Mediation Information System Engineering based on Hybrid Service Composition Mechanism

Nicolas Boissel-Dallier, Frederick Benaben, Jean-Pierre Lorré, Hervé Pingaud

Mediation Information System Engineering based on Hybrid Service Composition Mechanism☆

Nicolas Boissel-Dallier[a,b,*], Fréderick Bénaben[a], Jean-Pierre Lorré[b], Hervé Pingaud[a]

[a] *Université de Toulouse - Mines Albi, 4 route de Teillet, 81000 Albi, France*
[b] *Linagora Toulouse, 3 Av Didier Daurat, 31000 Toulouse, France*

Abstract

With the expansion of service-based information systems and the need for organizations to collaborate with external partners, the alignment of business with IT has become crucial. This paper presents a hybrid service composition mechanism, which couples logic-based and syntactic matchmaking of services and messages in order to transform business processes into executable workflows by ensuring service interoperability. This mechanism is based on both top-down and bottom-up approaches, with a view to meeting business requirements for access to available technical services, using a generic semantic profile as a pivotal model. Whereas service matchmaking focuses on functional coverage of generated workflow, message matchmaking involves the generation of the required message transformation.

Keywords: semantic web services, hybrid matchmaking, business process management, interoperability, mediation, information systems

1. Introduction

In today's wide-open business ecosystem, the ability to collaborate with potential partners is a crucial requirement for organizations. Furthermore, collaboration between organizations must be reactive (to seize any relevant collaboration opportunity) and flexible (to remain pertinent to the ongoing situation). In addition, due to the crucial position of information systems (ISs) in organizations nowadays, collaboration between organizations is highly dependent on the ability of their ISs to collaborate [1][2]. For these reasons, the concept of mediation could be extended to the IS domain through a mediation information system (MIS) in charge of providing interoperability to the partners ISs. Such a

☆This document is a collaborative effort.

*Corresponding author
Email addresses: boisseld@mines-albi.fr (Nicolas Boissel-Dallier), benaben@mines-albi.fr (Fréderick Bénaben), jean-pierre.lorre@linagora.com (Jean-Pierre Lorré), pingaud@mines-albi.fr (Hervé Pingaud)

mediator should be able to deal with information management, function sharing and process orchestration in a reactive and flexible way (to support the potential collaboration efficiently). The MISE 2.0 project (for MIS Engineering) deals with the design of such an MIS. MISE 2.0 is based on a model-driven engineering (MDE) approach, dedicated to providing reactivity (by ensuring automated model transformation) and is structured according to the following layers:

- Knowledge gathering (situation layer): collect information concerning the collaborative situation,
- Process cartography design (solution layer): design the processes according to the knowledge gathered,
- MIS deployment (implementation layer): implement an IT structure able to run the process cartography.

The transitions between these layers are the sticking points of this approach: driving an MDE approach in order to deploy a SOA system requires (i) designing a relevant business process cartography from the dedicated situation (first gap) and (ii), from the obtained business processes involving business activities, designing a set of executable workflows invoking technical services (second gap). The first gap is managed at the abstract level of MISE 2.0 while the second is managed at the concrete level of MISE 2.0. By filling both these gaps, one can assume that reactivity can be managed.

Furthermore, MISE 2.0 deployment is based on a service-oriented architecture (SOA) dedicated to providing flexibility (by bridging the gap between design-time and run-time). The use of SOA allows deployment, on the same platform, of both services dedicated to design-time and services dedicated to run-time. Design-time services include a collaborative situation editor (dedicated to gathering collaborative knowledge and to modelling the collaborative situation), a business transformation service (dedicated to deducing the process cartography from the collaborative situation model) and a technical transformation service (dedicated to building the executable workflow files from the process cartography). Run-time services include all the technical services that can be invoked to orchestrate the workflow files. Based on this architecture, flexibility can be assumed: if an adaptation is required, run-time workflow orchestration can be interrupted and, depending on the nature of the adaptation required, any design-time service can be invoked in order to re-design new run-time workflows.

This article will focus on reactivity issues at the concrete level. Our methodology, explained in Section 3 after a presentation of the study context (Section 2), aims at generating executable workflows from business processes. In order to perform this transformation, we first have to bring semantic information to our source models, thanks to our semantic annotation mechanism presented in Section 4. After that, we can make use of this knowledge to find technical services which fit our business needs (see Section 5). In order to ensure good communication between these services, we then use our message matchmaking engine, detailed in Section 6. Finally, Section 7 concludes and presents research perspectives.

2. Positioning and related work

2.1. Mediation Information System Engineering and Interoperability

The first iteration of the MISE project (MISE 1.0, 2006-2010), did provide four design-time services which aimed to (i) characterize a collaborative situation based on a dedicated ontology[3], (ii) extract and transform the knowledge embedded in that ontology in order to model a collaborative business process[3], (iii) transform that collaborative business process model into a logical UML (Unified Modeling Language) model of the mediation information system [4] and finally (iv) transform this logical model into the files required for the deployment of an ESB (Enterprise Service Bus) giving efficient support to the collaborative situation[5]. However, in the first iteration, there were some weak points:

- The collaborative situation ontology is mainly based on the MIT process ontology handbook[6] and is thus essentially dedicated to manufacturing contexts.
- There is only one single collaborative process diagram deduced at the business level (merging decisional, operational and support considerations).
- Matching between business activities (included in the business collaborative process model) and technical services (included in the workflow model and deployed on an ESB) is a manual action.
- Detection of events requiring adaptation and flexibility mechanisms is also a human task.

The second iteration of MISE (MISE 2.0, started in 2009) aims at using the results of MISE 1.0 in order to improve these four drawbacks. As explained in the introduction, it is based on two levels (abstract and concrete), dealing with the two transitions between three layers (situation, solution and implementation).

- By defining our own collaborative situation ontology, we aim to overcome the first drawback[7].
- By using the ISO standard, we aim to deduce a full process cartography (covering decisional, operational and support levels) and thus mitigate the second drawback[7].
- By using semantic annotation of business activities, business information, technical services and technical information, we seek to manage the semantic reconciliation mechanisms [8], in order to avoid the third drawback.
- By using an event-driven architecture, we aim to manage events and automatically detect any needs for adaptation[5]. The fourth drawback can then be avoided.

The current article specifically addresses the third point concerning semantic reconciliation.

2.2. Suitable solution for multiple application domains

Thanks to its flexible and adaptable mechanism, this approach is suitable for any application domain. A first iteration of this mechanism was developed during the ISTA3 project, which focused on sub-contractor collaboration and insisted on information system interoperability as collaboration support [9]. This project only addressed the question of semantic reconciliation (service discovery and message transformation generation) and focussed on the industrial environment (limited agility needs, few potential services, basic performances). This implementation is currently being improved in order to meet more demanding activity domains such as nuclear crisis management[10] or transport issues[11]. As well as those projects needing service and message matchmaking, these studies also took an interest in the other features of MISE 2.0 such as business process generation and event-driven architecture. These features allow the system to detect technical or business problems quickly, analyse them, and then adapt itself accordingly.

2.3. Semantic reconciliation: state of the art

With the expansion of SOA architectures and BPM approaches, web service discovery and composition has become a key factor for IT alignment. A lot of projects focus on it, most of them taking an interest in SWS possibilities.

Service discovery and selection focuses on 1-to-1 service searches depending on functional or non-functional parameters. All service searches are seen independently: neither service exchanges nor message formats are involved here. The SAWSDL-MX2 [12], METEOR-S [13] and FUSION [14] projects take an interest in SAWSDL service discovery. Whereas [13] focuses on syntactic matching based on input and output study only, [14] and [12] both provide a hybrid semantic matchmaking (i.e. coupling logic-based reasoning and syntactic similarity measurement results for better precision), using additional information such as service function. OWLS-MX [15] and WSMO-MX [16] use mechanisms similar to that used by [12] and (hybrid semantic matchmaking), respectively for OWL-S and WSMO. These semantic representations allow a higher expression range than SAWSDL. They allow [15] and [16] to use more information, such as service precondition and effect, to perform service matchmaking. They generate heavy algorithms but improve precision and research possibilities. While previous discovery mechanisms only focussed on functional properties, YASA-M [17] also considers non-functional requirements thanks to its semantic annotation model YASA, an extension of SAWSDL.

Service composition aims at combining several technical services in order to meet a wider need (such as a business requirement) or, at least, brings in process execution to deal with exchanged messages. WSMX [18], IRS-III [19] and SUPER [20] are based on WSMO representation and also use, like [16], most WSMO features such as precondition and effect semantic descriptions. While [18] and [19] settle for "1-to-1" logic-based service matching by means of, respectively, WSMO Choreography and a UPML knowledge model (Unified Problem-solving Method description Language) for process description, [20] uses

dedicated ontology, called sBPMN, to express process logic and provides "1-to-n" composition in order to deal with granularity differences between business activities and technical services. SOA4All [21] defines a lightweight SWS representation based on WSMO (called WSMO-Lite) and its executive environment based on a light WSMX, improving algorithm to provide high-performance "1-to-n" composition. [22] provides service composition based on I/O matching using SAWSDL formalism in order to ensure runtime message exchanges. Finally, [23] takes an interest in service composition regardless of SWS representation using Cartesian products on available services, based on both operation and I/O. These frameworks handle process generation and execution, and thus have to deal with data management between services. [19] chose to entrust message transformation to the user through a graphical interface whereas [18] and [20] aim at generating the required transformation. [18] focuses on a specific semantic data description and [20] on SAWSDL I/O semantic concepts to bind message parts together and uses ontology axioms to generate value transformations. [24] focuses on semantic data matching to generate message transformations. It only considers tagging affiliation and does not handle format or value divergences.

Service discovery projects bring out interesting ideas, especially about hybrid approaches, which improve research precision and recall. However, these projects only look at service-to-service matching whereas our model transformation needs business activity to service matchmaking, which implies management of granularity differences. Additionally, they usually focus on only one SWS representation, whereas we try to embrace several standards (potentially different between organizations). Furthermore, they do not include the area of message management, which is essential for interoperability between partners. Apart from [20], service composition mechanisms use specific languages to express process logic. This forces organizations to redesign existing processes to align them with information systems. Moreover, most of them make do with tag-matching or entrust the user with the task of manually creating message transformation. Our work aims to provide complete message transformation generation, based on a classic BPM standard.

3. From business processes to executable workflows

3.1. Matchmaking methodology

At a concrete level, the main goal is to generate a mediation information system based on business process cartography (from solution layer) and focussed on SOA principles (Service Oriented Architecture). In order to execute abstract processes, we need to generate the appropriate BPEL processes. BPMN 2.0 specification already suggests a BPMN to BPEL syntactic mapping. This mapping allows us to transform processes from one meta-model to another but it does not bring execution-needed information such as real service endpoints or exact exchanged messages. This abstract to concrete transformation involves taking into account both the granularity and conceptual differences.

This model transformation uses both top-down and bottom-up approaches. It is made up of three steps (see Figure 1):

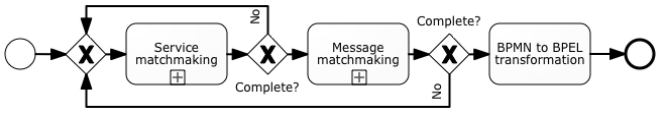

Figure 1: Macro process of hybrid matchmaking.

Step 1 concerns the matching of business activities to technical services. Abstract processes are designed using business activities, which differs from technical services (granularity levels, semantic concepts, etc.). It involves an "n-to-m" matching and ontology reconciliation of concepts from different levels. The syntactic BPMN to BPEL transformation (cf. step 3) is already handled by our BPMN management library, which can then execute it by a Petri-based orchestration engine. So we mainly need to focus on the business to technical process transformation. Therefore, we chose to exploit both the BPMN 2.0 extension mechanism (which brings semantic annotations into business activities, including their inputs, outputs and internal behaviour) and SWS representations (which also brings equivalent semantic annotations into web services). In order to provide reusability and acceptable performances, we have also based our process transformation on a pattern database populated with previous successful tries. The whole process transformation follows these steps:

Step 1.1. We search for existing patterns in our database;

Step 2.2. For uncovered activities, we study the semantic descriptions of these activities and of the available web services (to drive a semantic reconciliation);

Step 3.3. If some activities are still uncovered, we inform the user. He can then choose to develop a new web service, find another partner who already owns it, or entrust our library to generate GUI-based services, which handle expected messages but confide the added-value treatment to an external (human) actor.

Step 2 is dedicated to enabling "on the fly" data transformation. The discovery of web services that fit our functional needs is not enough to generate executable processes and ensure good communication. We also have to provide data interoperability between them. We selected three steps to apply to each web service in order to generate transformation for its inputs:

1. We try to match input concepts with previous service outputs using semantic matching.
2. Thanks to our ontology, we decompose every concept into sub-concepts that are as low as possible.
3. Using our databases, we define syntactic transformation rules (for instance from American date to British date) or mathematic transformation rules (for instance, from Celsius temperature to Fahrenheit temperature).

Steps 1 and 2 are complementary: in order to perform message transformation, we need technical information about inputs and outputs (I/O), i.e. to

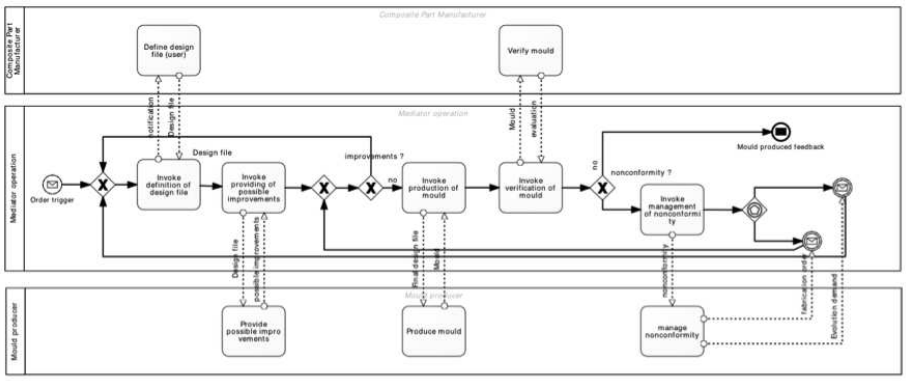

Figure 2: Industrial collaborative use case.

know which technical service is used. Then, step 2 uses step 1 results. If no transformation is possible, we need to find another service to use, through the step 1 mechanism.

Step 3 uses our BPMN-to-BPEL library to create the final BPEL file from the overall BPMN structure of the business process (initial process cartography), selected web services (from step 1) and data transformations (from step 2).

Each step is made up of an automated search for best solutions, using our hybrid approach, then a manual selection, completion or validation by the business user. Except for the n-to-m matchmaking, the whole approach described in this paper has already been implemented as a prototype. A collaboration definition platform was produced for the ISTA3 project. Two main components were used for this: Petals BPM, an online BPMN 2.0 collaborative designer and EasierSBS (for Semantic Business Services), our hybrid matchmaking engine.

3.2. Concrete illustration

In order to illustrate our work, we examine the co-design of composite parts in the Aerospace industry. This example comes from the ISTA3 project. Figure 2 represents a simplified collaboration between two partners: a composite-part manufacturer and its mould producer sub-contractor. This example treats the stages from the definition of the mould design file to the reception of the conforming mould. The collaboration is supported by a mediator, which centralizes the business process and deals with interoperability issues.

4. Semantic management in heterogeneous models

Matching business models with technical ones requires workable links between them. To this end, our matchmaking mechanism proposes using available semantic information from both models. However, in order to facilitate semantic matchmaking between business activities and multiple SWS representations, we have defined a common semantic profile. It embeds the required information

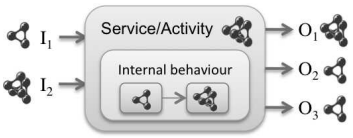

Figure 3: Semantic profile representation.

and is fillable by semantic models (associated to WSDL service descriptions or BPMN 2.0 business process).

4.1. Semantic profile

Before defining a common semantic profile, we need to study our semantic requirements. This common semantic profile aims to facilitate service matching, and is the matchmaking pivot between business activities and technical services. Therefore, it has to bring enough information to allow our matchmaking engine to differentiate services, and then propose the most appropriate ones.

Firstly, our profile needs to express the functional aspects of models in order to cover activity requirements and service capabilities. While a simple functional description slot is enough to enable 1-to-1 matchmaking, it could be interesting to allow a more detailed description in order to improve engine abilities such as n-to-m matchmaking. Then, our semantic profile embeds an internal behaviour description, composed of a sequence of unit activities. Each of these unit activities is represented by a list of semantic concepts, such as functional description.

Secondly, our profile must include high-level semantic description of inputs and outputs. Technical details do not matter for service matchmaking: interoperability problems will be solved during message matchmaking, once real services are chosen, by focusing on XSD annotated schemas or other technical representations. For instance, if we search for an accounting service, which produces an invoice from a reference to current business, we just search for a web service that fills our business needs, without needing to consider the business reference or invoice format.

Figure 3 represents our generic semantic profile. We have defined three distinct parts: inputs, outputs and operation, which embeds description of its internal behaviour. The input and output parts are made up of several semantic elements, whereas semantic unit activities compose the internal behaviour description. This abstract semantic profile can be easily implemented and fulfilled by most SWS representations, or by the following semantic annotation mechanism for BPMN 2.0.

4.2. Semantic annotation for BPMN: SA-BPMN

Whereas a lot of annotation mechanisms exist for web services, the recent BPMN 2.0 is still devoid of a semantic standard. However, among its useful features, such as high design range (from very high level processes to executable workflows), this recent modelling standard brings an XML representation and

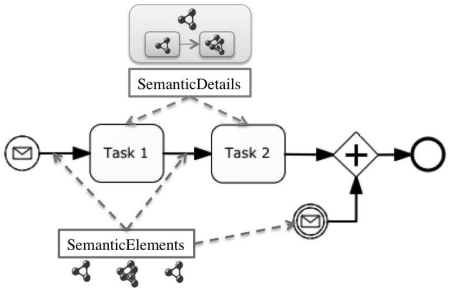

Figure 4: Semantic annotation for BPMN 2.0 (SA-BPMN).

its extension mechanism. Therefore, we decided to propose our own annotation mechanism, called SA-BPMN 2.0 and based on a previous semantic profile. Figure 4 represents our two new XML tags.

The first element, called SemanticDetails, allows any type of BPMN 2.0 task (receive, send, service, user) to be described. It embeds both functional and internal behaviour description. Each one contains a name and a list of URIs (corresponding to semantic concepts from any ontology).

```
<SemanticDetails>
  (<FunctionalDescription name= xs:string
      modelReference= xs:anyUri+ >)?
  (<InternalBehavior>
    (<UnitActivity name= xs:string
       modelReference= xs:anyUri+ />)+
  </internalBehavior>)?
  (<xs:any />)*
</SemanticDetails>
```

The second element, SemanticElements, allows messages and sequencing flows to be described, attaching a list of expected elements. Each element then contains the syntactic name coupled with a list of concepts.

```
<SemanticElements>
  (<SemanticElement name= xs:string
      modelReference= xs:anyUri+ />)+
</SemanticElements>
```

5. Parametrized hybrid service matchmaking

The proposed service matchmaking approach aims to compare business activities and technical services. It is based on a 1-to-1 hybrid matchmaking mechanism and focuses on semantic profile comparison. Semantic distance between profiles is performed thanks to a logic-based reasoning coupled with a syntactic similarity measurement. These measurements use main profile parts: operation (service capability or activity requirement) and I/O.

5.1. Semantic matchmaking

After extracting the semantic profile of the activity, we first compute the semantic distance measurement between this profile and available service profiles. Figure 5 shows the four steps needed to perform this computation for one potential technical service.

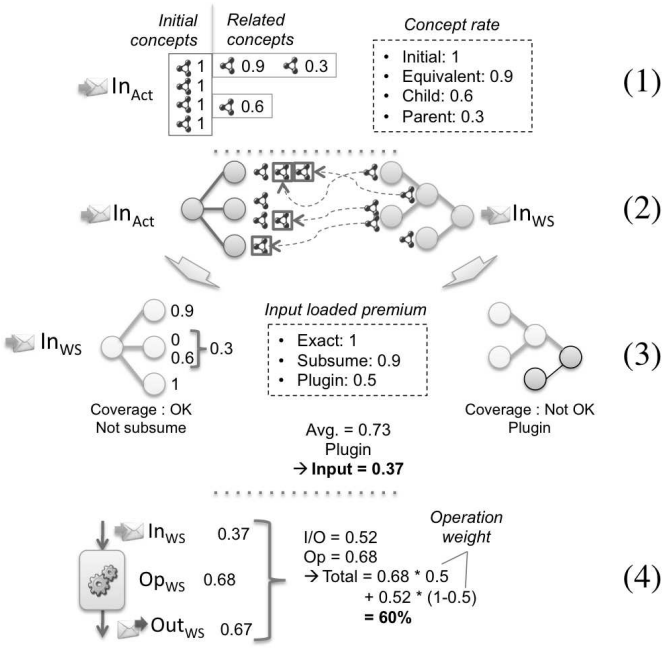

Figure 5: Semantic service matchmaking in details.

Step 1. For each semantic concept involved in the activity profile (noted "Act" in Figure 5), we search for related concepts in available ontologies. We associate a rating to each concept, according to its relation to the initial concepts. We distinguish four types of relations: "exact" for the initial concept, "equivalent" for close concepts (owl:equivalentClass, owl:equivalentProperties or instances of the same class), "child" for specialized classes or instances and "parent" for generalized concepts. Related concept rate (noted r_c) can be cumulated to express more complex relations:

$$r_c = r_{c_{\text{init}},c} = \max_{\text{possible paths}} r_{P(c_{\text{init}},c)} \quad \text{where } r_{P(c_i,c_j)} = \prod_{p \in P(c_i,c_j)} r_p.$$

with c_{init} and c initial and current concepts, $P(c_i, c_j)$ a path from c_i to c_j and p a direct relation between two concepts (e.g. a "parent" relationship).

For example, if we find a specialized (or child) concept in our ontology, we bring this concept into our related list with a 0.6 grade. In the same way, if we find a concept two degrees above (e.g. a parent of parent), we will associate the rate 0.09 (0.3 x 0.3). While exact and equivalent concepts are well rated, child and parent ones are penalized in order to estimate the semantic proximity with the initial concept. If our initial concept was MouldProduction, a subclass IronMouldProduction may be more specific whereas the parent concept Production can be too general for our research.

Step 2. Then, for each initial concept of search profile, we search for the closest concept in the target technical service profile (noted "WS" in figure 5). All related concepts were found by the reasoner in the previous step, so this search involves a very simple (and fast) matching. The rate associated to an initial concept c_i for this WS is defined as: $r_{c_i} = \max_{c \in C_{\mathrm{Act}_{pp}} \cap C_{\mathrm{WS}_{pp}}} r_c$, with $pp \in \{\mathrm{Op}, \mathrm{In}, \mathrm{Out}\}$ the profile part of c_i, $C_{\mathrm{Act}_{pp}}$ and $C_{\mathrm{WS}_{pp}}$ the set of semantic concepts (including inferred ones) involved in the part pp of activity profile (respectively WS profile).
For instance, if the target profile contains one concept corresponding to an equivalent concept of the search profile, we associate the rate 0.9. If two target concepts refer to the same search concept, the best rate is kept.

Step 3. Next, we calculate the rates of semantic distance for each profile part (inputs, outputs and operation), considering semantic concept rates and part coverage. Four cases are taken into account: part coverage is considered as exact if target concepts exactly cover the initial ones, "plugin" if target concepts cover more than expected concepts, "subsume" if any search concept is uncovered and "fail" if there is no concept in common. These four cases considered are similar to those of [12], although the coverage computation and the treatment are different. The penalty applied differs as a function of the studied part. A small penalty is applied for operation and output in the case of "plugin" coverage, a bigger one is used in "subsume" whereas the opposite rule applies to input parts. For output or operation, a "plugin" coverage means the target service provides more functionalities or results than expected whereas "subsume" indicate a lack of functionality. For input, "plugin" means some unsolicited information is expected whereas "subsume" indicate that less information is needed. Algorithm 1 aims at evaluating the coverage of an element tree, returning true if an element or all its children (recursively) are semantically linked ($r_{elem} \neq 0$), and false otherwise. If this function returns false when applied to the expected root element (from expected activity), it means we are in a "subsume" case, whereas if it returns false when applied to the target root element (from web service), we are in a "plugin" case.
Whereas the final rating of the operation is computed thanks to a simple arithmetic average ($r_{\mathrm{Op}} = \frac{1}{|C_{\mathrm{Op}}|} \sum_{c \in C_{\mathrm{Op}}} r_c$, with C_{Op} the set of semantic concepts attached to the operation), the rating of the input or output

Algorithm 1 isCoverageAcceptable

```
function isCoverageAcceptable(element, coverageMap)
    return isCoverageAcceptable(element, coverageMap, new List)
end function

▷ This second function is never called directly but avoids infinite loops due to recursive elements
function isCoverageAcceptable(element, coverageMap, studiedList)
    if element.rate in coverageMap ≠ 0 then
        return true
    else
        if element.children ≠ ∅ AND element ∉ studiedList then
            Add element to studiedList
            for all element.children as child do
                if ¬ isCoverageAcceptable(element, coverageMap, studiedList) then
                    return false
                end if
            end for
            return true
        end if
        return false
    end if
end function
```

part must take into consideration element hierarchy and semantic coverage. In this view, the final rating of input (or output) is computed as a recursive arithmetic average, to which we apply a coverage loaded premium:

$$r_{\text{In}} = P_{\text{In}} * r_e(root_{\text{In}})$$

$$\text{where } P_{\text{In}} = \begin{cases} P_{subsume} & \text{if "subsume"} \\ P_{plugin} & \text{if "plugin"} \\ P_{subsume} * P_{plugin} & \text{if both} \\ 1 & \text{otherwise} \end{cases}$$

$$\text{and } r_e(elem) = \max\left(\frac{1}{|C_{elem}|}\sum_{c \in C_{elem}} r_c, \frac{1}{|Ch_{elem}|}\sum_{child \in Ch_{elem}} r_e(child)\right)$$

with P_{In} the penalty applied to this input, $root_{\text{In}}$ the root element of input, C_{elem} the set of semantic concepts associated to element $elem$ and Ch_{elem} the set of $elem$'s children.

Step 4. Lastly, the final semantic rating is computed by applying an importance weighting between the three rates (inputs, outputs, operation):

$$r_{\text{WS}} = w_{\text{Op}} * r_{\text{Op}} + (1 - w_{\text{Op}}) * r_{\text{I/O}}$$

$$\text{where } r_{\text{I/O}} = \begin{cases} \frac{r_{\text{In}} + r_{\text{Out}}}{2} & \text{if activity has output(s)} \\ r_{r_{\text{In}}} & \text{otherwise} \end{cases}$$

with $w_{\text{Op}} \in [0; 1]$ the importance weighting of the operation.

Of course, steps 2 to 4 are computed for each technical service profile in order to obtain a semantic distance measurement for each available service. A semantic threshold can be defined by the user to automatically eliminate

irrelevant services before the next step then lighten the syntactic matchmaking computation.

As stated above, all weights and penalties can be changed between computations by the user according to IS and ontology specificities.

5.2. Syntactic matchmaking

Our syntactic similarity measurement uses classic similarity metrics such as Extended Jaccard or Dice [25]. All these metrics focus on word occurrence comparison. For this computation, we no longer use logic-based mechanisms focussing on relations between semantic concepts, but instead we only study used words, both technical names used in service description and concept names themselves. The same three groups of concepts are considered: inputs, outputs and operation. Figure 6 represents the four steps of our similarity measurement approach:

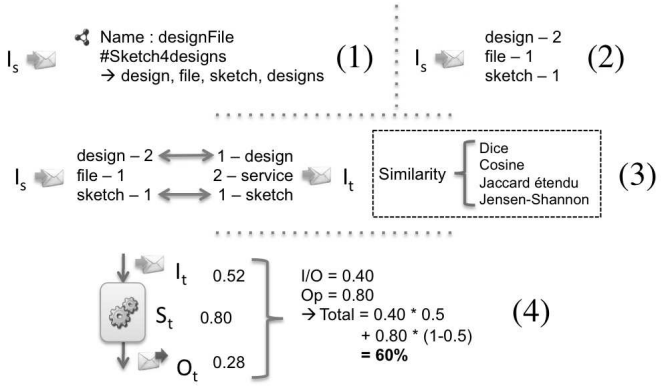

Figure 6: Syntactic service matching in detail.

Step 1. For each group of both business activity and technical service profiles, we extract potential words from its name and concept syntactic names. In this view, we make use of main writing conventions, such as camel case or punctuation uses. For instance, we can extract 4 words from concept name "#very_importantConcept4instance": very, important, concept and instance.

Step 2. Then, we enumerate word occurrences taking into account small differences. In this view, we use as an error margin the Levenshtein distance [26], which computes the number of necessary changes to transform one word into another. For example, the Levenshtein distance between "design" and "designs" is only 1 so we consider those as quite identical. The Porter algorithm [27], which compares word roots, is often preferred in information retrieval mechanisms to compare word proximity thanks to its wider recognition of close words. However, it only works on English words whereas Levenshtein can be applied to every language.

Step 3. Thanks to similarity metrics, we can then measure the distance between groups of words. Several similarity measurement techniques exist and we implemented four of them: Cosine, Dice, Extended Jaccard and Jensen-Shannon. However, large-scale syntactic matchmaking shows similar results for these four metrics. By default, we chose Dice similarity measurement, which is slightly faster than the others. This computation returns a proximity rate, between 0 and 1, for each profile part: $sim_{pp}^{Dice} = \frac{2|A \cdot B|}{|A|^2+|B|^2}$, with A and B the normalized vectors for the activity profile part and target web service, containing occurrence numbers of each word that appears in one or the other (enumerated during the previous step).

Step 4. Finally, once the three groups have been rated for a specific technical service (input, output and operation), we can compute its final mark as we did for semantic distance, using the same importance weighting.

Whereas the first two steps can be performed once for the search profile, others must be done for each potential target service (according to semantic matchmaking). Another threshold mechanism allows our matchmaking engine to automatically exclude irrelevant services before sending results to the user.

Next, we combine the two previous ratings using an importance weighting of the logic-based computed mark compared to the syntactic one, to obtain a final distance rate: $r_{\text{final}} = w_{\text{logic}} * r_{\text{logic}} + (1 - w_{\text{logic}}) * r_{\text{syntact}}$, with $w_{\text{logic}} \in [0; 1]$ the importance weighting of the semantic, r_{logic} and r_{syntact} the semantic and syntactic ratings. Finally, results are sent to the user who can choose the most appropriate technical services from among only a few relevant services rather than the thousands of available ones.

5.3. Evaluation of performance

In order to evaluate our service matchmaking engine performances, we used the SAWSDL test collection SAWSDL-TC2[1]. This test suite contains 42 queries and their expected responses over 1080 services and 38 ontologies. It allows us to conduct a retrieval performance evaluation on a large base of semantically annotated service descriptions covering several application domains, such as travel, education, food or medical care. Even if these SAWSDL only contain semantic annotations on a I/O schema description, it remains the best way to test a service matchmaking engine while there is currently no other test collection as complete as this one available. The performance tests were performed on a machine with 2Ghz i7 CPU and 2Go RAM on Mac OS X and Java 6.

All performance measurements are based on two classic retrieval information indicators: precision and recall (noted P and R in the following equations). The precision indicator represents the quality of results (percentage of relevant documents among retrieved ones) whereas the recall indicator shows the sufficiency of the result (percentage of retrieved documents among relevant ones). From

[1] http://semwebcentral.org/projects/sawsdl-tc/

these two indicators, we calculated the average $F_{measure}$ for all queries and the Mean-Average Precision (MAP):

$$F_{measure} = 2 \cdot \frac{P \cdot R}{P + R}$$

$$\text{where } P = \frac{|retrieved \cap relevant|}{|retrieved|} \text{ and } R = \frac{|retrieved \cap relevant|}{|relevant|}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \qquad \text{where } AP = \frac{\sum_{k=1}^{n} P(k) \cdot Rel(k)}{|relevant|}$$

where $AP(q)$ is the average precision for the query q, Q is the number of queries, $P(k)$ represents the precision for the first k results (from the ordered retrieved documents) and $Rel(k)$ is equal to 1 if the kth document is relevant, 0 otherwise. While $F_{measure}$ combines recall and precision for a whole query result, MAP represents precision evolution along ordered results. Finally, we depicted the precision/recall graph, representing the evolution of precision functions of recall, computed from precision and recall values for each k^{th} first retrieved documents of each query result.

Table 1: Evaluation performance of our service matchmaking engine

	Semantic	Syntactic	Syntactic (Leven.)	Hybrid	Filtred Hybrid
MAP	0.59	0.52	0.52	0.54	0.58
$F_{measure}(\in [0;1])$	0.45	0.48	0.48	0.57	0.48
Avg. time (ms)	512	283	971	751	664

We studied five service matchmaking configurations: semantic only, syntactic only (with and without Levenshtein limit), hybrid and hybrid with filter between semantic and syntactic selection. Table 1 summarizes average times, MAP and $F_{measure}$ for all cases, whereas Figure 7 shows recall/precision graphs for these configurations. Results for both syntactic matchmaking cases coincide. For this reason, we depicted only one curve in the figure.

First, these results show the inefficiency of the Levenshtein distance margin in our matchmaking algorithm. Results are quite similar with or without this feature while Levenshtein recognition increases the average execution time four-fold. Secondly, we notice that both hybrid- and semantic-only approaches provide interesting results. The logic approach offers better document ordering (represented by recall/precision graph and MAP value) whereas hybrid approach retrieves more precise document set (higher $F_{measure}$). Finally, these benchmarks indicate that the setting-up of a filter between semantic and syntactic matchmakings reduces the interest of the hybrid approach, the results being close to the semantic-only approach in spite of additional computation.

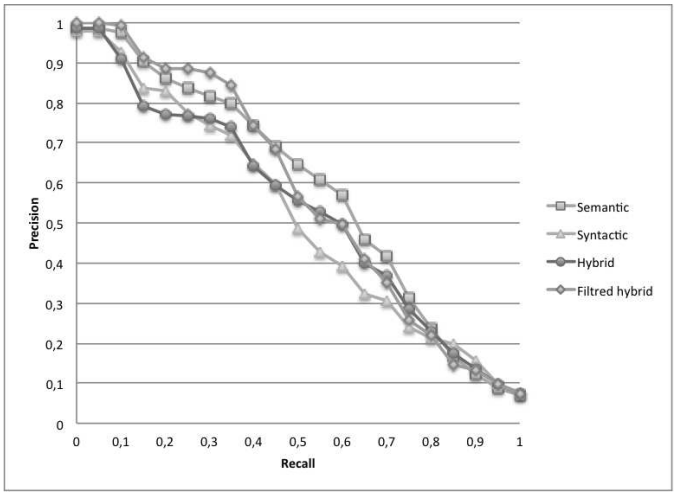

Figure 7: Service matchmaking performance results.

According to these benchmark results, the advantages of a hybrid approach appear limited. However, it should be remembered that SAWSDL-TC only contains fully annotated service descriptions, which cannot be the case in real industrial situations.

5.4. The quest for process reconciliation

Thanks to internal behaviour descriptions coupled with the previous hybrid mechanism, it became possible to create semantic profiles of atomic functions in order to match them. These profiles are only partially filled because of lack of internal information descriptions but they still provide some mapping trails that we can confirm using available inputs/outputs description. Unfortunately, this approach involves a combinatorial computation. For each possible group of activities, we theoretically have to test each possible group of technical services. In order to limit possible associations, we considered some simple filters:

- Some information about collaboration, such as target partner, activity domain and non-functional requirements are provided by both business process and service descriptions and can be used as filters. These filters do not require time-consuming computation and can be performed before our hybrid matchmaking while this information is contained in our technical registry. For instance, if the collaboration involves one preselected mould producer and the business process specifies the need to use secured exchanges, we can perform matchmaking that only considers secured services from this particular service provider.

- We cannot consider bringing together activities that are randomly selected. The process logic can also reduce possible groups. Any business process implies connections between activities that we cannot avoid (i.e.

sequences, gateways). Figure 8 illustrates possible logic groups of business activities that can be considered for semantic reconciliation. From millions of possible groups, there are only 22 useable groups.

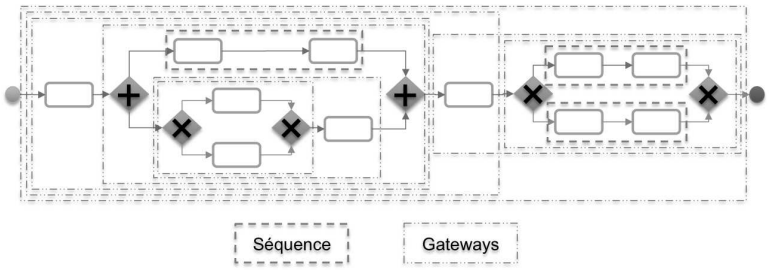

Figure 8: Process logic study for combinatorial computation reduction.

For now, this n-to-m matchmaking remains very simple. Implementation and tests are in progress and the list of possible improvements is still lengthening. Some recent external research, such as [28] is contributing promising new ideas for improving the precision of this approach, and we are still working on these improvements too.

6. Ensuring message interoperability

Once the user has selected technical services, we can focus on real data mapping. The discovery of web services that fit our functional needs is not enough to generate executable processes and ensure good communication between partner' ISs. We also have to provide interoperability between these services.

6.1. Technical databases

Each service invocation needs a specific input message to be correctly executed. Expected tags are described in the service description (name, data type) but they must be filled with relevant data in exact format. They usually dont bring information, but organise it. Furthermore, service descriptions can only depict standard data types. For expected input described as a simple string, the real service can expect a date in an exotic format but service descriptions are not precise enough to express it.

Semantic business information is not sufficient for message matchmaking. One business concept such as delivery date can be expressed in many formats (XML Datetime, US date format). This choice belongs to the service developer who can also use classic XML Datetime, declared as such in the service description, or choose to use an exotic one, declared as a simple string. In order to solve this problem, we propose a technical ontology focused on format concepts and linked to three technical databases filled with syntax representation and conversion formulae. The factorization database expresses possible decompositions using concept URIs as value representations. For instance, using the

three decomposition examples below, we can factorize a complex concept such as XsDatetime into unit concepts and its embedded syntax (#XsYear-#XsMonth-#XsDayT#XsHour:#XsMinute:#XsSecond).

```
#XsDatetime = %%#XsDate%%T%%#XsTime%%
#XsDate = %%#XsYear%%-%%#XsMonth%%-%%#XsDay%%
#DateUs = %%#XsMonth%%-%%#XsDay%%-%%#XsYear%%
```

The formula database is close to the previous one, except that it expresses conversion formulae between two technical concepts. Here is an example of conversion between millimetre and inch measurements:

```
#Milimeter = %%#Inch%% * 25.4
```

Finally, the regex database links those technical concepts to their regular expressions (e.g. #XsYear = [0-9]4). This technical information will allow our transformation engine to treat real values in run time. Those technical concepts can then be used to annotate input/output descriptions in order to support our transformation generation mechanism.

6.2. Message matchmaking approach

Each web service can be invoked using a specific XML format as input. This format is described in its WSDL description, which stipulates names, data types and tag hierarchy. Ensuring service interoperability involves creating the expected input, using the available data in correct format. In this view, our message matchmaking mechanism aims at generating the correct message transformation, considering all parameters such as process logic, complex data formats, unknown/undefined data format, etc. The proposed matchmaking process, presented in Figure 9, tries to cover all these possible cases in order to provide a complete and adaptable solution:

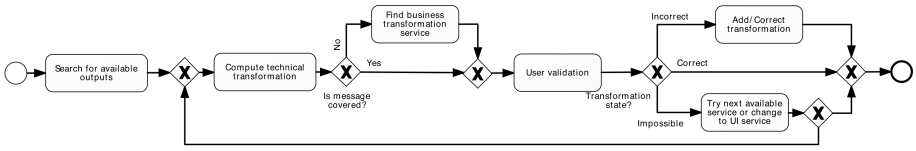

Figure 9: Overview of message transformation approach.

Step 1. First we search for available outputs using process logic. We have to find out which previous output messages can be used to create our input target message, according to previous activity states at target activity call.

Step 2. Then, using this available data, we try to compute the whole message transformation according to the semantic link between tags, format descriptions and technical information about known transformations.

Step 3. If the whole message is not covered by the computed transformation, we first try to find an available transformation service using our service matchmaking mechanism described above. This mechanism works

well for simple messages, but is limited for really complex information such as CAD formalisms where thousands of tags are involved. However, this kind of business transformation is usually performed by high value-added services provided by software editors themselves (e.g. file transformation between two major versions of a specified software).

Step 4. Result transformation is submitted to the user for validation or completion. Three cases are then possible:

- Computed transformation is correct and covers the whole input message. In this case, the user can directly validate it. The transformation is generated in computer-readable language then added into the workflow description.
- The transformation is feasible using data from the previous service but is incorrect or incomplete for now. In this case, the user can manually complete or correct the transformation. The technical database is then updated and this new information will then be useable for future transformations.
- Transformation is impossible according to available information. The user can then choose to (i) try with another possible service (result from service matchmaking), (ii) search for a new partner who proposes expected service or (iii) generate a business service, a simple user interface, which allows the user to bring in the missing information. In every instance, the chosen service needs a dedicated message transformation. The whole generation process is then resumed from transformation computation.

The following part details steps (1) and (2) of this generation process, which concentrate the majority of our added value. First, we look at the selection of available messages based on process logic study, then the computation of the transformation file.

6.3. Selection of available messages

The first step in this generation is the selection of available data from formerly sent messages. According to the process logic, some technical services will not be invoked depending on gateways and runtime conditions. Here again, we distinguish several cases, which cover most possible scenarios. Figure 10 illustrates those cases, the considered service being the one annotated with a "?":

- Some services (shaded dark in the figure) will inevitably be executed before the target service, whatever the runtime conditions or values in each process execution may be. All these service outputs can be used to create the target input. If two available outputs are identical, the most recent will be favoured by the generation engine.
- Others may or may not be called during the process execution. Their output messages (and associated values) may not be available (light ones in the figure):

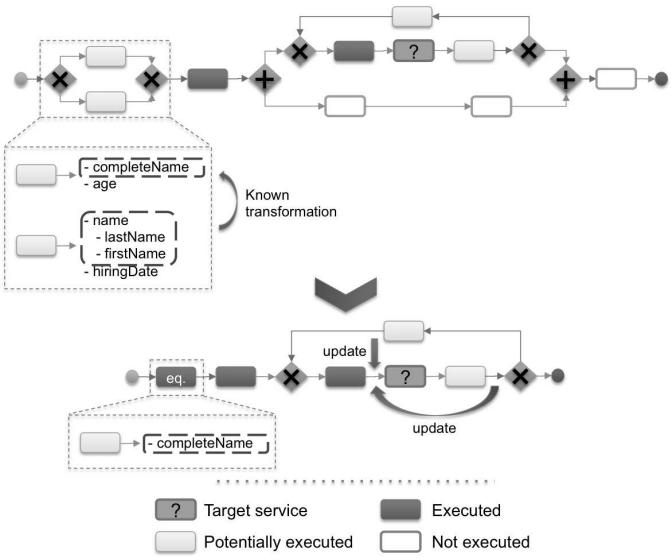

Figure 10: Selection of available messages.

  - Services from prior conditional branches: their outputs can be considered only if all process branches share the same semantic concepts, i.e. produce the same data (possibly with different values). Hybrid matchmaking is then performed in order to compare available outputs. If a transformation can be computed by our engine, a new variable is created in the process description in order to store the runtime value in a common format. This variable will be filled at all process executions and can then can be used to create the target input.
  - Services placed downstream from our target service but within a common loop: their outputs are not available at the first call but can be used if the way back is used to update previous values. The mechanism is quite similar to the previous one: hybrid matchmaking is performed to determine which data can be updated by the concerned services and the workflow description is consequently modified.

- Finally, remaining services (the white ones in the figure) cannot be called at all before the target service. These outputs are simply excluded from considered data.

Once all available output has been selected, with associated ratings depending on their distance from the target service, we can begin the transformation computation (Step 2).

6.4. Generation mechanism

In our specified environment, and according to the low coupling principle, all exchanged messages are written in XML. Few methods focus on XML transformations, but one of them has broken away from the others thanks to its maturity and flexibility: XSLT (for Extensible Stylesheet Language Transformations). This mechanism aims at generating accurate XSLT transformation for each service using previously selected available messages and their semantic annotations.

Service I/O are described into XML Schema formalism embedded in WSDL files. This format description is limited to standardized data types and all exotic formats are only specified as simple string. Unfortunately, added semantic information is not sufficient to overcome the lack of format description. One business concept such as delivery date can be expressed in many formats (XML Datetime, US date format), according to the service developers choice. In order to solve this problem, we propose a technical ontology focused on format concepts and linked to a technical database filled with syntax representation and conversion formulae. This database contains the following tables.

- The factorization table details possible concept decompositions using URIs as value representation. For example, a complex concept such as #XsDatetime (in XML format) can be factorized into unit concepts as #XsYear-#XsMonth-#XsDayT#XsHour:#XsMinute:#XsSecond.
- The formula table, which expresses conversion formulae between two technical concepts. For instance, conversion between millimetre and inch measurements (#Millimetre = #Inch x 25.4).
- Finally, the regular expression table links those technical concepts to their syntactic description (e.g. #XsYear = [0-9]{4} means the year is written with 4 digits). This technical information will allow our transformation engine to treat real values at runtime.

The following generation mechanism is based on both XML schema syntactic description and its annotated technical concepts. The generation mechanism can be split into three steps (see Figure 11):

Step 2.1. First, thanks to our hybrid matchmaking engine and list of available data (see section 6.3), we match target tag concepts with available source concepts. This enables us to find equivalent tags, without considering data format for the moment. For instance, the selected service expects, among others, a date in a XML Datetime format and a length in millimetres. Thanks to the hybrid matchmaking engine, we find some close potential tags from selected outputs: Date (US format) and Time that cover the Datetime embedded concepts and LengthInch, which corresponds to the expected length expressed in inches.

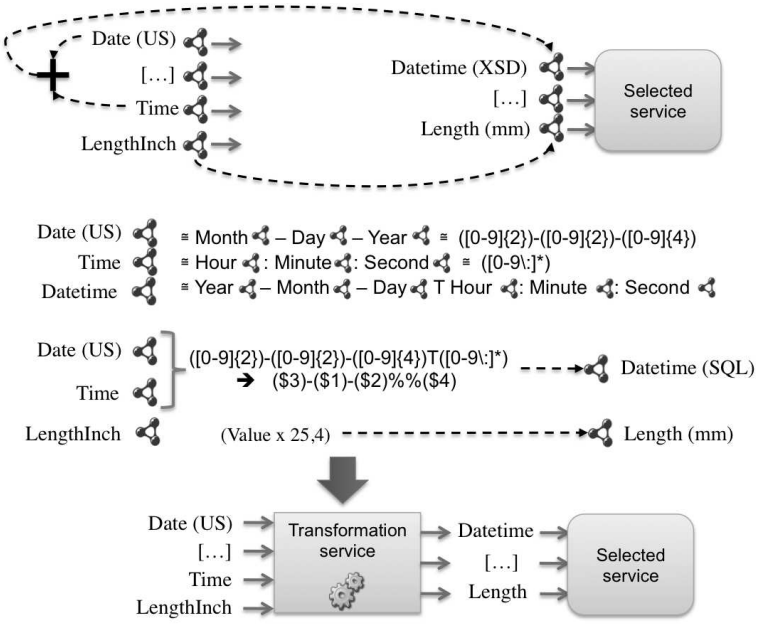

Figure 11: Generation of XPath functions.

Step 2.2. Next, once previous values fitting our functional needs have been selected, we can focus on the data and format matching, using the technical description of formats from our databases (described above), we can now deduce the XPath function, which allows us to transform available values into expected ones. To this end, we first factorize both available and expected concepts into finest-grained concepts in order to compare concepts of similar granularity. We then try to match target unit concepts with previous output concepts, generating the corresponding XPath function.
As you can see in Algorithm 2 (in Annexe Appendix A, an example is given in the comments on XsDatetime generation from Date US and XsTime), we first use the factorization database to decompose the expected value description into unit concepts (L4). For each potential value we also decompose it into unit technical concepts in order to match concepts of the same granularity (L12). If one concept from this potential value corresponds to an expected unit concept, we replace this concept in its decomposition with its regular expression and associate the expected concept to a unique key (needed for final XPath replacement, L16-22). Once all potential concepts have been treated, we search for uncovered expected concepts in order to find a formula conversion (L30-33). If we find one for each uncovered concept, we can generate the XPath transformation using collected information (used potential values, their regular expression and associated key or

formula).). Thus, this algorithm produces a general XPath formula to create an expected value from previously available values (L39-57). In real implementation, ids of potential values correspond to the real value path in the message and concepts are referred to by their complete URIs. Incidentally, the "searchFormula" method is not detailed here, but it simply checks coverage of concepts in formulas with potential concepts.

Step 2.3. Finally, once all target elements have been covered by generated XPath functions, we turn to the element structure itself (tag hierarchy) in order to generate the corresponding XSLT. The XPath function allows only conversion from one tag to another (e.g. a delivery date unique in the message) but is not sufficient for complex element management. In order to manage such hierarchy complexity it is necessary to use both BPMN and XSLT features. Table 2 summarizes the required XSLT or BPMN functions depending on source and target tag occurrences (resp. vertically and horizontally):

- If both source and target are not mandatory, transformation is possible but the source value must be tested (using XSLT if function) to avoid generation of a useless tag in the target.
- Else if source value is not mandatory whereas the target is, we cannot ensure the transformation. In this case, the source must be changed or completed.
- Else if source value is both mandatory and unique, no test is needed and transformation only uses XPath features.
- Else if source value is not unique (e.g. some lines in an estimate description) but the target tag allows one value maximum, the target service must be invoked once per possible value. In this case, we must change the BPMN activity type to a loop.
- In the last case (multiple source and target values), we must use the XSLT function "for-each", which allows our transformation engine to copy several values from an XML message to another, applying the same XPath function for each.

Table 2: XSLT tags according to element occurrence

in/out ↬	0..1	1..1	0..∞	1..∞
0..1	if ∃	X	if ∃	X
1..1	✓	✓	✓	✓
..∞	↻ BPMN	↻ BPMN	for-each	for-each

These generated transformations cover most format divergences. They are included directly in the BPMN definition as XSLT transformation in order to

be exported as external services during the BPMN to BPEL automated transformation. If this transformation is not complete after this computation, the message-matchmaking process continues as described in section 6.2.

7. Perspectives and conclusion

Driving a BPM approach is a very common solution to improve the maturity of organizations and enterprises. Such an approach is usually dedicated to (i) obtaining a certification, (ii) diagnosis of some dysfunctions and improvement of the overall behaviour or (iii) providing requirements to improve the IS. However, concerning point (iii), there is still a large gap between the business considerations regarding the obtained process cartography and the technical world of IS deployment. This work aims to fill that gap with tangible theoretical mechanisms and efficient software tools. To this end, we propose a complete and fully implemented methodology, based on SWS features, which covers transformation from business process cartography to executable workflows. The semantic issues, described in [29] are addressed on the three required consideration levels (information, functions, processes). The most important perspectives mainly concern the use of non-functional requirements (and governance issues) in the reconciliation mechanism in order to enlarge the considered criteria during the selection phase. Elements such as latency, security or efficiency requirements may be embedded in business process models in order to be taken into account during the semantic reconciliation step (this step might thus be refined and improved). Documentation and source code of these research components is available at: http://research.petalslink.com. For any information about other products such as Petals ESB and EasyBPEL, our orchestration engine, please visit: http://docs.petalslink.com.

References

[1] F. Bénaben, W. Mu, S. Truptil, H. Pingaud, J. Lorré, Information systems design for emerging ecosystems, in: Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on, 2010, pp. 310–315.

[2] F. Bénaben, N. Boissel-Dallier, J. P. Lorré, H. Pingaud, Semantic reconciliation in interoperability management through Model-Driven approach, in: Collaborative Networks for a Sustainable World, 2010, pp. 705–712.

[3] V. Rajsiri, Knowledge-based system for collaborative process specification, Thse en systmes industriels, INPT - EMAC (2009).

[4] J. Touzi, F. Benaben, H. Pingaud, J. P. Lorré, A model-driven approach for collaborative service-oriented architecture design, International Journal of Production Economics 121 (1) (2009) 5–20.

[5] S. Truptil, Etude de l'approche de l'interoprabilit par mdiation dans le cadre d'une dynamique de collaboration applique la gestion de crise, Ph.D. thesis, INPT - EMAC, Albi (2011).

[6] T. Malone, K. Crowston, G. Herman, Organizing business knowledge: the MIT process handbook, the MIT Press, 2003.

[7] W. Mu, F. Bénaben, H. Pingaud, N. Boissel-Dallier, J. Lorré, A model-driven BPM approach for SOA mediation information system design in a collaborative context, in: Services Computing (SCC), 2011 IEEE International Conference on, 2011, pp. 747–748.

[8] H. Hoang, P.-C. Tran, T. Le, State of the art of semantic business process management: An investigation on approaches for business-to-business integration, in: N. Nguyen, M. Le, J. Swiatek (Eds.), Intelligent Information and Database Systems, Vol. 5991 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 154–165.

[9] N. Boissel-Dallier, F. Bnaben, H. Pingaud, Model-driven engineering of mediation information system: application to the ISTA3 use-case, in: Proceedings of the sixth international conference of I-ESA, Springer, Valencia, Spain, 2012.

[10] A.-M. Barthe, M. Lauras, F. Bnaben, 8th international conference on information systems for crisis response and management : From early-warning systems to preparedness and training, in: Proceedings of the 8th International Conference on Information Systems for Crisis Response and Management (ISCRAM), Lisbon, 2011.

[11] G. Mac Ramte, F. Bnaben, J. Lamothe, Towards an agile information decision support system in transport crisis context, in: Proceedings of the I-ESA conference, Valencia Spain, Springer, Valencia, Spain, 2012.

[12] M. Klusch, P. Kapahnke, I. Zinnikus, Sawsdl-mx2: A machine-learning approach for integrating semantic web service matchmaking variants, in: Web Services, 2009. ICWS 2009. IEEE International Conference on, 2009, pp. 335–342.

[13] K. Sivashanmugam, J. A. Miller, A. P. Sheth, K. Verma, Framework for semantic web process composition, International Journal of Electronic Commerce 9 (2) (2005) 71–106.

[14] S. Alexakis, M. Bauer, A. Pace, A. Schumacher, A. Friesen, A. Bouras, D. Kourtesis, Application of the fusion approach for assisted composition of web services, Establishing The Foundation Of Collaborative Networks (2007) 531–538.

[15] M. Klusch, B. Fries, K. Sycara, Automated semantic web service discovery with OWLS-MX, in: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 2006, pp. 915–922.

[16] F. Kaufer, M. Klusch, Wsmo-mx: A logic programming based hybrid service matchmaker, in: Web Services, 2006. ECOWS'06. 4th European Conference on, 2006, pp. 161–170.

[17] Y. Chabeb, S. Tata, A. Ozanne, Yasa-m: A semantic web service matchmaker, in: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 2010, pp. 966–973.

[18] F. Facca, S. Komazec, I. Toma, WSMX 1.0: A further step toward a complete semantic execution environment, The Semantic Web: Research and Applications (2009) 826–830.

[19] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, C. Pedrinaci, IRS-III: a broker-based approach to semantic web services, Web Semantics: Science, Services and Agents on the World Wide Web 6 (2) (2008) 109–132.

[20] M. Hepp, F. Leymann, J. Domingue, A. Wahler, D. Fensel, Semantic business process management: A vision towards using semantic web services for business process management, in: IEEE International Conference on e-Business Engineering, 2005. ICEBE 2005, 2005, pp. 535–540.

[21] F. Lécué, Y. Gorronogoitia, R. Gonzalez, M. Radzimski, M. Villa, SOA4All: an innovative integrated approach to services composition, in: Web Services (ICWS), 2010 IEEE International Conference on, 2010, pp. 58–67.

[22] F. Ishikawa, S. Katafuchi, F. Wagner, Y. Fukazawa, S. Honiden, Bridging the gap between semantic web service composition and common implementation architectures, in: Services Computing (SCC), 2011 IEEE International Conference on, 2011, pp. 152–159.

[23] N. Guermouche, O. Perrin, C. Ringeissen, A mediator based approach for services composition, in: Sixth International Conference on Software Engineering Research, Management and Applications, 2008. SERA '08, IEEE, 2008, pp. 273–280. doi:10.1109/SERA.2008.43.

[24] D. Gagne, M. Sabbouh, S. Bennett, S. Powers, Using data semantics to enable automatic composition of web services, in: Services Computing, 2006. SCC'06. IEEE International Conference on, 2006, pp. 438–444.

[25] W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string metrics for matching names and records, in: KDD Workshop on Data Cleaning and Object Consolidation, Vol. 3, 2003.

[26] V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet Physics Doklady, Vol. 10, 1966, pp. 707–710.

[27] M. Porter, An algorithm for suffix stripping, Program: electronic library and information systems 14 (3) (1980) 130–137.

[28] C. Gerth, M. Luckey, J. Küster, G. Engels, Precise mappings between business process models in versioning scenarios, in: Services Computing (SCC), 2011 IEEE International Conference on, pp. 218–225.

[29] F. Benaben, N. Boissel-Dallier, H. Pingaud, J.-P. Lorre, Semantic issues in model-driven management of information system interoperability, International Journal of Computer Integrated Manufacturing (IJCIM)doi:10.1080/0951192X.2012.684712.

Appendix A. XPath generation algorithm

Algorithm 2 generateXPath

```
function GENERATEXPATH(targetSyntax, potentialSyntaxList)
                                                    ▷ targetSyntax: %%#XsDatetime%%
                                ▷ potentialSyntaxList: $id1=%%#DateUs%%; $id2=%%#XsTime%%
    targetSyntax.factorize()
                                                    ▷ targetValue: %%#XsYear%%-%%#XsMonth%%-
%%#XsDay%%T%%#XsHour%%:%%#XsMinute%%:%%#XsSecond%%
    for all targetSyntax.concepts as targetConcept do
        Add targetConcept in coverageMap
        coverageMap.targetConcept.value ← 0
    end for
    key ← 1
    for all potentialSyntaxList.elements as potentialSyntax do
        potentialSyntax.factorize()
                                        ▷ $id1 = %%#XsMonth%%-%%#XsDay%%-%%#XsYear%%
        for all potentialSyntax.concepts as potentialConcept do
                                                                        ▷ #XsMonth
            if potentialConcept ∈ coverageMap then
                Replace potentialConcept with "(" + potentialConcept.regExp + ")" in
potentialSyntax
                                                    ▷ $id1 = ([0-9]2)-%%#XsDay%%-%%#XsYear%%
                coverageMap.potentialConcept.value ← "$" + key
                                                                    ▷ #XsMonth = $id1
                key ← key + 1
                Add potentialSyntax to usedPotentialSyntax
            else
                Replace potentialConcept with potentialConcept.regExp in potentialSyntax
            end if
        end for
                                                    ▷ $id1([0-9]2)-([0-9]2)-([0-9]4)
    end for
    for all coverageMap.targetConcepts as targetConcept do
        if coverageMap.targetConcept.value = 0 then          ▷ 0 = uncovered concept
            formula ← SEARCHFORMULA(potentialSyntaxList,usedPotentialSyntax)
            if formula ≠ null then
                coverageMap.targetConcept.value ← formula
            else                                                ▷ Concept still uncovered
                return null
            end if
        end if
    end for
    if usedPotentialSyntax.size > 1 then
        source ← "fn:string-join("
    end if
    for all usedPotentialSyntax.elements as potentialSyntax do
        source ← source + potentialSyntaxList.potentialSyntax.id + ","
        pattern ← pattern + potentialSyntaxList.potentialSyntax.value + "  "
    end for
    Replace last "," with "" in source
    source ← source+ "' ')"
    Replace last "  " with "" in pattern
                                                ▷ source: fn:string-join ($id1, $id2,' ')
                                ▷ pattern: ([0-9]2):([0-9]2):([0-9]2)  ([0-9]4)-([0-9]2)-([0-9]2)
    for all coverageMap.targetConcepts as targetConcept do
        Replace targetConcept with key in targetSyntax
    end for
                                                ▷ targetSyntax = ($4)-($5)-($6)T($1):($3):($2)
    xpath ← "fn:replace(" + source + "," + pattern + "," + targetSyntax + ");"
                                                ▷ fn:replace(fn:string-join ($id1,$id2,' '),
([0-9]2):([0-9]2):([0-9]2)  ([0-9]4)-([0-9]2)-([0-9]2),($4)-($5)-($6)T($1):($3):($2));
    return xpath
end function
```