An Intentional Approach to Service Engineering

C. Rolland, M. Kirsch-Pinheiro, C. Souveyet

Abstract— Despite its growing acceptance, Service-Oriented Computing (SOC) remains a computing mechanism to speed-up the design of software applications by assembling ready-made software services. We argue that it is difficult for business people to fully benefit of SOC if it remains at the software level. The paper proposes a move towards a description of services in business terms, i.e. intentions and strategies to achieve them and to organize their publication, search and composition on the basis of these descriptions. In this way, it leverages SOC to an intentional level, ISOC. We present ISM, the model to describe intentional services, and populate the service registry with their descriptions. We highlight its intention driven perspective for service description, retrieval and composition. Thereafter, we propose a methodology to determine intentional services that meet business goals and to publish them in the registry. Finally, the paper introduces a set of transformations to bridge the gap from the intentional level to the implementation one.

Index Terms— Intentional service-oriented architecture, Intentional service modeling service-oriented architecture, service-oriented computing.

1 INTRODUCTION

Service-Oriented Computing (SOC) is the software engineering paradigm that utilizes services as fundamental elements for developing software applications [23] [24]. SOC relies on the Service-Oriented Architecture (SOA) [1] that is a way of reorganizing a portfolio of legacy applications into services that are self-describing computational elements, which are platform independent, accessible through standard interfaces and can be assembled in complex compositions based on standard messaging protocols. As shown in Fig.1, the basic SOA defines an interaction between three kinds of software agents [24], namely the service provider, the service client and the service registry involving the publish, find and bind operations. Services are offered by service providers that procure the service implementations and supply their descriptions to a service registry. The service registry publishes services by exhibiting their descriptions. The service requester uses the find operation to retrieve the service description matching his functional needs and uses it to bind with the service provider and invoke the service.

SOC is a way of designing a software system that is function-driven. Services perform functions implemented in software, wrapped with formal documented interfaces which provide the mechanism by which services can communicate with one another in compositions to perform higher level functions. The service interface (that provides the signatures of the available operations) is central to the SOA view as it is the only thing which is exposed to the client to invoke the function.

However, it shall be noticed that interface descriptions are low level, technical statements (cf. WSDL statements) [41] that are understandable by software professionals but far to be comprehensible by business people. At the same time, the notion of a service is familiar to the management world [27] and with the growing acceptance and popularity of SOC, computing systems now aim to extend far beyond the firewall to automate enterprise-wide business processes, covering sales, supply chain, manufacturing, delivery, payment, human resources, and more. To attain this, it is necessary to adapt SOC and its underlying architecture, SOA to a mainstream practitioners’ level and bridge the gap between high level business services and low level software services. This is a position acknowledged by authors who have been instrumental in the SOC world such as [2] and [43].

The position adopted in this paper suggests a move from the function-driven SOC to intention-driven SOC. Whereas the former lies on a functional view of services, the latter proposes to spell out the purpose, the intention behind a service. In other words, to describe a service through the need of its business user that it allows to fulfill instead of organizing its description to facilitate software reuse when developing applications. As a consequence, interfaces of these services will bring out the business goal that the service allows to fulfill instead of defining the signatures of basic operations that can be invoked on class objects. Our belief is that this position should avoid the current mismatch of languages between low level service expressions such as WSDL statements and business perceived services. We refer to these services as intentional services and present in this paper ISM, a model for intentional service modelling.

This view of services implies a move from the traditional SOA towards an intentional level. While complying with the SOA model, our model, the intentional SOA (ISOA) is a proposal for leveraging the 3-SOA tuple <Publish, Find, Bind> to an intentional level matching the business mainstream needs. In adapting the roles and operations of the SOA, the ISOA (Fig. 2) introduces two main departures:

(i) in the interaction, business agents replace software
agents,
(ii) intentional service descriptions replace functional software service descriptions.

The ISOA implies that business-centric organizations offering e-business services shall describe their services in an intentional manner, and publish them to an e-business service registry that makes these descriptions available. Business agents who are searching for services use an intention matching mechanism to retrieve service descriptions fitting their needs and use them to bind to the e-business provider. Despite its departure from the traditional SOA, the ISOA is dealing with services as its core elements and not with other forms of components such as COTS components.

In this paper, we use two of the roles mentioned in the ISOA architecture to structure the discussion on our view of intentional service-oriented computing. For the *registry*, we introduce the notion of intentional service, highlight its relationship with software services and present *ISM*, a model for intentional service modelling. We show that an intentional service description shall include variability, i.e., propose alternative variations of a given component service as well as alternative compositions to achieve the same intention. The *ISM* model gives us the capability to populate the intentional service registry with intentional service descriptions. This is the subject of section 2.

It is for the *e-business provider* to define the services that are to be provided in the business. In other terms, it is the role of the e-business provider to determine which intentional services he would like to make available through the intentional service registry. We developed a methodology to support this task in three steps:

- To represent business intentions in a graphical representation called map. This map takes the form of an intention/strategy graph with intentions as nodes and strategies to achieve them as edges.
- To derive the intentional services that can be published from a map based on an algorithm and guidelines.
- To relate intentional services to software services that operationalize them.

This role of the provider is considered in section 3.

This work is based on a previous work [34] that it extends in two ways: (a) by adding quality aspects in the intentional service definition and (b) by presenting the methodology and the associated transformation rules to bridge the gap between the intentional level and the implementation one. This requires the introduction of an intermediary level where logical services are derived from intentional services and then, mapped onto software services. The paper presents *OSM* (Operational Service Model) and the set of upwards and downwards transformations that allow to derive operational services from intentional services and then, to map those onto software executable services.

### 2. Defining and Publishing Intentional Services

In this section we consider the intentional service registry. To this end, we clarify the notion of an intentional service and present the Intentional Service Model, *ISM*, to model different types of intentional services. We highlight the intentional nature of the intentional service interfaces that are made available in the registry.

#### 2.1. Defining intentional services

An intentional service is a service captured at the business level, in business comprehensible terms and described in an intentional perspective, i.e., focusing on the intention it allows to achieve rather than on the functionality it performs. Fig. 3 presents the *ISM* using UML notations. As shown by the colors used in the Figure, there are four different aspects in the description of an intentional service, namely the service interface, the service behavior and the service composition that deal with the functional aspects of the service, and the QoS dimension that introduces the quality goals and their qualitative and quantitative evaluation. We describe the four in turn.

#### 2.1.1. Service interface

First, central to Fig. 3 is the fact that a service permits the fulfillment of an intention, given an initial situation and terminating in a final situation. These three elements constitute the interface of an intentional service; the intention replaces the operations that are part of a typical software interface whereas the initial and final situations are the input and output parameters structured as business object classes.

We view an intention in the same sense as a goal. A
goal is ‘an optative’ statement [11] that expresses what is wanted i.e. a state that is expected to be reached or maintained. Thus, Make Room Booking is the intention to make a reservation for rooms in a hotel. The achievement of this intention leaves the system in the state, Booking Made. If Accept Payment is the intention of a service then, the initial situation refers to the booking and customer classes whereas the final situation comprises the payment class in addition.

To highlight the intentional nature of our services, we propose to name a service with the intention it permits to achieve. Thus, the service allowing us to Make Room Booking is named S_Make Room Booking.

The intention giving its identity to a service plays an important role when the business agent wants to retrieve a service matching his need (that reflects his intention). To permit a powerful search through intention matching we developed a template for formulating an intention shown in Fig. 4. We use a linguistic approach inspired by the Fillmore’s case grammar [10] and its extension by Dik [9] to define this template. An intention statement is composed of a verb and different parameters which play specific roles with respect to the verb. For instance, the ‘beneficiary’ parameter expresses who benefits from the achievement of the intention. The structure of an intention is shown in Fig. 4. More details can be found in [28].

2.1.2. Service Behaviour

Second, Fig. 3 shows that the behaviour of the service is specified through its pre and post conditions that are the initial and final sets of states characterizing the initial and the final situation, respectively. In the Accept Payment service example, <booking.state=‘OK’ ∧ customer.status=‘registered’> and <booking.state=‘paid’ ∧ payment.status = ‘done’> are the pre and post-conditions, respectively.

Obviously the retrieval of a service will complete the intention matching with states and situation matching.

2.1.3. Service composition

Third, Fig. 3 points out that services are classified as aggregate or atomic. The former are composed of other services whereas the latter are not. Atomic services have intentions that are fulfilled by SOA level functional services. This explains in Fig. 3 the operationalized by link between an atomic and software service. In contrast aggregate services have high-level intentions that need to be decomposed in lower level ISOA services till atomic intentional services are found. Therefore, it can be understood that aggregate intentional services lie on an intention-driven composition that is necessary to bridge the gap between the actual functionality (captured in the atomic service) and the high level perception of business executives for a service fulfilling their strategic/tactical intentions.

Fig. 3 shows that aggregate services are further refined. Aggregation of services can involve variants, i.e. services which are alternative to the others or result from simple composition, leading to composite services.

Composite services reflect the precedence/succession relationship between their intentions. For example, in the room booking case, to achieve the intention Make Confirmed Booking it is necessary to attain the two intentions Make Room Booking and Accept Payment, in this order.

The composition of the two services S_Make Room Booking and S_Accept Payment forming the composite service S_Make Confirmed Booking leads to the satisfaction of the intention Make Confirmed Booking. This form of composition is grounded on the AND goal decomposition as used in goal modelling [30].

The composition is denoted “•” when there is a sequential order between component services and “//” when they can run in parallel. Every service in a composition can be executed repeatedly, this is denoted by the “∗” symbol. Thus, the composite service to fulfil the Make Confirmed Booking intention is defined as follows:

\[ S_{\text{Make Confirmed Booking}} = \bullet (S_{\text{Make Room Booking}} \cdot S_{\text{Accept Payment}}) \]

We propose a graphical representation of composite services as shown in Fig. 5 with two examples. The first one visualises the composite service to Make Confirmed Booking whereas the second illustrates the reflexive definition of a service. Sₐ is a composite service with two component services Sᵢ (which is atomic) and Sₗ (which is itself a composite including two atomic services Sᵢ and Sₗ that can be enacted in parallel).

Introduction of variability in intentional service modelling is justified by the need to introduce flexibility in intention achievement and adaptability in intentional service execution. There are three types of variants in ISM, namely alternative, choice and multi-path.

An alternative variation corresponds to an XOR relationship between the service intentions involved. For example, assume that Accept Payment can be achieved in exclusively one of the following ways, By electronic transfer or By credit card or By cash. This leads to define the service S_Accept Payment as a variant aggregate with three alternative components. We use the symbol “⊕” to denote alternative and therefore:
A choice variation corresponds to an OR relationship between the service intentions involved. For example, assume that Investigate Candidate Booking can be achieved either On the Internet or By visiting a travel agent or by both. The aggregate service $S$ Investigate Candidate Booking is therefore defined as variant with two components $S$ Investigate Candidate Booking on the Internet and $S$ Investigate Candidate Booking by visiting a travel agent. The difference between alternative and choice lies on the fact that the former implies exclusion of variants whereas the latter authorizes the selection of several of the choice when the variant service is enacted. We use the symbol “\(\nu\)” to denote the choice variation and therefore:

$$S \text{ Investigate Candidate Booking} = \nu (S \text{ Investigate Candidate Booking on the Internet } \cup S \text{ Investigate Candidate Booking by visiting a travel agent})$$

Fig. 6 shows the graphical representation of variants. The first example presents the alternative for payment whereas the second example shows a composite service having one of its components being a choice service.

Finally, a multi-path variation occurs when several compositions of an intentional service allow to achieve the same intentional service. Let us assume in our example that it is possible that the customer gets a booking as a reward for loyalty to the hotel chain. Thus, there are two paths to providing the intentional service Make a Confirmed Booking: one by achieving the sequence of intentional services Make a Booking, Accept payment and the other one Get a Rewarded Booking. The multi-path is denoted “\(\cup\)” and the multi-path service $S$ Make Confirmed Booking is defined as follows:

$$S \text{ Make Confirmed Booking} = \bigcup (S \text{ Make Booking} \cup S \text{ Accept Payment} \cup S \text{ Get a Rewarded Booking})$$

Fig. 7 provides the graphical representation of the above example.

The foregoing shows that services are defined recursively; an aggregate service being possibly composed of other aggregate services; besides, components of an aggregate service can be related directly through composition links (\(\cup\), \(/\), \(\ast\)) or in a more complex manner through variation relationships (\(\cup\), \(\nu\), \(\ast\)). Overall, services are defined in an intention-driven manner focusing on the ‘whys’ of the functionality provided by the underlying SOA level software service. Moreover, composition is itself intention-driven and grounded in XOR, OR, AND relationships among intentional services. Thus, whereas the service interface exhibits the ‘whys’ of the service, its actual implemented functionality is embedded in the related atomic services.

2.1.4. Quality of Services

Whereas the three aspects of the intentional service description previously presented are dealing with the functional dimension of the service, the QoS element of the ISM shown in Fig. 3 introduces the quality properties of the intentional service. We adopt the view that quality aspects can be captured by quality goals and use the qualitative framework [7][18] for their definition. Quality goals are often referred to as soft goals because their satisfaction cannot be defined in a clear cut manner by opposition to hard goals for which this is possible. Soft goals are said satisfied (a term introduced by [36] and reused in the qualitative framework [18]) by opposition to hard goals that can be said satisfied. The quality goals of the ISM are seen as soft goals whose satisficing reflect the partial contribution of a service towards (or against) a particular quality goal. As shown in Fig. 3, the quality (QoS) of an atomic service is said simple as it reflects the contribution to the quality goals associated to it. Vice versa, the quality of an aggregate service is said global as it results of a compilation of QoS related to its constituent services. We detail these aspects in the following.

Simple QoS: this relates to an atomic service and expresses the service contribution to quality goals of this service. We adopt a dual expression comprising a qualitative part and a quantitative part, both borrowed/inspired from the qualitative framework [7][18].

The qualitative evaluation corresponds to the following satisficing degrees: “Very satisfied”, “Satisfied”, ‘Not Satisfied” and “Not at all satisfied”, noted respectively as “++”, “+”, “-”, and “---”. We selected these notations for their understandability and readability by business agents. For example, the QoS of the two atomic services, $S$ Accept Payment and $S$ Make Room Booking are expressed as follows: $QoS(S$ Accept Payment $) = (<Performance,++>,<Confidentiality,++>)$ and $QoS(S$ Make Room Booking $) = (<Confidentiality,++>,<Cost,++>)$.

However, as this satisficing scale is subjective we supplement this evaluation by introducing the concept of indicator (Fig. 3). Each quality goal is associated with one or more indicators which facilitate the quantification of the quality goal satisficing. Each indicator has a set of reference values that are associated to the qualitative scale. For instance, a quality goal “Confidentiality” can be associated with different indicators, such as "Fraud Rate
and Security Level. The Fraud Rate indicator can be associated with a qualitative scale in which the satisficing degree “++” is obtained when the Fraud Rate is lower than 0.02%, whereas the satisficing degree “−” is obtained when the Fraud Rate is greater than 1%.

The expression of qualitative and quantitative satisficing degrees allow the service provider to indicate the extent to which a service can guarantee a quality goal satisfaction. Such values can then be matched with customers’ (business agents) quality requirements. The integration of the qualitative dimension in the intentional service description aims at a customer-centered adaptability of service composition by selecting the service composition that satisfies customers’ quality requirements. Hence, it is important to calculate the quality of the intentional service as a whole. This is presented in the following.

**Global QoS:** this is applicable to an aggregate service and is based on the QoS of the component services the aggregate is made of. In order to calculate a global QoS we defined a set of rules inspired from [18]. These rules consider the different types of composition (“•”, “/”) and variability (“∅”, “v”, “∪”). For instance:

- **Rule R1:** in the case of a composite service (using the “•” operator), where all component services contribute to satisficing a quality q with different satisfying degrees (dι), the computation of the global QoS corresponds to the minimum satisfying degree guaranteed by the component services. This is denoted as: QoS (service) = min (qι dι) … qι dι).

- **Rule R2:** in the case of a variant service (“∅”), where all component services contribute to satisficing a quality q with different satisfying degrees (dι), the global QoS is calculated as the maximum satisfying degree associated to one of the component services. This is denoted as: QoS (service) = max (qι dι) … qι dι).

These rules are justified by the semantics attached to the AND/OR operators. In an ANDed construction, all component services will be executed and consequently only the minimal value associated to the components can be guaranteed. Instead, in an ORed assembly, the component service with the best guaranteed quality can always be chosen. For example as QoS(SAccept Payment) and QoS(SMake Room Booking) share the quality goal ‘confidentiality’ with different degrees (see above), the composite service SMake Confirmed Booking = • (S Make Room Booking, S Accept Payment) will have a quality degree for ‘confidentiality’ set to the minimum that is (confidentiality, +).

- **Rule R3:** in another case of a composite service, where all component services contribute to satisfice several qualities qι with different satisfying degrees (dι), the global QoS is expressed as an union (∪) of the satisfying degrees of the component services. This is denoted as: QoS (service) = ∪(qι dι) … qι dι). For example, for the composite service SMake Confirmed Booking = • (S Make Room Booking, S Accept Payment) given rules R3 and R1, the corresponding QoS is as follows: (confidentiality, +), Cost, ++,<Performance, ++).

### 2.2. Publishing intentional services in the registry

Once we have defined intentional services, it is necessary to consider the issue of populating the intentional service registry with service descriptions. As indicated in Fig.2, ISOA proposes an intentional service registry, in which every service must be made available. It is worth noting that the registry does not contain software services. Only a reference to such services is kept as part of the atomic intentional service description, as defined in the LSM (Fig. 3). Unlike traditional SOA registry, which is function-driven and fits software engineering needs, the intentional registry considers only an intentional view, fitting particularly business agent needs. Such users are used to reason about business goals they need to reach. For these users, it is more natural to search services based on their intentions than searching based on service functionalities.

Thus, every atomic and aggregate intentional service is kept in the intentional service registry. For an aggregate, information about composition links and relationships is kept. This enables (Fig. 2) retrieval of complete aggregate services, their binding and adaptation to conform to the task at hand. For an atomic service, information described in the LSM is also stored in the registry.

As mentioned earlier, retrieval is merely based on intention matching and thereafter on situation and condition matching. That is, given the need to find a service with intention I, the registry is searched to retrieve a service with the same or similar intention. The linguistic approach (section 2.1.1) to formulate intentions provides a rich mean to support such a matching. Once such a service is found, one drills down to assure oneself that the pre and post conditions match. Finally, the initial and final situations yield the input and output parameters.

For sake of space, we do not consider the qualities aspects of intentional services in the following.

### 3. DISCOVERING INTENTIONAL SERVICES FOR PUBLICATION

We believe that the intentional services that populate the registry arise in the business of organizations. Services to be provided relate to business objectives and, indeed, help to achieve these. This requires that a model of the business can be developed giving to the E-business provider (Fig. 2) a way to identify services for publication. In this section, we propose the use of the MAP formalism [31] to represent businesses in intentional terms and provide guidelines to determine services from this representation. We use Materials Management (MM) to illustrate service publication (see [31] for details of the MM map).

#### 3.1. Capturing business intentionality in maps

MAP is a representation system that was originally developed to represent a process model expressed in intentional terms. It provides a representation mechanism based on a non-deterministic ordering of intentions and strategies. We will use it here as a means for modelling intention-driven composition of services.

A map is a labeled directed graph with intentions as nodes and strategies as edges (Fig. 8). An edge enters a node if its strategy can be used to achieve the intention of
the node. Since there can be multiple edges entering a node, the map is capable of representing many strategies that can be used for achieving an intention. A map as a whole is associated to an intention that expresses the global purpose of the business modeled as an intention/strategy graph. The MM map has Satisfy Material Need Efficiently as its intention.

An Intention is a business goal that can be achieved by the performance of a process. For example, the MM business map (Fig.8) has Purchase Material and Monitor Stock as intentions. Further, each map has two special intentions, Start and Stop, to start and end the process.

A Strategy is an approach, a manner to achieve an intention. A strategy $S_{ii}$ between the couple of intentions $I_i$ and $I_j$ represents the way $I_j$ can be achieved once $I_i$ has been satisfied. In Fig. 8, By reorder point planning is a manner to place an order to Purchase Material, any time the stock of this material falls under the reorder point.

A Section is a triplet $<I_i, I_j, S_{ii}>$ and represents a way to achieve the target intention $I_j$ from the source intention $I_i$ following the strategy $S_{ii}$. For instance $<\text{Start, Purchase Material, Manual strategy}>$ couples a source intention (Start) to a target intention (Purchase Material) through a strategy (Manual strategy) and represents a way to achieve the target intention Purchase Material from the source intention Start following the Manual strategy. Each section of the map captures the situation needed to achieve an intention and a specific manner in which the process associated with the target intention can be performed.

Sections in a map are connected to each other by four kinds of relationships, namely the multi-thread, bundle, path and multi-path relationships.

Multi-Thread relationship: A target intention can be achieved from a source intention in many different ways. Each of these ways is expressed as a section in the map and these sections are said to be in a thread relationship with one another. In Fig. 8, the Planning strategy and the Manual strategy are in a multi-thread relationship.

Bundle relationship: Several sections having the same pair $<$source, target$,>$ which are mutually exclusive, are in a bundle relationship. The group of these sections constitutes a bundle. Notice that the difference between a multi-thread and a bundle relationship is the exclusive OR of sections in the latter versus an OR in the former. For example, the Planning strategy is a bundle consisting of the Reorder point strategy and Forecast based strategy. Similarly, the Inventory balance strategy is a bundle of periodic, continuous and sampling strategies.

Path relationship: This establishes a precedence/succession relationship between sections. For a section to succeed another, its source intention must be the target intention of the preceding one. For example the two sections, $<$Start, Purchase Material, Manual strategy$,>$, $<$Purchase Material, Monitor Stock, Out-In strategy$>$ constitutes a path.

Multi-path relationship: Given the three previous relationships, an intention can be achieved by several combinations of sections. Such a topology is called a multi-path. In general, a map from its Start to its Stop intentions is a multi-path and contains multi-threads and bundles. For example, there is a multi-path to achieve the intention Monitor Stock; either the path from Start to Monitor Stock via Purchase Material can be followed or the direct path from Start to Monitor Stock can be used.

Finally, it is possible to refine a section of a map into an entire map at a lower level of abstraction. For example, Fig. 9 shows the refinement of the section $<$Purchase Material, Monitor Stock, Out-in strategy$>$ as a map. This refinement mechanism leads to model business intentionality as a hierarchy of maps. The source intention of the refined section, namely Purchase Material, corresponds to the Start intention of the refined map. Similarly, the Stop intention of the refined map corresponds to the target intention of the refined section, Monitor Stock in this example. The refined map includes several intentions and provides several strategies to achieve each of them. In this example, there are two refined intentions Accept Delivery and Enter Goods in Stock and several strategies to achieve them.

It shall be noticed that in contrast to most process modeling formalisms that do not employ a goal construct as an integral part of the model, the MAP formalism follows the human intention of achieving a goal as a force that drives the process. Consequently, goals to be accomplished are explicitly represented in the process model together with the alternative ways for achieving them. Traditional process modeling formalisms emphasize an internal view of a process, focusing on how the process is performed and externalizing what the process is intended to accomplish in the goal [17]. In contrast, intention-oriented process modeling focuses on what the process is intended to achieve, thus providing the rationale of the process, i.e. why the process is performed. We think that
this position helps capturing the essence of the business and avoids being bothered with too many details of the ‘how’ that are not necessary to develop at this stage. In addition, making strategies explicit allow showing the different ways to achieve a goal, thus permitting the map to capture variability through multi-threads and multipaths. We will see in the next section how these map topologies can be used as an architectural style for determining business services and their composition.

3.2. Deriving intentional services from maps
Having represented business intentionality in a hierarchy of maps, we now proceed to determine services and their composition according to the LSM. We propose a methodological process organized in four steps as follows:

Given a map \( m \) having \( I_m \) as its intention and \( S_{1m} \) as its associated service:

Step1: Associate every non-refined section to an atomic service.
Step2: Calculate all the paths of a map using an adaptation of the MacNaughton and Yamada’s algorithm [15].
Step3: Determine the aggregate service using the following correspondences between sections relationships in maps and service composition operators in \( \text{ISM} \langle \text{path - composite}\rangle, \langle \text{bundle - alternative}\rangle, \langle \text{multi-thread - choice}\rangle, \langle \text{multi-path - multi-path}\rangle \).

Since the entire map is, in general, a multi-path, it corresponds to an aggregate service.

For each refined section:

Step4: Repeat steps 1, 2 & 3 to the refining map \( m_r \).
We consider the four steps in turn and illustrate them with the MM map.

3.2.1. Deriving atomic services from every non-refined section
The derivation starts by considering the MM map presented in Fig. 8 that we denoted as map \( m \). \textit{Satisfy Material Need Efficiently} is the intention of this map and \( S \) \textit{Satisfy Material Need Efficiently} the corresponding service.

**Table 1. List of atomic services embedded in \( S \) satisfying material need efficiently**

<table>
<thead>
<tr>
<th>Sections</th>
<th>Atomic services</th>
</tr>
</thead>
<tbody>
<tr>
<td>ab(_1)</td>
<td>S Purchase Material with reorder point strategy</td>
</tr>
<tr>
<td>ab(_2)</td>
<td>S Purchase Material with forecast strategy</td>
</tr>
<tr>
<td>ab(_3)</td>
<td>S Purchase Material Manually</td>
</tr>
<tr>
<td>ac(_1)</td>
<td>S Receive stock by bill for expenses</td>
</tr>
<tr>
<td>cc(_1)</td>
<td>S Move stock</td>
</tr>
<tr>
<td>cc(_2)</td>
<td>S Evaluate value of stock</td>
</tr>
<tr>
<td>cc(_3)</td>
<td>S Inspect stock</td>
</tr>
<tr>
<td>cc(_4)</td>
<td>S Conduct Physical Inventory continuously</td>
</tr>
<tr>
<td>cc(_5)</td>
<td>S Conduct Physical Inventory by sampling</td>
</tr>
<tr>
<td>cc(_6)</td>
<td>S Conduct Physical Inventory periodically</td>
</tr>
<tr>
<td>cd(_1)</td>
<td>S Verify invoice against delivery.</td>
</tr>
</tbody>
</table>

To save space, we use a textual notation in which intentions in a map are named by letters of the alphabet, strategies are numbers and therefore, a section named \( ab_1 \) designates a way to achieve a target intention denoted \( b \) from a source one denoted \( a \) following a strategy denoted 1. Thus, the section \( \langle i, I, S_{ij} \rangle \) is denoted \( ab_1 \) where \( a \) is the code of the intention \( I \), \( b \) is the code of the intention \( I \) and \( I \) is the code of the strategy \( S_{ij} \). The letters referring to intentions and the numbers referring to strategies are shown in both Fig. 8 and Fig. 9.

Assuming that map \( m \) contains only one refined section, namely bc\(_1\), step1 leads to derive 11 atomic services from the MM map as mentioned in Table 1.

It can be seen that the name of each service reflects the business intention that can be achieved as well as the strategy to achieve it. As bc\(_1\) is a refined section, its derived services will be identified in step 4.

3.2.2. Calculating all paths
Step 2 leads to the identification of all the paths in map \( m \), the MM map in this case. In order to avoid errors in the elicitation of paths in a map and to ensure completeness, we propose to replace the manual elicitation of paths by an automatic generation based on an adaptation of the MacNaughton and Yamada’s algorithm [15]. We are aware of the limitation of this algorithm to large nets. However, our experience with maps in a large number of real cases showed that a map contains usually 3 to 5 intentions and 15 to 30 strategies, thus a map is never a large net. The adaptation of the algorithm to maps leads first, to the two following formulas:

Let \( s \) and \( t \) be the source and target intentions, \( Q \) the set of intermediary intentions including \( s \) and \( t \) and \( P \) the set of intermediate intentions excluding \( s \) and \( t \).

The initial formula \( Y_{s, Q, t} \) used to discover the set of all possible paths using the three operators that are the union ("\( \cup \)"), the composition operator ("\( . \)") and the iteration operator ("\( \ast \)"), is:

\[
Y_{s, Q, t} = \cdot \left( X^s \cup \cdot \left( X^s, Q, \{s, t\}, t \right), X^t \cup \cdot \left( X^t, Q, \{s, t\}, t \right) \right)
\]

And given a particular intention \( q \) of \( P \), the formula \( X_{s, P, t} \) applied to discover the set of possible paths is:

\[
X_{s, P, t} = X_{s, P, \{q, \}, t} \cup \cdot \left( X_{s, P, \{q, \}, t}, X_{s, P, \{q, \}, \{q, X_{q, P, \{q\}, t} \right)
\]

Secondly, we specialize the \( X_{s, P, t} \) into paths, multipaths, multi-threads and bundle relationships that we note as follows:

A \textit{Bundle relationship} between two intentions \( k \) and \( l \) is denoted: \( B_{kl} = \odot (k_1, k_2, \ldots, k_n) \) where the \( k_i \) are the exclusive sections related by an XOR relationship indicated by the operator \( \odot \). In Fig. 8, the bundle of planning strategies is \( B_{ab} = \odot (ab_1, ab_2) \).

A \textit{Multi-thread relationship} between two intentions \( k \) and \( l \) is denoted: \( MT_{kl} = \lor (k_1, k_2, \ldots, k_n) \) where the \( k_i \) are the sections and \( \lor \) expresses the OR related relationship between the \( k_i \) of the multi-thread. Thus, the multi-thread represented in Fig. 8 between the intentions \( \text{Start} \) and \( \text{Purchase Material} \) is denoted \( MT_{ab} = \lor (B_{ab}, ab_3) \).

A \textit{Path relationship} between two intentions \( k \) and \( l \) is denoted \( P_{k, Q, l} \) where \( Q \) designates the set of intermediary intentions used to achieve the target intention \( l \) from the source intention \( k \). A path relationship is based on the sequential composition operator "\( \ast \)" between sections. As an example, the path relationship in Fig. 8 between \( \text{Start} \)
and Monitor Stock is denoted \( P_{a,(b,c)} = \bullet (MT_{ab}, bc) \).

A Multi-path relationship between two intentions \( k \) and \( l \) is denoted \( MP_{a,(b,c)} \), where \( Q \) designates the set of intermediate intentions used to achieve the target intention \( l \) from the source one \( k \). A multi-path relationship is based on the union operator “\( \cup \)” between alternative paths. Thus, the multi-path in Fig. 8 between Start and Monitor Stock is denoted \( MP_{a,(b,c)} = \cup (ac_{1}, P_{a,(b,c)}). \)

The initial formula generating all the paths between the intentions \( a \) and \( d \) of the MM map in Fig. 8 map is:
\[
Y_{a,[b,c,d]} = \bullet (X_{a,[b,c,d]}, a)* , X_{a,[b,c,d]} , X_{a,[b,c,d]} , X_{a,[b,c,d]} .
\]

Using this formula, the algorithm leads to elicit the paths as shown in Table 2. Map \( m \) is made of two paths \( P_{a,(b,c)} \) and \( P_{a,(b,c)} \). The latter contains a multi-path \( MP_{a,(b,c)} \), a multi-thread \( MT_{ab} \), and the section \( cd \), whereas the former contains a multi-thread \( MT_{ab} \) and the section \( bc \). Both multi-threads are made of bundles and sections.

### Table 2. List of Section Relationships in the MM Map

<table>
<thead>
<tr>
<th>Type of relationship</th>
<th>Identified relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path ( P_{a,(b,c)} = \bullet (MT_{ab}, bc) )</td>
<td>( P_{a,(b,c)} = \bullet (MT_{ab}, bc) )</td>
</tr>
<tr>
<td>Multi-Path ( MP_{a,(b,c)} = \cup (ac_{1}, P_{a,(b,c)}) )</td>
<td>( MP_{a,(b,c)} = \cup (ac_{1}, P_{a,(b,c)}) )</td>
</tr>
<tr>
<td>Bundle ( B_{ab} = \otimes (ab, a) )</td>
<td>( B_{ab} = \otimes (ab, a) )</td>
</tr>
<tr>
<td>Multi-thread ( MT_{ab} = \nu (B_{ab}, ab) )</td>
<td>( MT_{ab} = \nu (B_{ab}, ab) )</td>
</tr>
</tbody>
</table>

### 3.2.3 Determine aggregate services

The relationships between sections in a map \( m \) with intention \( I_{m} \) provide an architectural basis to compose services in the corresponding aggregate service \( S_{I_{m}} \). To derive the composition in a systematic way, we establish a correspondence between types of section relationships in the map \( m \) and aggregate service types in \( S_{I_{m}} \). This correspondence is as follows: <path - composite>, <bundle - alternative>, <multi-thread - choice>, <multi-path - multi-path>. Table 3 presents the variant and composite services associated to the aggregate service \( S_{\text{Satisfy material need efficiently}} \) derived from the MM map. These are expressed with the set of variant and composite operators, namely \( \nu, \otimes, \cup, \bullet \) introduced earlier in section 2.

### Table 3. Components of the Aggregate Service \( S_{\text{Satisfy material need efficiently}} \)

<table>
<thead>
<tr>
<th>Aggregate Types</th>
<th>Identified Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant services</td>
<td>( S_{\text{Purchase Material Planning strategy}} = \otimes (S_{\text{Purchase Material Planning}}) )</td>
</tr>
<tr>
<td>( S_{\text{Purchase Material Planning strategy}} )</td>
<td>( S_{\text{Purchase Material Planning strategy}} )</td>
</tr>
<tr>
<td>( S_{\text{Conduct Physical Inventory}} = \otimes (S_{\text{Conduct Physical Inventory}} \otimes S_{\text{Physical Inventory by periodically}} \otimes S_{\text{Physical Inventory by sampling}}) )</td>
<td>( S_{\text{Conduct Physical Inventory}} )</td>
</tr>
<tr>
<td>( S_{\text{Purchase Material}} = \nu (S_{\text{Purchase Material Manually}} \otimes S_{\text{Purchase Material Planning}}) )</td>
<td>( S_{\text{Purchase Material}} )</td>
</tr>
<tr>
<td>( S_{\text{Monitor Stock}} = \nu (S_{\text{Conduct physical inventory}} \otimes S_{\text{Inspect stock}} \otimes S_{\text{More stocks}} \otimes S_{\text{Evaluate value of stock}}) )</td>
<td>( S_{\text{Monitor Stock}} )</td>
</tr>
<tr>
<td>( S_{\text{Receive stock}} = \nu (S_{\text{Receive stock by bill for expenses}} \otimes S_{\text{Receive stock normally}}) )</td>
<td>( S_{\text{Receive stock}} )</td>
</tr>
</tbody>
</table>

### 3.2.4 Determine Services of a Refining Map \( M_{bc1} \)

In this step, the three steps illustrated above shall be performed to each refined section in order to determine the composition of the aggregate service associated to the corresponding \( m \). In the MM business case, one section is refined, namely section \( bc \). The refining map \( M_{bc} \) is presented in Fig. 9. Its intention is \( \text{Receive Stock of purchased material} \) and the corresponding service is \( S_{\text{Receive stock of purchased material}} \). Table 4 presents the results of the application of step 1 that is a set of atomic services. None of the \( m \) sections is refined; thus the presence of 8 atomic services in Table 4.

### Table 4. Atomic Services Derived from \( M_{bc1} \)

<table>
<thead>
<tr>
<th>Sections</th>
<th>Atomic services</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ab_{1} )</td>
<td>( S_{\text{Accept delivery with reconciliation under/over delivery}} )</td>
</tr>
<tr>
<td>( ab_{2} )</td>
<td>( S_{\text{Accept delivery with reconciliation of unit difference}} )</td>
</tr>
<tr>
<td>( ab_{3} )</td>
<td>( S_{\text{Accept delivery with okay strategy}} )</td>
</tr>
<tr>
<td>( bc_{1} )</td>
<td>( S_{\text{Accept delivery with reconciliation by PO recovery}} )</td>
</tr>
<tr>
<td>( bc_{2} )</td>
<td>( S_{\text{Accept delivery with reconciliation by PO recovery}} )</td>
</tr>
<tr>
<td>( cd_{1} )</td>
<td>( S_{\text{Accept delivery with Out-in consumption strategy}} )</td>
</tr>
<tr>
<td>( cd_{2} )</td>
<td>( S_{\text{Accept delivery with Out-in storage-based strategy}} )</td>
</tr>
<tr>
<td>( bd_{1} )</td>
<td>( S_{\text{Stop by completeness strategy}} )</td>
</tr>
</tbody>
</table>

Using steps 2 & 3 lead to the derivation of the composition of services in \( S_{\text{Receive stock of purchased material}} \) (Table 5). These are variants on one hand, and composite services on the other hand, with complex nesting among them.

The aggregate service \( S_{\text{Receive stock of purchased material}} \) associated to the refined map \( M_{bc1} \) is a composition of two services. The first one is a variant service offering four alternative ways to achieve the intention \( \text{Accept delivery} \). The second one is a variant proposing two alternative ways to \( \text{Enter Goods in Stock} \).
in direct consumption strategy
$S_{\text{enter}}$ goods in stock by Out-in-storage-based strategy

$S_{\text{enter}}$ goods in stock = \bigcup (S_{\text{receive}}$ delivered material normally, $S_{\text{reject}}$ delivered material)

### Composite services
$S_{\text{receive}}$ delivered material normally = $*$ (S_{\text{enter}}$ goods in stock, $S_{\text{stop}}$ by completeness strategy)

$S_{\text{receive}}$ stock of purchased material = $*$ (S_{\text{accept delivery}}, $S_{\text{enter}}$ goods in stock)

### 3.3. Operationalizing intentional services into software services

Having identified intentional services and their classification into aggregate and atomic, we proceed now to determine software services that operationalize atomic services. This operationalization is a necessary condition for the enactment of intentional services as it determines the software functionality required to support the achievement of the atomic service intention. It is important to note that this activity is carried out by a software developer and is hidden from the business agent. In contrast, the business agent is only aware of intentional services and our ultimate aim is to provide an environment in which he/she can perform all aspects of service computing at the intentional level (see the proposal of an agent platform in the conclusion). In order to perform this operationalization, we developed guidelines to support transformation from the intentional level (represented by ISM) to an intermediary logical level (represented by the Operational Service Model - OSM), and from this latter, to the implementation level. These guidelines support a two-phase process to

1. operationalize intentional services, and
2. implement their operationalizations into software services.

![Fig. 10. From intentional services to software services.](image)

This process, shown in Fig. 10, is inspired by the MDA approach. MDA [20] advocates for successively transforming platform independent software models into platform specific ones, thus allowing the development of a software specification independently of its execution platform and its easy and fast implementation on multiple platforms. Similarly to MDA, we propose guidelines to transform ISM descriptions into platform independent OSM descriptions that are in turn, mapped into platform specific ones. In our current development of this work, we selected the remote portlets technology for the latter. Nevertheless, it is worth noting that the guidance process is intended to allow the mapping of OSM descriptions to other service-oriented platforms, such as OSGi [21].

We present the two phases of this process in turn and sketch the associated guidelines.

#### 3.3.1. Operationalizing atomic services

This stage aims at transforming atomic intentional service descriptions into functional specifications that we refer to as operational service descriptions. These functional descriptions conform to the Operational Service Model (OSM), which is a platform-independent model. As shown in Fig.10, a description is composed of three parts, namely the interaction, coordination and function parts. Each of them corresponds to a given concern. The first one describes the sequence of user interactions. The third corresponds to the business logics whereas the second one establishes the coordination between the two others. This coordination ensures the synchronization of invoked functions and carries out data interoperability.

ISM based services emphasize the intention to be achieved but do not describe the process to achieve it. They characterize the process transition through pre and post conditions but do not provide sufficient information about the process logics to infer the corresponding software functionality. This phase aims to capture additional knowledge for the operationalisation of intentions of atomic services thus providing the input for transforming this knowledge into OSM descriptions. The transformation of intention atomic service descriptions into OSM functional specifications is thus organized in two steps to:

1.1. Capturing intention operationalisation knowledge and,

1.2. Generating OSM descriptions.

**Capturing intention operationalisation with scenarios.** We adopted the L’Ecritoire scenario-based approach [30][32] to support the first step. Our choice was motivated by the fact that (a) scenarios proved to be an efficient way to concretize goals and (b) due to numerous successful experiences with L’Ecritoire.

L’Ecritoire combines informal narrative prose to express scenarios with natural language parsing mechanisms for scenarios analysis and conceptualisation. It proposes semantic patterns, which allows defining the semantic of the clauses and sentences of scenarios [30][32].

Besides, L’Ecritoire is an approach based on a bidirectional coupling between goals and scenarios. In the forward direction a scenario is seen as a concretization/realization of a goal, whereas in the backward direction a scenario serves to discover sub goals of the initial goal. Therefore, a scenario is authored in an informal narrative prose any time a goal has been elicited and then, transformed and completed through a linguistic analysis and interpretation of its narration. Such conceptualized scenario serves as a basis to discover new goals. To support these two key activities of scenario authoring and goal discovery, L’Ecritoire provides two types of rules to (a)
transform and complete a scenario and map its content onto a scenario structure [32] and (b) reason on the scenario contents for deriving sub goals, therefore helping to explore the functionality, discover its variants and exception cases.

Thus, a systematic application of L’Ecritoire rules guides towards the specification of the functionality required to achieve the service intention. In order to briefly illustrate the approach, let us consider the atomic service $S_{\text{Accept delivery with okay strategy}}$ having $\text{Accept delivery with okay strategy}$ as its intention (see Table 5). The narrative scenario describing the standard behavior to achieve this intention is shown in Table 6.

**TABLE 6. AN EXAMPLE OF TEXTUAL SCENARIO**

| When the employee activates the transaction $\text{accept delivery with the okay strategy}$, a prompt for authentication is displayed to him (her). He (she) enters his (her) login and password. This information is sent to the authentication provider who checks if the user is authorized to accept delivery. If he (she) is authorized, the MM portal requests the purchase order Id. The employee provides the Id of the purchase order. The MM portal sends the Id to the MM-IS (Material-Management-Information System) provider. If the MM-IS retrieves the purchase order and validates its pending status, the content of the purchase order (purchased item, quantity and pricing) is extracted and displayed to the employee. If every purchased item is delivered as it is planned, the employee confirms the complete delivery of the purchase order. The MM-IS asks for acknowledgment before registering the delivery acceptance. If the data are valid, the employee confirms the complete delivery and the MM-IS registers the delivery acceptance of the purchase order in the system. |

L’Ecritoire meta-model (Fig. 11) considers that every action is performed by an actor to another one affecting some parameter object [33]. Every agent identified in L’Ecritoire scenarios is translated into an OSM agent, and every object of the type resource is transformed into an OSM resource.

Considering the OSM coordination meta-model, the distinction between atomic and flow of actions in L’Ecritoire permits a mapping of an atomic action to an OSM basic activity and of flow of actions into an OSM structured activity. The correspondence also exploits L’Ecritoire semantic patterns. Semantic patterns identify respective roles played by action components, and the semantic of verbs [33]. Among all semantic patterns identified in [33], the communication pattern is particularly observed. This pattern states for verbs expressing the transition of an
object from a source to a destination. It allows us to identify actions involving the OSM coordination part, since each activity from this part is responsible for the communication between the interaction part and the functional part. Such coordination actions correspond to service invocations, data processing tasks, exception handling, and other actions used to communicate and to coordinate interaction among final users and business logic services. For instance, considering the scenario in Table 6, the request for user authorization verification is translated into a communication activity between MM Portal and authentication provider agents.

The correspondence between the scenario meta-model and the OSM interaction and function meta-models is based on a similar translation mechanism. For instance, if a human agent is directly involved, a scenario atomic action is translated into an OSM interaction. The OSM interaction meta-model defines two kinds of atomic interactions, namely user interactions and back office interactions. The former correspond to actions for data exchange between the user agent and the interface part, whereas the latter correspond to the communication between the interface and the coordination part. For instance, in the scenario presented in Table 6, the action between the employee and the MM-portal requesting the Id of the purchase order is translated into an OSM user interaction between these agents, while the action of sending this Id to the MM-IS is translated into a back office interaction between MM-portal and MM-IS.

3.3.2. Implement operational service descriptions into remote portlets

Starting from the independent-platform description of an operational service (OSM) obtained from the previous phase, it is possible to generate different platform specific implementations. In this paper, we illustrate the guidelines that we propose to support such transformation into a remote portlet implementation. The WSRP (Web Service for Remote Portlets) specification introduces a set of presentation-oriented interfaces that include the interaction with the content of a Web service. Such a presentation-oriented Web service provides then both application logic and presentation logic [19]. The choice of the WSRP technology was motivated by its user-centric nature and its ability to combine an interface service with Web services (Fig. 13). This technology fits our operational view of an intentional atomic service, which requires both business rules (embedded in Web services) and interactions with the users (the interface part) to achieve its intention.

The translation of an OSM specification to a specific platform (remote portlets in this case) follows the OSM division in three parts: the interaction part is translated into the portlet descriptor (with the corresponding user interface in JSP), the coordination part is translated into a composite Web service (with its WSDL and BPEL descriptions), while the function part of OSM is implemented as a set of atomic Web services, also described in WSDL. Thus, OSM back office interactions are translated into the corresponding requests to the coordination service, represented by a composite Web service. User interactions are translated into JSP views containing the requests identified by the back office interactions. In other words, the OSM interaction part is transformed into a user-interface service delegating the applicative logic to the composite web service.

Fig. 13 SAccept delivery with okay strategy Remote portlet implementation.

<table>
<thead>
<tr>
<th>TABLE 7. WSDL DESCRIPTION REFEREEING TO THE WEB SERVICE DELIVER TRANSACTION.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <code>&lt;definition&gt;</code></td>
</tr>
<tr>
<td>2. <code>xmlns:xsd = &quot;http://www.w3.org/1999/XMLSchema&quot;</code></td>
</tr>
<tr>
<td>3. <code>xmlns:soap = &quot;http://schemas.xmlsoap.org/soap/&quot;</code></td>
</tr>
<tr>
<td>4. <code>&lt;message name = &quot;deliverInput&quot;&gt;</code></td>
</tr>
<tr>
<td>5. <code> </code>&lt;part name = &quot;expression&quot; type = &quot;xsd:string&quot;/&gt;`</td>
</tr>
<tr>
<td>6. <code> &lt;/message&gt;</code></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>13. <code>&lt;portType name = &quot;DeliverTransactionType&quot;&gt;</code></td>
</tr>
<tr>
<td>14. <code>&lt;operation name = &quot;deliverTransaction&quot;&gt;</code></td>
</tr>
<tr>
<td>15. <code>&lt;input message = &quot;deliverInput&quot;/&gt;</code></td>
</tr>
<tr>
<td>16. <code>&lt;output message = &quot;deliverOutput&quot;/&gt;</code></td>
</tr>
<tr>
<td>17. <code> &lt;/operation&gt;</code></td>
</tr>
<tr>
<td>18. <code> &lt;/portType&gt;</code></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>44.<code>&lt;/definition&gt;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 8. FRAGMENT OF ATL RULES FOR TRANSFORMING OSM INTO WSDL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule businessService2wsdlDef{</td>
</tr>
<tr>
<td>From bftc : BusinessService!BusinessFunction to def : WSDL!Definition</td>
</tr>
<tr>
<td>name :&lt;- 'Web_Service.' + bftc.name</td>
</tr>
<tr>
<td>targetNamespace :&lt;- 'urn://'+ bftc.name +'/'.</td>
</tr>
<tr>
<td>'wsdl'+'/' , type :&lt;= t) : WSDL!Type(), service :&lt;- bftc.name +</td>
</tr>
<tr>
<td>'Service'; port :&lt;= p),</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Rule Operation&amp;Call2Operation{</td>
</tr>
<tr>
<td>from opCall: BusinessService!Operation to Operation : WSDL!Operation</td>
</tr>
</tbody>
</table>
| name :<- opCall.feature -> select(e | e.ooclIsKindOf(BusinessService!Operation)), typeApel :<- op-
| Call.feature -> select|
| e | e.ooclIsTypeOf(BusinessService!TypeOfCall))) } |
| } |

Finally, the coordination part activities are translated into invocations of the appropriate operations in the functional part. Fig. 13 sketches the remote portlet implementation of the SAccept delivery with okay strategy intentional atomic service. It shows two atomic Web services Authentication and Delivery of a Purchase Order (PO) transaction, which correspond to the OSM function operations for the Accept delivery with okay strategy example. It also shows one composite Web service that coordinates the interaction of
these atomic Web services and one interface service, which is coded by a set of JSP and java files together with a portlet descriptor. Additionally, Table 7 shows a fragment of the WSDL description generated for the service Deliver transaction, proposed in the example scenario (Table 6), and which was generated by ATL [3] rules, partially represented in Table 8.

3.4. Summing up

The Material Management example used in this paper illustrates how the intention/strategy structure of the map provides architecture for service composition reflecting business needs. Each section of the map identifies a service and the various section topologies of the map help assembling services in a comprehensible way for business people because they are independent of any technical aspect but just exhibit business characteristics. Further, the example points out how the section relationships in a map and particularly the thread and bundle relationships help identifying service variability. Again service variability is driven by business needs for process adaptability. Indeed, we distinguish two kinds of variability:

(i) Variability provided by the multiple strategies to satisfy the same intention and, (ii) Variability in the combinations of strategies to fulfill the same intention,

These two kinds of variability express flexibility in the way the business can be performed. The first kind is expressed by the multi-thread and bundle relationships in the map. The second kind is expressed by the multi-path relationship.

It shall be noticed that composition as expressed through ISM aggregate services is intentional in the sense that it allows to reason about different ways to achieve a high level, strategic goal by achieving its sub-goals that can be themselves fulfilled thanks to a composition of lower level sub-goals till operational goals and atomic services are found.

Those atomic services are technical software services and we overviewed the guidelines supporting a transformation process by which atomic intentional services can be converted into operational software service descriptions. This process is platform independent, allowing the adoption of different service-oriented technologies for implementing these operational software services.

We illustrated a possible implementation of software services as remote portlets (WSRP). This platform has been chosen since it permits a uniform and homogeneous implementation of both the interface aspect and the business logic aspect in one single solution. Finally, it is worth noting that a prototype of a software tool named OSM-WSRP generator tool, for automatic generation of remote portlet implementation from operational service description has been developed and is under testing.

4. RELATED WORKS

Generally speaking, research on service description, composition and adaptation in the SOA community is relevant to our work [1].

Typical descriptions of services are based on finite state formalisms, e.g., in [17] services are represented by state-charts, in [5] services are modeled as Mealy machines and in [4], services are represented as finite state machines. The ISM shares with these approaches the need to describe services to ease their retrieval but differs from their function driven perspective to propose an intention drive of service description. As a consequence, ISM service descriptions will bring out the business intention that the service allows to fulfill and pre and post conditions instead of defining the signatures of operations that can be invoked on class objects. We believe that this contribute to avoid the current mismatch of languages between low level services descriptions, such as WSDL statements, and business perceived services.

Obviously our approach shares similar concerns with semantic web services that enrich SOA-like service descriptions with semantic properties to ease their understanding and retrieval. A number of authors have contributed to this perspective [14], [39]; several languages have been defined such as OWL-S [22], WSML [8], DAML+OIL [12] and extensions have been proposed [37] to add semantics to WSDL and UDDI using ontologies. If the need for semantic service description languages is widely acknowledged, it has also been recognized that current solutions do not really relate to stakeholders’ desires and needs for services [25] and therefore, do not actually ease the retrieval of services matching their goals. We believe that the ISM shares with these approaches the view of semantically rich descriptions but overcomes the aforementioned drawback by leveraging the entire notion of a service to the intentional level, thus matching the business manager’s view for services.

A significant amount of research has focused on the composition of web semantic services, all exploiting the idea of flow-composition in which services are black boxes exchanging input/output parameters [42], [4]. In contrast, our approach borrows from goal driven approaches in requirements engineering [40], [42] the idea of goal decomposition and goal refinement through AND/OR graphs. This leads to an intention driven service composition: an ISM aggregate service has a high level, strategic intention as its key characteristic and its composition is reflecting the intention decomposition into sub-intentions that can be themselves fulfilled thanks to a composition of lower level sub-intentions till operational intentions related to atomic services are found. We believe that intention driven composition is easy to understand for business agents because it mirrors their strategic/tactical thinking strategies.

A large body of research work [6], [16], [17], [5] deals with service execution: (i) the peer-to-peer architecture in which the individual service interact among them and with the client directly, and (ii) the mediated architecture in which the control over the available services is centralized. Our approach fits best to the peer-to-peer perspective but needs a specific mechanism to cope with the adaptation issue. This is conforming to the architecture view of the semantic web service community that recommends a specific conceptual architecture [29], [38].
Our plan is to develop an agent architecture as an enactment mechanism that will allow the business agent to navigate directly in the intentional composition of services and to configure this composition ‘on the fly’ by selecting the variants which are best fitting the situation at hand. We believe that it is useful for business people to perform this adaptation because most business processes are theological in nature and require constant decision making.

From a methodological viewpoint, our proposal is close to [26] as both share the idea to capture service needs from exploring business goals. In [26] a revised Tropos design process is used to support service discovery and composition by offering a roadmap that relates stakeholder goals to collections of services available in different directories. It has some similarities with the framework for integrating business processes and business requirements [13] where a GORE approach serves as a support for moving from goal driven business requirements to BPEL process definitions. Our proposal diverges in two ways: we aim at supporting the move (a) to different implementation platforms and (b) in a more MDA driven manner.

In [44] Zisman proposes a discovery framework that supports the identification of various service operations from distinct web services that together can fulfill a system functionality. The discovery process is based on design models that are amended and reformulated by using the discovered services. This approach might be transposed to maps leading thus to an incremental and iterative construction of maps based on an initial design that is amended and modified when intentional services matching sub intentions of the map intention are discovered.

Finally, the work presented in this paper extends an initial proposal presented in [34] and [35]. First, the quality aspect of intentional service is considered here and secondly, guidelines to support the process of converting intentional services into software services are introduced.

5. CONCLUSIVE COMMENTS

In this paper we introduced the notion of intentional service as the fundamental element of intentional service oriented computing, i.e. intention driven service description, composition and execution. An intentional service is described in terms of the business intention it allows a business agent to achieve. We also showed that service composition can be intention driven in order to reflect business needs. This is in accordance with our view that business people must be provided with a description of services available in a service portfolio that is adapted to their own perceived needs.

The paper considered in some detail two aspects of ISOC in relationship with the roles of the ISOA architecture that supports our intention drive of service computing:

a) E-business provider, who looks at a business, identifies its intentions, derives and publishes services in the intentional service registry.

b) Intentional service registry where service are available. The descriptors of services and the typology of services being kept are modelled in the ISM.

Whereas the two roles of ISOA correspond to the service provider and the registry roles of the SOA, it is to be noted that ISOA services, aside from supporting business intentions, are also more complex than SOA ones. This is because of the aggregate variants that provide flexibility to the business agent when performing the task at hand. In contrast, SOA services are fixed and are available on a ‘take it or leave it’ basis.

The proposed approach is still a work in progress. We are currently considering the third role, namely the business agent role (see Fig.2). Our position is that in carrying his role, the business agent must be provided with the appropriate execution architecture particularly to handle variability and to allow the enactment to be driven by the intentional service composition. In this view, the business agent who retrieves an intentional service matching his needs will only know the intentional composition as expressed in the ISM model and will drive the selection and the execution of the component services by navigating in the intention/strategy map. In order to monitor the navigation among the composition of services and offer to the business agent the choice of variants he wants to execute, we are developing an agent architecture based on a hierarchy of agents managing service relationships and handing over the execution of atomic services. This will allow the business agent to dynamically navigate through aggregate services composition graphs using the agent platform. We consider that this solution is compatible with the business-oriented view of the ISM and the need for business agents to adapt decision making ‘on the fly’. We believe that it is possible for business people to perform this adaptation. This is because knowledge of the business characteristics and an analysis based on these is enough to make the adaptation decision. Our current research also aims at developing (a) an intention driven search mechanism for the selection of services on the basis of the business goal they allow to fulfill and (b) a software tool to guide the discovery of aggregate service through maps and c) a solution to deal with the non functional aspects all over the process.

REFERENCES


