# Description and composition of e-learning services

Oussama Kassem Zein and Yvon Kermarrec
ENST Bretagne, Dept. LUSSI
Technopôle Brest Iroise, BP 832, 29285 Brest Cedex, France
Email : {Oussama.Zein,Yvon.Kermarrec}@enst-bretagne.fr
Tel : + 33 2 29 00 12 85 Fax : + 33 2 29 00 10 30

***Abstract*— In this paper, we present our approach to describe and compose services for distant learning and research activities. For this purpose, we propose a metadata model for indexing services with three dimensions: a service can be viewed as a learning resource, as a resource that contribute and help researchers to perform their tasks in a community and as a service with a large scope. This model can be used by clients (learners/teachers or researchers) to query and discover services via for example a facilitator we have developed. We propose an approach for automatic composition of services which is based on facilitator. It relies on the semantic description of service behavior. We describe the latter with an automaton. This includes the description of the interface through the inputs/outputs and conditions (post and pre-conditions) that control the operations. We extend the functionalities of the facilitator to become a composition engine that enables services or resources to be composed automatically by comparing their inputs/outputs flows to help learners/researchers in their distant learning and research activities.**

**Index terms: service description, service discovery, service composition, facilitator.**

## I. Introduction and Context

Many different approaches for describing, managing and providing learning resources have been developed over the years to accommodate the massive flow of information and learning materials of the internet like Dublin Core [1] and LOM (Learning Object Metadata) [2]. Nevertheless, a clear understanding and consensus about what constitutes a learning resource has not yet been reached. Even though, we can find analogies with software engineering and components.

These approaches provide a set of properties that characterize a learning resource. These properties can be used by clients (learners/teachers) to discover e-learning services via UDDI [3] or a trader we have developed [4].

In this paper, we propose to extend the nature of a learning resource to be a service and to make it accessible via through an interface, as it is the case for a learning service. In this context, our proposal is to make a learning resource not only as a downloadable thing- that is only executed on the client machine with no interaction with servers during execution time- but also as a service accessible by an interface. We define a metadata model for learning services description which can be stored in UDDI or in our trader [4]. We have used ontologies [5] to index and store this model as for our previous R&D activities. An ontology is a consensus between clients and service providers and it enables cooperation and sharing a common vocabulary. Clients and providers that can be teachers or students can use this model to query and publish learning services through ontologies.

We extend the metadata model to make a client not only a learner or a teacher but also a researcher (a PhD student, a professor, etc.) to make distant research activities. In this context, a researcher can publish his services (software, tools, simulation results, etc.) that can contribute and help other researchers in their research activities. So, how can we describe these services to enable the distant research activities between researchers?

We have implemented this work in a network of excellence named *Kaleidoscope* [6] which is composed of a community of researchers/educators. A goal of Kaleidoscope is to provide a shared virtual laboratory aimed a community of researchers and educators to interact and to share information.

Based on SDL [7] and Interface Automaton [8], we have described the service behavior with an automaton and its inputs/outputs [9]. In our previous work, we describe the inputs/outputs by their names which are sequences of characters or strings without any semantic information (about their types and their compatibilities in particular). In this paper, we extend this description to address semantic aspects. It includes the description of post and pre-conditions and extends the description of the inputs/outputs by concepts in the ontology. We present the advantages of our approach compared to existing approaches like WSMO [10].

To invoke the service operations, a client needs information about the service interface. In this context, we have described the service interface (service operations, their parameters and so on). This information is necessary for client that discovers services at run-time to understand how he/she can formulate his/her request to invoke services dynamically by using for example the DII (Dynamic Interface Invocation) [11] of CORBA.

Based on our trader [9] and the service behavior description, we have developed an approach for service

composition [12]. It allows us to combine services by connecting their inputs/outputs which are strings. This approach is simple and powerful because the services can be easily queried. In this paper, we extend the composition mechanisms by connecting the inputs/outputs which are described by concepts in the ontology. We have chosen to use and develop facilitators based on ontologies that allow the indirect interactions between clients and service providers. So, the client (learner/teacher or researcher) looks for a service and the facilitator searches and invokes it and returns the result to the client directly. We extend the functionality of the facilitator to be a composition engine. In this context, the facilitator has the same functions of the trader an it can also execute a composed service, without intervention of the client, and returns the final result to the client. With the facilitator, the composition becomes automatic. Therefore, clients can benefit from the functionalities of many services to get the appropriate results in their distant learning and research activities and access resources made available by them in the community.

This paper is organized as follows. Section 2 presents our proposal for a metadata model that describes and characterizes learning and research services. Section 3 illustrates our approach of facilitator based on ontologies that allows the service discovery and composition. It provides interfaces that allow clients and servers to query and advertise learning services based on the underlying use of ontologies. Section 4 presents our approach for semantic description of the service behavior. It takes into account the description of post and pre-conditions and the inputs/outputs flows. Section 5 illustrates our approach for service interface description. Section 6 presents our approach of service composition based on facilitator when a service search is unsuccessful. It allows services to be composed automatically. Section 7 illustrates an application using our approach to describe documents. Section 8 presents related work and on-going trends and directions. Finally, the conclusion raises issues and presents our future work in section 9.

## II. Our Proposed Metadata Model

In this section, we present our metadata model for learning services. We propose to describe a learning service with three dimensions: as a learning resource, as a service that contributes and helps researchers to perform their tasks in a community and as a general purpose service. This model provides a set of properties that characterizes learning services. We describe the various dimensions :

1) **Dimension 1: learning resource**

We have based our model on Dublin Core [1] and IEEE LOM [2] to ensure compatibility with existing platforms and to allow the description of a learning resource in general. They define a set of elements like:

- **Title**: the name given to the resource by its creator.
- **Creator**: The person(s) or organization(s) primarily responsible for the intellectual content of the resource.
- **Subject**: the topic of the resource, or keywords or phrases that describe the subject or content of the resource.
- **Description**: a textual description of the content of the resource, including abstracts in the case of document and objects and content descriptions in the case of visual resources.
- **Publisher**: the person(s) or organization(s) in addition to those specified in the Creator element who have made significant intellectual contributions to the resource but whose is secondary to the individuals or entities specified in the Creator element (for example, editors).
- **Date**: the date the resource was made available in its present form.
- **Language**: language(s) of the intellectual content of the resource.
- **Format**: the data representation of the resource, such as text/html, Postscript file, JPEG image and so on.
- **Rights**: the content of this element is intended to be a link (a URL for example) to a copyright notice, a rights-management statement and so on.

2) **Dimension 2: research resource**

Our investigation is to describe services that contribute and help researchers to collaborate, to work together and to exchange the information in their research activities. A service can be a software (a program/application or a component), a document and so on that is provided by a researcher (the author) and can be used by other researchers to help them in their research activities. For this purpose, we must describe the context of the component precisely and we have identified three major fields: the research areas covered by the service, the researchers and the type of service (software, document) provided and so on.

A research area can be described by:

- **Indexes**: key words that describe the research area.
- **References**: a set of papers, documents and URLs that are useful in the research area.
- **Links**: which are towards other areas and topics which are linked to the research area.
- **Description**: a short textual description of the research area.

A researcher (the author of the service) can be described by:

- **Personal information**: name, email, phone number(s), web site, CV, etc.
- **Quality and title**: PhD thesis, professor or engineer for example.
- **Research areas**: the list of the research areas.
- **Teaching areas**: the list of the teaching areas.
- **Publications**: references to the published papers.
- **Department**: to which the researcher is attached.

These indexes can have default values according to provider profile.

A research department can be described by the following information :

- **Name**: of the department and the university or organization to which the department is attached.
- **Address**: the location of the department.
- **Description**: a short description of the department.
- **Research areas**: the list of the research orientations of the department.
- **List of researchers** attached to the department.
- **Research groups** constituted in the department.

3) **Dimension 3: service with a large scope**

We have based our proposal on [13], [14], [15], [16] to describe a service with a large scope. We can identify a service by a set of characteristics like: delivery systems (in the case of learning services), payment and pricing, location, quality of service, requesting and delivery channels (PDA, mobile phone, etc.), helping tools (chatting room, mailing list, etc.) and so on.

We present a few of these properties:

- **Provider and Location**: the Provider defines the characteristics of the service provider including his name and his address (e.g. the name of his city, street names, postal codes, and country codes, etc.). The service location can be a company address or an URL address, and so on.
- **Request and delivery channels**: with the introduction of the internet and new communication devices (mobile phones, pagers, etc.), there has been an increase in the number of request and delivery channels available to consumers. A channel is the means by which a user requests a service or receives the resulting output from a service.
- **Payment and pricing**: payment is the business process defined by the service provider for collecting the price of the service from the consumer. It can be conducted in single or multiple stages (i.e, installments), using various mediums (e.g, direct cash exchange, credit or debit card, cheque, direct debit, account, etc.). Payment can be made before delivery, at delivery, after delivery, or any combination of the above.
  Pricing is the charge for the service being provided. It is largely at the discretion of the service provider and as such, we consider a service to have a nominal price.
  Pricing and payment can include the identifier of the entity to which the payment is addressed, which can be different from the identifier of the service provider. They include the payment channel. This is the method used for conducting the payment (e.g, internet, email, post and phone, etc.).
- **Environment** : it determines the characteristics of the environment used to exploit the service like the bandwidth and which may appear as requirements.
- **Delivery system**: like visio-conference in the case of learning resources.
- **Helping tools**: the tools (like chatting room, mailing list and so on) that help clients to use the service.
- **Service type** : can be a learning resource, a software, a program, a document and so on.

The end users of our metadata model can be students (“dimensions 1 and 3”, for example to get a document or a course ), teachers (“dimensions 1, 2 and 3”, for example to get course materials ) and researchers (“dimensions 2 and 3”, for example to get simultation results).

The metadata model provides a complete vision about learning service description since it allows clients and servers to query and advertise services using in 3 different points of views. It is flexible because it is compatible with existing approaches like IEEE LOM, Dublin Core and DAML-S. These latters are complex and the indexation stages require an enormous burden. For this purpose, we have developed a tool allowing the automatic but partial indexation. It takes as input a service interface, like an IDL interface, and provides as output the index and the properties that describes the interface. This alleviates the manual indexation which is a complex task.

## III. Our approach of facilitator based on ontologies

Our approach for implementing the facilitator relies on ontologies. We have selected ontologies because they

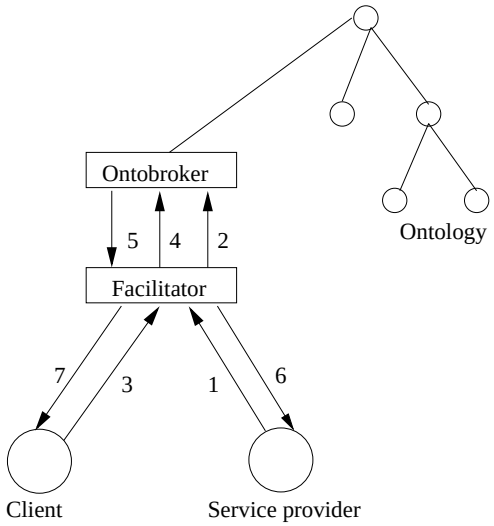


Figure 1. Interactions between the facilitator, Ontobroker, a client and a service provider

make it possible to provide a shared consensus that can help share/reuse [5] resources and this is highly valuable in the internet. We have selected Ontobroker [17] as our support engine for ontologies (see figure 1). It provides a complete management system to query, store, delete and modify information in the ontology and an inference engine to trigger various processing and events. We create an ontology describing the service properties defined in the metadata model that we have proposed.

We present the facilitator interfaces that we have developed to manage service ontologies. They allow clients/servers to query/add services through the ontoloy.

- *Importing function* : It is used by clients. It takes a query written in a logic language (e.g, F-Logic) [18] as the input, makes a request to Ontobroker for the service required, invokes the service and returns the result to the client. It allows client to get the result in an implicit way without direct interaction with the server.
- *Exporting function* : It is used by service providers. It allows a fact to be added to the ontology. A service offer is an instance of a service type. It is described by a fact. The fact includes the service properties and the service address (like WSDL file address with web services, IOR address with CORBA) which is necessary to access and communicate with the service.
- A function is provided to *remove* a fact from an ontology.

Finally, we declare an object which implements all these functions and we start it on a server. This object represents the facilitator. The clients and the servers use this object to contact Ontobroker and to add/retrieve services through the ontology.

The sequence of interactions between a client, a service provider and the facilitator to locate a service is illustrated in Figure 1:

1) The facilitator receives a service offer from a service provider. A service offer includes the service type, a network address (an address to be used when accessing the service) and a set of qualifying properties.
2) The facilitator sends the service offer to Ontobroker to store it in the ontologies.
3) The facilitator receives a service import request from the client. This request includes the type of service desired and a list of desired attributes. A service request is an expression of required service characteristics made by a client when a service is needed.
4) The facilitator sends the request to Ontobroker to search the required service offer.
5) Ontobroker returns to the facilitator the required service offers (if they exist) with their references after applying selection criteria.
6) The facilitator invokes a service selected by interacting with its provider.
7) The facilitator sends the result to the client.

## IV. Semantic description of service behavior

In this section, we present our approach for semantic description of service behavior. We build an ontology [9] that is constituted of concepts to index and store the properties of the services that we need to be accessed and shared. Service description (and indexing) is central and the underlying metadata model should be designed with care with the help of end users. So that services can be retrieved with adequate properties.

### *A. Behavior description*

The behavioral description of a service is critical since it provides the information enabling a client to use the service and to understand how its invocation needs to be performed. This includes, for example, the correct sequences in which the operations of the service need to be invoked. Once a client discovers a service by querying its static properties like its location, it can query its behavior as a function or a relation between inputs and outputs and which sequence of operations to be invoked to get an output from a given input.

SDL [7] and Interface Automata [8] have been designed to specify and to describe the functional behavior of telecommunication systems. They describe a process behavior by an automaton which can then be used to validate the specification. Therefore, based on SDL and Interface Automata, we describe the service behavior via a finite state machine (“an automaton”) that models the allowed operation sequences. Invocation of the service must satisfy these sequences. For example,

a seller may require a buyer to log in before ordering something. Thus, we can describe valid sequences of operation invocations which constitute a path to obtain a result. The interactions between the service operations occur through message exchanges. These messages are the inputs and the outputs of the operations. We can describe the automaton representing the service behavior as: a black box which carries a relation between its inputs and its outputs (external behavior) and as a white box which indicates the allowed interactions between the service operations to provide outputs from given inputs (internal behavior).

The automaton describing the service behavior is composed of a set of states and transitions between the states. Each service operation is represented by a state. For a current state, the successor state belongs to the set of its matching states: i.e., the output of the current state can be connected to the input of its next. Therefore, a transition connects a state to one of its next valid state.

### *B. inputs/outputs and post/pre-conditions description*

In a previous work [9], we have described the inputs/outputs of a service as strings (i.e., sequences of characters). Based on the experiences we made, this presented limitations to describe complex services which take for example, several inputs (having different types) in the same time to provide an output. In this work, we propose to extend this approach and to describe inputs/outputs by concepts in the ontology. So, we describe each input and output by a concept which is composed of a set of attributes. An output (described by a concept "O") of an operation can be connected to an input (described by a concept "I") of other one if the attributes of "O" are included and compatible with the attributes of "I". Figure 2 shows an example of an automaton that describes a service behavior: It verifies if a client (learner/teacher or researcher) has sufficient credit to buy a book or an article which he/she needs in his/her studying or research activities. This service provides two operations "BarnesGetPrice" and "CheckCredit". "BarnesGetPrice" takes as input the "ISBN" of the book and provides "price" as output to indicate the price of the book. "CheckCredit" takes as input a 2-uple ("price" and "Cust Id") the price and the identifier of the customer to check if the customer has sufficient credit to buy the book. So, we can describe the output of "BarnesGetPrice" by the following concept written in F-Logic [18] :

*PRICE [ price =>> Integer].*

and the input of "CheckCredit" by the following concept :

*PRICE_CUSTOMERID [ price =>> Integer;*
*Customer_id =>> Integer].*

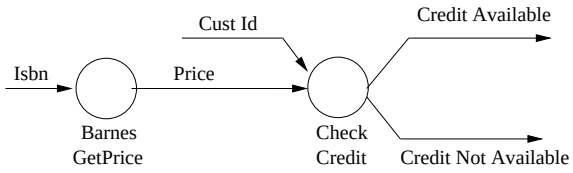


Figure 2. An example of service behavior

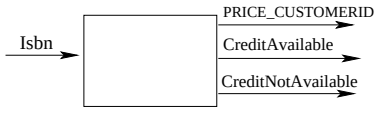


Figure 3. The service description as a black box

"PRICE" can be connected to "PRICE_CUSTOMERID" because its attribute(s) ("price") is included and compatible with attribute(s) ("price") of "PRICE_CUSTOMERID". So, "BarnesGetPrice" has "CheckCredit" as a next valid state because they can be connected by the output/input : "PRICE"/"PRICE_CUSTOMERID". Therefore, a user can invoke "BarnesGetPrice" has the price as result. It takes this as input and it can invoke "CheckCredit" by indicating its identifier. We can define two concepts "CREDITAVAILABLE" and "CREDITNOTAVAILABLE" to describe the outputs of "CheckCredit".

Thus, we can describe this service as a black box (Figure 3) which carries a relation between its inputs and its outputs by the following concept :

*Black_box1 [*
*Input =>> ISBN;*
*Output1=>>PRICE_CUSTOMER; (or PRICE)*
*Output2 =>> CREDITAVAILABLE;*
*Output3 =>> CREDITNOTAVAILABLE;*
*].*

Our principal investigation in this area compared for example to WSMO [10] is to describe a service as a white box. It allows client to discover all the sequences of operations enabling to obtain an output from a given input and he/she can select the appropriate sequence (maybe the shortest sequence, the sequence having the cheaper cost and so on.). For this, we can describe each operation by a concept that includes the name of the operation and its inputs/outputs. So, the client can construct the sequences of operations allowing to get an output from a given input by comparing their inputs/outputs and by verifying their compatibility. We describe the service as a white box by the following concepts:

*White_box1 [*
*Operation1 =>> BARNESGETPRICE;*
*Operation2 =>> CHECKCREDIT*

*].*

*BARNESGETPRICE[*
*Name =>> STRING "BarnesGetPrice";*
*Input =>> ISBN;*
*Output=>> PRICE ].*

*CHECKCREDIT[*
*Name =>> STRING "CheckCredit";*
*Input=>> PRICE_CUSTOMER;*
*Output1=>>CREDITAVAILABLE;*
*Output2=>>CREDITNOTAVAILABLE*
*].*

Moreover, we can define post and pre-conditions to add a semantic level when using the service [10]. They are conditions on inputs/outputs and service parameters. This takes a closer look at what the service offers to a client (post-condition), when some conditions are valid (pre-conditions). A pre-condition holds the pre-requisite so that the service call can be made whereas the post-condition describes its effect. By comparing the pre and post conditions, the user can obtain valuable information on the effects of a service and thus on its behavior.

The post/pre-conditions can be expressed by axioms in the ontology. For example, when the service execution is successful, the credit card intended to be used for paying is a valid one (post-condition) if the Isbn of the requested book is positive (an example). The post and pre-conditions can be expressed by axioms which are logical expressions in the ontology, for example (by using F-Logic) :

- pre-condition : *x[hasIsbn>0]* for each instance x of the service, which guarantees a form of coherence. "hasIsbn" is a parameter of the service.
- post-condition : *x[hascreditcard->>TRUE]* for each instance x of the service, which "hascreditcard" is a parameter of the service.

## V. Description of a Service Interface

As each service has an interface, we define a concept that describes the service interface. This concept is considered as an attribute of a service. It contains the descriptions of all the operations, the exceptions, the attributes, the type definitions and the constant definitions of the service. As the number of these latter attributes varies, we define each of them with a list. For example, each element of the operations list contains a list of parameters, specific attributes (type of return, number of parameters, and so on) and a reference to the next element of the list (recursive). Thus, the Operation concept is composed of the following attributes:

*Operation[ name =>> STRING;*
*nbr_parameters =>> INTEGER;*
*parameter =>> Parameter;*
*return_type =>> STRING;*
*operation =>> Operation ].*

For example, we can describe the operation "BarnesGetPrice" as an instance of the above concept (in F-Logic) :

BarnesGetPrice:Operation[
name ->> "BarnesGetPrice",
nbr_parameters ->> 1,
parameter ->> Param,
return_type ->> "Integer",
operation ->> CheckCredit ].

"Param" is an instance of the below concept "Parameter" which describes the list of "BarnesGetPrice" parameters and "CheckCredit" is the next operation of "BarnesGetPrice".

Each element of the parameters list includes a name, a type (integer, string, etc), a position and a mode that indicates the direction in which the parameter is being passed during a dynamic invocation (input 'in', output 'out', input/output 'inout') and a reference to the next element of the list.

*Parameter[ name =>> STRING;*
*type =>> STRING;*
*position =>> INTEGER;*
*mode =>> STRING;*
*parameter =>> Parameter ].*

For example, we can describe the parameters "BarnesGetPrice" as an instance ("Param") of the concept "Parameter":

Param:Parameter [ name ->> "Isbn",
type->> "String",
position ->> "First parameter",
mode ->> "in",
parameter ->> Null ].

As the operation takes only one parameter, we have indicated "Null" as next parameter.

This interface description is very similar to the interface repository of CORBA [11] and WSDL [19] of web services. It is used to invoke dynamically a service, for example by using the DII (Dynamic Invocation Interface) of CORBA or of web services. This interface allows clients to build dynamically operation requests for any interface that has been stored in the interface repository and invoke any operation of this interface at run-time. By using this interface, clients are not restricted to use the services that were defined at the time the client was compiled. The interface description that we have defined allows clients to get the necessary information to invoke dynamically a service.

## VI. Automatic composition of services

The composition of services is critical since it provides novel services and functionalities to the client. By

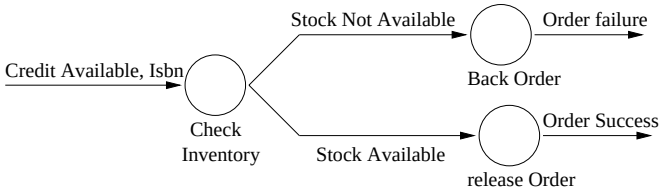


Figure 4. Other example of service behavior

combining services, the client can obtain a desired output from a given input not provided yet by any service but by several combined services. In our approach, we consider that two services can be composed if an output of one is equal to an input of the other one. Since we have described the inputs/outputs by concepts in the ontologies, we have to compare the attributes of these concepts and check their compatibilities. In this case, two services can be connected if the attributes of a concept output of one include and compatible (by their names and types) with the attributes of the concept input of the other one. For this, we extend the functionalities of the facilitator to be a composition engine. When a client queries a service from the facilitator and if the service doesn't exist, the facilitator tries to combine and connect two or more existing services (based on their inputs/outputs) to compose a new service that fulfills user's expectations.

To illustrate our approach, we provide an example. Figure 4 describes the behavior of a service that takes as input the "Credit Available" of a client and the "Isbn" of the book to do a buying order. This service contains 3 operations : "CheckInventory", "BackOrder" and "releaseOrder". For example, "CheckInventory" takes as input "CreditAvailable" of the user and the "Isbn" of the book and provides as output "StockNotAvailable" or "StockAvailable". So, we can describe the input of this operation by the following concept in the ontology:

*Credit_Isbn[*
*CreditAvailable =>> CREDITAVAILABLE;*
*Isbn =>> ISBN*
*].*

We can describe the inputs/outputs of the other 2 operations by other concepts in the ontology.

As a white box, we describe the service by the following concept :

*white_box2[*
*Operation1 =>>CHECKINVENTORY;*
*Operation2 =>> BACKORDER;*
*Opertion3 =>> RELEASEORDER*
*].*

For example, the concept "CHECKINVENTORY" can be composed of the following attributes :

*CHECKINVENTORY[*
*Name =>> STRING "CheckInventory";*
*Input =>> Credit_Isbn;*
*Output1 =>> STOCKNOTAVAILBLE;*
*Output2 =>>STOCKAVAILABLE*
*].*

As a black box, the service can be described by the following concepts:

*Black_box2 [*
*Input =>> Credit_Isbn;*
*Output1=>>STOCKNOTAVAILABLE;*
*Output2 =>> STOCKAVAILABLE;*
*Output3 =>> Order_failure;*
*Output4 =>> Order_success*
*].*

"STOCKNOTAVAILABLE", "STOCKAVAILABLE', "Order_failure" and "Order_success" are the concepts that describe the inputs/outputs of the operations "BackOrder" and "releaseOrder".

If a client queries a service type (as a black_box) from the facilitator that takes as input an "Isbn" of a book and as output "OrderSuccess" by the following query written in F-Logic and this service type doesn't exist :

*Forall x <- x[input=>>ISBN ] and x[output=>>OrderSuccess].*

This query allows to find all the black boxes ("*values of x*") that takes as input "ISBN" and output "OrderSuccess". If no offers exists, the facilitator tries to combine and compose existing service types to satisfy the client's request. It can query all the service type having "ISBN' as input by the following query :

*Forall x <- x[input =>> ISBN].*

It takes the outputs of the returned services and tries to compare them with the output of other services. It can obtain the service represented by Figure 2 with "CREDITAVAILABLE" as output. It takes this output and tries to compare it with the concepts inputs of other services, it means to compare their attributes and their compatibility. It takes the attribute (Credit Available "Figure 2") of the concept "CREDITAVAILABLE" and compare it with the attributes ("CreditAvailable" and "Isbn") of the concept "Credit_Isbn" of the service described by Figure 4. Therefore, the output "Credit Available" of the first service is compatible and is included in the output of the other service. Thus, the facilitator can compose them. The second service provides "OrderSuccess" as output. When the facilitator compose the two services, it can satisfy the client's request. It invokes the first service by getting "ISBN" as input. It takes the output "Credit Available" as input and invokes the second service. Finally, it returns the output "OrderSuccess" to the client. Therefore, the

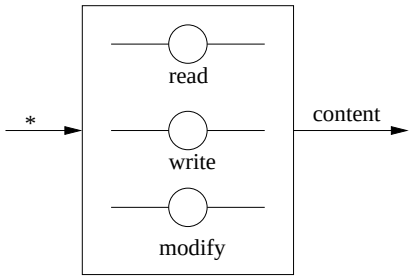


Figure 5. An automaton describing the document behavior

composition becomes automatic and implicit to the client which is one of our initial aims.

## VII. Application

We present an application we have developed using our approach to discover and describe documents. It allows documents to be reused and shared by using the approaches defined for web services. A client can discover a document which is downloadable on the client machine with no interaction with its provider. A document doesn't have an interface. This approach is used in the semantic web domain. Our idea is to direct and view a document as a service, i.e., a document accessible via an interface. This allows clients to use the same type of interface to interact with different documents (web documents, databases, etc.). In this context, we can add an interface to the document that allows clients to read, write and modify a document.

We describe a document with our proposed metadata (Section 2) as :

- A learning resource: title, version, etc. We can use the metadata proposed by Dublin Core [1] to describe documents.
- A service helping researchers : it includes the description of the research area of the document, the description of its author, and so on.
- A service in general: it includes delivery and request channels, payment and pricing and so on.
- By behavior: we describe the inputs/outputs of a document. We consider that a document doesn't have an input. A client doesn't need inputs to interact with a document. The output of a document can be its content. Figure 5 describes the document behavior with an automaton. The three operations (read, write, modify) don't have inputs ("*") and can provide as outputs the content of the document after reading, writing or modifying.

## VIII. Related Work

Several different approaches for describing and discovering services have been developed. For example, DAML-S [20] provides a set of characteristics that can be used as index or properties of a service. This description addresses only the static properties of a service. It does not describe the behavior of a service and its interface. WSMO [10] describes the service behavior by inputs/outputs, post and pre-conditions. It uses ontologies to describe the service behavior. It doesn't indicate how clients can query services based on their inputs/outputs and how she/he can select the appropriate sequence of operations allowing to get an output from a given input (white box). Our approach is innovative since it is the first one to describe the service as a white box and to enable clients to select the appropriate sequence of operations (the sequence having the cheaper cost, the shortest one, or any other comparison criterion).

To discover a service, we can select for example UDDI [3] which, is a registry that allows a Web service to be discovered via a yellow page style of search. It allows a service to be discovered by querying only its static properties. It provides a static schema for service description and the service provider cannot modify this schema or create databases to advertise its services offers. Then, in our approach of facilitator, each service provider can define its databases and can advertise its service offers by using their static properties, their behaviors and their interfaces. The clients use the same interface to query these three levels of service description. The service indexing and discovery become more sophisticated. Our facilitator allows clients to get the result of the service in an implicit way and to not interact with the service provider. It is compatible with UDDI and CORBA as we have demonstrated in [9] and [12].

Many different approaches for service composition have been developed like [21], [22], [23] but the client must invoke all the service components of the composed service. In this context, the client must be familiar with the service composition. Our approach for service composition using facilitator is innovative since it allows services to be composed automatically in an implicit way to the clients. It can be used by clients that are not familiar with the service composition and aren't interested by the intermediate steps of the service composition process. The description of the inputs/outputs of a service by concepts in our approach made the composition more powerful and flexible by comparing the attributes of the inputs/outputs, evaluating their compatibility and the possibility to compose and connect them. Our approach of semi-dynamic composition model is the first one that exceeds the performance and complexity problem of the dynamic composition model [12].

## IX. Perspectives and Conclusions

In this paper, we have presented our current related work in the indexing and searching of learning and research services. For this latter operation, we have proposed a metadata model for describing learning services. The current model that we have proposed, describes a learning service with three dimensions: as a learning resource, as a service that contribute and help

researchers and as a general service. This model is a powerful one since it provides a complete description of learning services and extends services to be used by researchers/teachers and not only by teachers. It is flexible because it ensures compatibility with existing approaches like IEEE LOM, Dublin Core and DAML-S. The clients and the servers of this metadata model can be teachers, researchers and students. We have implemented this model with ontologies, consisting of concepts representing the characteristics of a service. Flexibility is introduced since the ontology and its content can be changed and adapted to usage.

In this paper, we have presented a facilitator we designed and developed : it is based on ontologies and knowledge representation. It allows a service provider to advertise a service offer and a client to discover a service by querying its static and dynamic (behavior) properties in distributed systems. Flexibility is introduced since the ontology and its content can be changed and adapted to usage. When a client discovers a service offer by querying its static properties, it can obtain useful information about the service behavior enabling it to use the service discovered and to understand how the invocation of the service operations needs to be processed.

Based on SDL, Interface Automata and WSMO, we can describe the service behavior which enables the client to use the service discovered and to understand how the invocation of the service operations needs to be performed. We have described the inputs/outputs by concepts in the ontologies and we have proposed to describe a service : as a black box which carries a relation between its inputs and its outputs (external behavior) and as a white box which indicates the allowed interactions between the service operations to provide outputs from given inputs (internal behavior). Our proposal is the first one as far as we know that allows describing a service as a white box. We have described the service interface (operations, parameters, exceptions and so on) that provides necessary information allowing clients to invoke dynamically the service operations by using for example the DII (Dynamic Invocation Interface) of CORBA or of web services.

Based on the service behavior description and facilitators, we have presented in this paper an approach that allows the composition of services in distributed systems. This approach is a powerful one since it allows the client to benefit from the functionalities of more than one service to get novel functionalities and novel services. It is an innovative one since it allows facilitator to compose services in an implicit way to the clients. It is interesting because the client may not be familiar with the service composition (like elderly people, disabled people, novice, etc.) or if services are becoming more complex even an experienced client will have problems in composing services.

As perspectives, we will propose an approach of facilitator federation to compose services. In this context, when a client queries a service from a facilitator and this service doesn't exist, the trader can interact with other traders to search and compose services to satisfy the client's request. So, this approach will allow the composition of services located on different facilitators.

## REFERENCES

[1] A. Powell. Dublin Core in RDF. Technical Report, 1998.
[2] IEEE. Draft standard for learning object metadata. In *http://Itsc.ieee.org/wg12/*. Technical Report, 2002.
[3] *www.uddi.org*.
[4] O. K. Zein and Y. Kermarrec. An Elaborate and Flexible Trader Based on Ontologies. In *IFIP WG6.7 Woskshop and EUNICE Summer School on Adaptable Networks and Teleservices*, pages 103–108, Trondheim, Norway, September 2002.
[5] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledege Acquisition, 5 : 199-220*, 1993.
[6] Kaleidoscope: a European network of excellence. http://www.noe-kaleidoscope.org.
[7] ITU-T. ITU-T Recommendation Z.100. Specification and Description Language (SDL). 2002.
[8] L. De Alfaro and T. A. Henzinger. Interface Automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*. ACM Press, 2001.
[9] O. Kassem Zein and Y. Kermarrec. An approach for describing, discovering services and for adapting them to the needs of users in distributed systems. In K Sycarra and T Payne, editors, *In the proceedings of AAAI Spring Symposium on Semantic Web Services*, Stanford, California, March 2004.
[10] *http://www.wsmo.org/*.
[11] Object Management Group. *The Common Object Request Broker : Architecture and Specification, 2.0 edition*, July 1995.
[12] O. K. Zein and Yvon Kermarrec. Static, semi-dynamic and dynamic composition of services in distributed systems. In *IEEE International Conference on Internet and Web Applications and Services (ICIW'06)*, Guadeloupe (French Caribbean), 17-25 February 2006.
[13] M. Dumas, J. O'Sullivan, M. Heravizadeh, D. Edmond, and A. Hofstede. Towards a semantic framework for service description. *In Proceedings of the 9th International Conference on Database Semantics, Hong-Kong*, April 2001.
[14] M. Merz, M. Witthaut, and S. McConnel. Catalogue and Service Architecture. http://osm-www.informatik.uni-hamburg.de/osm-www/public/docs. *OSM D8*, 1997.
[15] W. Ng, G. Yan, and E. Lim. Heterogeneous product description in electronic commerce. *ACM SIGeCom Exchanges, 1(1):7-13*, 2000.
[16] *http://www.daml.org/services/*.
[17] D. Fensel, S. Decker, M. Erdmann, and R. Stude. Ontobroker : The very high idea. *In Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida, USA*, May 1998.
[18] M. Kifer, G. Lausan, and J. Wu. Logical foundations of object-oriented and frame-based language. *Journal of the ACM, 42(4) : 741 - 843*, 1995.
[19] web service definition language. http://www.w3.org/tr/wsdl.
[20] *http://www.daml.org/services/*.

[21] J. Yang, M. P. Papazoglou, and W. den Heuvel. Tackling the Challenges of Service Composition in E-Marketplaces. In *The 12th International Workshop on Research Issues in Data Engineering : Engineering e-Commerce/e-Business Systems*. RIDE 2002.
[22] M. Pistore et al. Planning and monitoring web service composition. In *AIMS*, 2004.
[23] D. Berardi et al. Automatic service composition based on behavioral description. In *IJCIS*, 2005.

## BIOGRAPHY

**Oussama Kassem Zein** received the PhD degree in computer science form ENST Bretagne, France, in 2005. He is in a postdoctoral position at ENST Bretagne. his research interests include web services, distributed systems, knowledge representation, e-learning.

**Yvon Kermarrec** received the PhD degree in computer science form Rennes 1 University, France, in 1988. He is a professor at ENST Bretagne. His research interests include distributed systems, software engineering, web services, parallel architectures, e-learning.