Patterns for e-Service Composition

Marie-Christine Fauvet[1*] **Marlon Dumas**[2] **Fethi Rabhi**[3]
Boualem Benatallah[4]

[1] LSR-IMAG, University of Grenoble
BP 72, 38402 Saint-Martin d'Hères, France. E-Mail: Marie-Christine.Fauvet@imag.fr

[2] Centre for Information Technology Innovation, Queensland University of Technology,
GPO Box 2434, Brisbane QLD 4001, Australia. E-Mail: m.dumas@qut.edu.au

[3] School of Information Systems, Technology and Management,
University of New South Wales, Sydney NSW 2052, Australia. E-Mail: f.rabhi@unsw.edu.au

[4] School of Computer Science & Engineering, University of New South Wales,
Sydney NSW 2052, Australia. E-Mail: boualem@cse.unsw.edu.au

Abstract

Due to the growth of the Internet, the concept of e-service has gained a considerable momentum as a paradigm for supporting Business-to-Business (B2B) collaboration. In this paper, we propose two patterns dedicated to the composition of e-services, namely the Service Wrapper Pattern and the Service Composition Pattern. By *service*, we mean any class of immaterial products whose provision involves the execution of a set of human and/or computational activities within an organisation, or across several organisations. An e-service is a service which is accessible through electronic means (e.g., a web interface).

1 Introduction

Internet and Web technologies have opened new means of doing business cheaply and more efficiently. Established enterprises are continuously discovering new opportunities to form alliances with other enterprises, by offering value-added integrated e-services. By *service*, we mean any class of immaterial products whose provision involves the execution of a set of human and/or computational activities within an organisation, or across several organisations (Dumas, O'Sullivan, Heravizadeh, Edmond & ter Hofstede 2001). An e-service is a service which is accessible through electronic means (e.g., a web interface).

At the same time, the lack of high level abstractions and functionalities for e-service integration has triggered a considerable amount of research and development efforts. This has resulted in a number of systems and standards addressing different aspects of service integration (Dogac, editor 1998, Brodie 2000, Weikum 2001) (e.g., XML, CORBA, workflow-based systems). However, there is still a need to provide rigorous software methodologies and tools supporting the rapid integration of e-services. Ad hoc and proprietary solutions on the one hand, and lack of canonical methodologies for abstracting, searching, composing, executing, monitoring, and evolving e-services on the other hand, have largely hampered a faster pace in deploying Web-based B2B applications.

We propose two complementary design patterns related to activities involved during the service composition life-cycle. Among others, these activities are:

- *Wrapping native services*: ensuring that a native/proprietary service (e.g., legacy application) can be invoked regardless of its underlying data model, message format and interaction protocol.
- *Assembling and executing composite services*: identifying component services, specifying their interactions at a high level of abstraction, and deriving an execution model that satisfies the specifications (e.g., data flow and control flow).

The proposed design patterns are *Service Wrapper* and *Service Composition*. Both patterns come with a range of implementation strategies that depend on the underlying technology. These strategies explain the trade-offs involved, possible optimisations, etc.

The rest of the paper is organised as follows. Section 2 presents the Service Wrapper Pattern. The Service Composition Pattern is described in section 3. Finally, Section 4 provides some concluding remarks and bring both patterns together to show their dependencies.

2 Service Wrapper Pattern (SWP)

2.1 Intent

The purpose of the SWP is to ensure that a pre-exiting service specification (e.g., service's interface) is separated from its implementation (e.g., a stand-alone program, an ERP application, or a workflow).

*Work conducted while the author was Visiting Fellow at the School of Computer Science & Engineering, University of New South Wales, Australia.

2.2 Problem description and forces

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 processes.

Forces. The specific forces that arise in this situation are:

- Given that the data model and format in which a business document is generated generally differs from that in which it is interpreted, there is a need for specifying mappings between formats and data models. Specifying these mappings requires an understanding of the meaning of the terms used in the formats and data models.
- Given that the applications are likely to use different interaction protocols (e.g., different message names, semantics and sequencing), these protocols need to be aligned. In other words, conversions between sequences of messages need to be specified.
- Given that the applications belong to different organisations, and that they are likely to exchange critical business information, properties such as the confidentiality, integrity and non-repudiation of these exchanges need to be ensured.
- The number of services consumed and provided is typically large and volatile. Some of the applications which provide and consume services within the organisation are implemented by legacy systems which are not cost-effective to be modified to accommodate new interaction requirements. Even those applications that can undergo modifications require a considerable amount of programming effort to adapt to new requirements.

2.3 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 organisations. Being developed by separate teams, for different purposes, and at different times, these information systems are heterogeneous both from the managerial and technological viewpoints. For similar reasons, an organisation requiring an external e-service needs to make sure that its information system is capable of inter-operating 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. 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 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.

2.4 Related patterns

The Service Wrapper Pattern specialises the Gateway Pattern (Buschmann, Meunier, Rohnert, Sommerlad & Stal 1996) by explicitly taking into account the issues of security, document format, and conversational protocol heterogeneity. The Service Wrapper Pattern can also be seen as a combination of the Façade pattern with the Proxy Pattern (Gamma, Helm, Johnson & Vlissides 1995). In fact, the Service Wrapper acts as a proxy which handles calls to remote servers on behalf on an application and offers as well a unified entry point to a set of services offered by an organisation. Also notice that the scope of the Service Wrapper Pattern is far more specialised than those of the above two patterns.

2.5 Solution

The idea of the SWP is to structure a service wrapper into 3 different modules (see Figure 1):

- *Security manager*: e-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 e-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 (Dogac & Cingil 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 e-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.

2.6 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 time to time, a given partner may be replaced by another.

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 processes from time to time, 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, the company may change its own service interfaces, whether to enhance or to simplify them, or to cope with internal policy changes. Figure 1 shows the internal architecture of the WSP 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 SWP of this company. Symmetrically, every request for any of the four services provided by *Traduit-Tout* transits through the SWP.

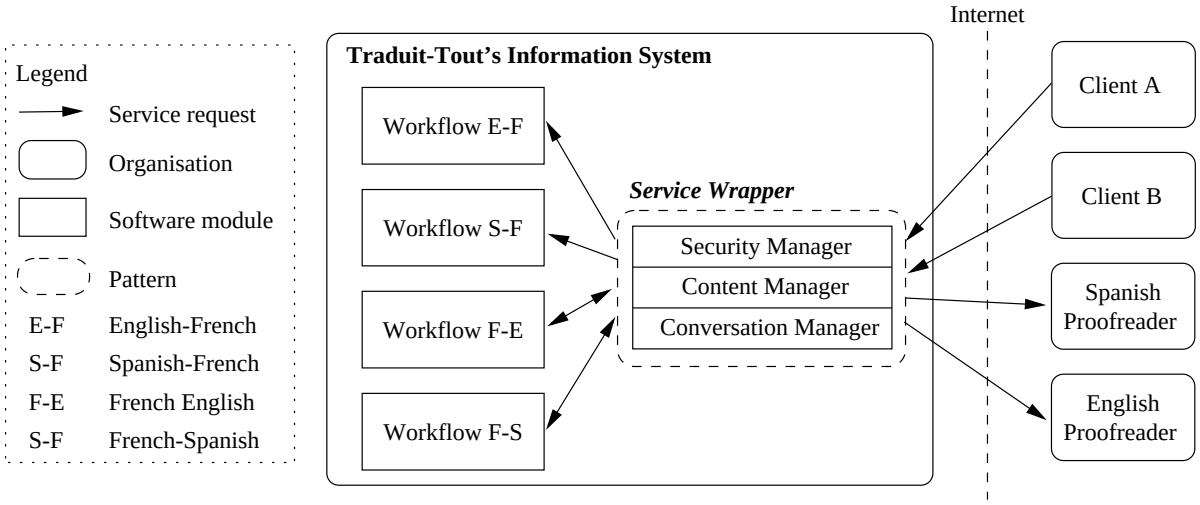

Figure 1: The SWP of the *Traduit-Tout* company.

2.7 Implementation aspects

In the following, we present techniques for handling document format and conversational protocol heterogeneity, two of the three aspects addressed by the SWP. The issue of security is not discussed here as it can be handled using established cryptography techniques, secure communication protocols (e.g., HTTP-S), 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 remains an open area of research.

In the context of XML-based business document standards, for example, the XSLT language provides a means of expressing transformations from documents abiding to a given standard, into documents abiding to another standard. The specification of these transformations in XSLT is 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 uses XSLT as the underlying transformation 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 (Omelayenko & Fensel 2001). The authors 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:

- Abstraction: translate XD into an RDF document RD abiding to the data model of S.
- Conversion: Translate RD into another RDF document RD' abiding to the data model of S'.
- Refinement: Translate RD' into an XML document XD' abiding to the syntax of S'.

2.8 Related Work

As shown in (Omelayenko & Fensel 2001), 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 open. While B2B standards defining conversational protocols have recently emerged (e.g., RosettaNet's PIPs [1]), the issue of mapping a conversation in a given protocol into an "equivalent" conversation in another protocol is an open problem.

Web-Service Conversation Language (WSCL) could be used for specifying conversational protocols (Kuno, Lemon, Karp & D 2001). The authors show that these specifications can be used to automatically build a conversation controller, one of the two components of the conversational protocol manager.

In (Sayal, Casati, Dayal & Shan 2001), the authors 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 conversations manager, which handles the conversion of internal workflow variables into external documents.

3 Service Composition Pattern (SCP)

3.1 Intent

The issues addressed by the SCP are, first to facilitate the description of integrated services and second to derive from its specification, a suitable execution model for the resulting composite service.

3.2 Problem description and forces

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. The task involved in this context is that of integrating services (either pre-existing or composite services) in order to build a composite service offering new integrated features and to make it ready to be executed when requested by users.

Forces. The important forces that play a role in service composition are summarised as follows:

- Given that a solution which allows fast integration and easy maintenance is required there is a need for a high-level approach for describing interactions amongst services, without referring to any implementation or execution model. Even if a low level approach is more flexible it may require significant coding effort.
- Given that services could be either reused in composition or decomposed in order to ease their design and implementation, arbitrary nesting of composite services needs to be supported.
- A high-level approach can accommodate arbitrarily large compositions of services but makes composite service execution a complex issue. A low-level approach may not scale easily but control over the composite service execution is maintained.
- In some situations service composition is only known at run-time. How to dynamically discover and identify services and to bind them at run time.

The latter force is not addressed by the Service Composition Pattern.

3.3 Context

The dynamic integration of business processes is an essential requirement for organisations in the context of the Internet. Unfortunately, as electronic commerce applications are most likely autonomous and heterogeneous, connecting and coordinating them in order to build inter-organisational services is a difficult task. So far, development of integrated B2B services is largely ad hoc, time-consuming and requires an enormous effort of low-level programming. This approach is not only tedious, but also hardly scalable because of the volatility of the Internet, and the dynamic nature of business alliances.

3.4 Related patterns

Two of the Business-to-Business integration (B2Bi) Patterns reported in (eBizQ n.d.) are related to the SCP. Specifically, the Open Process B2Bi Pattern models a B2B integrated service as a global process, in which each participant contributes with its own internal process. The participants interact in a peer-to-peer way. In the Closed Process B2Bi Pattern on the other hand, there a principal participant responsible for managing the global process. The other participants are secondary: they do not have visibility into the global process, nor do they actively manage it. These two patterns can therefore be seen as specific realisations of the SCP: in the former the collaboration between participants is conducted in a peer-to-peer way, and in the latter it is conducted by a central authority.

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

3.5 Solution

The solution distinguishes between *elementary* and *composite* services. Composite services are recursively defined as an aggregation between other composite services and elementary services, which are referred to as *component services*. Elementary services are pre-existing services, whose execution is entirely under the responsibility of the SWP (see Section 2). 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. On the other hand the data-flow captures the flow of data between component services.

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 This scheduler is responsible for:

- 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 wrapper if they are native services) in the order and under the conditions specified in the control-flow.
- While the service S is available, the scheduler receives and processes service requests.
- The scheduler is also responsible for handling and processing data according to the data semantics of the composite service.

The derivation of the scheduler from the composite service specification could be based upon two execution models: (i) The components of a composite service are coordinated by a central scheduler hosted by the provider of the composite service (components may be distributed), (ii) The entities participating in a composite service coordinate the execution in a distributed manner (e.g., through peer-to-peer communication). In the former, the scheduler of a composite service S is responsible for initiating the execution of the components of S according to the control-flow associated with S. In the latter, the responsibility of coordinating the execution of a composite service is distributed across the providers which host the components of the composite service.

3.6 Example

As an example, we consider an organisation that wants to offer an on-line travel planner called "Travel Solutions". This planner will allow users to build their own itinerary in a given city. To do so, the organisation must integrate the following independent services:

- Flight booking: this service searches for a flight and when the most suitable flight is found, initiates the booking and the payment. This service can be assigned to an individual provider (e.g., an airline company).
- Accommodation booking: this service searches for different styles of accommodation (hotels, hostels, bed & breakfast, camping, etc). It is assigned to a community of providers that federates entities such as public central booking and private booking sites. When a service request is addressed to the community, its representative forwards it to one of its members.
- Tourist attractions searching: this service gives information about the main tourist attractions (schedules, venues, etc.).
- Bicycle hire and car rental booking: this service gives the user the choice to ride a bike or drive a car. This choice is based upon the distance from the booked accommodation to the major tourist attractions. Both of these services are assigned to an individual provider.
- Event attendance planner: this service is decomposed into two others: event searching and ticket purchasing (if ticket pre-purchasing is required for the selected event).

3.7 Implementation aspects

A natural way of 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 consists 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) (Coalition 1996) 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. The use of formal notations for workflow specification has been considered in (Aalst 1998) and (Muth, Wodtke, Weissenfels, Dittrich & Weikum 1998). (Aalst 1998) discusses several advantages of using Petri-nets for describing the control-flow perspective of workflows, such as their expressive power. However, many designers find the Petri-net formalism difficult to grasp, and, 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, (Muth et al. 1998) advocates the use of statecharts instead (Harel & Naamad 1996). For example, the statechart

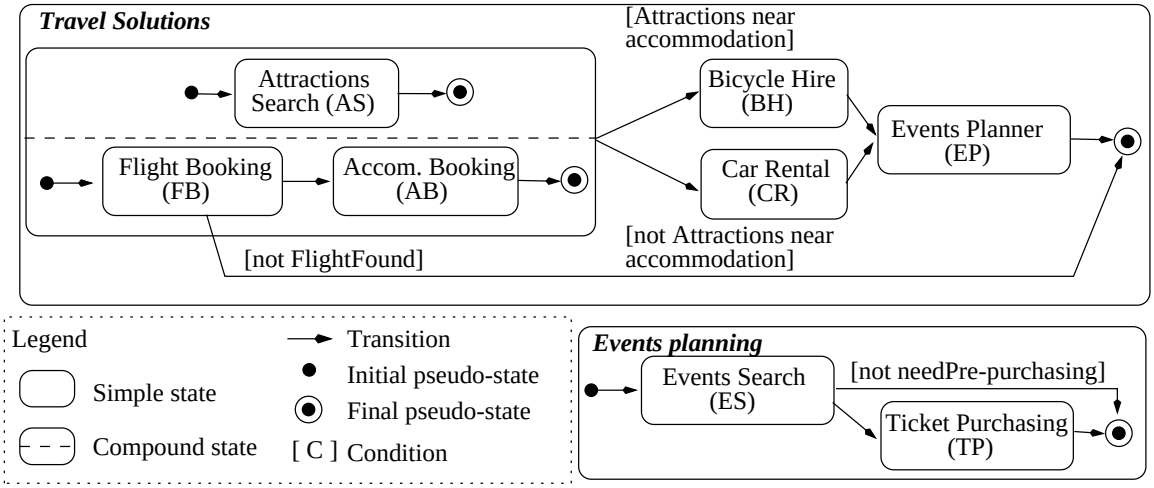

Figure 2: Example of a control flow specification using statecharts

depicted in Figure 2 specifies the composite service "Travel Solutions" described earlier. 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) as the foundation of many intra and inter-object process modeling constructs.

There are three other implementation approaches. The first one is to use cross-organisational workflows whose objective 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. The second approach is that of component-based frameworks (e.g., CORBA, J2EE) which provide for the connection and coordination of data and operations among services. The description of a composite service is worked out and agreed off-line. After that, the global description of a composite service is generally spread through the implementation code of every component.

Finally, the third approach, which includes EDI and XML-based integration frameworks, specifies the interactions among the components of a composite service using 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, RosettaNet, 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.

3.8 Related Work

CMI (Schuster, Georgakopoulos, Cichocki & Baker n.d.) is a platform for modeling and managing inter-enterprise business processes. A service is modeled by a state machine that specifies possible states of a service and their transitions. Transitions are caused by service operation (service activity) invocations or internal service transitions.

EFlow (Casati, Ilnicki, Jin, Krishnamoorthy & Shan n.d.) 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. In both CMI and eFlow the execution model is based on a centralised process engine.

WebBIS (Benatallah, Medjahed, Bouguettaya, Elmagarmid & Beard 2000) 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 customisation 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).

CPM (Chen & Hsu 2001) 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 sub-processes that `P` is responsible for.

In Mentor (Muth et al. 1998) the idea is to partition a global workflow specified as a statechart into a number of sub-workflows, each encompassing all the activities that are to be executed by a given entity within an organisation. Each of these sub-workflows is itself specified as a statechart and is executed by the corresponding organisation.

In SELF-SERV (Benatallah, Dumas, Sheng & Ngu 2002), 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 offering, 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 trigger-less 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).

In (Tut & Edmond 2002), the authors study the application of generic process patterns to service composition. The patterns considered in this work are essentially process templates whose activity nodes can be associated to specific service descriptions in order to yield composite services that satisfy a given set of functional and non-functional requirements. With respect to the patterns that we have presented, the patterns presented in this approach are at a finer level of ganularity: they are intended to facilitate the development of composite services fulfilling very specific user needs.

4 Conclusion

This text has discussed two complementary 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:

- Identify native services and make them elementary services through a wrapper (SWP)
- Specify the control and data flow semantics of the new service based on these elementary services or other composite services, called component services (SCP).

The structural elements brought in by these patterns and the dependencies between them are summarised by the UML (Rumbaugh, Jacobson & Booch 1999) class diagram depicted in Figure 3.

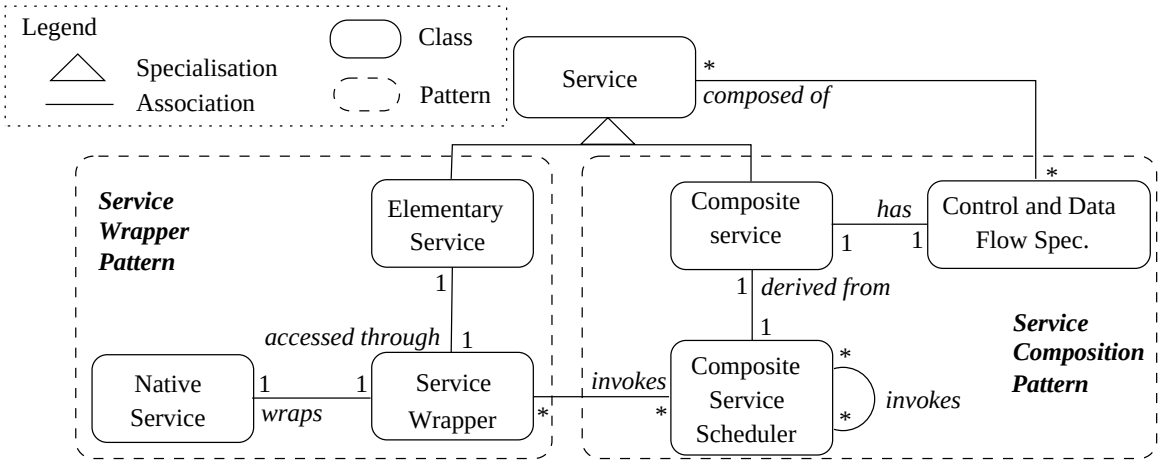

Figure 3: Patterns and issues for service composition

Figure 3 also relates these patterns to some issues neither the SWP nor the SCP deal with. However, in order to provide a framework for services management these issues must be addressed (Benatallah, Dumas, Fauvet & Rabhi 2002). They are listed below:

- *Setting outsourcing agreements*: negotiating, establishing, and enforcing contractual obligations between partner services.
- *Discovering services*: instead of statically binding e-services to each other, dynamically discovering new e-services with the right set of features and bind them at run time.
- *Monitoring services execution*: supervising service execution (e.g., state changes, contract violation), and measuring performance (e.g., time) and predicting exceptions.
- *Evolving services*: adapting services to accommodate actual business climate (e.g., economical or organisational changes) or to take advantage of new technological opportunities.

Acknowledgements

We thank our shepherd Brian Wallis for his valuable comments during the shepherding process. We are also very grateful to our workshop mates who gave us interesting and useful feedback during the conference, that helped us to improve our text for its final publication.

References

Aalst, W. v. (1998), Three good reasons for using a Petri-net-based workflow management system, *in* T. Wakayama, ed., 'Information and Process Integration in Enterprises: Rethinking documents', Kluwer Academic Publishers, Norwell MA, USA, pp. 161–182.

Benatallah, B., Dumas, M., Fauvet, M.-C. & Rabhi, F. (2002), Towards patterns of web services composition, *in* F. Rabhi & S. Gorlatch, eds, 'Patterns and Skeleton for Parallel and Distributed Computing', Springer Verlag (UK), chapter 10.

Benatallah, B., Dumas, M., Sheng, Q.-Z. & Ngu, A. (2002), Declarative composition and peer-to-peer provisioning of dynamic web services, *in* I. C. Society, ed., 'Proceedings of ICDE'02 Conference', San Jose, California.

Benatallah, B., Medjahed, B., Bouguettaya, A., Elmagarmid, A. & Beard, J. (2000), 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 VLDB', Cairo, Egypt.

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.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (1996), *Pattern-Oriented Software Architecture: A System Of Patterns*, John Wiley & Sons, West Sussex, UK.

Casati, F., Ilnicki, S., Jin, L.-J., Krishnamoorthy, V. & Shan, M.-C. (n.d.), Adaptive and dynamic service composition in eFlow, *in* 'Proc. of CAiSE'00', Springer Verlag, Stockholm.

Chen, Q. & Hsu, M. (2001), Inter-enterprise collaborative business process management, *in* 'Proc. of the Int. Conf. on Data Engineering (ICDE)', Heidelberg, Germany.

Coalition, W. M. (1996), Terminology and glossary, Technical Report WFMS-TC-1011, Workflow Management Coalition, Brussels - Belgium.

Dogac, A. & Cingil, I. (2001), 'A survey and comparison of business-to-business e-commerce frameworks', *ACM SIGecom Exchanges* **2**(2).

Dogac, editor, A. (1998), 'Special Issue on Electronic Commerce', *ACM SIGMOD Record* **27**(4).

Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D. & ter Hofstede, A. (2001), Towards a semantic framework for service description, *in* 'Proc. of the 9th Int. Conf. on Database Semantics', Kluwer Academic Publishers, Hong-Kong.

eBizQ (n.d.), 'eBiz integration', `http://b2b.ebizq.net/ebiz_integration/yee_1.html`. Last access on March 2002.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Readings MA, USA.

Harel, D. & Naamad, A. (1996), 'The STATEMATE semantics of statecharts', *ACM Transactions on Software Engineering and Methodology* **5**(4), 293–333.

Kuno, H., Lemon, M., Karp, A. & D, B. (2001), Conversations + interfaces =3d business logic, *in* 'Proc. of the 2nd Workshop on Technologies for E-Services (TES)', Roma, Italy.

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

Omelayenko, B. & Fensel, D. (2001), 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)', Springer Verlag, Munich, Germany.

Rumbaugh, J., Jacobson, I. & Booch, G. (1999), *The Unified Modeling Language reference manual*, Addison-Wesley.

Sayal, M., Casati, F., Dayal, U. & Shan, M. (2001), Integrating workflow management systems with Business-to-Business interaction standards, Technical Report HPL-2001-167, HP Labs.

Schuster, H., Georgakopoulos, D., Cichocki, A. & Baker, D. (n.d.), Modeling and composing service-based and reference process-based multi-enterprise processes, *in* 'Proc. of CAiSE'00', Springer Verlag, Stockholm.

Tut, M. & Edmond, D. (2002), The use of patterns in service composition, *in* 'CAiSE Workshop on Web Services, e-Business, and the Semantic Web (WES)', Toronto, Canada. http://pcsiwa12.rett.polimi.it/ pernici/WSeBT/papers/TutAndEdmond.pdf.

Weikum, G., ed. (2001), *Special Issue on Infrastructure for Advanced e-Services*, IEEE Data Engineering Bulletin. volume 24(1).