A Model-driven Method to Design SoaML Services from BPMN Models: Principles, Proof-of-concept, and Validation

Redouane Blal
LATECE Research Center
Université du Québec à Montréal
blal.redouane@courrier.uqam.ca

Abderrahmane Leshob
LATECE Research Center
Université du Québec à Montréal
leshob.abderrahmane@uqam.ca

Imen Benzarti
École de technologie supérieure
imen.benzarti@etsmtl.ca

Hafedh Mili
LATECE Research Center
Université du Québec à Montréal
mili.hafedh@uqam.ca

Omar K Hussain
University of New South Wales
o.hussain@adfa.edu.au

Abstract

Today's business processes are increasingly complex as they cross organizational boundaries. To execute their business processes, organizations develop software applications called Process-Aware Information System (PAIS). PAIS designers must consider complex scenarios involving multiple partners. Consequently, the architectural design of high quality PAIS is complex and requires vast amounts of knowledge and skills both in software architecture and in the business domain. This paper proposes a model-driven method to design the architecture of PAIS using the service-oriented architecture (SOA) style. The proposed method generates SOA-based design models expressed in SoaML from the specifications of collaborative business processes expressed in BPMN. We developed a prototype tool using the Eclipse Modeling Framework (EMF) ecosystem. We tested the method on a set of processes from the Enterprise Resource Planning literature to assess its effectiveness. Our results show that 80.95% of the identified services were relevant and corresponded to what architecture specialists expected.

Keywords: Service-oriented architecture, Business process, Business pattern, Model transformation

1. Introduction

Modern organizations are process-oriented. They rely on improving their processes to gain a competitive advantage. Some business processes cross the boundaries of organizations by involving multiple partners, which increases their complexity. We call these processes *Collaborative Business Processes* (CBP).

van der Aalst (2009) refers to information systems built by organizations to manage and execute CBPs as *Process-Aware Information System* (PAIS). *Business Process Management Systems* (BPMS) are examples of PAISs. Designing PAISs is complex and requires designers with advanced technical skills and an extensive experience as one must take into account complex scenarios that involve multiple partners. One common approach to design PAISs is to use the Service-Oriented Architecture (SOA). For Laskey and Laskey (2009), SOA is a paradigm for organizing and packaging clusters of functionality as services that are available via their interfaces, to solve business problems.

A number of studies have tried to design SOA-based PAISs from CBPs models. Some were limited to service *identification*, without addressing the specification of these services (Bianchini et al., 2014; Azevedo et al., 2013). Others have tackled the latter (Delgado et al., 2018; Nikaj et al., 2019; Daghaghzadeh and Babamir, 2021), yet generated specifications were too generic. Further, most existing approaches suffer from a number of limitations including complexity and usability.

This paper proposes an end-to-end method to identify and specify SOA services from CBP models specified with the *Business Process Model and Notation* (BPMN) (OMG, 2011). BPMN is the de facto standard for modeling business processes (OMG, 2011). To specify SOA services, our method relies on *business patterns* detection, and uses the *Service-oriented architecture Modeling Language* (SoaML) (OMG, 2009); SoaML is an OMG specification that extends UML and provides a profile to specify services (OMG, 2009).

This paper extends the work presented in (Leshob et al., 2019) by, 1) refining and improving the method to design adequate set of services, 2) presenting the design of a proof-of-concept implementation, and 3) presenting the results of experiments to validate the effectiveness of

URI: https://hdl.handle.net/10125/103339
978-0-9981331-6-4

the method.

The remainder of this paper is organized as follows. Section 2 surveys the related work. Section 3 describes the approach to identify and specify SOA-based services from CBPs models. Section 4 presents the prototype implementation. Section 5 presents the validation of the method regarding three aspects: 1) the pertinence of the identified services, 2) the correctness of the SoaML specification and 3) the effectiveness of the services to automate the CBPs. We conclude in Section 6.

2. Related Work

A number of studies proposed methods, with different levels of automation, to assist software designers with services identification and specification from process models. Azevedo et al. (2013) proposed a method that uses a set of heuristics to derive and classify services from an EPC (Event Process Chain) process model. The syntactic and semantic analysis of the process activities generates a list of candidate services. The list is then consolidated and prioritized. Finally, the services are optimized based on their granularity. Bianchini et al. (2014) proposed a three-phase semi-automatic approach. The first phase builds an initial set of services by analyzing the process from two perspectives, a value analysis to identify exchanged values and a task dependency analysis to identify the data and the flow dependencies between the tasks. The second phase refines the service list based on an evaluation of their overall cohesion and coupling metrics. The last phase eliminates redundant services when similarities between their descriptors are detected. The above methods are limited to service identification as they don't elaborate on how the services are specified.

Other research work proposed methods to identify and specify services from CBP models. Gonzalez-Huerta et al. (2017) presented a three-step method to derive software analysis and design artifacts from BPMN models. First, the as-is BPMN model is refined by detecting missing elements and inconsistencies. Second, a to-be process model is generated by applying re-engineering patterns to the refined BPMN models. The resulting to-be BPMN model is used to derive software models. Nikaj et al. (2019) presented a semi-automatic approach to generate RESTful choreographies from BPMN choreography models. To derive *REST Verbs (GET, POST, PUT, DELETE)* for each Web service, the method uses natural language analysis to derive interactions from the textual information within the process models. The outcome of this approach depends heavily on the accuracy of synonyms. Ouyang et al. (2009) proposed an approach to derive BPEL definitions from BPMN models. The approach executes BPMN-to-BPEL translations using mappings between pattern blocks of BPMN model elements and block-structured BPEL processes. Zafar et al. (2019) proposed an MDA approach to identify and elaborate Web services from BPMN process models. The method identifies and specifies Web services using model-to-model transformation rules from BMPN XMI to SoaML. It then generates executable code using model-to-text transformation from the SoaML models. Delgado et al. (2018) proposed an MDA approach to derive SOA services from collaborative BPMN models. The method uses a set of one-to-one mappings between BPMN model elements and SoaML constructs to derive a service model. We argue that such mappings yield models that are too generic and that lack business semantics; by contrast, our method uses a business pattern detection approach to accurately specify identified services. Daghaghzadeh and Babamir (2021) proposed a model-driven approach to identify services from business process models. The approach builds a service portfolio by clustering and partitioning the business process activities. Then it optimizes the services based on cohesion, coupling, and granularity metrics. Existing approaches were able to identify services and generate their specification from process models. However, they show a number of limitations related to their complexity and usability. This work proposes an end-to-end and a usable method to design SOA-based PAISs from process models using SoaML.

3. Services Identification and Specification from Collaborative BPMN Models

Our goal is to provide organizations with a method and tools to design SOA-based PAISs. Figure 1 illustrates the proposed four-step method to identify and specify services from CBP models. The first step derives a choreography process model. The second step identifies the services using the choreography tasks of the choreography process model. The third step links the identified services to business patterns. The last step consolidates the set of services by removing duplicates and then specifies them with SoaML according to the associated pattern specifications.

The method is illustrated with the B2B collaborative process shown in Figure 2. Let us consider an *Insurance Company (IC)* that needs to acquire a Robotic Process Automation (RPA) software to automate its processes. IC publishes a *Request For Quote* (RFQ) intended for RPA Software Providers. The providers prepare and send back their *Quotes*. IC analyzes the Quotes,

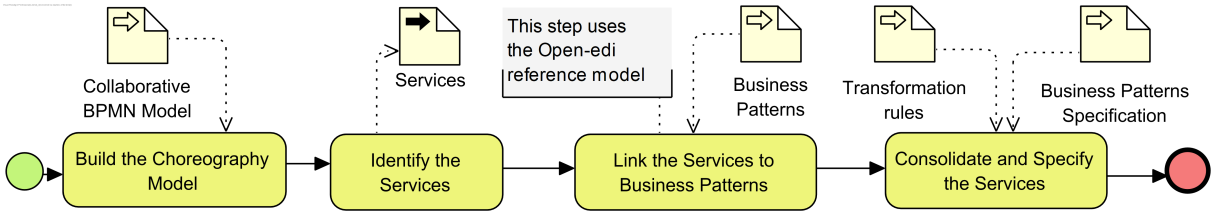

Figure 1. Proposed method for services identification and specification from BPMN models.

selects compliant providers, and invites them to perform a demonstration. Then, a demonstration session is scheduled for selected providers. IC proceeds with a final provider selection to identify the best offer then sends a *Purchase Order* (PO). The selected provider sends PO reception *Confirmation* then an *Invoice*. IC pays the invoice via a financial institution (FI) by sending a *Payment Order*. FI sends the *Payment* to the provider and a *Confirmation* to IC. When the Payment is received, IC is granted access to the RPA tool.

3.1. Step 1: Producing the choreography model

A choreography model is a flow of choreography tasks representing activities (tasks or a sub-processes) from the original BPMN model that have at least one message exchange between partners. When there is more than one message, the initiating message is marked and the return messages are shaded with a light fill (OMG, 2011). Choreography models are BPMN models that show how partners coordinate their interactions. In contrast with orchestration models that focus on activities performed by each of the partners, choreography models focus on the exchanged messages between them (OMG, 2011).

To build the choreography model, we track messages exchanged between the partners. The orchestration of the choreography tasks corresponds to the sequence flow defined in the BPMN model. Figure 3 shows the choreography model obtained from the BPMN model of the running example (see Figure 2). For example, *Process RPA Tool Quote* is a choreography task that handles two messages exchanged between IC and the provider. IC (the initiator) sends the *RPA Requirements Specification* message (initiating message). The RPA provider responds with *Quote* message. *Quote* is shaded to indicate the return message. The partner (IC) who initiates the choreography task is not shaded.

3.2. Step 2: Identification of services

According to OMG, a service is a value delivered to a partner, through an interface, by another partner in exchange for some other value. One option to identify services from a CBP models consists of extracting one service per message. Such a design decision identifies too many services with granular responsibilities, which leads to an anti-pattern called *tiny services* (see Palma et al., 2019). Instead, a service is identified for each *choreography task*, considering that each choreography task represents a set of one or more exchanged messages (One-Way or Callbacks). In the running example, the method identifies six services, each of which automates the corresponding choreography task regardless of the number of exchanged messages.

Table 1. identified services.

Service	Patterns
Quote Management Service	Contract/Commitment
Demo Schedule Service	Schedule
Ordering Service	Contract/Commitment
Billing Service	Claim/Claim Materialization
Payment Service	Exchange/Claim
Access Manager Service	Exchange

3.3. Step 3: Mapping business patterns to services

This step links identified services to business patterns[1] using the Open-EDI reference model (ISO, 2011). This model describes a business transaction as a set of five phases: *planning, identification, negotiation, actualization*, and *post-actualization*. During the *planning* phase, participants choose actions and resources to be yielded or acquired during the collaboration. This phase ends when the consumer sends a formal request for a quote. During the *identification* phase, partners are selected in order to establish a one-to-one link. Activities of the negotiation phase lead to an explicit agreement between the partners. During the *actualization* phase, partners exchange resources. When the resource exchange is completed, the *post-actualization* phase starts. To

[1] A business pattern is a reusable solution for a recurrent business scenario/problem.

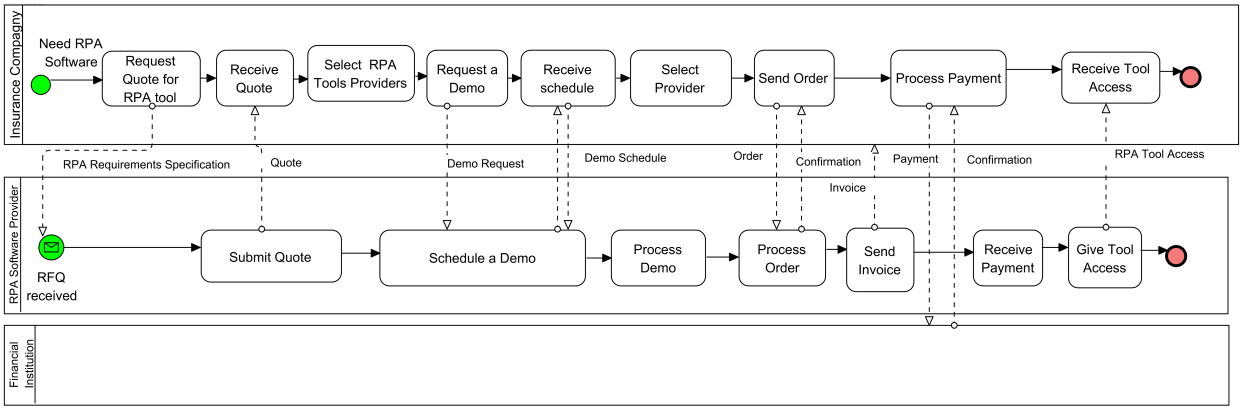

Figure 2. Collaborative procurement process.

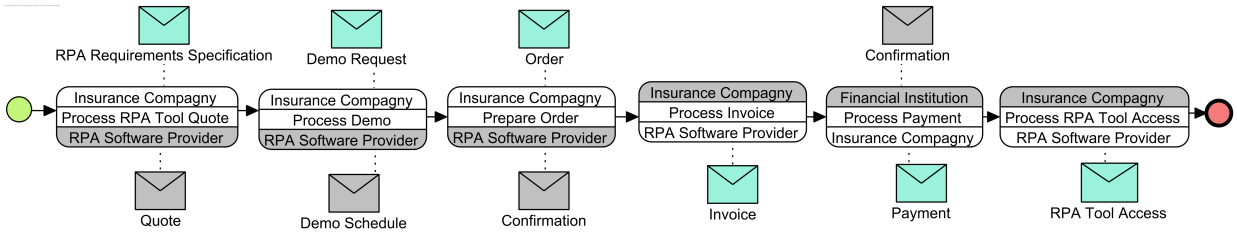

Figure 3. Choreography diagram obtained from the running example.

identify the boundaries of the Open-EDI phases, we use a semi-automated approach that asks the user (e.g. a business analyst) a minimal set of questions (Leshob et al., 2019). To map a service to a business pattern, the underlying choreography task is mapped to a business pattern from (Kartseva et al., 2009; Hruby, 2006). For example, services supporting choreography tasks that exchange resources (e.g. *Cash for RPA tool access*) during the actualization phase, are mapped to the *exchange* pattern. Services supporting the creation of a contract (e.g. *Prepare Order Service*) between the partners during the negotiation phase, use the *commitment/contract* pattern. For the remaining services, we rely on user annotations. For example, we ask the user to annotate the choreography task that manages the claim[2] during the actualization phase; then we map the service that supports the task to the *claim* pattern (see Hruby, 2006). Services that support the choreography tasks *'Process Quote*' and *'Process Demo',* performed to establish a one-to-one link between IC and a solution provider during the identification phase, are mapped respectively to the *commitment/contract* and *schedule* patterns. Figure 4 illustrates Open-EDI phases and business pattern annotations in the choreography model of the example.

[2] A claim occurs when the exchange of the resources does not occur simultaneously. This temporary imbalance results in a claim (e.g. invoice, credit note).

If some tasks are left without a link to a business pattern, we use a generic pattern in order to define the service CRUD operations (REST verbs) based on the message object. Table 1 shows the identified services and their corresponding business patterns.

3.4. Step 4: Consolidation and Specification of services

This step starts by removing duplicate services. Two services are considered duplicated if: 1) they share the same messages, 2) they are mapped to the same patterns, and 3) they are both one-way or bi-directional. Next, consolidated services are specified with SoaML (OMG, 2009). There are three design approaches to specify services with SoaML: 1) *Simple Interface,* which supports the design of unidirectional services, 2) *Service Interface,* which supports bidirectional services (i.e. services with callbacks from the provider to the consumer) by specifying the interface offered by the provider and the expected interface from the consumer, and 3) *Service Contract*, which specifies how business partners collaborate to exchange resources. The last approach is not yet supported by our method. Services are specified using their mapped business patterns. More precisely, the method specifies the services by applying transformation rules according to the specification of their corresponding *business patterns*. Figure 7 shows

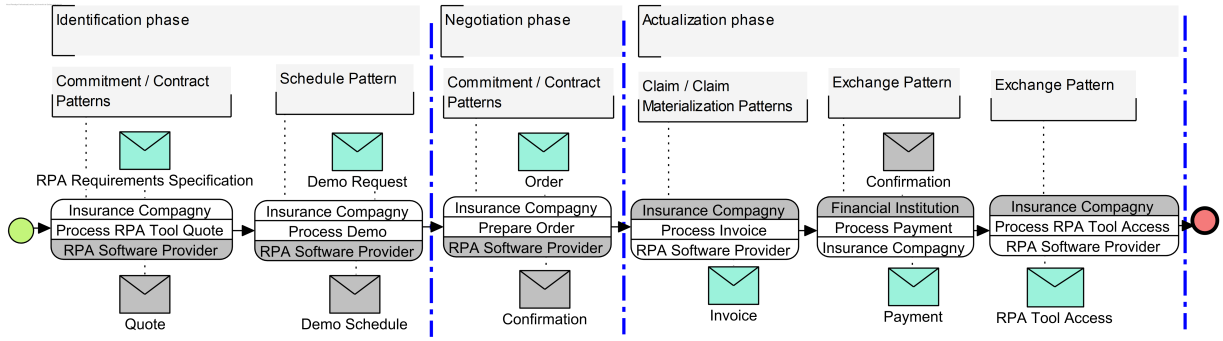

Figure 4. Choreography model task annotations.

the transformation rule that specifies the *Ordering Service,* which is mapped to the *Contract pattern* (see Figure 5 [3]). Figure 6 illustrates the generated SoaML specification of the *Ordering Service* from the running example. The classifiers *Order Receiver* and *Order Requester* represent the interfaces for the RPA software provider and IC, respectively.

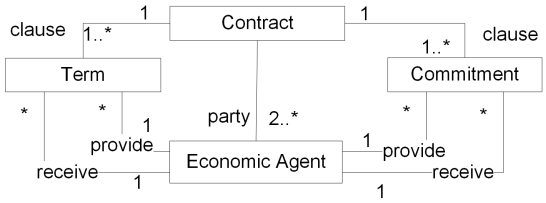

Figure 5. Contract business pattern (adapted from (Hruby, 2006)).

4. Prototype Implementation

The first sketch of the prototype that supports the proposed method is implemented as a plugin within the Eclipse Modeling Framework™ (EMF). EMF is a Java-based modeling framework that implements EMOF (Essential Meta-Object Facility). As for now, the prototype is based on a set of unit tests. It uses rule-based transformations to identify and specify services. It is implemented with the *Red Hat Drools*, an open-source business rules management system (BRMS). Each rule manipulates BPMN and SoaML objects using two parts: *a when* part and a *then* part. The *when* part matches model elements, such as business patterns and choreography task objects. The *then* part builds SoaML models. The transformation process starts by initializing the Drools facts base with an annotated choreography model. To identify and specify the services, the transformation process uses the approach presented in Section 3.

The prototype extracts a service for each choreography task. For each extracted service, it sets its context with: 1) the corresponding business pattern, 2) the type of its SoaML specification (unidirectional or callback), based on the type of the choreography task (i.e. one-way or two-way), and 3) the JSON representation of the exchanged messages (i.e., resources). To specify the services, we developed a transformation rule per business pattern per type of choreography task. Each rule specifies the SoaML service (e.g., *provided interfaces, required interfaces, roles, and behavior*) using its context. For example, the transformation that specifies the service that supports a two-way choreography task of the negotiation phase is based on the *contract pattern*. Since it's a two-way task, the transformation rule uses the *Service Interface* based approach of SoaML to allow *"callbacks"* from the provider of the service to the consumer. This rule is shown in pseudo-code format in Figure 7.

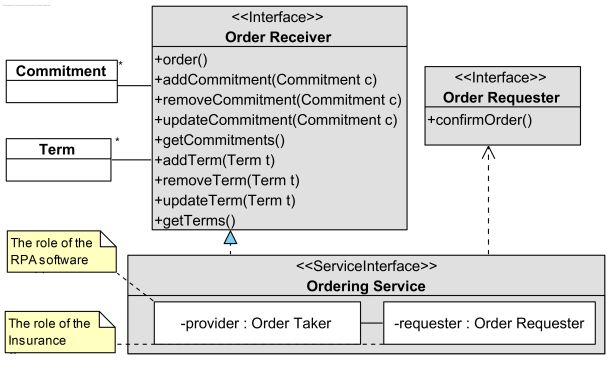

Figure 6. The SoaML Service Interface specification of the Ordering Service.

[3]For the sake of simplicity, the methods and the attributes of the pattern classes are hidden.

```
Rule "Bidirectional Contract"
When
$service: Service(context.pattern==Pattern.CONTRACT, context.type==Direction.TWO_WAY)
Then
// Specify the SoaML Service Interface
1. si <- Add an SoaML Service Interface class (Stereotype <<ServiceInterface>>) to $service
// Add the interface that the provider offers to the consumer
2. si.add (provided interface according to the Contract pattern)
// Add the interface required by the the provider
3. si.add (required interface according to the Contract pattern)
// Add the SoaML provider roles to the service interface (It defines the role of the provider in the service)
4. si.add(provider role)
// Add the SoaML consumer role to the service interface (It defines the role of the consumer in the service)
4. si.add(consumer role)
```

Figure 7. The specification rule of contract-agreement services.

5. Experimental Evaluation

An empirical study was conducted to evaluate the effectiveness of the method. This experiment was conducted following the Goal Question Metric (GQM) approach (Basili and Rombach, 1988). GQM defines 1) the goal of the experiment, 2) the set of questions used to characterize the way to attain the goal, and 3) the set of metrics used to answer the questions. Because the goal of this experiment is to assess the effectiveness of generated services, we needed participants with expertise that allows them to walk through the method.

5.1. Experts Selection

We used the purposeful sampling technique to select the participants (Seidman, 2019). We considered 15 senior professionals from various backgrounds and performed a short interview with each candidate. The criteria to select the participants included: i) the depth of the experience in solutions architecture; ii) the expertise with business process automation and SOA design (especially with SoaML), and iii) their availability. From the initial list, we recruited six experts. The following is a short description of their profiles:

- Expert 1 is a senior solutions architect at a major Canadian bank, with extensive experience in SOA and business process automation with SAP.
- Expert 2 is a senior enterprise architect in a leading IT consulting company with more than 20 years experience. He also worked as a business and software architect for many years.
- Expert 3 is a senior solutions architect with more than 15 years of experience in manufacturing and finance CBPs. He has a solid expertise in business process automation with the BPM approach.
- Expert 4 is a senior enterprise architect at an international regulation organization. He accumulates more than 30 years of experience in the IT and management field.
- Expert 5 is a senior solutions architect with a major consulting firm. He holds many years of experience in both software architecture and design with the SOA architectural style .
- Expert 6 is a senior enterprise architect in a major Canadian public organization. He has a strong experience in SOA and process automation.

Table 2. Selected CBPs.

Collaborative Business Process	Type
Buy a software (P1)	Procurement
Sales & Distribution (P2)	Sales and service management
Product Return (P3)	Supply chain management
B2B Financial Loan (P4)	Financial management

5.2. Experimental Processes

To carry out the experiment, we selected four common CBPs modeled using the BPMN language. Table 2 lists the selected business processes.

5.3. Experimental Design

The goal of this experiment is to evaluate the effectiveness of the proposed method from the point of view of the experts. We established three objectives:

1. Verify if the method generates the adequate set of services in the context of each CBP. For that, we assessed the *pertinence* of each service (*aspect 1*). A service is considered *pertinent* if, a) it is *relevant*, and b)there are no other services that provide the same functionalities through their provided interfaces or their composition.
2. Verify the *correctness* of SoaML specifications of the services (*aspect 2*). Thus, we asked the experts to assess whether i) the transformation rule uses the appropriate SoaML specification approach (i.e. Simple Interface or Service Interface) and 2) the SoaML specification is correct.
3. Validate whether the generated services *effectively support* the business collaboration between the partners (*aspect 3*). A service is considered effective if 1) its interfaces provide methods whose behaviors allow supporting the collaboration and 2) these methods have sufficient business semantics that are aligned with business objectives/goal of the collaboration.

Table 3. Identified SOA services.

CBP	Serv.	Soa Service	Business patterns
P1	S1	Quote Management	Contract/Commitment
	S2	Demo Schedule	Schedule
	S3	Purchase Order	Contract/Commitment
	S4	Billing	Claim/Claim Materialization
	S5	Payment	Exchange
	S6	Access Manager	Exchange
P2	S7	Quote Management	Contract/Commitment
	S8	Order Management	Contract/Commitment
	S9	Distribution Order	Contract/Commitment
	S10	Tracking	Identification/Notification
	S11	Shipping	Conversion/Location
	S12	Billing	Claim/Claim Materialization
	S13	Payment	Exchange
P3	S14	Return Request	CRUD/Notification
	S15	Return Order	Contract/Commitment
	S16	Distribution Order	Contract/Commitment
	S17	Shipping	Conversion/Location
	S18	Billing	Claim/Claim Materialization
P4	S19	Quote Management	Contract/Commitment
	S20	Contract Management	Exchange
	S21	Funds Transfer	Exchange

Therefore, we encoded a research question for each objective as follows.

1. RQ1: Are the services pertinent in the context of the selected CBPs?
2. RQ2: Does the method generate correct SoaML specification?
3. RQ3: Do the generated services effectively automate the business exchange?

The context of the experiment is determined by: i) the experimental CBPs, ii) the experts and iii) the identified services. During the evaluation, we presented the list and specifications of the generated services S1 through S21 to the participants (see Table 3). Then we asked them to answer the questions (RQ1, RQ2, RQ3) for each service in the context of their CBPs.

5.4. Experiment Operation and Execution

The experiment was performed during two different sessions of approximately 3 hours each. Sessions took place using separated Zoom virtual rooms. A first session was conducted with experts 1, 2 and 3. The second session was scheduled later with experts 4, 5 and 6. The materiel[4] used to conduct the experiment includes: 1) BPMN models of the experimental CBPs, 2) training slides with an overview of the method, 3) identified services with their specifications and associated patterns, and 4) the questionnaire for gathering the data.

During the sessions, the experiment conductor answered the questions raised by the participants. In addition, he joined the Zoom classrooms when requested. The experiment had the following hypotheses:

$H1i_0$: The service Si was not found pertinent/ $H1i_a = \neg H1i_0$; For i = 1 to 21

$H2i_0$: The SoaML specification of the service Si was found incorrect/ $H2i_a = \neg H2i_0$; For i = 1 to 21

$H3i_0$: The service Si does not effectively automate the business exchange/$H3i_a = \neg H3i_0$; For i = 1 to 21

Thus, the evaluation consists of three experiments. Each experiment validates one of the 3 hypotheses for the 21 identified services. Every participant was asked to answer 63 questions (3 X 21). Additional space was available to allow the experts to add comments in order to explain their answers.

- Evaluation of the pertinence of the services: The experts were asked to answer the question: *Is the service* Si *pertinent in the context of its CBP?*. The value 0 means that the architect finds the service not pertinent; the value 1 means that the architect finds the service pertinent.
- Evaluation of the correctness of the SoaML specification: We asked the question: *Does the method generate a correct SoaML specification of the* Si *service?* The value 0 means that the architect finds the SoaML specification of the service to be incorrect; the value 1 means that the architect finds the SoaML specification of the service to be correct.
- Evaluation of the effectiveness of the services: We asked the question: *Does the service* Si *effectively automate the business exchange?*. The value 0 means that the architect finds the service does not automate the business exchange; the value 1 means that the architect finds that the service does, indeed, automate the business exchange.

5.4.1. Analysis of the Results

Analysis of the pertinence of the services: Table 4 depicts the evaluation of the pertinence of each service analyzed considering the set of generated services for each process. D_i means that the architect found the

[4]The material is available at: https://tinyurl.com/ytd9yw6m

Table 4. Evaluation of service pertinence.

Experts	Services																				
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
Architect 1	D3	1	1	1	1	1	D8	1	1	1	1	1	1	1	1	1	1	1	D20	1	1
Architect 2	M3	1	1	1	1	1	M8	1	1	1	1	1	1	1	1	1	1	1	M20	1	1
Architect 3	D3	1	1	1	1	1	D8	1	1	1	1	1	1	1	1	1	1	1	D20	1	1
Architect 4	1	1	1	1	1	1	1	1	1	M11	1	1	1	1	1	1	1	1	M20	1	1
Architect 5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	M20	1	1
Architect 6	1	1	1	1	1	1	1	1	1	M11	1	1	1	1	1	1	1	1	M20	1	1

service is a duplicate of the service S_i. M_i means that this service should be merged with S_i.

Based on the results, we observe that in most cases architects agreed that the services were pertinent. They confirmed that 80.95% of the services were *pertinent*. These observations allowed us to reject the null hypotheses $H1i_0$ for $1 \leq i \leq 21$ and $i \notin \{1, 7, 10, 19\}$ and to accept their alternative hypotheses.

Further analysis of architect's answers to open questions revealed that architects 1 and 3 found that the *Quote Management services* (S1, S7 and S19) are duplicate services of the *Contract Management services* (S3, S8 and S20) respectively. They argued that the quotes have the same structure as contracts except that quotes have not been accepted yet by the partners. Architect 2 proposed merging services S1 and S7 with services S3 and S8 respectively. Architects 2, 4, 5 and 6 suggested merging S19 with S20. In addition, architects 4 and 6 suggest merging the *Tracking Service* (S10) and *Shipping Service* (S11).

Analysis of the correctness of the SoaML specification: Table 5 shows the assessments of the correctness of the SoaML specifications. The value 1 with gray background means that the architects perceived the SoaML specification as correct *but more generic* than what they expected. The value NA means that the architect *did not evaluate* that service specification.

A closer look at the results shows that the architects 1, 2 and 3 suggest that services designed to support the negotiation phase activities (i.e., S3, S8, S9, S15, S16, S20) had to be specified using the SoaML *Service Contract* approach instead of the *Service Interface* approach, as proposed by our method. They pointed out that interactions between partners in a contract (e.g., purchase order, distribution order) will be defined separately from the participants in the *Service Contract* approach (OMG, 2009). The latter defines the obligations of all the participants while the *interface-based approach* defines the obligations individually on each participants' service and request (OMG, 2009). We agree with the architects[5]. It is worth noting that architects 2, 3 and 6 found that the *payment services* (S5 and S13) *are too generic* as they are based on the *Exchange* business pattern. They suggested designing/using a new pattern that is specific to payment transactions. Furthermore, architects 2, 3 and 5 comments revealed that S14 (*Return Request Service*), which uses the generic CRUD operations designed as REST verbs, must be based on a specific *Rollback/Compensation* business pattern(s).

Based on the results, we rejected the null hypotheses $H2i_0$ for $1 \leq i \leq 21$ and $i \notin \{3, 8, 9, 15, 16, 20\}$ and accepted their alternative hypotheses, meaning that 71.43 % of the services specifications were perceived by the architects as *correct*, genericity notwithstanding–see previous paragraph.

Analysis of the effectiveness of the services: Each expert evaluated the ability of the services to effectively automate the collaboration activities for the CBPs. As shown in Table 6, architects agreed that the services will support the collaborations at hand, except for *Return Request Service* (S14) and *Tracking Service* (S10). For S14, architects argued that, while the CRUD operations manage the return request message as the HTTP methods do for Rest services, these operations do not indicate the purpose of the message (i.e., initiating a return process of a product). They suggest that the service had to be linked to a business pattern with the *Return/Rollback* (Architects 2, 3, 4, 5 and 6) or *Compensation* semantics (Architect 1). We plan to apply *Compensation patterns* from (Boubaker et al., 2015) to return type processes. Moreover, architects 1 and 3 found that the (S10) *automates partially* the collaboration (value 1 with gray background in the table 6). Architects explained that, while the *identification* pattern generates a unique identifier and links it to the resource (i.e. product), and the *notification* pattern notifies the partner accordingly, the message does not inform other services (e.g. *Shipping Service*)

[5]Our transformation rules *do not* support the *Service Contract-based* approach

Table 5. Evaluation of the correctness of SoaML specifications.

Experts	Services																				
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
Architect 1	1	1	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	0	1
Architect 2	NA	1	0	1	1	1	NA	0	0	1	1	1	1	1	1	0	1	1	NA	0	1
Architect 3	1	1	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	0	1
Architect 4	1	NA	1	NA	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Architect 5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
Architect 6	1	1	1	1	1	1	1	1	1	1	1	1	1	NA	NA	1	1	1	1	1	1

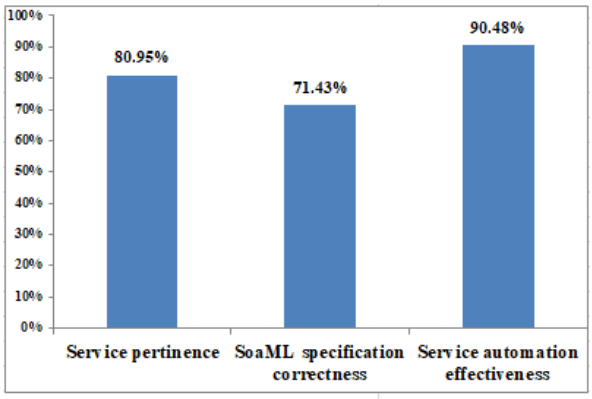

Figure 8. Results of the evaluation.

that this number is for tracking the resource during the shipping. From their point of view, architect 2, 4 and 6 found that S10 does not effectively automate the tracking of the product during the shipping. They proposed to compose S10 with S11 (*Shipping Service*) to be able to effectively automate the collaboration.

Based on the above, we rejected the null hypotheses $H3i_0$ for $1 \leq i \leq 21$ *and* $i \notin \{10, 14\}$ and accepted their alternative hypotheses, meaning that the architects agreed that 90.48% of the services will *effectively* support the execution of the collaboration process.

Figure 8 summarizes the results of the evaluation of the effectiveness of the proposed method from the point of view of the experts. Among the 198 evaluations (21 services x 3 architects x 3 aspects to evaluate), experts confirmed that: 1) 80.95% of the services were perceived as pertinent, 2) 71.43% of their SoaML specifications were perceived to be correct, and 3) 90.48% of them effectively support the execution of the collaboration process.

5.5. Threats to the Validity

This section discusses the threats to the internal and the external validity of the experimental study (see Cook and Campbell, 1979).

5.5.1. Threats to Internal Validity The threats to internal validity are relevant when the study's goal is to establish a causal relationship between those variables. In this study, they are mainly related to the participants' experience and the information exchange among them.

To mitigate the threats related to experts' profiles, we defined a minimum skill set to be met by participants. Experts' selection was based on strong professional experience in business and solutions architecture. To mitigate the impact of information exchange, the experiment took place in a controlled environment where the participants could not communicate.

5.5.2. External Validity Threats to external validity compromise the generalization of the obtained results. In this experiment, the external threat arises from the limited set of the selected collaborative processes. To address this issue, we selected common exchange processes. To improve the generalization of the results, authors plan to conduct further replication studies.

6. Conclusion and Future Work

Organizations develop PAIS to support CBPs. Designing PAIS is complex and requires designers with extensive experience. One emerging solution to design PAISs is to use the SOA style. This paper proposed a model-driven method to design PAISs by identifying and specifying services from CBPs expressed in BPMN. Identified services are specified with SoaML. We conducted an experimental study to evaluate the effectiveness of the method in the context of CBPs from the ERP literature. Results showed that 80.95% of the services were deemed pertinent and corresponded to what the experts expected. It also confirmed that 71.43% of the SoaML specifications were perceived to be correct and that 90.48% of the services effectively automated the underlying collaboration activities.

This research represents a new step to reach our long-term research goal to provide organizations with a method and tools to automate the design of service-oriented PAISs from CBP models. Future

Table 6. Evaluation of service automation effectiveness.

Experts	Services																				
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
Architect 1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
Architect 2	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1
Architect 3	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
Architect 4	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1
Architect 5	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
Architect 6	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1

work will include: i) automating the mapping between services and business patterns, ii) extending the method to support flexible CBPs expressed with the CMMN (Case Management Model and Notation), and iii) developing a web-based solution to support the method.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Azevedo, L. G., Santoro, F., Baião, F., Diirr, T., Souza, A., De Souza, J. F., and Sousa, H. P. (2013). A Method for Bridging the Gap between Business Process Models and Services. *iSys - Brazilian Journal of Information Systems*, 6:62–98.

Basili, V. R. and Rombach, H. D. (1988). Tame Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6):758–773.

Bianchini, D., Cappiello, C., De Antonellis, V., and Pernici, B. (2014). Service Identification in Interorganizational Process Design. *IEEE Transactions on Services Computing*, 7(2):265–278.

Boubaker, A., Mili, H., Leshob, A., and Charif, Y. (2015). Towards Automating Business Process Compensation Scoping Logic. In *International Conference on E-Technologies*, volume 209, pages 20–36. Springer, Cham.

Cook, T. D. and Campbell, D. T. (1979). *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin, Boston.

Daghaghzadeh, M. and Babamir, S. M. (2021). A model driven and clustering method for service identification directed by metrics. *Software - Practice and Experience*, 51(2):449–484.

Delgado, A., Ruiz, F., and García-Rodríguez de Guzmán, I. (2018). A reference model-driven Architecture linking Business Processes and Services. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, volume January, pages 4651–4660.

Gonzalez-Huerta, J., Boubaker, A., and Mili, H. (2017). A Business Process Re-Engineering Approach to Transform BPMN Models to Software Artifacts. In Aïmeur, E., Ruhi, U., and Weiss, M., editors, *E-Technologies: Embracing the Internet of Things*, pages 170–184, Cham. Springer International Publishing.

Hruby, P. (2006). *Model-Driven Design Using Business Patterns*. Springer-Verlag, Berlin/Heidelberg.

ISO (2011). Information technology: Business operational view- Part 1: Operational aspects of open-edi for implementation. Technical Report ISO/IEC 15944-1:2011, International Organization for Standardization, Geneva, Switzerland.

Kartseva, V., Gordijn, J., and Tan, Y. H. (2009). Designing value-based inter-organizational controls using patterns. In *Lecture Notes in Business Information Processing*.

Laskey, K. B. and Laskey, K. (2009). Service oriented architecture. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):101–105.

Leshob, A., Blal, R., Mili, H., Hadaya, P., and Hussain, O. K. (2019). From BPMN Models to SoaML Models. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 123–135. Springer.

Nikaj, A., Weske, M., and Mendling, J. (2019). Semi-automatic derivation of RESTful choreographies from business process choreographies. *Software and Systems Modeling*, 18(2):1195–1208.

OMG (2009). Service oriented architecture Modeling Language (SoaML) Version 1.0.

OMG (2011). Business process model and notation (BPMN 2.0).

Ouyang, C., Dumas, M., Aalst, W. M. P. V. D., Hofstede, A. H. M. T., and Mendling, J. (2009). From business process models to process-oriented software systems. *ACM Transactions on Software Engineering and Methodology*, 19(1):1–37.

Palma, F., Moha, N., and Gueheneuc, Y. G. (2019). UniDoSA: The Unified Specification and Detection of Service Antipatterns. *IEEE Transactions on Software Engineering*, 45(10):1024–1053.

Seidman, I. (2019). *Interviewing as Qualitative Research: A Guide for Researchers in Education and the Social Sciences*. ERIC.

van der Aalst, W. M. P. (2009). Process-Aware Information Systems: Lessons to Be Learned from Process Mining. In Jensen, K. and van der Aalst, W. M. P., editors, *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, pages 1–26. Springer, Berlin, Heidelberg.

Zafar, I., Azam, F., Anwar, M. W., Maqbool, B., Butt, W. H., and Nazir, A. (2019). A Novel Framework to Automatically Generate Executable Web Services From BPMN Models. *IEEE Access*, 7:93653–93677.