Business Service Integration Using Pattern Composition

Jeewanie Jayasinghe Arachchige and Hans Weigand

Tilburg University, P.O.Box 90153,
5000 LE Tilburg, The Netherlands
{J.JayasingheArachchig,H.Weigand}@uvt.nl

Abstract. Smooth integration of services is one of the key benefits of service-oriented enterprises. But the key questions are "Does the selected service address the real business need?" and "How to select the right service?". Separating the business choices and technical choices is necessary in order to structure the solution to these questions. In this research, the two perspectives are handled by embedding service thinking at business level design and secondly, by supporting the integration with available services and process templates. To reach this combined goal, we present a pattern-based approach for business service integration.

Keywords: Business Service Pattern, Pattern Composition, BSRM, Business Service Integration.

1 Introduction

Nowadays organizations are rapidly adopting SOA into their information systems and the business strategy, because of its emerging advantages. This environment creates an opportunity to such organizations to interact dynamically as service consumers and service providers making use of a service marketplace to design, offer and consume services [1]. On the other hand, it is claimed [2] that "the full potential of SOA is only realized when it is applied as an architecture for business design". Hence, service oriented business thinking is important when integrating the software services. Therefore the objective of this research work is, first of all, how to embed service thinking at business level and, secondly, to support the integration of available services and process templates. To reach this combined goal, we present a pattern-based approach for business service integration.

There exists some relevant work on pattern-based service integration. The research in [1] presented a pattern-based modeling approach to achieve the unforeseen integration of services into extensible enterprise systems. This framework demonstrates service integration to the presentation (HCI) layer using adaptation patterns that group common patterns of model elements and their relationships. Even though this approach enables an integrator to model or design the relevant integration aspects on a higher abstraction level than implementation-level, service thinking at the business level is not addressed. The work is related to the concept of plug-in technologies that allow the development and installation of web 2.0 applications.

M. Bajec and J. Eder (Eds.): CAiSE 2012 Workshops, LNBIP 112, pp. 519–530, 2012.

Based on the example of SAP's Enterprise Services, authors of [3] describe a representational model that integrates both service and data by consolidating existing models and patterns used during the service design process. On top of this model, they created a metadata repository based on a list of ES and their respective metadata. Both representational model and metadata repository represent the basis of the iterative search of software services. Even though the business objects are one of the ingredients of the representational model, there is no clear basis of selecting business objects to their model and their focus is on the presentation layer of the ES by providing a pathway to service matchmaking.

There is also relevant research that incorporates business thinking to the service design and aims at mapping those designs to the software level. One good approach is value based service design. [4,5,6,7] present an MDA approach to design and transform services from CIM, PIM to PSM. However, the focus of all these approaches is on service design and transformation, not on service integration. Some other researches can be found in the literature for example [1] and [3], with a focus on service integration. One problem here is that business choices and technical choices are all dealt with together at a technical layer and have an exclusive software engineering perspective.

The original contribution of this paper is that we extend the intuitive business service patterns presented in [8] to facilitate modeling the business services at a conceptual business level, by proposing a model-driven method of using these patterns in service integration. Business service patterns are represented using our previous work [4], BSRM (business service and resource modeling) language, which is based on the REA business ontology [9].

This paper is structured as follows: Section 2 provides a background which describes the BSRM ontology – the basis of developing patterns. Section 3, consists of four parts. Part one and two describe the generic business service patterns for service outsourcing and the sub-services. Part three describes the pattern composition using operators and section three ends with providing design steps. Service integration is explained in the section 4. The conclusion is given in the section 5.

2 Background

Using an MDA approach, we have introduced a new business service and resource modeling language - BSRM based on the Resource-Event-Agent (REA) business ontology in our previous work [4]. BSRM is capable to design the business activities in a company with service perspective at CIM level using simple modeling notation. The constructs of the BSRM language and their relationships are grounded in a meta-model (Fig. 1) which provides comprehensive specification using UML notation. We distinguish two service specializations: exchange service and conversion service, corresponding to the two basic REA dualities. Each of them corresponds to a group of decrement and increment economic events in REA.

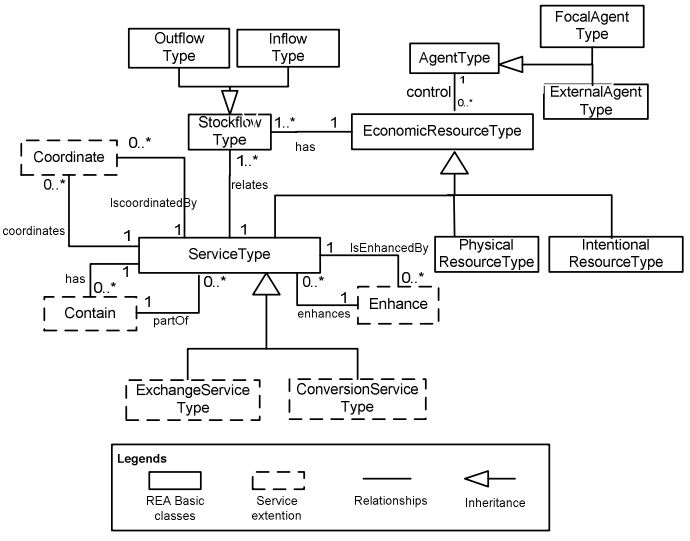

Fig. 1. BSRM meta-model

Based on the service classification model [6], we identified two different service roles. The concept of *enhancing* service which adds value to the any other service is introduced as one of them. Considering the situation where core-service realization involves multiple value activities and it makes sense to view these value activities as independent services that are shared by different contexts, we identified the next category of services as *sub-services*. A distinction is made between *core sub-services* and *coordination services*. The former are used in the realization of the composite service by manipulating physical resources and the later are used in the realization of the composite service, and coordination of the core sub-services, by manipulating intentional resources. The relationship between economic resource type and service type is defined as REA stockflow, further specialized as inflow and outflow. Following REA, we include the agent concept to the meta-model. An agent is an individual or organization capable of having control over economic resources. Agent type is classified into two, namely focal and external. The focal agent type is the individual or the central organization who intends to view its business with service perspective. External agent type represents the outside agents who are involved in the service value chain. The relationship between the economic resources and agent is "control". Note that agents are not explicitly represented in the BSRM model. A BSRM model always takes the perspective of a particular agent. The design decision to deemphasize the agents/owners is in line with the SD-logic approach in which co-creation is more important than ownership.

We used a simple modeling notation for BSRM. A summary of the modeling notation is given in the table 1.

Table 1. Summary of the modeling notation

Services are denoted as rounded rectangle (Conversion services are colored/filled rounded rectangles and exchange services are not filled with a color.)	
Physical resources are denoted as rectangles	
Intentional resource are denoted as dashed rectangles	
Part of Relationship Co-subservices are denoted as Coordination relationship is denoted as	 +
Enhancing relationship is denoted as	+
Stockflow relationship Inflow Outflow	

3 Business Service Patterns (BSP) and Composition

The purpose of using business service patterns is twofold. Firstly, the reuse of patterns reduces the designing time. Secondly, it assures that the designer does not violate the domain concepts. We introduced generic business service patterns for exchange and conversion processes in [8]. In this section, we present just two generic service patterns - service outsourcing and the service which has multiple sub-services. These patterns are modeled with BSRM notation and from the perspective of the focal agent.

3.1 Outsourcing a Service

Description:

Companies are outsourcing services to another party due to several reasons. Some of them are lack of own resources, lack of competencies and expertise knowledge and when the outsourcing is cost effective.

Pattern:

The business service pattern for service outsourcing is shown in Fig. 2. Outsourcing is based on the exchange pattern. The company has to pay a fee for the external agent to get their service. Hence there exist *give* and *take* relationships between *ServiceOutsourcingExchange* and the *money* and the service, respectively. The service which is outsourced has stock inflow and outflow relationships with resources. In particular, it *uses* resources (resource 1) and it *produces* another resource (resource 2) - the meaning of "produce" can be actual production or adding value.

Parameters:

s: ServiceType (the outsources service), FA, EA: agent (focal agent, external agent)

Constraints:

- At least one resource outflow of the service belongs to the focal agent. This ensures that the service has value for the focal agent.
  ∃r: ResourceType [produce(s,r) ∧ control(FA,r)]
- At least one resource inflow belongs to the external agent. This ensures that the external agent really contributes something.
  ∃r: ResourceType [use(s,r) ∧ control(EA,r)]

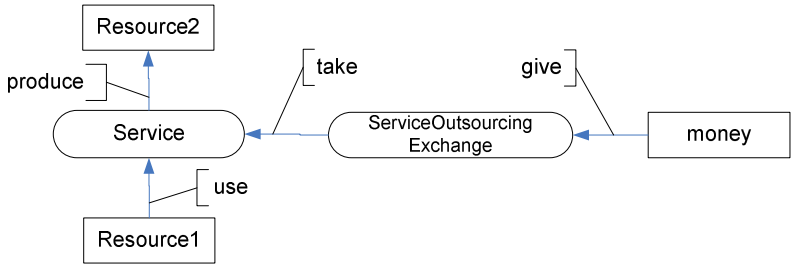

Fig. 2. Service Outsourcing pattern

3.2 Sub-service Pattern

Description:

Sub services are coming into play when the core service has to be realized by multiple activities. For example *produce* is a core service and it has *assemble*, and *inspection* sub-activities. When decomposing the composite service, it is important to analyze the relationship between resources with the sub-activities and their constraints.

Pattern:

The business service pattern for sub-service is shown in Fig. 3. Sub-services have a part-of relationship with the composite service.

Parameters:

s: ServiceType (the composite service), *s1* and *s2* are sub services

Constraints:

- At least two sub services are to be defined.
  ∃s1,s2: ServiceType [part-of(s1,s) ∧ part-of(s2,s) ∧ s1≠s2]
  Note:
  Number of sub-services can be more than 2. Following the pattern expansion method in [10], number of sub-services can be increased. In the pattern expansion, the model has a fixed part and a variable part; we consider the sub services and its related resources as the variable part.
- Intermediate outflow resources have to be consumed by another sub-service.
  ∀r: ResourceType [[∃s1: part-of(s1,s) ∧ outflow(s1,r)] ⇒ outflow(s,r) ∨ [∃s2: part-of(s2,s) ∧ inflow(s2,r)]]

- Intermediate inflow resources have to be produced by another sub-service. ∀r: ResourceType [[∃s1: part-of(s1,s) ∧ inflow(s1,r)] ⇒ inflow(s,r) ∨ [∃s2: part-of(s2,s) ∧ outflow(s2,r)]]
- Inflow resources must be consumed by a sub-service, outflow services must be produced by a sub-service.

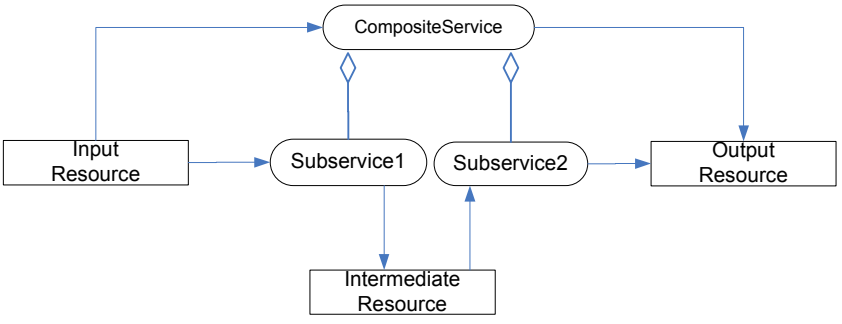

Fig. 3. Sub-service pattern

3.3 Pattern Composition with Business Pattern Operators (BPO)

The enterprise model is a composition of BSPs that always starts from a generic enterprise model (for trading company, for manufacturing etc). BSPs can be combined using business pattern operators each time the enterprise model is expanded. This section describes the operators used in business service pattern composition. A pattern can be viewed as a graph consisting of nodes and edges. In this section we present several operators to merge and expand patterns. The term "pattern" is used for both the basic patterns given in the library and the compositions of patterns. We follow the definitions of [10] for pattern composition, annotating and expansion which originate from the category theory.

1. Merge Operator

The merge operator enables to combine two patterns and compose a new pattern. We follow the definition b-4 in [10] in which the composition is described using the technique "pushout" by gluing the two objects along a common sub object. The following acronyms are used for easy representation of the merge operation.
BSP1, *BSP2*: business pattern 1 and 2 respectively;
K: common object to BSP1 and BSP2;
Morphisms m1, m2 to BSP1 and BSP2: BSP1 $\overset{m1}{\leftarrow} K \overset{m2}{\rightarrow}$ BSP2 respectively.
The possible candidates for the common object can be a *physical resource*, an *intentional resource,* a *conversion/ exchange service,* a *coordination service* or an *enhancing service*.

The following example (Fig.4) illustrates the merge operation. Consider the situation where we need to outsource the delivery service when self delivery is not possible.

Pattern (a) in Fig.4 represents the business service pattern of delivery service. Delivery is a service used in the sales and it adds value to the product. It uses forklift and truck 1 as physical resources. Pattern (b) in Fig.4 shows the delivery outsourcing pattern. It is an exchange service which takes the delivery service and gives money. Delivery service in the outsourcing pattern uses truck 2. Merging product delivery with delivery outsourcing is done through the common object "delivery service". We build the constraints of the new model as a union of constraints which relate to the common object.

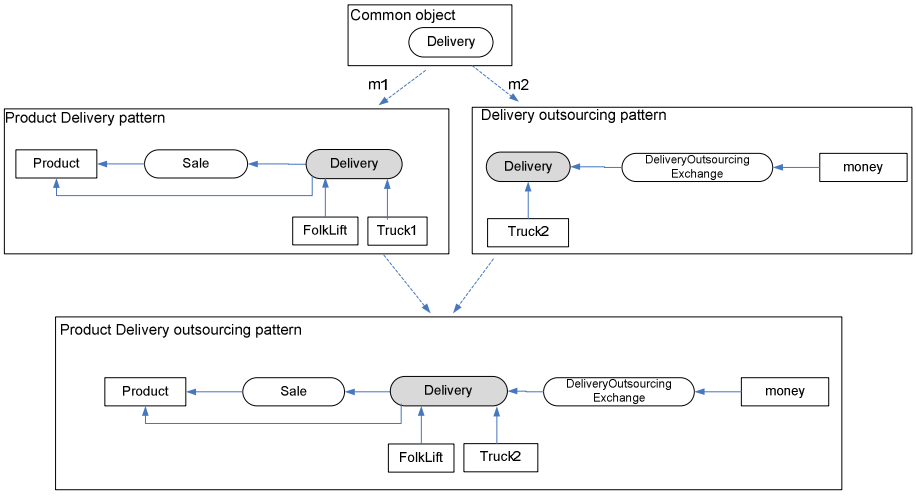

Fig. 4. Example for merge operator

Constraint of delivery service s in product delivery pattern (in fact, this constraint holds for *any* service type):

$\exists r1,r2$: ResourceType [inflow(s,r1) $\wedge$ outflow(s,r2) $\wedge$ r1$\neq$r2]

Constraint of delivery service s in delivery outsourcing pattern (cf. 3.2):

$\exists r$: ResourceType [produce(s,r) $\wedge$ control(FA,r)]

$\exists r$: ResourceType [use(s,r) $\wedge$ control(EA,r)]

These constraints are not inconsistent. The union can be written as:

$\exists r1,r2$: ResourceType [inflow(s,r1) $\wedge$ control(FA,r) $\wedge$ outflow(s,r2) $\wedge$ control(EA,r) $\wedge$ r1$\neq$r2]

In the result, the delivery service has one inflow resource from the external agent (truck 2) and one outflow resource of the focal agent (product).

2. Decompose operator

This operator is used when the service is realized by multiple activities (sub-services) that have to be distinguished, for instance, because they involve different actors and responsibilities, or because it allows for more flexible sourcing of the sub-services. Decomposition has to be defined in three steps.

Step 1: Pattern Expansion (if there are more than 2 sub-services)
In the first step, the sub- service pattern has to be expanded if there are more than 2 sub-service. Pattern is expanded as described in section 3.2 which is based on the pattern expansion definition [10].

Step 2: Derive the domain specific model
The second step is deriving the domain specific model by pattern annotation, again based on the definition in [10]. They use pattern annotation to represent the model with specialize vocabulary in a specific domain. Using a triple graph which consists of the source graph -domain specific pattern, target graph- vocabulary pattern and the correspondence graph- source and the target relates through morphisms, the specific model is derived. Here we use an example of bike producing which has assemble and inspection sub activities. According the constraints in sub-service pattern we do not need expansion to the generic sub- service structure because it has 2 sub- services. By adapting vocabulary pattern which consists of the names of specific domain- bike producing, we derive the decompose domain specific model for bike producing (Fig.5(b)).

Step 3: Merge the composite pattern with decomposed pattern
The third step is merging the decomposed pattern with the composite pattern as described in 'merge operator' section. Fig. 5 shows the decomposition. Producing a bicycle is a composite service and it can be realized through assemble and inspection sub services.

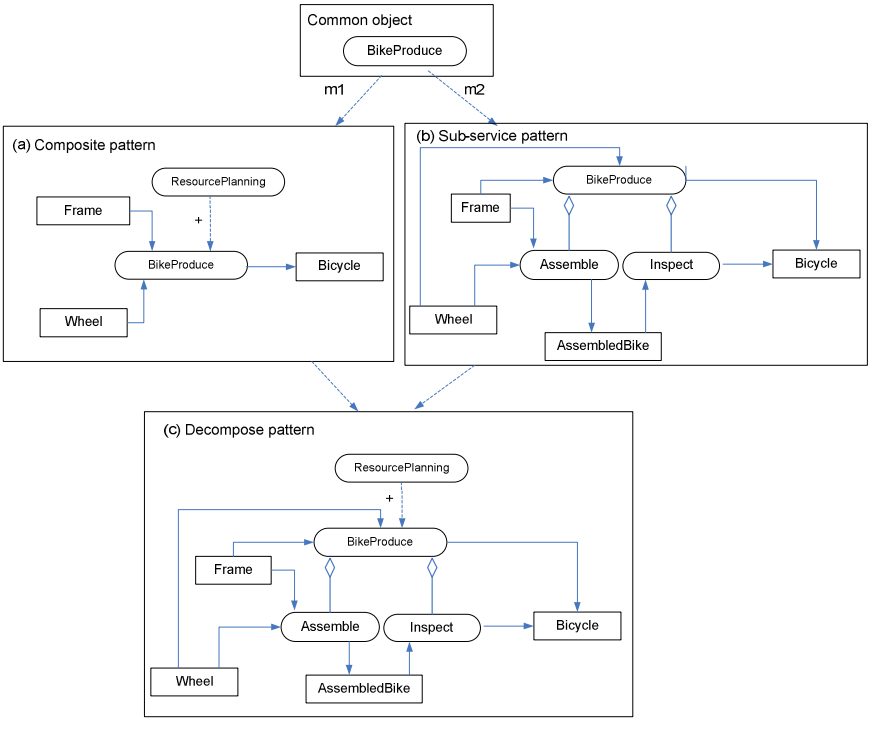

Fig. 5. Decomposition

Constraints for composite service: (*s*: ServiceType)

$\exists$r1,r2: ResourceType [inflow(s,r1) $\wedge$ outflow(s,r2) $\wedge$ r1$\neq$r2]

Constraints for sub-service pattern:

Inflow resources must be consumed by a sub-service, outflow services must be produced by a sub-service

$\exists$r1,r2: ResourceType [inflow(s,r1) $\wedge$ outflow(s,r2) $\wedge$ r1$\neq$r2] $\wedge$[inflow(s1,r1) $\wedge$ outflow(s2,r2) $\wedge$ r1$\neq$r2]

All the other constraints related the sub-service pattern remain unchanged in the decomposition model.

The union of above two constraint can be written as:

$\exists$r1,r2: ResourceType [inflow(s,r1) $\wedge$ outflow(s,r2) $\wedge$ r1$\neq$r2] $\wedge$[inflow(s1,r1) $\wedge$ outflow(s2,r2) $\wedge$ r1$\neq$r2]

3. Specialization operator

The specialize operator is used when a service is further qualified with a specific domain. It is also achieved in two steps. First the pattern has to be expanded with the number of specializations (if there is more than one). Then, using pattern annotation, the domain specific model has to be derived. For example, *bike produce* is a specialization of the *product produce* service.

Constraint for specialization of a service

- At least one specialization is to be defined.

$\exists$s1: ServiceType [specialize (s1,s) $\wedge$ s$\neq$s1]

Constraint for specialization of resources

$\forall$r: ResourceType[inflow(r,s)] $\Rightarrow$

$\exists$ r1: ResourceType inflow(r1,s1) $\wedge$ subtype(r1,r2)

$\forall$r: ResourceType[outflow(r,s)] $\Rightarrow$

$\exists$r2: ResourceType outflow(r2,s1) $\wedge$ subtype(r1,r2)

3.4 Design Steps for Enterprise Information Systems

Given the service patterns and pattern operators, a service design method can be developed. The following activities are intended to describe the design steps to achieve such model. All the steps after the first step are not strictly ordered.

Step 1: ϕ $\rightarrow$ enterprise model

Step 2: decomposition: $\{s_0\} \rightarrow \{s_0, s_1, \ldots s_n\}$, where part-of($s_i$,s_0)

Step 3: coordination: $\{s_{1,\ldots,} s_n\} \rightarrow \{ s_0, s_{1,\ldots,} s_n, c_i,\}$, where part-of($c_{i,}$ s_0), coordinate($c_{i,}$ s_0)

Step 4: extension: $\{s_{1,} s_2\} \rightarrow$ merge($s_{1,} s_2$)

Step 5: enhancing: $\{s\} \rightarrow \{s,e\}$, where enhance(s,e)

Step 6: specialization

Step 7: outsource, reengineer

The design starts top-down with the generic enterprise model, to ensure the completeness. In the second step, identify specific services by decomposing the enterprise model. This is done using domain models. Once multiple sub- services exist, they have to be coordinated. Hence third step introduces the coordination service. Step 4 extends a certain model. For instance, there may be a manufacturing service pattern that generates waste and a waste management pattern, then the first one can be extended by merging it with the second. Step 5 is adding enhancing services, such as management services, to other services where necessary. Note that the enhancing services can be decomposed, enhanced, etc. as well. Step 6 does specialization, for instance, from bike to race bike. Step 7 specifically supports model evolution. Model evolution includes outsourcing services, but also reengineering in the form of the reversal of any step 2-6. Our aim is to support meaningful model evolution steps, not arbitrary deletions and insertions.

4 Service Integration

The role of BSP is not limited to design business services. It can also be used in the discovery of the services in a service marketplace or service library. In this section we provide a meta-model for service integration and an example to illustrate the concept.

4.1 Meta-model for Service Integration

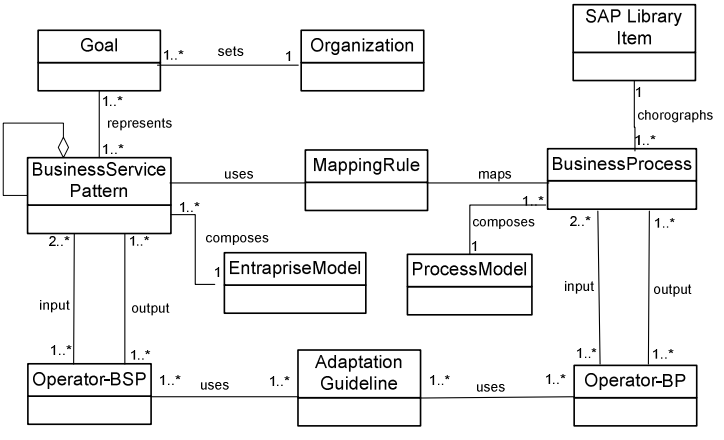

Fig. 6. Framework for pattern-based service integration

Fig. 6 shows the meta-model for composing the enterprise model and how these patterns are related in the service discovery. As soft goals (extra-functional requirements) should be optimized, service discovery should address these goals. Hence these goals are represented in the BSPs: some BSP may prioritize efficiency, another customer-intimacy. Between the BPSs and the business processes a mapping exists, as discussed in our previous work [4]. When using the BSPs in a service

marketplace setting, these mappings may also be defined manually. As explained above, BSPs can be combined using algebraic operators, including first of all "merge". Each time the enterprise model is transformed, using a BSP and BSP operator, the corresponding BPs are integrated as well, so that when finishing the enterprise model, the designer also has an integrated BP (to be more precise: a set of possible BPs as we assume that the enterprise model still leaves room for different process implementations). The adaptation guide lines describe the homeomorphism between BSPs operators and BP operators, including conditions and their use and pragmatic guidelines to the designer.

4.2 Example

The soft goal of customer intimacy can be optimized by delivering the product to the customer at right time. This can be achieved by having an option of outsourcing the delivery service when self delivery is not possible due to lack of resources. This requirement can be implemented by merging the basic "Product Delivery" pattern [Fig. 4(a)] with "Delivery Outsourcing" pattern [Fig. 4(b)]. These two patterns are combined using a merge operator with the common node e.g. delivery. The composite pattern is shown in Fig. 4(c).

The corresponding change at the business process side can be viewed as creating a relationship by means of a message flows ("delivery request") from the company to the external agent. It is the operator in business process side. Adaptation guidelines:

Rule 1:
For every merge operator in the BSP, there are one or more message flows in the BP.

Conditions:
If the merge operator is based on a service:
- If the service exists internally, the message flows run in between swim lanes of same pool.
- If the service belongs to another party, the message flows run in between swim lanes of different pools.

In the case that the services themselves can be found in the library, or an external market registry, the guidelines provide the requirements to glue them together in coherent processes.

5 Conclusion

In this research work we have presented an efficient modeling method for business services using service patterns. As these patterns are based on the well-established business ontology REA, it provides a truly service-oriented modeling step at the conceptual level. The design is further facilitated by providing a systematic way of using these patterns for pattern composition by means of operators and model completion using design steps. The role of business service patterns is not limited to the design of business services, but does also support service integration at

implementation level. The idea of transformation rules aligning the conceptual and PIM/PSM level is core to the MDA approach and not new, but the pattern-based approach extends this idea and makes it more realistic. In the future, we plan to evaluate the proposed method with a more general case and by means of further formal analysis.

References

1. Allgaier, M., Heller, M., Weidner, M.: Towards a Model-based Service Integration Framework for Extensible Enterprise Systems. In: Enterprise Resource Planning und Transformation von ERP-Systemen 1523, MKWI (2010)
2. Cummins, F.A.: BPM meets SOA. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management, vol. 1. Springer, Berlin (2010)
3. Roy, M., Suleiman, B., Weber, I.: Facilitating Enterprise Service Discovery for Non-technical Business Users. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 100–110. Springer, Heidelberg (2011)
4. Jayasinghe Arachchige, J., Weigand, H., Jeusfeld, M.: Business Service Modeling for the Service-Oriented Enterprise. International Journal of Information System Modeling and Design 3(1) (2012)
5. Zdravkovic, J., Ilayperuma, T.: A Model-driven Approach for Designing E-Services Using Business Ontological Frameworks. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 121–130 (2010)
6. Weigand, H., Johannesson, P., Andersson, B., Bergholtz, M.: Value-Based Service Modeling and Design: Toward a Unified View of Services. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 410–424. Springer, Heidelberg (2009)
7. De Castro, V., Marcos, E., Wieringa, R.: Towards a service-oriented MDA-based approach to the alignment of business processes with IT-systems: from the business model to a web service composition model. International Journal of Cooperative Information Systems 18(2), 225–260 (2009)
8. Jayasinghe Arachchige, J., Weigand, H.: A Pattern based Approach for Business Service Integration. In: Proc. 6th Int. Workshop on Value Modeling and Business Ontology, VMBO 2012 (2012)
9. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review, 554–575 (1982)
10. Bottoni, P., Guerra, E., de Lara, J.: A language-independent and formal approach to pattern-based modelling with support for composition and analysis. Int. Journal Information and Software Technology 52(8) (2010)