A Requirement-Centric Approach to Web Service Modeling, Discovery, and Selection

Maha Driss[1,2], Naouel Moha[1], Yassine Jamoussi[2], Jean-Marc Jézéquel[1], and Henda Hajjami Ben Ghézala[2]

[1] IRISA/INRIA,University of Rennes 1, France
{mdriss,moha,jezequel}@irisa.fr
[2] ENSI, RIADI-GDL Laboratory, University of Manouba, Tunisia
{yassine.jamoussi,henda.benghezala}@ensi.rnu.tn

Abstract. Service-Oriented Computing (SOC) has gained considerable popularity for implementing Service-Based Applications (SBAs) in a flexible and effective manner. The basic idea of SOC is to understand users' requirements for SBAs first, and then discover and select relevant services (i.e., that fit closely functional requirements) and offer a high Quality of Service (QoS). Understanding users' requirements is already achieved by existing requirement engineering approaches (e.g., TROPOS, KAOS, and MAP) which model SBAs in a requirement-driven manner. However, discovering and selecting relevant and high QoS services are still challenging tasks that require time and effort due to the increasing number of available Web services. In this paper, we propose a requirement-centric approach which allows: (i) modeling users' requirements for SBAs with the MAP formalism and specifying required services using an Intentional Service Model (ISM); (ii) discovering services by querying the Web service search engine Service-Finder and using keywords extracted from the specifications provided by the ISM; and(iii) selecting automatically relevant and high QoS services by applying Formal Concept Analysis (FCA). We validate our approach by performing experiments on an e-books application. The experimental results show that our approach allows the selection of relevant and high QoS services with a high accuracy (the average precision is 89.41%) and efficiency (the average recall is 95.43%).

Keywords: Service-Based Applications, Users' Requirements Modeling, Service Discovery, Service Selection, QoS, Formal Concept Analysis.

1 Introduction

Service-Oriented Computing (SOC) is an emerging paradigm for developing low-cost, flexible, and scalable distributed applications based on Web services (WSs) [1]. SOC is becoming broadly adopted as it offers the ability to build efficiently and effectively added-value Service-Based Applications (SBAs) by composing ready-made services. The basic idea of SOC is to understand users' requirements for SBAs first, and then discover and select WSs that fit closely expected functional and non-functional requirements. Functional requirements define functionalities provided by WSs. We refer to services which fit closely users'

P.P. Maglio et al. (Eds.): ICSOC 2010, LNCS 6470, pp. 258–272, 2010.

functional requirements as 'relevant' services. Non-functional requirements are expressed by the term Quality of Service (QoS) that refers to various properties such as availability, response time, security, and throughput [2]. If multiple WSs offer the same functionality, then a QoS requirement can be used as a secondary criterion for service selection.

Understanding users' requirements includes requirements elicitation, analysis, and modeling which provide a full support for SBAs engineering. Discovering services is achieved by querying a WS search engine to browse WSs using several criteria (e.g., functionalities and QoS). Among the set of services obtained by discovery, only services that best match users' functional and non-functional requirements are selected. Understanding users' requirements is already achieved by traditional requirement engineering approaches that are extended and refined to meet the SBAs characteristics. To this end, goal modeling techniques such as TROPOS [3], KAOS [4], and MAP [5] are used to model SBAs. However discovering and selecting relevant and high QoS WSs is still a challenging task because of two main issues. First, the growing number and diversity of WSs in addition to their publication over multiple public and private registries make service discovery difficult to accomplish. Second, the frequent large number of WSs returned by discovery requires costly and time-consuming selection of relevant and high QoS services.

In this paper, we propose a new requirements-centric approach for: (i) modeling SBAs in terms of functional and non-functional users requirements; (ii) discovering potential services that match expected requirements; and (ii) selecting relevant and high QoS services. This approach consists of three successive steps 1-3 as it is shown by Figure 1.

In the first step, our approach allows an intentional-driven modeling of SBAs using the MAP formalism [5]. The MAP elicits and analyzes users' requirements in a set of graphs composed of intentions and strategies, called *maps*. In previous work [6], an Intentional Service Model (ISM) is proposed to specify intentional services presented by maps. In this paper, the same model will be enhanced to include QoS aspects and will be used to specify intentional services. In the second step, our approach permits discovery of operational services by querying the WS search engine Service-Finder using keywords extracted from ISM models. To efficiently

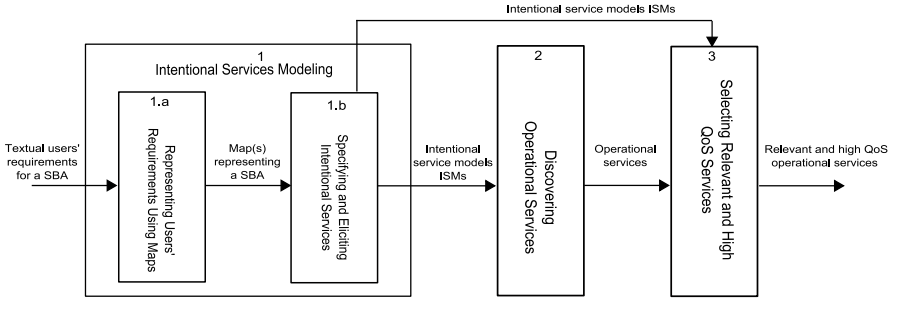

Fig. 1. A requirement-centric approach to WS modeling, discovery, and selection

discover relevant and high QoS services, we propose two-level filtration. In the first level, some QoS properties are considered namely validity (i.e., we verify if the service URI is valid) and availability (i.e., we verify if the service is operational). Services which pass the first level are filtered according to a semantic matchmaking between the intentional specification provided by ISM and the operational specification provided by WSDL. In the third step, the remaining set of services is classified into an ordered structure called concept lattice by applying Formal Concept Analysis (FCA) [7]. FCA is a formal framework that allows grouping individuals that share common properties and organizes them into concept lattices. The FCA will automate the selection task by providing a clear and organized view of potential services to enable users to easily check out relevant and high QoS services. We validate our approach by performing experiments on an e-books WS-based application. The experimental results show that our approach allows the selection of relevant and high QoS services with a high accuracy (the average precision is 89.41%) and efficiency (the average recall is 95.43%).

The remainder of the paper is structured as follows. In Section 2, we present the intentional-driven modeling of SBAs. In Section 3, we describe the WS discovery conducted using Service-Finder. In Section 4, we explain how to apply FCA to select relevant and high QoS WSs. Experimental results are documented in Section 6. Section 5 surveys related work. This paper ends with concluding remarks and future work.

2 Intentional Services Modeling

A considerable number of research efforts on SBAs modeling are conducted both in industry and academia. Indeed, these initiatives aimed to propose languages (e.g., BPEL4WS [8] and OWL-S [9]) and formalisms (e.g., Petri nets [10]) for modeling SBAs. All these initiatives adopt a 'function-driven' service modeling focusing on 'low level' technical statements (e.g., coordination messages, input/output parameters, and bindings) that are understandable by software programmers but far to be comprehensible by users. However, users need to interact with programmers to obtain SBAs satisfying their requirements. Thus, SBAs must be modeled in terms of users' requirements and not in terms of technical statements.

In this paper, we adopt a 'requirement-driven' approach that allows a 'high level' modeling of SBAs. This approach uses the MAP formalism to represent users' requirements. We refer to services presented by maps as intentional services. These services are specified by an Intentional Service Model (ISM).

In the following, we present an overview of the MAP formalism, we introduce the ISM model, and we provide guidelines to elicit intentional services and their composition from maps.

2.1 Representing Users' Requirements with Maps

A MAP is a meta-process formalism which allows designing several processes, i.e., maps, under a single representation. A map is a labelled directed graph

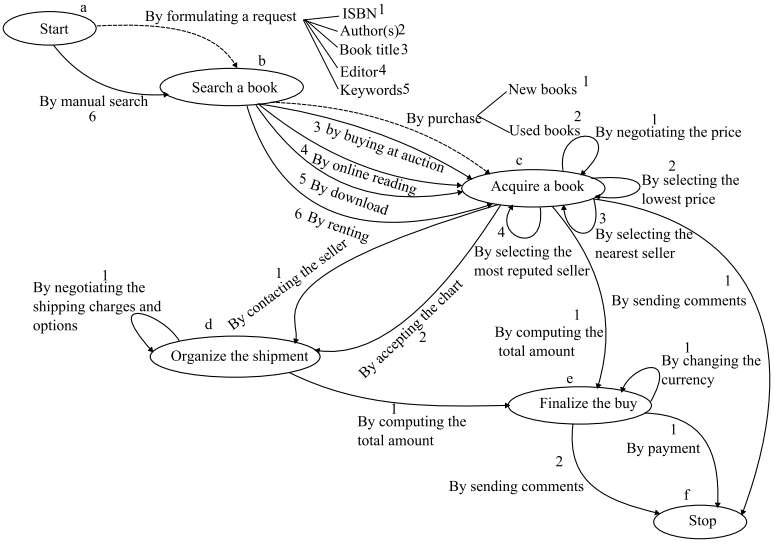

Fig. 2. The e-books application map

with intentions as nodes and strategies as edges between intentions. A strategy is a manner to achieve an intention. An intention is a requirement that can be achieved by following different strategies. Each map has two distinct intentions *Start* and *Stop* to respectively begin and end the navigation in the map. There are two main reasons for using the MAP formalism: first, the MAP was already applied to service modeling domain [11], so we can use previous knowledge and experiences. Second, the MAP permits to capture variability by focusing on the strategy to achieve an intention and the potential alternatives to accomplish the same intention. This explicit representation of variability offered by maps is missing in other requirement engineering formalisms such as TROPOS or KAOS.

Figure 2 represent users' requirements for an e-books application with the MAP formalism. The e-books application map has four key intentions to be achieved, namely *Search a book*, *Acquire a book*, *Organize the shipment*, and *Finalize the buy*. To achieve the *Search a book* intention, users can follow the *By manual search* strategy or the *By formulating a request* strategy.

A map is composed of one or more sections. A section is a triplet <Source Intention I_i, Target Intention I_j, Strategy S_{ij}> that captures a specific manner to achieve the target intention I_j starting from the source intention I_i with the strategy S_{ij}. For instance, <*Start*, *Search a book*, *By formulating a request*> represents a way to achieve the target intention *Search a book* from the source intention *Start* following the *By formulating a request* strategy.

There exist four relationships between sections: *bundle*, *multi-thread*, *path*, and *multi-path* relationships.

Bundle relationship: sections in a bundle are mutually exclusive; exactly one strategy can be used to realize the target intention. In Figure 2, *By formulating*

a request is a bundle consisting of five different strategies: *By ISBN*, *By author(s)*, *By book title*, *By editor*, and finally *By Keywords*.

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. One or more of these sections can be used to realize the target intention. For example, *By formulating a request* and *By manual search* are two different strategies to *Search a book*. These two strategies are in a *multi-thread* relationship.

Path relationship: this establishes a precedence/succession relationship between sections. For example, the sections <*Start*, *Search a book*, *By formulating a request*> and <*Search a book*, *Acquire a book*, *By purchase*> constitute a *path*.

Multi-path relationship: given the three previous strategies, a target intention can be achieved by combining several sections. For instance, there are two distinct paths to achieve the intention *Finalize the buy* from *Start*. The first is the path via *Search a book*, *Acquire a book*, and *Organize the shipping* intentions. The second is the path via only the *Search a book* and the *Acquire a book* intentions.

In general, a map from its *Start* to its *Stop* intention is a *multi-path* and may contain *multi-threads*. Finally, it is possible to refine a section of a map into another map. Refinement is an abstraction mechanism by which a complex assembly of sections at level i+1 is viewed as a unique section at level i. Figure 3 shows the refinement of the section <*Finalize the buy*, *Stop*, *By payment*> as a map. This map is composed of two key intentions *Sign in* and *Finish the payment* and it provides several strategies to achieve each of them.

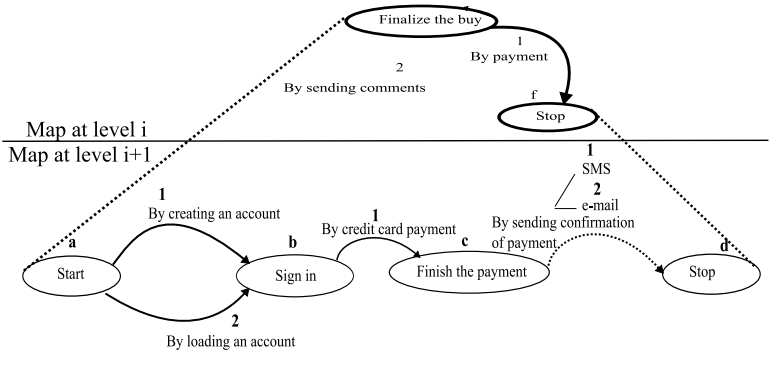

Fig. 3. The refinement of the section <*Finalize the buy*, *Stop*, *By payment*>

2.2 Specifying Intentional Services

Intentional services are services presented by maps. Intentional services allow the achievement of users' requirements represented as intentions using the MAP formalism. Intentional services are specified by the Intentional Service Model

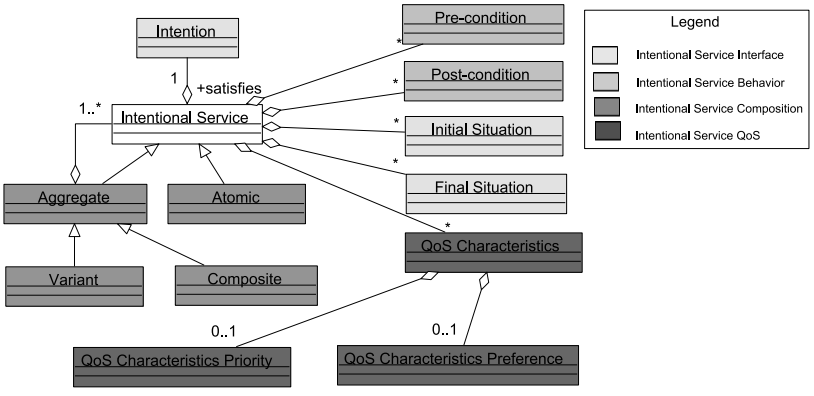

Fig. 4. The intentional service model (ISM)

(ISM). Figure 4 presents the ISM meta-model using UML notations. As shown by colors used in Figure 4, the ISM describes intentional services through four main aspects: the intentional service interface, behavior, composition, and QoS. We describe each of these aspects in the following paragraphs.

The Intentional Service Interface. There are three elements *that constitute* interface, namely *Intention*, *Initial Situation*, and *Final Situation*. The key idea of the ISM is that an *intentional* service allows the achievement of an intention given an initial situation and terminating with a final situation. The intention replaces the functionalities that can be achieved by the service. The achievement of an intention will ultimately lead to a state that is expected to be reached or to be maintained. The *Initial Situation* and the *Final Situation* represent respectively the input and output parameters of the intentional service.

The Intentional Service Behavior. *Pre-condition* and *Post-condition* describe the service behavior aspect. *Pre-condition* and *Post-condition* are respectively the initial and final state, i.e., the state requiring the achievement of the intention and the state resulting from its achievement.

The Intentional Service Composition. *Atomic* and *Aggregate* services are involved in the service composition aspect. An *Atomic* service has an operationalized intention that can be achieved directly by an atomic operational service. An *Aggregate* service has a high level intention that should be decomposed till atomic operational services are identified. Aggregation of services can result either by *Composite* or *Variant* services. *Composite* services express the precedence/succession relationship between intentional services. We distinguish three types of composition: *Sequential*, *Parallel*, and *Iterative*. *Variant* services describe variability needed for SBAs. There are three variants in the ISM, namely *Alternative*, *Choice*, and *Multi-path*.

The Intentional Service QoS. There are three elements specifying QoS of intentional services, namely *QoS Characteristic*, *QoS Characteristic Priority*, and

QoS Characteristic Preference. *QoS Characteristic* is the quality to be attained or preserved. *QoS Characteristic Priority* allows the expression of a priority over a *QoS Characteristic*. *QoS Characteristic Preference* allows the expression of a preference over a *QoS Characteristics*.

2.3 Eliciting Intentional Services from Maps

The MAP is used to represent intentional services satisfying users' requirements. To elicit intentional services and their composition from maps, we follow three key guidelines [6]:

1. The first guidline consists of associating every non-refined section of a map to an atomic service. In the case of the e-books application map, this correspondence leads to a set of 29 atomic services including for instance: $ab1_i \rightarrow S_{Search\ a\ book\ by\ ISBN}$ and $bc1_{i+1} \rightarrow S_{Pay\ with\ credit\ card}$[1].
2. The second guideline consists of identifying paths of a map by applying an algorithm that calculates paths in a graph. This algorithm is an adaptation of the MacNaughton and Yamada's algorithm [12]. This algorithm is based on formula that calculate the set of all possible paths between an initial node and a final node of a graph which, in our case, are respectively the intentions *Start* and *Stop*.
3. The third guideline consists of determining aggregate services by establishing the following correspondences between section relationships in the map and aggregate service types in the ISM model: <*Path-Composite*>, <*Bundle-Alternative*>, <*Multi-thread-Choice*>, and <*Multi-path-Multi-path*>.

3 Discovering Operational Services

Discovering operational services is proceeded by querying Service-Finder [13]. Service-Finder is a Web 2.0 platform for WS discovery. It is managed to search and access existing WSs on the Web. It searches among almost 25.000 WSs from more than 200.000 related Web pages [13]. In order to understand the problem of WS discovery and selection, we are going to demonstrate an example of a WS from the e-books application. Naturally, discovery and selection should be processed for all intentional services in order to build the e-books application. We consider the service $S_{Pay\ with\ credit\ card}$ presented by the section bc1 of the e-books application map at level i+1. This service performs payment with a credit card. Figure 5 shows the intentional model of the service $S_{Pay\ with\ credit\ card}$ according to the ISM.

$S_{Pay\ with\ credit\ card}$ is an atomic service that ensures the achievement of the intention *Pay with credit card*. $S_{Pay\ with\ credit\ card}$ takes 2 input parameters that are *Book* and *Card* and provides *Payment* as output. $S_{Pay\ with\ credit\ card}$

[1] For sake of conciseness, we use an abbreviated notation to refer to a section of a map. We refer to each intention by a letter starting from a and to each strategy by a digit starting from 1. Levels of maps start from i.

```
<?xml version="1.0" encoding="ASCII"?>
<ism:Map xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ism="http://ism/1.0"
xsi:schemaLocation="http://ism/1.0 ISM.ecore">
  <service xsi:type="ism:Atomic"
  intention="//@Intention.0"
  pre_condition="//@Pre-Condition.0"
  post_condition="//@Post-Condition.0"
  initial_situation="//@Initial Situation.0"
  final_situation="//@Final Situation.0"
qos_characteristic="//@QoS Characteristic.0"
qos_characteristic="//@QoS Characteristic.1"
  id="Pay with credit card"/>
  <Intention description="Pay with credit card"/>
  <Pre-Condition value="Book.Cart=true"/>
  <Post-Condition value="Payment.State=true"/>
  <resource name="Book"/>
  <resource name="Payment"/>
  <resource name="Card"/>
  <Initial Situation input="//@resource.0 //@resource.2"/>
  <Final Situation output="//@resource.1"/>
  <QoS Characteristic id="Respons time">
    <QoS Priority value="3"/>
    <Qos Preference value="Very low"/>
  </QoS Characteristic>
  <QoS Characteristic id="Availability">
    <QoS Priority value="1"/>
    <Qos Preference value="Very high"/>
  </QoS Characteristic>
</ism:Map>
```

Fig. 5. The intentional model of the service $S_{Pay\ with\ credit\ card}$

has as pre-condition *Book.Cart=true* and as post-condition *Payment.State=true*. $S_{Pay\ with\ credit\ card}$ is characterized by a very high availability and a very low response time. Response time is three time more important for the users than the availability. The QoS preferences and priorities are given by users.

We query Service-Finder using the keywords 'payment + credit + card'. We extract these keywords from the intentional model $S_{Pay\ with\ credit\ card}$ using a well known Information Retrieval (IR) metric called TF/IDF [14]. All words of a service specification are weighted with TF/IDF metric. TF/IDF metric allows us to filter out both stop words and low frequency words. Only meaningful words (i.e., keywords) having a high TF/IDF weight are maintained. Service-Finder returns a result set of 77 WSs[2]. To reduce this set, we process two-level filtration. In the first level, we consider some QoS properties namely validity and availability. These two properties are checked as follows: (1) validity is verified by checking whether the endpoint URI exists or not; (2) Availability is verified by checking if the service is operational or not. This first filtration level generates a new reduced set that contains 37 WSs. These services are passed to a second level filtration which is based on a semantic matchmaking between the intentional model of the service $S_{Pay\ with\ credit\ card}$ and WSDLs of the returned operational services. This is done by parsing every service WSDL in order to check if required operation with specified signature exists or not. An operation signature is the combination of the operation name, its input parameters, and its output. In our example, the required operation is *Pay with credit card*, its inputs are *Book* and

[2] The discovery result set is obtained on June 13^{th}, 2010.

Card, and its output is *Payment*. A final set of 16 WSs is obtained after the second level filtration. These remaining services will be organized into concept lattices using FCA in order to facilitate the selection task.

4 Selecting Relevant and High QoS Operational Services

To automate the selection of relevant and high QoS services, Formal Concept Analysis (FCA) [7] is used.

4.1 Introducing FCA

FCA offers a formal framework for clustering individuals along the properties they share. It describes clusters, called formal concepts, both extensionally and intentionally, i.e., as sets of individuals and sets of shared properties, and organises them hierarchically, according to a binary incidence relation, into a complete lattice, called the concept lattice [7]. FCA considers a dataset as being organised into a formal context, i.e., a triple $\mathcal{K} = (\mathcal{O}, \mathcal{A}, \mathcal{I})$, where $\mathcal{O}$ is a set of individuals, $\mathcal{A}$ is a set of properties, and $\mathcal{I}$ is the binary incidence relation between $\mathcal{O}$ and $\mathcal{A}$, $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{A}$.

For our problem, we define a context $\mathcal{K}$ where individuals represent services and properties represent QoS properties. We consider 2 QoS properties: availability and response time. Real time monitoring information of service availability and response time are provided by Service-Finder. The binary incidence relation is the *service-has-QoS property* relationship. To spread out availability and response time values, we use the boxplot statistical technique [15]. A boxplot splits a set of numerical values into four quarters called quartiles. We map these quartiles into a five-point Likert scale with the following ordinal values: very high, high, medium, low, and very low. Then, we associate QoS values with these ordinal values. As specified in the intentional model of the service $S_{Pay\ with\ credit\ card}$, users give three time more importance to the service response time than to the availability (see Figure 5). So, we define for each response time ordinal value three sub-values in order to express the priority. For instance, we consider the sub-values: very low 1, very low 2, and very low 3 for the ordinal value very low. Service having a very low response time should has an incidence relationship with the three sub-values of the ordinal value very low. Also, we consider that if a service has an incidence relationship with a response time ordinal value (availability ordinal value, respectively), it should has incidence relationship with all ordinal values that comes after (that comes before, respectively). For instance, if a service has a very low response time (very high availability, respectively) it has also low, medium, high and very high response time (high, medium, low , and very low availability, respectively). Relevant and high QoS of services are services which have more incidence relationships with context properties.

Table 1 illustrates the context drawn from the final obtained set of services. It shows the 16 services (individuals in rows), and their has-relationship links with availability and response time (properties in columns).

Table 1. Context $\mathcal{K}$ linking services to QoS characteristics

	(A1) Very Low Availability	(A2) Low Availability	(A3) Medium Availability	(A4) High Availability	(A5) Very High Availability	(RT11) Very Low Response Time 1	(RT12) Very Low Response Time 2	(RT13) Very Low Response Time 3	(RT21) Low Response Time 1	(RT22) Low Response Time 2	(RT23) Low Response Time 3	(RT31) Medium Response Time 1	(RT32) Medium Response Time 2	(RT33) Medium Response Time 3	(RT41) High Response Time 1	(RT42) High Response Time 2	(RT43) High Response Time 3	(RT51) Very High Response Time 1	(RT52) Very High Response Time 2	(RT53) Very High Response Time 3
(s0) XWebCheckOut	×	×	×	×																
(s1) PaymentWS	×	×	×						×		×	×	×	×	×	×	×	×	×	×
(s2) Pay-Service	×	×	×						×		×	×	×	×	×	×	×	×	×	×
(s3) Checkout	×	×	×	×	×							×	×	×	×	×	×	×	×	×
(s4) Book247XMLWebServices	×	×	×		×							×	×	×	×	×	×	×	×	×
(s5) ExigoAPI	×	×	×									×	×	×	×	×	×	×	×	×
(s6) FSI	×	×	×	×	×				×	×	×	×	×	×	×	×	×	×	×	×
(s7) CEPayProcessor	×	×	×									×	×	×	×	×	×	×	×	×
(s8) SmartPayment2	×	×	×									×	×	×	×	×	×	×	×	×
(s9) BasicOperations	×	×	×	×	×				×	×	×	×	×	×	×	×	×	×	×	×
(s10) Order	×											×	×	×	×	×	×	×	×	×
(s11) MemberServices	×	×	×	×	×				×	×	×	×	×	×	×	×	×	×	×	×
(s12) SmartPayments	×	×	×									×	×	×	×	×	×	×	×	×
(s13) SBWebServices	×	×	×	×	×							×	×	×	×	×	×	×	×	×
(s14) MWService	×											×	×	×	×	×	×	×	×	×
(s15) OMService	×																	×	×	×

4.2 Using Concept Lattices to Select Relevant and High Quality Operational Services

FCA organizes formal concepts into complete lattices, called concept lattices. The lattice structure allows easy navigation and search as well as optimal representation of information. Figure 6 depicts a simplified (reduced) labeling of the concept lattice derived from our context $\mathcal{K}$. This lattice is built using the Galicia (Galois Lattice Interactive Constructor) [16] tool. Galicia is a multi-tool open-source platform for creating, visualizing, and storing concept lattices. Our lattice contains information that can be interpreted using the following set of rules:

- The concepts are represented using the intent (I) and the extent (E) sets.
  - A service that appears in the extent set E of a concept is inherited by all the concepts that are above it.
  - A QoS value that appears in the intent set I of a concept is inherited by all the concepts that are below.
- When the extent set E is not empty, the concept represents exactly the service(s) that is/are in the extent set E with its/their QoS value(s) in the I set.
- When the extent set E is empty, this signifies that the concept represents a service specification that does not exist in the services set.

From these rules, we can conclude that relevant and high QoS services are services presented by the concept which is in the bottom of our lattice (node 4). The extent set of this concept includes 4 services: *Basicoperations*, *FS*I, *MemberServices*, and *Pay-Service*. These services have a high availability (A4) and a low response time (RT21, RT22, and RT23).

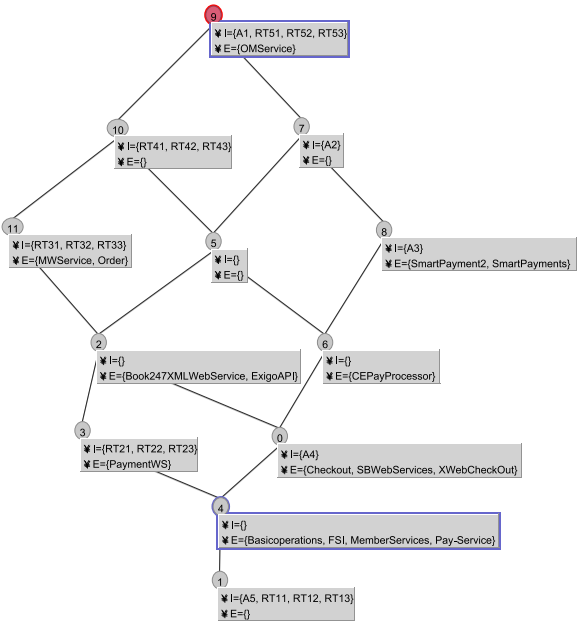

Fig. 6. The lattice of the context $\mathcal{K}$

5 Experiments

We validate our approach on the 29 intentional services of the e-books application. We verify manually that the services returned by our approach correspond to real relavant and high QoS services. We recast our work in the domain of information retrieval and we use the precision and recall measures [17]. Precision assesses the number of true relevant and high QoS operational services identified among the returned set of operational services, while recall assesses the number of returned operational services among the existing relevant and high QoS services, according to the following equations:

$$\text{precision} = \frac{\left|\{\text{true relevant and high QoS operational services}\} \cap \{\text{returned operational services}\}\right|}{\left|\{\text{returned operational services}\}\right|}$$

$$\text{recall} = \frac{\left|\{\text{true relevant and high QoS operational services}\} \cap \{\text{returned operational services}\}\right|}{\left|\{\text{true relevant and high QoS operational services}\}\right|}$$

Due to the limitation of paper space, we show only, in Table 2, experimental results related to 10 intentional services. However, we provide the precision and recall average of the 29 intentional services of the e-books application. In Table 2, the first column corresponds to intentional services. In the second column, we list, first, the keywords used to query Service-Finder; then, the number of

Table 2. Precision and recall of services of the e-books application

Intentional services	Service-Finder			Our Approach			Precision	Recall
	Keywords	Returned operational services	True relevant and high QoS operational services	Discovery	Selection	True relevant and high QoS operational services		
$ab1_i \rightarrow S_{Search\ a\ book\ by\ ISBN}$	'search + book + isbn'	7	2/7	4/7	2/4	2/2	2/2 (100%)	2/2 (100%)
$ab2_i \rightarrow S_{Search\ a\ book\ by\ author(s)}$	'search + book + author'	16	3/16	10/16	4/10	3/4	3/4 (75%)	3/3 (100%)
$ab3_i \rightarrow S_{Search\ a\ book\ by\ title}$	'search + book + title'	31	5/31	22/31	6/22	5/6	5/6 (83.33%)	5/5 (100%)
...	...	...	...	...	...	...	...	...
$cc2_i \rightarrow S_{Sort\ books\ by\ price}$	'sort + price'	64	11/64	52/64	9/52	8/9	8/9 (88.89%)	8/11 (72.73%)
$cc3_i \rightarrow S_{Sort\ books\ by\ seller\ location}$	'sort + location'	58	12/58	40/58	11/40	11/11	11/11 (100%)	11/12 (91.67%)
...	...	...	...	...	...	...	...	...
$ee1_i \rightarrow S_{Change\ the\ currency}$	'change + currency'	55	8/55	30/55	7/30	6/7	6/7 (85.71%)	6/8 (75%)
$ab1_{i+1} \rightarrow S_{Create\ an\ account}$	'create + account'	330	26/330	203/330	25/203	22/25	22/25 (88%)	22/26 (84.62%)
$ab2_{i+1} \rightarrow S_{Load\ an\ account}$	'load + account'	29	4/29	20/29	5/20	4/5	4/5 (80%)	4/4 (100%)
$bc1_{i+1} \rightarrow S_{Pay\ with\ credit\ card}$	'payment + credit + card'	77	4/77	16/77	4/16	4/4	4/4 (100%)	4/4 (100%)
$cd1_{i+1} \rightarrow S_{Send\ sms}$	'send + sms'	162	17/162	105/162	18/105	15/18	15/18 (83.33%)	15/17 (88.24%)
						Average	89.41%	95.43%

services returned by Service-Finder; and finally, the number of services identified manually as true relevant and high QoS services. In the third column, we enumerate the number of services obtained, first, after the discovery step of our approach, and then, after the selection step, and finally the number of true services among those returned after selection step. The two last columns correspond to the precision and the recall. For example, the two keywords of the intentional service $S_{Send\ sms}$ are: 'send + sms'. The query returns an initial set of 162 operational services. Among this set, only 17 services are verified manually as true relevant and high QoS services. The discovery step reduces the initial set to a second set of 105 services. The selection step using FCA reduces the second set to 18 services. Among these services, only 15 services are verified manually as true relevant and high QoS services. The precision of $S_{Send\ sms}$ is 83.33% and he recall is 88.24%.

Table 2 shows that the precision and recall of our approach are both very high. The average precision is 89.41% and the average recall is 95.43%.

To increase the robustness of our approach, we need to use more advanced semantic techniques for the filtration of the discovered operational services. Moreover our selection is based only on two QoS properties. Thus, we need to enhance our selection method with multiple QoS properties to identify more efficiently high QoS services.

6 Related Work

Recently, several requirements-driven approaches for service composition have been proposed. The work presented by Pistore *et al.* [18] is the first work that takes in the challenge of deriving service compositions by refining TROPOS requirements models. The key idea of this work is to enrich formal TROPOS models with BPEL4WS code and exploit model checking techniques to ensure verification and validation. More recently, the MAP formalism has been used in [6] to describe services compositions. Main contribution of this work takes on the proposition of an Intentional Service Model (ISM) to model, retrieve, and compose services in an intentional manner. Unfortunately, this model omits QoS aspects. All the works mentioned above focus only on modeling service composition in a requirement driven manner. They do not provide solutions for service discovery and selection to ensure the satisfaction of the users' requirements. A similar work to ours is provided by Zachos *et al.* [19]. The goal of this work is to align and refine requirements to the available services. The key idea of the proposed approach is to create an initial requirements specification, transform it into a registry query and use the query results to refine the specification. This work suffers from several problems: (i) the textual and incomplete requirements description; (ii) the lack of the non-functional requirements description; and finally (iii) the difficulty to refine the initial requirements based on service query result.

The solution that we present in this paper is complementary to [18], [6], and [19] since our approach permits not only SBAs modeling in terms of functional and non-functional requirements but also the discovery and the selection of services that fit closely expected functional requirements and offer a high QoS. To handle the problem of service selection we apply the FCA. Several approaches using FCA in the context of WSs have been proposed. We detail these approches in the following paragraphs.

Peng *et al.* [20] present an approach to classify and select services. They build lattices starting from contexts where individuals are WSs and properties represent the operations of these services. The approach allows similar services clustering by applying similarity search techniques that compare operation descriptions and input/output messages' data type. The contributions of this work are complementary to ours, insofar as they do not deal with QoS properties in the selection task. Azmeh *et al.* [21] present a similar approach to classify and select services using the FCA. They propose WSPAB (Web Service Personal Address Book) tool that permits the discovery, the automatic classification, and the selection of WSs. This tool processes by multiple successive steps. It first queries the service registry to find a first set of candidate services. Secondly, it filters this service set according to functional and non-functional criteria. Finally, the set of remaining services is classified into a service lattice using FCA. Classification is accomplished by defining a binary relation between services and operation signatures. The obtained lattice can be used to identify relevant services and their substitutes. In contrast to our approach, this work does not consider QoS properties to process WS selection. Also, this work is purely syntactic (signature-based) while our work proposes a semantic filtration of discovered services based on the intentional models.

7 Conclusion

In this paper, we presented a requirement-centric approach to WS modeling, discovery, and selection. This approach consists of three successive steps. First, our approach allows an intentional-driven modeling of SBAs using the MAP formalism. Intentional services, that are presented by maps, are specified with the ISM model. Secondly, our approach permits discovery of operational services by querying the WS search engine Service-Finder using keywords extracted from specifications provided by ISM. To efficiently discover relevant services, we propose two-level filtration. In the first level, some QoS properties are considered namely validity and availability. Services which pass the first level are filtered according to a semantic matchmaking between the intentional specification provided by ISM and the operational specification provided by WSDL. Finally, the remaining set of services is classified into concept lattices using FCA. We consider contexts with services as individuals and QoS characteristics as properties. The obtained concept lattices are used to check out relevant and high QoS services. We validate our approach by performing experiments on an e-books application. The experimental results show that our approach allows the selection of relevant and high QoS services with a high accuracy (the average precision is 89.41%) and efficiency (the average recall is 95.43%). Future work includes: (i) studying semantic similarity techniques to enhance the filtration of the discovered operational services; (ii) considering multiple QoS properties, during the selection step, to identify more efficiently high QoS services; and (iii) applying Relational Concept Analysis (RCA) to identify high QoS composite services since the presented approach is applicable to atomic services.

Acknowledgments. This work has been supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube). (`http://www.s-cube-network.eu/`).

References

1. Huhns, M.N., Singh, M.P.: Service-oriented computing: Key concepts and principles. IEEE Internet Computing 9(1), 75–81 (2005)
2. Menascé, D.A.: Qos issues in web services. IEEE Internet Computing 6(6), 72–75 (2002)
3. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
4. Lamsweerde, A.V., Letier, E.: Handling obstacles in goal oriented requirements engineering. IEEE Transactions on Software Engineering 26(10), 978–1005 (2000)
5. Rolland, C., Prakash, N.: Bridging the gap between organizational needs and erp functionality. Requirements Engineering 5(3), 180–193 (2000)
6. Rolland, C., Kaabi, R.S., Kraeim, N.: On isoa: Intentional services oriented architecture. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 158–172. Springer, Heidelberg (2007)

7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, New York (1999)
8. BPEL4WS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
9. OWLS, http://www.w3.org/Submission/OWLS/
10. Mecella, M., Presicce, F.P., Pernici, B.: Modeling e-service orchestration through petri nets. In: Buchmann, A., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. (eds.) TES 2002. LNCS, vol. 2444, pp. 38–47. Springer, Heidelberg (2002)
11. Kaabi, R.K., Souveyet, C., Rolland, C.: Eliciting service composition in a goal driven manner. In: ICSOC 2004, New York, USA, pp. 308–315 (2004)
12. MacNaughton, R., Yamada, H.: Regular expressions and state graphs for automata. IEEE Transactions on Electronic Computers 9(1), 39–47 (1960)
13. Service-Finder, http://www.service-finder.eu/
14. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
15. Chambers, J.M., Clevelmd, W.S., Kleiner, B., Tukey, P.A.: Graphical methods for data analysis. Wadsworth & Brooks / Cole, Belmont (1983)
16. Galicia, http://galicia.sourceforge.net/
17. Frakes, W.B., Baeza-Yates, R.: Information Retrieval: Data Structures and Algorithms. Prentice-Hall, Englewood Cliffs (1992)
18. Pistore, M., Roveri, M., Busetta, P.: Requirements-driven verification of web service. In: WSFM 2004, Pisa, Italy, pp. 95–108 (2004)
19. Zachos, K., Maiden, N., Zhu, X., Jones, S.: Discovering web services to specify more complete system requirements. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 142–157. Springer, Heidelberg (2007)
20. Peng, D., Huang, S., Wang, X., Zhou, A.: Management and retrieval of web services based on formal concept analysis. In: CIT 2005, Shanghai, China, pp. 269–275 (2005)
21. Azmeh, Z., Huchard, M., Tibermacine, C., Urtado, C., Vauttier, S.: Wspab: A tool for automatic classification & selection of web services using formal concept analysis. In: ECOWS 2008, Dublin, Ireland, pp. 31–40 (2008)