol (e.g., SOAP).

—*security manager*: Web services may need to cross corporate firewall and security systems in order to access partners' services. The purpose of this module is to handle security issues including single mutual authentication corporate wide, fine grain authentication, and access auditing and authorisation, communication integrity, confidentiality, and non-repudiation.

—*content manager*: It is likely that Web services use disparate information formats and semantics. For example, if an internal application uses xCBL to represent business documents, and this application needs to interact with an external service which expects documents in cXML [Bussler 2001], the conversion between these two formats should be handled by the content manager.

—*conversation manager*: is concerned with the conversational interactions (i.e, joint business process) among Web services. For instance, a given service may require a login procedure to be performed prior to any access to the service's functionalities, while another service 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.

Implementation aspects. The issues of security and communication can be handled using established or emerging protocols (e.g., HTTP-S, SAML). The issue of handling document format heterogeneity at the syntactical level can be addressed to some extent by existing content-management technologies (e.g. XML-based ontologies such as cXML and xCBL). However, the issue of interoperability in the presence of semantic heterogeneity is still an open problem [Brodie 2000].

While B2B standards defining conversational protocols are emerging (e.g., RosettaNet's PIPs), the issue of mapping a conversation in a given protocol into an "equivalent" conversation in another protocol still needs to be addressed on a case by case basis. Application integration solutions (e.g., TibcoSoftware's TIB/Active Enterprise Suite, CrossWorlds) provide abstract interfaces, remote operation invocations, connectors for back-end systems, etc. [Brodie 2000]. By doing so, they provide the connection and coordination of data and operations among applications. Some frameworks (e.g. eCO) use XML-based messages in documents to describe the interactions that make up the B2B processes [Dogac and Cingil 2001].

[Sayal et al. 2002] describe 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. At runtime, this workflow interacts with external service providers through a conversation manager, which handles the conversion of internal workflow variables into external documents.

4. SERVICE DISCOVERY PATTERN

Description. The space from which users can locate Web services may be large and highly dynamic. In particular, on-the-fly B2B integration is better supported by establishing online catalogues for Web services. These catalogues should provide capabilities for brokering and dynamic collaboration. Instead of statically binding Web services to each other, catalogues should allow to dynamically discover new Web services with the right set of features and bind them at run time. Selecting a partner should consider the available Web services, characteristics, organisational policies and resources that are needed to accomplish the integrated Web service.

Solution. The purpose of Service Discovery Pattern (SDP) is to facilitate the automatic discovery of services. Given that 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 services based on constraints over their meta-data.

Implementation aspects. A number of industry efforts to define standards that provide common building blocks for Web service discovery and integration emerged recently including UDDI (*Universal Description, Discovery, and Integration*), WSDL (*Web Services Description Language*), and ebXML (see for example [Casati et al. 2000]). UDDI provides an XML-based registry for advertising businesses and services. The advertisement and discovery of services and businesses in UDDI exploit keywords categorisation. 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. WSDL is an XML-based language for describing the content and capabilities of Web 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 to UDDI. However, ebXML focuses on business processes from a workflow perspective. Other approaches to service advertisement and discovery include IETF's Service Location Protocol and Sun Microsystem's JINI.

In any case, it is essential to consider that services are characterised by numerous functional and non-functional parameters [O'Sullivan et al. 2002]. Hence, search interfaces over service registries should be able to handle complex criteria.

5. SERVICE NEGOTIATION PATTERN

Description. A service has a number of interrelated parameters. A prospective consumer and a provider need to set up the values of these parameters through a mutual agreement i.e., a *service contract*. This agreement is the result of a tradeoff between the possibly conflicting constraints and preferences of each party. The parties are generally reluctant to disclose these preferences to each other.

Solution. The purpose of the Service Negotiation Pattern (SNP) is to abstract contracts for a given service into *contract templates* to avoid building contracts from scratch. 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 negotiation between a prospective consumer and a 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 [Jennings et al. 2000]. In the extreme case, when only one parameter is variable, and the domain of this parameter can be modelled as a set of numbers (e.g., the price), the negotiation can be reduced to an online auction. If new parameters can 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.

Implementation aspects. In the general case, the effective representation and negotiation of contracts for service integration and provisioning remains an open issue. Currently, service integration platforms such as Microsoft's BizTalk and Extricity's Alliance, do not address the negotiation of service contracts. On the other hand, negotiation tools such as Attricom and Ozro Negotiate, which provide basic support for online negotiations, do not take into account the specificities of service contracts.

6. COMPOSITE SERVICE SPECIFICATION PATTERN

Description. The fast integration of business processes is an essential requirement for organisations 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 purpose of the Composite Service Specification (CSS) Pattern is to facilitate this type of relationship. The latter form of partnership does not assume an *a priori* trading relationship among partners and will be the subject of another pattern (Service Discovery Pattern).

Solution. We distinguish between *elementary* and *composite* services. An elementary service is a pre-existing service whose execution is entirely under the responsibility of the service wrapper pattern. A composite service is recursively defined as an aggregation among other services whether composite or elementary, which are referred to as *component services*. The purpose of the CSS Pattern is to specify the interactions among the components of a composition service without referring to any implementation or execution model. The *specification* of interactions among services must include descriptions about both *control-flow* and *data-flow*. The 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. The data-flow captures the flow of data between component services.

Implementation aspects. A natural way of describing the control-flow and data flow of composite services, is to use an existing process-modelling language. The Workflow Management Coalition (WfMC) 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 formal semantics, making it difficult to compare their capabilities and expressiveness in order to make an objective choice between them [Muth et al. 1998]. More recently, industry initiatives such as WSFL and XLANG are emerging as promising alternatives to specify the control and data-flow dependencies between the components of composite services (e.g., the composite service's choreography).

Cross-organisational workflows [Yang and Papazoglou 2000] focus on the automation of 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.

Component-based frameworks [Dogac 1998; Brodie 2000] support 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 spread through the implementation code of every component. Thus the composition of services in this approach is mainly ad-hoc.

In document-based approaches such as EDI and XML-based frameworks [Dogac and Cingil 2001; Yang and Papazoglou 2000; Casati et al. 2000], the interactions among the components of a composite service are specified by the shared document definitions. The components are interconnected in terms of agreed upon documents. Interactions between components may be carried out according to a specific B2B standard (e.g., EDI, OBI, RosettaNet, cXML) or bilateral agreements.

7. CONCLUSIONS AND DIRECTIONS

Web service composition is a promising area of research and development. From a research point of view, the key issues that need to be investigated relate to the facilitation of Web service composition in large, autonomous, heterogeneous, and dynamic environments. For B2B E-commerce to really take off, there is a need for effective and efficient means to *search*, *abstract*, *compose*, *analyse*, *execute*, and *evolve* Web services in appropriate time-frames.

In this paper, we provided an overview of several proto-patterns for architecting and managing composite Web services. These proto-patterns aggregate results from previous efforts in the area of Web service composition, into guidelines for addressing design issues related to the various activities in the life cycle of a composite service. This contribution is a starting point towards a pattern-oriented service composition methodology. The solutions provided by the proto-patterns need to be validated through the implementation of supporting tools and the development of case studies, and refined according to the feedback obtained from this validation.

As a step further in this direction, our ongoing work in the context of the *SELF-SERV* project [Benatallah et al. 2002, second] aims at providing high-level modelling constructs and supporting tools to *search, compose, execute, monitor, and evolve Web services*. SELF-SERV provides a framework in which services can be declaratively composed and the resulting composite services can be executed in a peer-to-peer way within a dynamic environment. One of the main objectives of the project is to devise novel integration techniques that allow fast development of new services from existing ones. Appropriate information discovery techniques will be used to efficiently locate and exploit services in a dynamic and constantly growing environment. Advanced service management techniques will monitor services, dynamically adapt their components, and relationships, and bring them to mutually acceptable states. These techniques will also support peer-to-peer collaborative service execution.

REFERENCES

Benatallah, B., Dumas, M., Fauvet, M.-C., and Rabhi, F. 2002. Towards patterns of web services composition. In *Patterns and Skeletons for Parallel and Distributed Programming*, F. Rabhi and S. Gorlatch, Eds. Springer Verlag, London, UK.

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

Booch, G., Jacobson, I., and Rumbaugh, J. 1999. *The Unified Modeling Language user guide*. Addison-Wesley.

Brodie, M. 2000. 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.

Bussler, C. 2001. B2B protocol standards and their role in semantic B2B integration engines. *Bulletin of the Technical Committee on Data Engineering 24,* 1.

Casati, F., Georgakopoulos, D., and Shan, M., Eds. 2000. *Proceedings of the 2nd VLDB Workshop on Technologies for E-Services*. Springer Verlag, Rome, Italy.

Casati, F. and Shan, M.-C. 2001. Dynamic and adaptive composition of e-services. *Information Systems 26,* 3 (May), 143–162.

Dogac, A., Ed. 1998. *ACM SIGMOD Record: Special Issue on Electronic Commerce*. ACM SIGMOD RECORD. ACM. 27(4).

Dogac, A. and Cingil, I. 2001. A Survey and Comparison of Business-to-Business E-Commerce Frameworks. *ACM SIGecom Exchanges 2,* 2 (June), 14–25.

Jennings, N., Norman, T., Faratin, P., O'Brien, P., and Odgers, B. 2000. Autonomous agents for business process management. *Journal of Applied Artificial Intelligence 14,* 2, 145–189.

Muth, P., Wodtke, D., Weissenfels, J., h, A. D., and Weikum, G. 1998. From centralized workflow specification to distributed workflow execution. *Journal of Intelligent Information Systems 10,* 2 (March).

O'Sullivan, J., Edmond, D., and ter Hofstede, A. 2002. What's in a Service. *Distributed and Parallel Databases 12,* 2–3 (September), 117–133.

Sayal, M., Casati, F., Dayal, U., and Shan, M. 2002. Integrating workflow management systems with Business-to-Business interaction standards. In *Proc. of the International Conference on Data Engineering (ICDE)*. IEEE Press, San Jose CA, USA.

WebServices.Org. The Web Services Community Portal. `http://www.webservices.org`.

Yang, J. and Papazoglou, M. 2000. Interoperation support for electronic business. *CACM 43(6)*, 39–47.