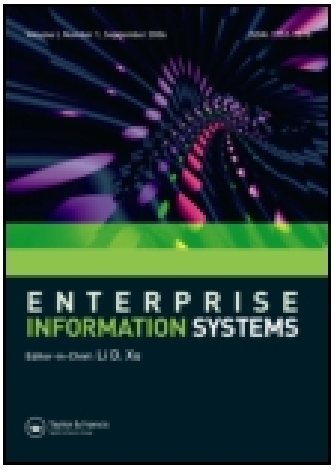
Supporting BPMN choreography with system integration artefacts for enterprise process collaboration



Enterprise Information Systems
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/teis20

Supporting BPMN choreography with system integration artefacts for enterprise process collaboration

Hongchao Nie[a], Xudong Lu[a] & Huilong Duan[a]
[a] Department of Biomedical Engineering, Zhejiang University, Hangzhou 310027, China
Published online: 28 Jan 2014.

To cite this article: Hongchao Nie, Xudong Lu & Huilong Duan (2014) Supporting BPMN choreography with system integration artefacts for enterprise process collaboration, Enterprise Information Systems, 8:4, 512-529, DOI: 10.1080/17517575.2014.880131

To link to this article: http://dx.doi.org/10.1080/17517575.2014.880131

Enterprise Information Systems, 2014
Vol. 8, No. 4, 512–529, http://dx.doi.org/10.1080/17517575.2014.880131

Taylor & Francis
Taylor & Francis Group

Supporting BPMN choreography with system integration artefacts for enterprise process collaboration

Hongchao Nie, Xudong Lu* and Huilong Duan

Department of Biomedical Engineering, Zhejiang University, Hangzhou 310027, China

(*Received 3 December 2012; accepted 31 December 2013*)

Business Process Model and Notation (BPMN) choreography modelling depicts externally visible message exchanges between collaborating processes of enterprise information systems. Implementation of choreography relies on designing system integration solutions to realise message exchanges between independently developed systems. Enterprise integration patterns (EIPs) are widely accepted artefacts to design integration solutions. If the choreography model represents coordination requirements between processes with behaviour mismatches, the integration designer needs to analyse the routing requirements and address these requirements by manually designing EIP message routers. As collaboration scales and complexity increases, manual design becomes inefficient. Thus, the research problem of this paper is to explore a method to automatically identify routing requirements from BPMN choreography model and to accordingly design routing in the integration solution. To achieve this goal, recurring behaviour mismatch scenarios are analysed as patterns, and corresponding solutions are proposed as EIP routers. Using this method, a choreography model can be analysed by computer to identify occurrences of mismatch patterns, leading to corresponding router selection. A case study demonstrates that the proposed method enables computer-assisted integration design to implement choreography. A further experiment reveals that the method is effective to improve the design quality and reduce time cost.

Keywords: BPMN; choreography; integration solution design; behaviour mismatch; enterprise integration patterns

1. Introduction

Enterprise information systems (EISs) play an increasingly important role to support daily operations within and even across enterprises, and the need for their integration becomes prominent to satisfy business requirements (Xu 2011a). Enterprise application integration (EAI) technology uses various methods and tools to enable the distributed EIS to connect to each other so that they work as a whole (Xu 2011b; Panetto and Cecil 2013; Wang et al. 2012). The recent technology trend is to merge the traditional middleware-based EAI approaches with emerging disciplines such as Business Process Management (BPM) and Service-oriented Architecture (SOA) (Xu 2011b; He and Xu 2014). These technologies offer higher-level understanding and description of EIS integration as orchestration (i.e. composing distribute services into flexible processes to meet enterprise goal) and choreography (i.e. coordinating interactions between processes to enhance enterprise collaboration) (Xu et al. 2009).

*Corresponding author. Email: lvxd@zju.edu.cn

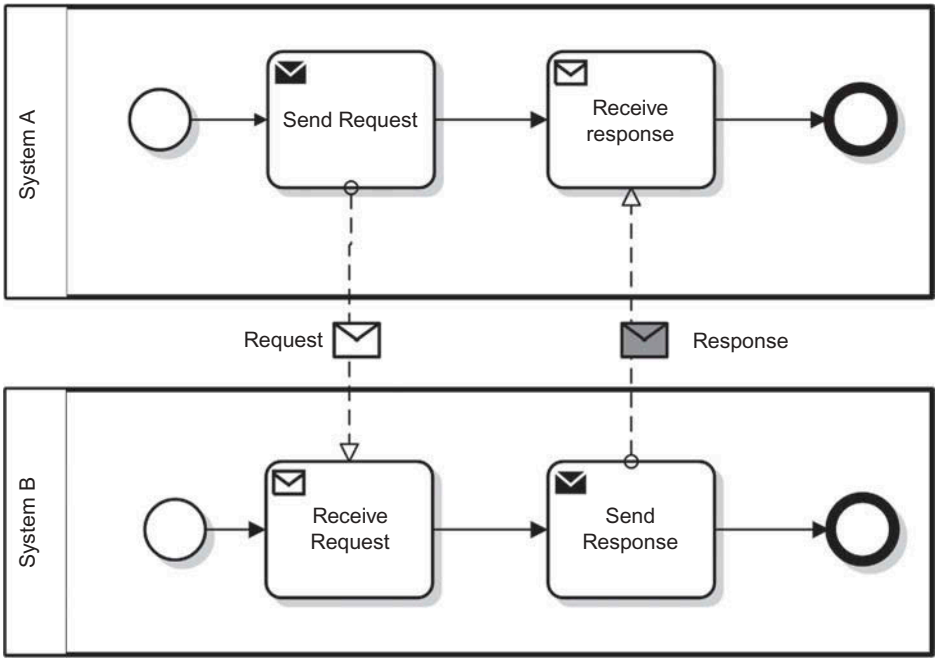


Figure 1. A BPMN model of choreography.

Choreography modelling depicts externally visible message exchanges between the autonomous processes of collaborating EIS (Weske 2012). As a standard for process modelling, the Business Process Model and Notation (BPMN) (OMG 2011) provides choreography modelling capability in terms of collaboration diagrams (Kopp, Leymann, and Wagner 2011). In BPMN collaboration diagrams, individual systems are modelled as control-flow process models, and collaborative interactions between them are modelled as message flows across the processes. For example, Figure 1 shows a BPMN collaboration diagram which choreographs processes of two systems (called participants), with request–response interactions. The message exchange activities of systems are modelled as rounded rectangles, called send tasks (with dark envelope icons) and receive tasks (with light envelope icons). A message flow (the dashed arrow) connects a send task in one process to a receive task in another process to establish an interaction. As the collaboration evolves, the choreography model can become quite complex. Modelling choreography using BPMN provides a mechanism to describe interconnections between systems in a bird's-eye view.

Choreographing two processes faces the challenge of behaviour mismatches. Since the EISs are autonomous, their external behaviours, i.e. the message exchange interactions and their dependencies may differ significantly. Choreographies modelled using BPMN collaboration diagrams can also represent intents of coordinating processes with behaviour mismatches. In this paper, such type of choreography model is termed *coordinated choreography*. In coordinated choreography, the domain expert interprets the relations between the message exchange activities in the process models, and coordinates the processes by connecting matching activities to specify the correct way of collaboration. For example, Figure 2 shows a coordinated choreography model: a process A has two send tasks to send messages A and B in sequence, and then waits for a response. Another

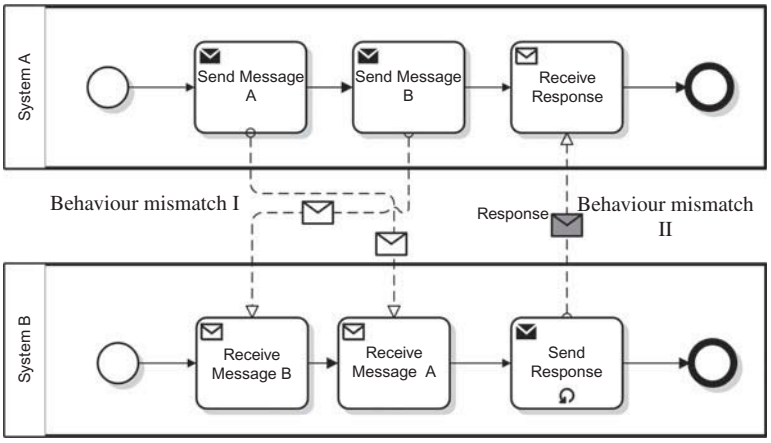


Figure 2. A BPMN model of coordinated choreography.

process B has two receive tasks to firstly receive message B and then message A, and then replies with partial responses in loop. The message flows are defined by the business analyst based on interpretation of the relations between tasks. Such interconnections reflect the business intent to coordinate the mismatching behaviours.

While the choreography model specifies business-level design of collaboration between processes, it needs to be supported by EAI technology at technical level. Processes of different systems may vary in their use of message semantics and behaviour models (Barker, Walton, and Robertson 2009; Guo et al. 2012). Since modifying EIS to exactly match with each other is always costly, sometimes impossible, mediation is necessary to translate between different message formats and to coordinate different message exchange behaviours (Wang and Xu 2008; Quartel et al. 2009). In EAI technology, such mediation design produces an integration solution, which tells the EAI middleware (broker engine or Enterprise Service Bus (ESB) (Chappell 2004)), how to connect to the interfaces exposed by systems and how to implement data flows between them according to requirements while mediating the differences. Enterprise integration patterns (EIPs) (Hohpe and Woolf 2004) provide a rich set of domain-specific design artefacts for messaging-oriented integration solution. Since most integration solutions are built on messaging-oriented EAI middleware, the EIP artefacts are widely adopted in integration solution design (Scheibler, Mietzner, and Leymann 2009). In EIP-based integration design, all the interfaces of systems (called endpoints) are connected to the integration system, and data flows between interfaces are implemented using message channels. To mediate the differences between interfaces, the message translators and routers support intermediate message transformation and routing. This paper focuses on the behaviour aspect of integration solution design.

Designing integration solution to implement choreography may seem straightforward, but actually is not, particularly in coordinated choreography cases. To implement the message flows in Figure 1, the integration designer can directly design message channels to implement data flows between the interfaces of the two systems, as shown in Figure 3. But if the model represents coordinated choreography, the mismatching behaviours require specific design of message routing. For example, to support the choreography in Figure 2, the designer interprets the choreography model, and identifies that a rearrangement of the sequence of messages A and B is necessary. Therefore, a specific router,

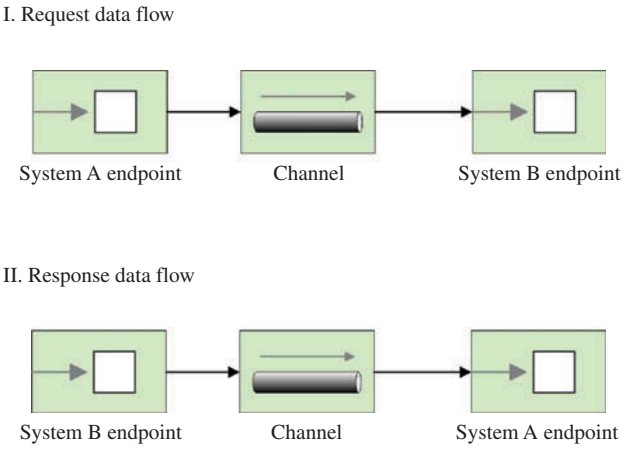


Figure 3. The integration solution to support the choreography.

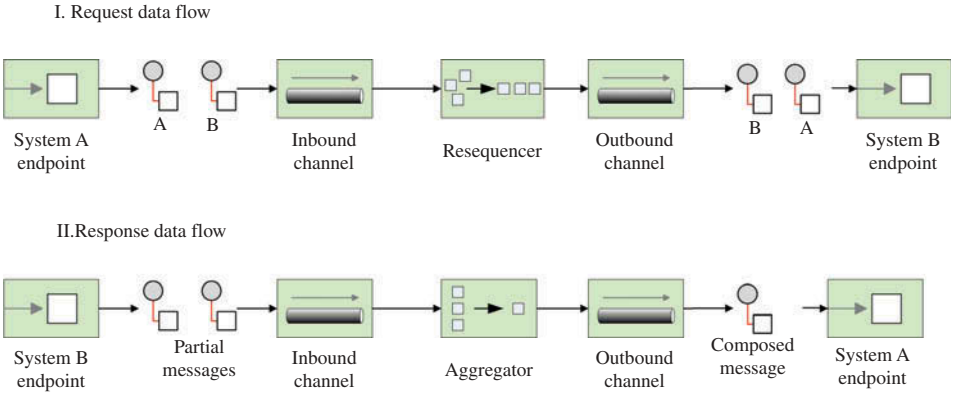


Figure 4. The integration solution to support the coordinated choreography.

called *resequencer*, is assigned to mediate the order mismatch of message exchanges. Also, since multiple partial responses need to be aggregated into a single comprehensive interaction, another type of router, called *aggregator*, is assigned to implement such aggregation. The result integration solution is shown as Figure 4.

It can be observed that designing integration solutions to support coordinated choreography takes two steps: (1) the designer identifies routing requirements from the BPMN choreography model; and (2) the designer addresses the identified requirements using specific types of EIP message routers. Currently, this design process is case-by-case and totally manual, based on the designer's interpretation of routing requirements and expertise of integration design. Heavy dependence on personal expertise may lead to errors in case of insufficient knowledge, and manual work costs considerable time. As collaboration scales and complexity increases, such manual work hampers quality and efficiency of integration solution design. Therefore, the research problem of this paper is to investigate if this design process can be computerised, i.e. to use the computer to identify routing requirements automatically and correspondingly design routers in EIP-based integration solution. To address this problem, recurring scenarios of behaviour mismatches in BPMN

choreography model are analysed and summarised as computer-readable patterns, and corresponding solutions of these patterns are proposed as specific types of EIP message routers. Upon detecting occurrences of patterns, computer can automatically select appropriate routers to address the routing requirements. This enables computer-assisted design and reduces human efforts. As validation, applicability of the method is investigated within a case study of a typical process collaboration scenario in the health-care industry. Also, the effectiveness of the method is validated with an experiment to compare the design results of human designers and the proposed automated method. As the experiment shows, the method is effective to improve design quality and reduce time cost. The main contribution of the proposed method is to enable computer-assisted integration solution design to support choreography implementation.

The rest of the paper is organised as follows. Section 2 introduces the research background and existing related work. Section 3 proposes the method with a solution framework overview followed by detailed method descriptions. Section 4 validates the method using a case study and an experiment. Finally, Section 5 concludes the paper.

2. Background

Viewing EIS from process-oriented perspective leads to prosperous research efforts in academia, such as process modelling (Xu et al. 2008; Viriyasitavat, Xu, and Martin 2012; Li et al. 2014), process analysis and management (Tan et al. 2008; Xu et al. 2012), and process implementation, particularly by service discovery and composition in SOA (Hachani, Gzara, and Verjus 2013; Paulraj, Swamynathan, and Madhaiyan 2012; Tao et al. 2012). Considering the specification and implementation of inter-process, collaboration is an emerging research trend (Xu et al. 2009; Xu 2011a). Choreography modelling formally describes collaboration between autonomous processes using message exchanges (Decker et al. 2009). By depicting relations between the individual messaging behaviours, choreography modelling provides to the business analysts a perspective to describe the whole picture of continuous interactions between EIS (Peltz 2003). Choreographies can be defined in top-down or bottom-up approaches. In the top-down approach, the choreography model (as shown in Figure 1) defines a common agreement on message exchange behaviours between systems which implement processes and interfaces conforming to the choreography. In the bottom-up approach, the existing systems are analysed to represent their external interaction behaviours as autonomous processes. Then, message exchange activities within different processes are connected to enable information sharing and process collaboration, as shown in Figure 2. Since the behaviours of systems may differ, such choreography model reflects coordination requirement at business level.

BPMN (OMG 2011) provides the capability of choreography modelling (Kopp, Leymann, and Wagner 2011). As the de facto process modelling standard, BPMN aims to provide a standardised bridge for the communication gap between the business expert, who defines the processes, and the technical designer, who implements the technology that supports the processes. BPMN achieves this by providing intuitive graphical notations, which can be easily understood by the business expert, and associates the notations to modelling objects defined in a metamodel. BPMN models choreography specifications as collaboration diagrams. BPMN collaboration diagram shows a collection of collaborative processes (i.e. participants) and their interactions. The external interactions between the participants (i.e. entities involved in the collaboration) are modelled as message flows connecting their communication activities such as send tasks and receive tasks. With the

independence of process models, BPMN collaboration diagrams are capable to represent both top-down and bottom-up choreography models.

After business-level design, the choreographies need to be implemented to realise actual message exchanges between systems. Since behaviour mismatches exist, the coordinated choreography model cannot be implemented by directly invoking concrete interfaces provided by the individual systems. In such cases, EAI technology enables designing and deploying an integration solution to implement the data flows between EIS interfaces and mediate their differences (Linthicum 2000; Izza 2009; Vernadat 2010). The design of the integration solution involves selecting and composing reusable integration artefacts according to the integration requirements, i.e. the expected message flows between the systems and the differences to be mediated. EIPs (Hohpe and Woolf 2004) summarise recurring problems in integration design, and provide corresponding solutions to these problems as a set of domain-specific artefacts using messaging-oriented mechanism. EIPs represent abstract knowledge of basic EAI functionalities, including transporting messages through message channels and adding intermediate processing steps such as message transformation and routing. Benefiting from the decoupled nature of the Pipes-and-Filters (PaF) architecture and visual notations, EIP-based integration solutions are easy to understand and maintain. Thus, EIP artefacts are widely adopted by EAI middleware products to design integration solutions.

Since the choreography is modelled using BPMN and integration solution is designed using EIPs, there is still a knowledge gap between business-level requirement specification and technical-level solution design. Currently, this gap is bridged by intellectual work of the integration designer who: (1) obtains business knowledge by understanding the integration requirements specified in the choreography model, particularly routing requirements in coordinated choreography cases; and (2) transfers the business knowledge into appropriate integration solution design to address the requirements based on integration expertise, particularly by designing message routers to coordinate the mismatching behaviours. Therefore, the research problem addressed in this paper is to explore a computerised, automated method to bridge this knowledge gap. Automating this design process by computer assistance requires that: (1) the computer can identify routing requirements, i.e. behaviour mismatches, and (2) the computer has the design knowledge of router usage to deal with specific types of routing requirements.

Behaviour mismatch, also termed as protocol mismatch or process mismatch, etc., considers differences between message exchange behaviours. Some efforts have been made to mediate processes with incompatible behaviours. It has been recognised that totally automating the mediation at runtime is difficult, and complex scenarios still require human decision (Benatallah et al. 2005). Therefore, designing process mediation solution is unavoidable, and can at most be semi-automated. Semi-automatic design of process mediation can be categorised as simulation-based or pattern-based approaches. Simulation-based approaches traverse simulated state transitions to find possible solutions (Nezhad et al. 2007). Like the runtime mediation approaches, since not all mismatches can be handled automatically, the simulation cannot proceed if some unresolvable mismatches exist, and the remaining processes cannot be mediated. In contrast, the pattern-based approach adopts the divide-and-conquer strategy to identify some recurring situations and solving them individually with predefined solution which can be composed, and allows human refinement rather than demanding the complete solution (Benatallah et al. 2005). The COSMO framework summarises some behaviour mismatch cases and designs corresponding conceptual mediators using a model-driven approach (Quartel et al. 2009). However, the developer still needs to manually identify these mismatch cases and design

mediators. This paper goes further in this direction to investigate how to automatically identify mismatches in the context of BPMN choreography modelling and how to solve them by designing EIP message routers. This aspect has not been addressed by existing works.

3. Method

Traditionally, designing integration solution to implement the choreography scenario is manual. In this section, a computer-assisted method is proposed to identify routing requirements from BPMN choreography models and to correspondingly design routing solutions. The overall solution framework is shown in Figure 5. First, recurring behaviour mismatch scenarios with routing requirements are analysed and summarised as behaviour mismatch patterns. These patterns can be identified by looking for particular structural characteristics represented in the BPMN choreography models. These patterns provide reusable knowledge for computerised requirement analysis. Then, for each behaviour mismatch pattern, a routing solution using specific types of message routers is proposed to address the routing requirement. These solutions represent EAI design knowledge of router selection based on requirements. To utilise this method during integration solution design to support choreography implementation, a computer analyses routing requirements in the overall choreography model by automatically extracting a set of occurrences of behaviour mismatch patterns, and automatically selects corresponding types of EIP message routers to meet each specific requirement.

The patterns in this paper mainly concern differences in cardinality and order between the senders and the receivers. Here, senders and receivers refer to send tasks and receive tasks connected by the related message flows in a pattern, respectively. The patterns 1, 2 and 3 concern one-to-many scenarios in which one sender has multiple receivers. The patterns 4 and 5 concern many-to-one scenarios in which multiple senders interact with one receiver. The pattern 6 considers sequence differences between multiple senders and receivers. Note the BPMN diagrams used to illustrate the patterns just show partial choreography models.

Pattern 1: Loop-receiver pattern: a receiver expects to receive several partial messages, but the sender sends a composed message. In BPMN, this occurs when a send task with no loop attribute is connected to a looped receive task (Figure 6 left). Occurrences of this pattern can be extracted as:

$$LoopReceiver \leftarrow \{mf \mid \text{loop}(\text{target}(mf)) = \text{true} \wedge \text{loop}(\text{source}(mf)) = \text{false}\}.$$

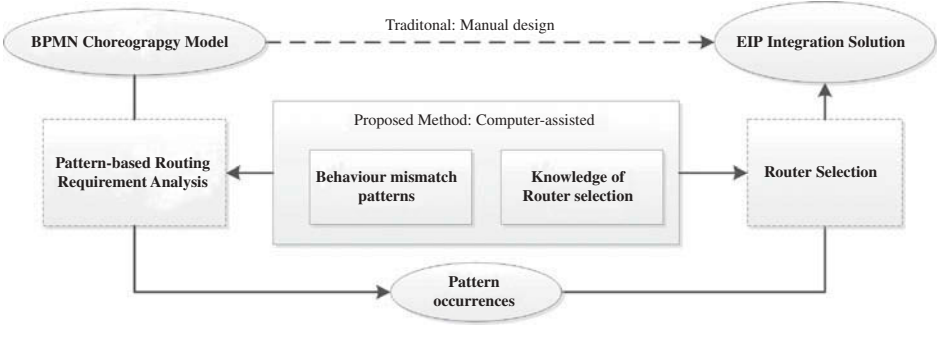


Figure 5. Overview of the solution framework.

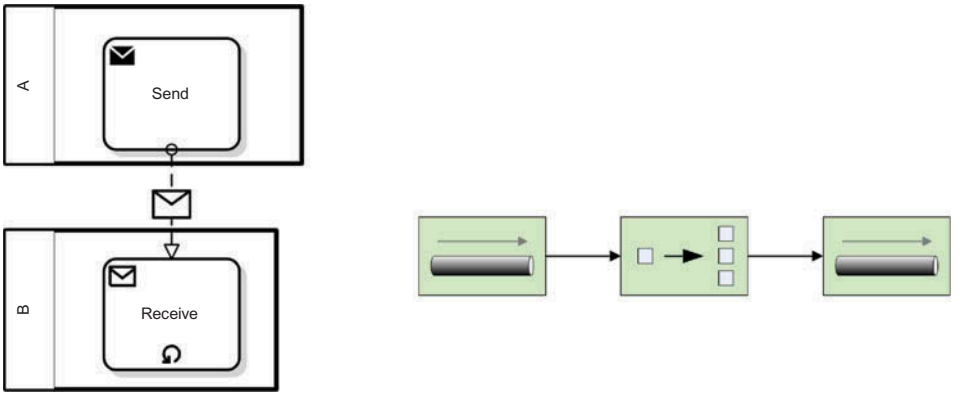


Figure 6. The loop-receiver pattern.

Here, *mf* denotes a message flow that connects the sender and receiver. In EIP, a splitter router breaks out a composite message into a series of individual partial messages. Thus, the requirement in this pattern can be addressed by a splitter to split the message into multiple parts (Figure 6 right).

Pattern 2: Exclusive-receiver pattern: a sender has multiple receivers which are mutually exclusive in the same process model. In BPMN, this occurs when the receive tasks connected to the same sender are proceeded by the same exclusive gateway (Figure 7 left). This pattern can be extracted by the rule:

$$ExclusiveReceivers \leftarrow \{mf\}, \quad \text{where}: \quad \forall mf_i, \quad mf_j \in ExclusiveReceivers,$$
$$\text{source}(mf_\text{i}) = \text{source}(mf_\text{j}) \wedge \text{exc}\big(\text{target}(mf_\text{i}), \text{target}(mf_\text{j})\big).$$

This definition means any two message flows in the identified set have the same source but their targets are exclusive. The *exc* function examines exclusive relation between two tasks in the same process model. The requirement in this pattern can be addressed by a Content-based Router (CBR) to route each message to only one receiver depending on content-based conditions (Figure 7 right). The condition of selecting a

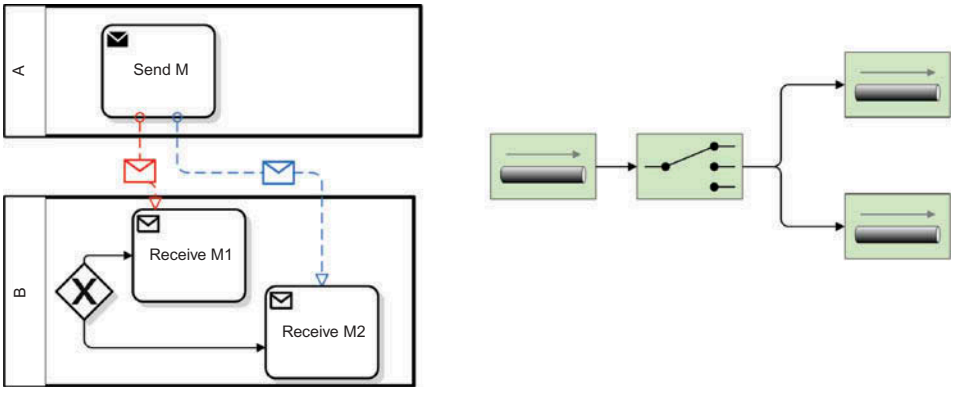


Figure 7. The exclusive-receivers pattern.

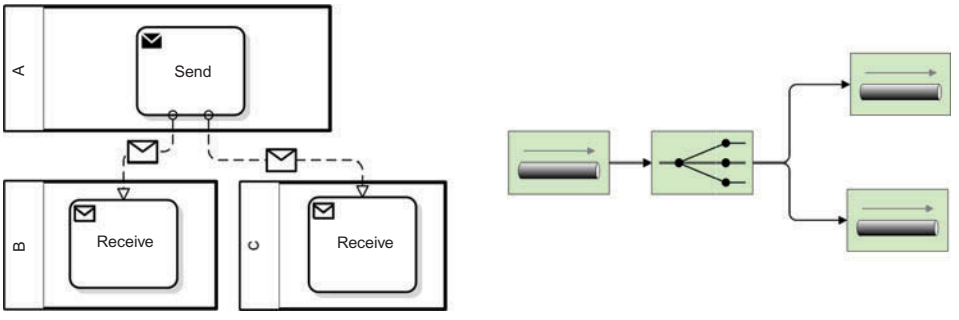


Figure 8. The multiple-receivers pattern.

particular branch relies on the message content and therefore needs to be manually specified by the user.

Pattern 3: Multiple-receivers pattern: the message from one sender is of interest to multiple receivers. In BPMN, this pattern occurs when a send task is connected to multiple receive tasks within different participants (Figure 8 left). This pattern can be identified as:

$$MultipleReceivers \leftarrow \{mf\} \quad \text{where}: \quad \forall mf_i, \quad mf_j \in MultipleReceivers,$$
$$\text{source}(mf_i) = \text{source}(mf_j) \wedge \text{process}(\text{target}(mf_i)) \neq \text{process}(\text{target}(mf_j)).$$

This definition means that any two message flows in this pattern have the same source but different target participants. In EIP, a recipient list router broadcasts an incoming message to all output channels. Thus, the requirement in this pattern can be addressed by a recipient list router (Figure 8 right).

Pattern 4: Loop-sender pattern: a receiver expects a composed message, but the sender sends partial messages individually in loop. In BPMN, this occurs when a send task has a loop property while the receiver does not (Figure 9 left). This pattern can be identified as:

$$LoopSender \leftarrow \{mf \,|\, \text{loop}(\text{source}(mf)) = \text{true} \wedge \text{loop}(\text{target}(mf)) = \text{false}\}.$$

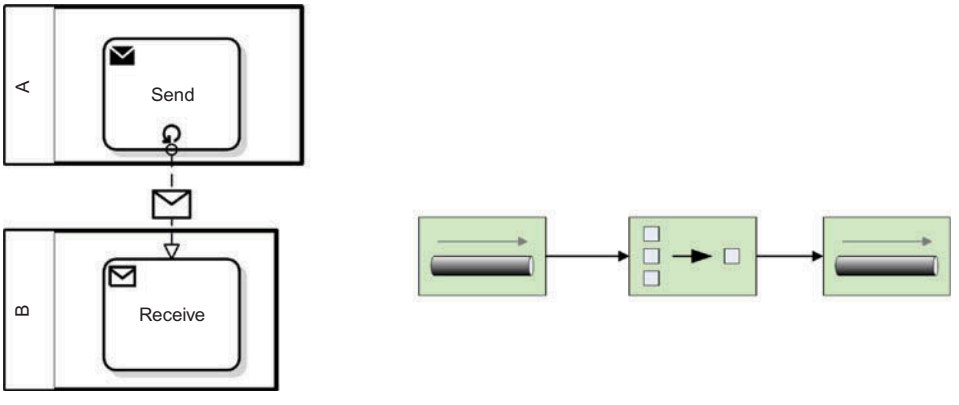


Figure 9. The loop-sender pattern.

In EIP, an aggregator collects and stores individual messages until a complete set of related messages has been received, and then combines the stored individual messages into a composed one. Thus, the requirements in this pattern can be addressed using an aggregator router that waits for multiple inbound messages and aggregates them to output a merged message to the receiver (Figure 9 right).

Pattern 5: Multiple-senders pattern: multiple message flows are connected to the same receive task (Figure 10 left). This pattern can be identified as:

$$MultipleSenders \leftarrow \{mf | \text{target}(mf) = r\},\ \textit{where } r \textit{ is a specific receive task}.$$

This rule finds a set of message flows that connect multiple send tasks to a particular receive task. This pattern implies both routing and message processing requirements; not all messages may be consumed, and the output message may contain contents from multiple sources. To meet these requirements, a combination of a message filter and an aggregator can be used (Figure 10 right). The message filter decides which messages can be processed, while the aggregator processes the received messages to output a single message to the receiver.

Pattern 6: Order-mismatch pattern: The order of the source send tasks is different from that of the target receive tasks. For example, a sender may use two send tasks to send messages M1 and M2 sequentially, while the receiver may use two receive tasks to firstly receive M2 and then M1 (Figure 11 left). Occurrences of this pattern can be identified as:

$$OrderMismatch \leftarrow \{mf\} \text{ where } \quad \forall mf_i, mf_j \in OrderMismatch, \text{seq}(\text{source}(mf_i), \text{source}(mf_j)) \wedge \text{seq}(\text{target}(mf_j), \text{target}(mf_i)).$$

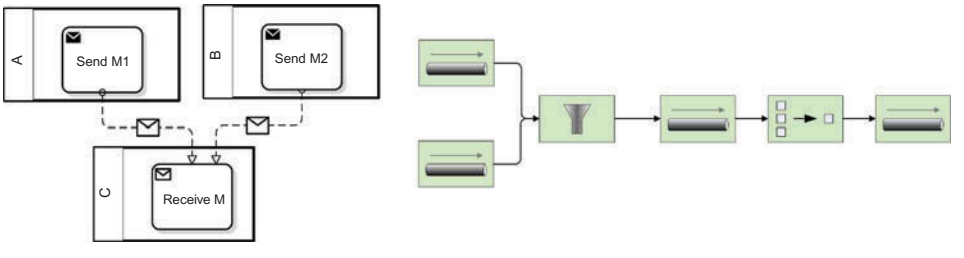


Figure 10. The multiple-senders pattern.

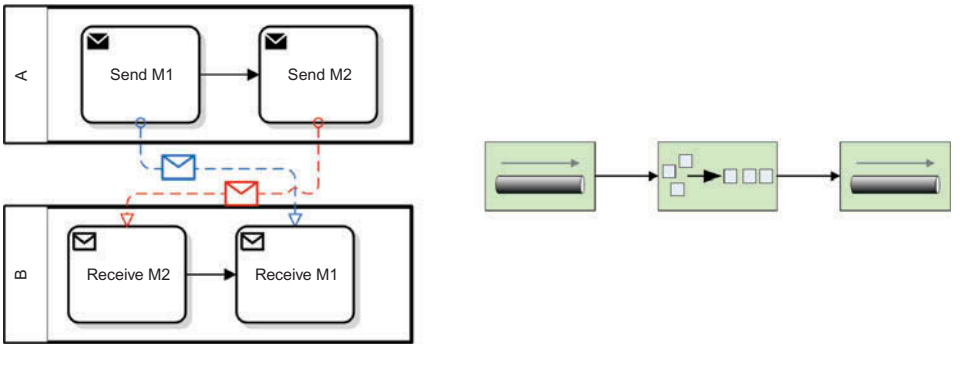


Figure 11. The order-mismatch pattern.

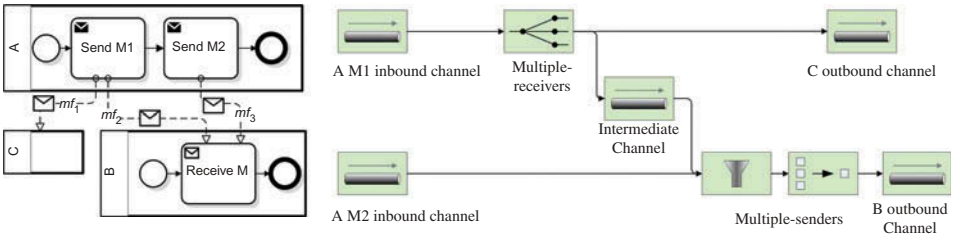


Figure 12. An example of pattern composition.

This means for any two message flows in the identified set, the order of their source tasks is different from that of their target receive tasks. The *seq* function examines if the first task proceeds the second task in a process model, i.e. only after the completion of the first task can the second task start. This pattern implies the requirement to coordinate the message sequence. In EIP, a resequencer router collects and reorders messages so that they are reorganised in a specified order. Thus, to meet the requirement in this pattern, a resequencer can be used to reorder received messages (Figure 11 right).

In the previous sections, six basic patterns of behaviour mismatches that may occur in BPMN choreography models have been identified, and their routing requirements have been addressed using specific types of EIP message routers. The pattern list is not meant to be exhaustive. However, they can serve as basic units of design and can be composed to resolve complex problems. Composition of solutions of these patterns can be achieved using the composition mechanism of the PaF architecture, i.e. connecting the routers using intermediate channels. For example, a sender (the system A) sends the message M1 to both systems B and C, and also sends the message M2 to the system B. On the other hand, the system B uses a single receive task to receive information from both M1 and M2. Their coordination is modelled as in Figure 12 (left). This represents a combination of the multiple-receivers pattern and the multiple-senders pattern. To support this choreography, according to the multiple-receivers pattern, a recipient list router is assigned to broadcast M1 to the two receivers. Also, according to the multiple-senders pattern, a combination of a filter and an aggregator is assigned. Then, the two partial solutions need to be composed. Since the message flow mf_2 goes through both the recipient list router and the filter, an intermediate channel is created to connect the two routers. The composed solution is shown in Figure 12 (right).

4. Validation

In this section, the proposed method is validated. First, a health-care choreography scenario is studied to show how the method applies in real case. Second, results of an experiment are reported to evaluate the effectiveness of the method in assisting routing design to mediate behaviour mismatches in common choreography scenarios.

4.1. Case study

As enterprises in other industries, hospitals gradually purchase information systems from various vendors. Integration of the distributed EIS in health-care enterprise is important to support hospital-wide information sharing and process collaboration (Li et al. 2008; Yin, Fan, and Xu 2012). In this section, a real process collaboration scenario within a health-

care enterprise is used as a case study to validate the applicability of the proposed method. This scenario represents collaboration between the clinical information system (CIS), which supports clinical diagnosis workflow, and the radiology information system (RIS), which supports radiology departmental workflow. The two systems were independently purchased, and their message exchange behaviours differ. The CIS sends order request consisting of multiple order items using a single send task, and then expects to receive responses using two different receive tasks: one for acceptance response and the other for rejection response. If the response indicates that the order is accepted, the CIS expects to receive a comprehensive final study result (i.e. the radiology report). In comparison, the RIS accepts individual order items using a looped receive task. Then it sends the order response using a single send task indicating either acceptance or rejection within the message content. If the order is accepted, the RIS sends study results using two successive send tasks: one for the preliminary report (e.g. image interpretation by a preliminary radiologist) and the other for the confirmation supplement (e.g. diagnosis by an expert radiologist). These behaviours are analysed by the business analyst and modelled as BPMN processes. By interpreting the correspondences of their message exchange activities, the business analyst coordinates behaviours of the two systems by specifying message flows to connect the tasks between their process models. Such coordination specification produces a coordinated choreography model, as shown in Figure 13. To implement such collaboration, the integration designer designs an integration solution using message routing capabilities provided by the ESB to actually implement the message flows. Traditionally, this involves human interpretation of routing requirements and manual selection of appropriate message routers.

Using the proposed method, a computer programme takes the BPMN model as input, extracts occurrences of mismatch patterns, and automatically assigns corresponding message routers to the integration solution. First, the message flow to send the order request (mf_1) represents the loop-receiver pattern. Therefore, a splitter is assigned. Second, the two message flows related to the order response (mf_2 and mf_3) represent the exclusive-receivers pattern. Thus, a CBR is assigned to exclusively invoke the two receiving operations provided by the CIS. Third, the reporting message flows (mf_4 and mf_5) represent the multiple-senders pattern. Correspondingly, combination of a filter and an

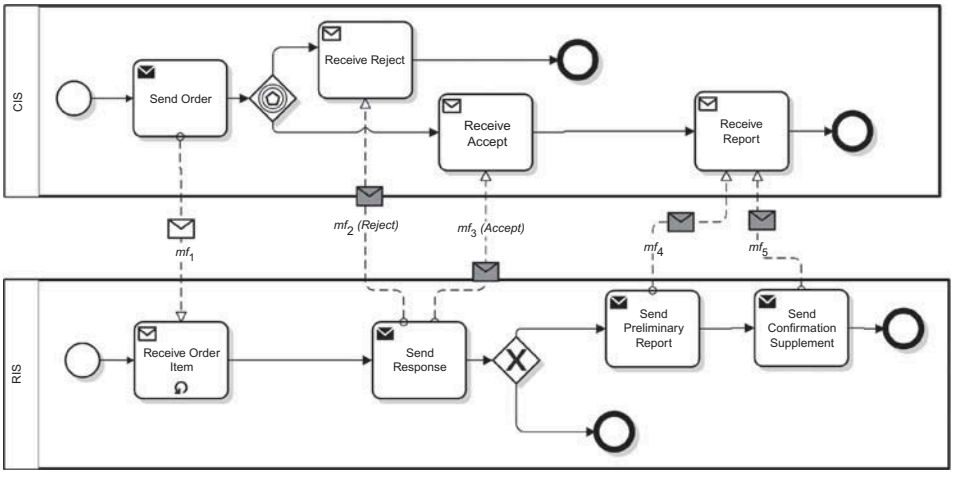


Figure 13. The choreography model of the case study scenario.

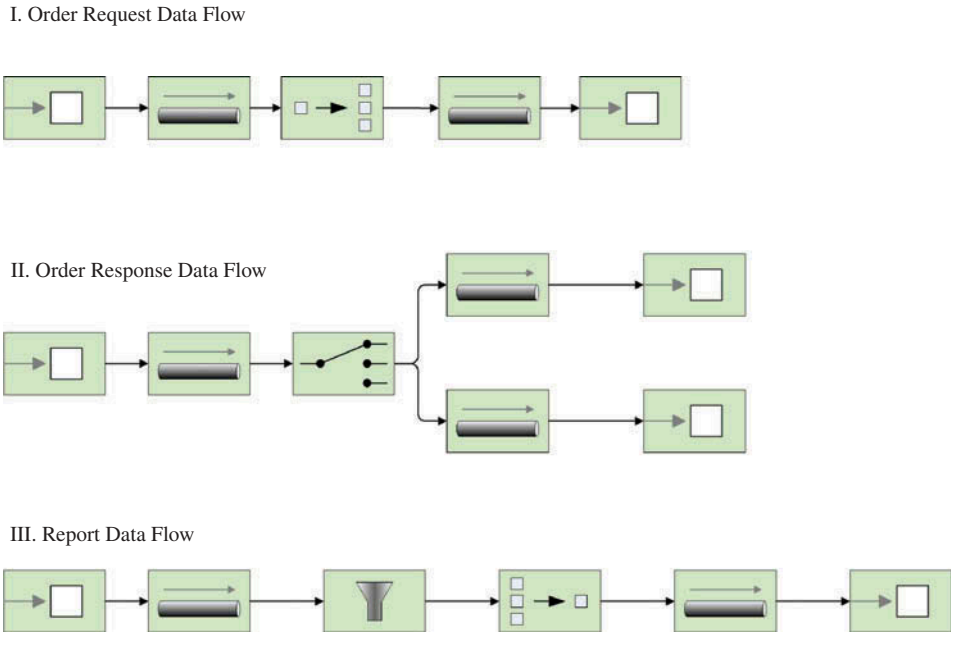


Figure 14. The result of computer-assisted integration solution design of the case study.

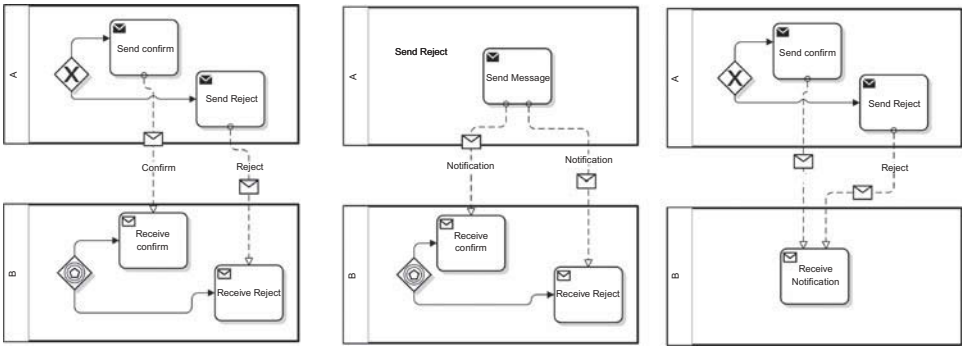
aggregator is assigned to build a comprehensive report from the two partial report messages. The auto-generated integration solution containing three data flows with routers is shown in Figure 14.

The automated programme produces a basic skeleton as preliminary integration solution consisting of routers, endpoints and channels. Then, the integration designer further configures the routers to specify concrete routing logics. In this case, the splitter in the first data flow needs further configuration of the method to split the composed order message into individual items. The CBR in the second data flow needs configuration of branch conditions to evaluate the result values in response messages. The filtering criteria and aggregation strategies in the third data flow also need to be further specified depending on the designer's interpretation.

This case study shows the ability of the proposed method to incorporate computer assistance into integration solution design. Traditionally, the integration designer must manually select routers and compose them based on interpretation of integration requirements defined in the BPMN choreography model and personal expertise of design knowledge. In comparison, the proposed method enables this step to be automated. This method automates the structural aspect of integration design (i.e. selection and composition of message routers), although the semantic aspect, i.e. message-based routing logic design, still relies on human interpretation and design. Nevertheless, the design of integration solution has been semi-automated.

4.2. Comparison experiment

In this part, results of an experiment to evaluate the effectiveness of the proposed method are reported. This experiment compared the integration solutions designed by the method to those designed by human designers.



(a) The racing incoming messages SIP (b) Variation I (c) Variation II

Figure 15(a–c). The racing incoming messages SIP and two variations modelled using BPMN.

To find a test set of collaboration scenarios with behaviour mismatches, service interaction patterns (SIPs) (Barros, Dumas, and ter Hofstede 2005) were adopted as a basis. SIPs represent a commonly recognised catalogue of typical behaviours of systems in process collaboration, and are regarded as well-established knowledge of choreography scenarios (Decker et al. 2009). By altering behaviours of participants in each pattern, a set of commonly encountered behaviour mismatch scenarios was derived. These scenarios with behaviour mismatches were modelled as BPMN-coordinated choreographies. For example, in the *racing incoming messages* SIP, a participant expects to receive one among a set of messages which may come from different types of participants. In BPMN, the racing condition of this pattern can be modelled using an event-based gateway to wait for a specific type of message (Figure 15(a)). When the first received message triggers a specific event, the process proceeds to that branch, and later received messages may be discarded. Altering the behaviour of the message sender or the receiver can produce two variations of this pattern with behaviour mismatches. First, if the message sender does not distinguish different message types, the choreography model connects a single send task of one participant to multiple racing receive tasks of another participant (Figure 15(b)). Second, if the receiver does not distinguish reception of different message types by using only one receive task, the choreography model connects multiple send tasks of different senders to a single receive task, which expects only one message with no racing behaviour (Figure 15(c)). By altering behaviours of participants in such ways, 22 behaviour mismatch scenarios were derived in total. Their BPMN coordinated choreography models were used as the test set of this experiment.

Then a group of integration engineers, with different integration knowledge levels, was asked to interpret routing requirements and design integration solutions for these scenarios. The group consisted of five designers, including one expert, two designers with medium expertise and two beginners. They were asked to record all identified routing requirements in the test scenarios, and then to design routing solutions. These results were compared to those produced by the computer programme using the proposed method. The results of designers with medium expertise and those of beginners were averaged respectively.

Table 1 shows the results of the experiment. The first row shows total numbers of identified routing requirements. Note that the number identified by the proposed method is the most close to the result of the expert. The differences are further analysed in the next three rows, using the results designed by the expert as references. As shown in row 2,

Table 1. The results of the experiment.

Criteria	Expert	Medium (avg.)	Beginner (avg.)	The proposed method
1. Total routing requirements identified	28	25	21	27
2. Missing requirements	/	5	12	1
3. Unnecessary requirements	/	2	5	0
4. Router selection differences of correctly identified requirements	/	6	15	1
5. Requirement identification and router selection time cost (days)	1.5	2.5	3	Automated
6. Routing logic design time cost (days)	1.5	3	3.25	/

some requirements identified by the expert were not identified by designers with less expertise. The programme using the proposed method failed to identify one requirement related to behaviour mismatch in message-based routing scenarios. A further examination reveals that this requirement occurred in the scenario where one system sends messages to a fixed end point, while the other system utilises dynamic routing to pass the references of other participants (e.g. addresses) within its exchanged messages. This mismatch is not structural but related to message content, which is only interpretable by human. Thus, the computer did not identify this mismatch when detecting the behaviour mismatch patterns proposed in this paper. In comparison, the designers with less expertise successfully identified this requirement, but they missed several requirements in other scenarios. Row 3 shows a comparison with regard to unnecessary requirements, i.e. those that were identified but actually do not necessarily need routing solution (according to the expert). In this criterion, while the proposed method produced no unnecessary results, the designers with less expertise misinterpreted some requirements. Row 4 shows a comparison of accuracy of router selection to solve the correctly identified requirements. The only difference between the solutions designed by the expert and the proposed automated method was in the scenario where two mutually exclusive send tasks connect to one receive task, as a variation of the racing incoming messages SIP (Figure 15(c)). To solve this mismatch (as the multiple-senders pattern), the computer automatically assigned a filter together with an aggregator to process incoming messages. Since the two send tasks are mutually exclusive, the expert deemed the filter to be unnecessary. This result reveals the method needs to further decompose the multiple-senders pattern into sub-patterns to consider variation of senders' behaviours. In comparison, the designers with less expertise produced more errors in router selection without computer assistance. In general, these results of this preliminary experiment show that the proposed method can be considered to be effective to improve the design quality, whereas the expert still cannot be replaced.

The last two rows of Table 1 compare results of time cost. The overall integration design consisted of two steps: (1) requirement identification and router selection; and (2) router configuration. Row 5 shows a comparison of time cost in identifying routing requirements and correspondingly selecting routers. It can be noted that the automated method can significantly reduce the time cost in this step, while the time cost of human design varied with expertise level. Row 6 shows a comparison of the time cost of configuring the selected routers with concrete routing logic. This step was totally manual, and the time cost varied with design expertise. In general, these results of this preliminary experiment show that the method of automated requirement detection and router selection can reduce human efforts in the first design step, and thus improves design efficiency.

5. Conclusion

This paper presents an approach to computer-assisted design of integration solutions in order to implement choreographies. It focuses on the automatic identification and analysis of routing requirements in BPMN choreography models. Behaviour mismatch scenarios are analysed and summarised as patterns, and their corresponding solutions are proposed as EIP routers. From the feasibility perspective, validation using a case study in a health-care enterprise shows that the method is capable of designing routing solutions to support BPMN choreography models semi-automatically. A further experiment in comparison to human design efforts shows that the method is effective to improve quality of integration solution design and to reduce time cost. The proposed method takes an initial step forward to bridge the gap between the business-level specification of choreographies and the technical-level design of EAI solutions.

Ongoing work aims to further assess and improve the proposed method. As the experiment shows, routing requirements with regard to message contents cannot be discovered by analysing only the structural characteristics of BPMN models, but requires understanding meanings of message contents. To this aim, the message schemas referenced by the BPMN message objects can be annotated using domain ontologies. The annotations allow computers to interpret the meanings of message contents to further enable detection of routing requirements in such cases. It is envisioned that explicit description of message semantics might lead to a new approach to configure the concrete routing logics.

References

Barker, A., C. D. Walton, and D. Robertson. 2009. "Choreographing Web Services." *IEEE Transactions on Services Computing* 2 (2): 152–166. doi:10.1109/TSC.2009.8.

Barros, A., M. Dumas, and A. ter Hofstede. 2005. "Service Interaction Patterns." In *Business Process Management*, edited by W. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, Vol. 3649, 302–318. Berlin: Springer.

Benatallah, B., F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani. 2005. "Developing Adapters for Web Services Integration." In *Advanced Information Systems Engineering*, edited by O. Pastor and J. F. e Cunha, 415–429. Berlin: Springer.

Chappell, D. A. 2004. *Enterprise Service Bus*. Sebastopol, CA: O'reilly Media.

Decker, G., O. Kopp, F. Leymann, and M. Weske. 2009. "Interacting Services: From Specification to Execution." *Data & Knowledge Engineering* 68 (10): 946–972. doi:10.1016/j.datak.2009.04.003.

Guo, J., L. D. Xu, G. Xiao, and Z. Gong. 2012. "Improving Multilingual Semantic Interoperation in Cross-Organizational Enterprise Systems through Concept Disambiguation." *IEEE Transactions on Industrial Informatics* 8 (3): 647–658. doi:10.1109/TII.2012.2188899.

Hachani, S., L. Gzara, and H. Verjus. 2013. "A Service-Oriented Approach for Flexible Process Support within Enterprises: Application on PLM Systems." *Enterprise Information Systems* 7 (1): 79–99. doi:10.1080/17517575.2012.688221.

He, W., and L. Xu. 2014. "Integration of Distributed Enterprise Applications: A Survey." *IEEE Transactions on Industrial Informatics* 10 (1): 35–42. doi:10.1109/TII.2012.2189221.

Hohpe, G., and B. Woolf. 2004. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA: Addison-Wesley.

Izza, S. 2009. "Integration of Industrial Information Systems: From Syntactic to Semantic Integration Approaches." *Enterprise Information Systems* 3 (1): 1–57. doi:10.1080/17517570802521163.

Kopp, O., F. Leymann, and S. Wagner. 2011. "Modeling Choreographies: BPMN 2.0 Versus Bpel-Based Approaches." Proceedings of the 4th international workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011, Hamburg, September 22–23.

Li, L., L. Xu, H. Jeng, D. Naik, T. Allen, and M. Frontini. 2008. "Creation of Environmental Health Information System for Public Health Service: A Pilot Study." *Information Systems Frontiers* 10 (5): 531–542. doi:10.1007/s10796-008-9108-1.

Li, Y., B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin, and Z. Wu. 2014. "An Efficient Recommendation Method for Improving Business Process Modeling." *IEEE Transactions on Industrial Informatics* 10 (1): 502–513. doi:10.1109/TII.2013.2258677.

Linthicum, D. S. 2000. *Enterprise Application Integration*. Reading, MA: Addison-Wesley Longman.

Nezhad, H. R. M., B. Benatallah, A. Martens, F. Curbera, and F. Casati. 2007. "Semi-Automated Adaptation of Service Interactions." In *Proceedings of the 16th International Conference on World Wide Web, Banff, AB*, May 8–12, 993–1002. New York: ACM.

OMG. 2011. "Business Process Model and Notation (BPMN) Version 2.0." Accessed December 11, 2012. http://www.omg.org/spec/BPMN/2.0/

Panetto, H., and J. Cecil. 2013. "Information Systems for Enterprise Integration, Interoperability and Networking: Theory and Applications." *Enterprise Information Systems* 7 (1): 1–6. doi:10.1080/17517575.2012.684802.

Paulraj, D., S. Swamynathan, and M. Madhaiyan. 2012. "Process Model-Based Atomic Service Discovery and Composition of Composite Semantic Web Services Using Web Ontology Language for Services (OWL-S)." *Enterprise Information Systems* 6 (4): 445–471. doi:10.1080/17517575.2011.654265.

Peltz, C. 2003. "Web Services Orchestration and Choreography." *Computer* 36 (10): 46–52. doi:10.1109/MC.2003.1236471.

Quartel, D. A. C., S. Pokraev, T. Dirgahayu, R. M. Pessoa, M. W. A. Steen, and M. van Sinderen. 2009. "Model-Driven Development of Mediation for Business Services Using COSMO." *Enterprise Information Systems* 3 (3): 319–345. doi:10.1080/17517570903045591.

Scheibler, T., R. Mietzner, and F. Leymann. 2009. "Emod: Platform Independent Modelling, Description and Enactment of Parameterisable EAI Patterns." *Enterprise Information Systems* 3 (3): 299–317. doi:10.1080/17517570903042770.

Tan, W., W. Shen, L. Xu, B. Zhou, and L. Li. 2008. "A Business Process Intelligence System for Enterprise Process Performance Management." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38 (6): 745–756. doi:10.1109/TSMCC.2008.2001571.

Tao, F., H. Guo, L. Zhang, and Y. Cheng. 2012. "Modelling of Combinable Relationship-Based Composition Service Network and the Theoretical Proof of its Scale-Free Characteristics." *Enterprise Information Systems* 6 (4): 373–404. doi:10.1080/17517575.2011.621981.

Vernadat, F. B. 2010. "Technical, Semantic and Organizational Issues of Enterprise Interoperability and Networking." *Annual Reviews in Control* 34 (1): 139–144. doi:10.1016/j.arcontrol.2010.02.009.

Viriyasitavat, W., L. D. Xu, and A. Martin. 2012. "Swspec: The Requirements Specification Language in Service Workflow Environments." *IEEE Transactions on Industrial Informatics* 8 (3): 631–638. doi:10.1109/TII.2011.2182519.

Wang, C., and L. Xu. 2008. "Parameter Mapping and Data Transformation for Engineering Application Integration." *Information Systems Frontiers* 10 (5): 589–600. doi:10.1007/s10796-008-9112-5.

Wang, S., L. Li, K. Wang, and J. Jones. 2012. "E-Business Systems Integration: A Systems Perspective." *Information Technology and Management* 13 (4): 233–249. doi:10.1007/s10799-012-0119-8.

Weske, M. 2012. "Process Choreographies." In *Business Process Management*, edited by M. Weske, 243–291. Berlin: Springer.

Xu, L., H. Liu, S. Wang, and K. Wang. 2009. "Modelling and Analysis Techniques for Cross-Organizational Workflow Systems." *Systems Research and Behavioral Science* 26 (3): 367–389. doi:10.1002/sres.978.

Xu, L., W. Tan, H. Zhen, and W. Shen. 2008. "An Approach to Enterprise Process Dynamic Modeling Supporting Enterprise Process Evolution." *Information Systems Frontiers* 10 (5): 611–624. doi:10.1007/s10796-008-9114-3.

Xu, L. D. 2011a. "Enterprise Systems: State-Of-The-Art and Future Trends." *IEEE Transactions on Industrial Informatics* 7 (4): 630–640. doi:10.1109/TII.2011.2167156.

Xu, L. D. 2011b. "Information Architecture for Supply Chain Quality Management." *International Journal of Production Research* 49 (1): 183–198. doi:10.1080/00207543.2010.508944.

Xu, L. D., W. Viriyasitavat, P. Ruchikachorn, and A. Martin. 2012. "Using Propositional Logic for Requirements Verification of Service Workflow." *IEEE Transactions on Industrial Informatics* 8 (3): 639–646. doi:10.1109/TII.2012.2187908.

Yin, Y. H., Y. J. Fan, and L. D. Xu. 2012. "EMG and Epp-Integrated Human–Machine Interface Between the Paralyzed and Rehabilitation Exoskeleton." *IEEE Transactions on Information Technology in Biomedicine* 16 (4): 542–549. doi:10.1109/TITB.2011.2178034.