Specifying Banking Transactions using Web Services Modeling Language (WSML)

Omid Sharifi  
Department of Computer Engineering  
Eastern Mediterranean University  
Famagusta, Cyprus  
omid.sharifi@emu.edu.tr

Zeki Bayram  
Department of Computer Engineering  
Eastern Mediterranean University  
Famagusta, Cyprus  
zeki.bayram@emu.edu.tr

ABSTRACT
We present a semantic web service specification of banking transactions so that client software can declare the functionality it desires of the web service without actually knowing which web service will be able to answer its request. The semantic web service infrastructure can then determine the correct web service to call, and make the call, taking care of details about the interaction between the client and server. Our approach represents a new genre of use case scenarios for semantic web services, since it envisions using them for specifying web services within an organization, rather than assuming the existence of numerous web services provided by different organizations, which would make the job of discovery an almost insurmountable task.

Keywords
Semantic web services, Ontology, E-banking, WSML, WSMO

1. INTRODUCTION
Web services are computational units on the World Wide Web (WWW), and can be called through standard interfaces and protocols, such as HTTP [2] and SOAP [1]. They represent a paradigm shift in Computer Science, where abstraction from hardware to software has been replaced by abstraction from software to service-ware in terms of Service Oriented Computing [9]. Web services allow an organization to open up part of its internal computing infrastructure to the outside world in a controlled way. External software can then use the functionality provided by the web service and interact with the organization’s computing infrastructure for very specific tasks. There are several drawbacks to “normal” web services though. Their specification is purely syntactic, so automatic discovery is unrealistic, and they need to be invoked manually. Any semantic specification is done informally, using natural language, which does not lend itself to reliable machine understanding. Similarly, orchestrating several web services so that they work in concert to achieve a goal needs to be done manually as well.

Semantic Web services attempt to remedy the drawbacks of regular web services by providing rich, machine interpretable semantic descriptions of Web services [10] based on logic which give a formal specification of their functionality and behaviour so that the whole process of web service discovery, orchestration or composition, and execution can be automated through appropriate semantic web service frameworks. Discovery aims to support the autonomous identification of appropriate services by software agents, to provide the satisfaction of their goals[17]. SWS description elements such as preconditions, effects, assumptions, and postconditions provide the means for carrying out the discovery activity.

Web Service Modelling Ontology (WSMO) [5] is a framework for semantic description of Semantic Web Services based on the Web Service Modelling Framework (WSMF) [8]. Its four main components are ontologies, web services, goals and mediators. Web Service Modelling Language (WSML) [11] is a language for modeling web services, ontologies, and related aspects of WSMO framework, to provide the description of semantic web services so that automatic discovery and invocation becomes possible. Five language variants of WSML exist based on Description Logic and Logic Programming. Each language variant provides different levels of logical expressiveness [11]. The variants are: WSML-Core, WSML-DL, WSML-Flight, WSML-Rule and WSML-Full.

The notion of an ontology is key to the semantic web[12]. It is an explicit formal shared conceptualization of a domain of discourse is an ontology, in which it facilitates semantic interoperability the main concepts and relations that a community shares over a particular domain [7][13]. Basically, an ontology acts like a dictionary, defining the common terminology in some domain. Ontologies form a very significant foundation of the semantic web on which other components are built [4]. One of the major components of an ontology is the “concept.” Concepts are used to establish the basic elements of the agreed terminology for a problem domain. From a high-level perspective, a concept is described by a concept definition and provides attributes with names and types [15]. A “concept” corresponds pretty much to the “class” construct in object-oriented programming languages.

WSML has an ontology component that acts like an intelligent, object-oriented database system that the other components of the framework utilize for “common understanding” of the data and terminology involved in the web service discovery and invocation process [19]. WSMO-based discovery engines make extensive use of ontologies as well [14][7][16][6][18].
Another major SWS framework is OWL-S[3]. In our work we have used WSMO and WSML since it is the more developed of two.

In this paper, we use WSML to semantically specify a sample banking transaction, namely initiating a transfer of money from one account to some other account. We first lay the groundwork for the semantic specification by defining the ontologies for representing domain knowledge, such as accounts and customers, as well as domains needed for information exchange between the client and the service.

The reminder of the paper is organized as follows. Section 2 depicts a portion of E-banking ontology in WSML. This ontology contains concepts, instances, and relations of the E-banking domain. Sections 3 contains the semantic web service specification for the “money transfer” transaction. One possible goal for requesting a “money transfer” transaction is given in 4. Finally, section 5 is the conclusion and future work.

2. E-BANKING ONTOLOGY

In this section we give the concept definition of the specification. Note that some concepts that are not made use of in the “money-transfer” use case are also presented in order to give a more complete picture.

2.1 E-banking Ontology concepts

This section describes some portion of the E-banking concepts that are used in defining the web services and goals. We shall elaborate on some of the essential concepts the following subsections.

2.1.1 “Customer” concept

The “Customer” concept includes attributes “customerId,” “customerName,” “customerStreet” and “customerCity.” Also in this concept, two attributes “gender” and “marriedTo” to represent the one-to-one relationship in between two customer objects, in case such a relationship between customers exists. Figure 1 illustrates customer concept. Note that the “marriedTo” attribute was defined as symmetric, since a person being married to another person implies that the second person is married to the first.

2.1.2 “Account” concepts

In Account concept two sub concepts are presented: “saving-account” and “checking-account” with common attributes “account-number” and “balance.” Figure 2 depicts the definition of these concepts in WSMO. The concepts “SavingAccount” and “CheckingAccount” inherit from the base concept “Account.” The “SavingAccount” concept has the attribute “interestRate” and the “CheckingAccount” concept has the attribute “overdraftAmount.”

2.1.3 “Loan” concept

The Loan concept has five attributes. Figure 3 shows loan concept that is a concept of banking ontology. However, for the total participation “loan-payment” relationship, we need an axiom which enforces the constraint that a payment object cannot exist unless it is related to a loan object. This constraint is given as an axiom in figure 3.

2.1.4 “RequestLoan” concept

“RequestLoan” concept in the ontology has attributes “requestLoanAmount” and “customerId,” which links loan objects to customer. The “RequestLoan” concept is depicted in figure 4.

2.1.5 “LoanAcknowledgment” concept

”LoanAcknowledgment” concept in the ontology has attributes “loanNumber”, “amount”, and “inBranch” which denotes assigned loan to a customer. The “RequestLoan” concept is depicted in figure 5.

2.1.6 “RequestLoanPayment” concept

“RequestLoanPayment” concept in the ontology has attributes “customerId”, “paymentAmount”, and “accountNumber” which is enabled to have payment for a loan from a customer account, as well the attribute “forLoan” which links “RequestLoanPayment” objects to loans. The “RequestLoanPayment” concept is depicted in figure 6.

2.1.7 “LoanPaymentAcknowledgment” concept

”LoanPaymentAcknowledgment” concept in the ontology has five attributes. This concept supports objects which are used in order to update balance of customers after a loan payment from customer accounts. The “customerId”, “pay-
2.1.8 “TransferRequest” concept

“TransferRequest” concept is considered in the ontology to transfer money from a sender account to another account. It has attributes “customerId”, “senderAccountNumber”, “receiverAccountNumber”, and “transferAmount” to enable sending money from a customer account to the receiver account. The “TransferRequest” concept is depicted in figure 8.

2.1.9 “TransferAcknowledgment” concept

“TransferAcknowledgment” concept in the ontology has five attributes. This concept is considered to support objects which are used in successful transfer operation. The “balance” attribute is updated after transfer operation. The “LoanPaymentAcknowledgment” concept is depicted in figure 9.

2.2 E-banking relations

A relation can be defined between concepts in the ontology. Membership in a given relation can be specified logically, through the definition if axioms, which basically are logical expressions with a name. In WSML, axioms preceded by “!-“ are constraints, meaning that the logical expression that follows “!-“ must never be true, otherwise an error condition is reported.

Usually, relationships can be mapped to attributes in concepts. When a relationship has attributes, however, and is a

manypo-to-many relationship, then we need a relation on the ontology side as well. This is the case with the “depositor” relationship, which has an “accessDate” attribute.

Also, to provide flexibility in dealing with relations with attributes, it is good practice to be able to access individual members of the relation as objects. This can be done by providing a concept for the relationship, and a mapping between the concept and the relation.

2.2.1 E-banking relation “depositor”

Figure 10 illustrates the “depositor” relation, which denotes a many-to-many relationship set between “Customer” and “Account,” with “Calendar” being the attribute of the relationship.

2.2.2 E-banking relation “borrower”

The “borrower” is a one-to one relationship from a customer to a loan to determine a assigned loan to a customer. Figure 11 denotes the corresponding relation that specifies every loan to be related with a specific customer(no loan can exist without a related customer).

2.2.3 E-banking relation “accountBranch”

The “accountBranch” is a one-to many relationship from account to branch to determine a branch accounts. Figure 12 denotes the relation that specifies every branch to be related with specific customer accounts(a customer can have several accounts in a branch).
3. E-BANKING WEB SERVICE: “SERVICE-TRANSFERMONEY”

In this part We have specified the last web service of banks to transfer money from a customer account to another customer account. Figure 13 depicts the WSMO specification of transfer money web service from a customer account. It includes the variant of WSML used (in this case WSML- Rule), namespace, name of the web service (“ServiceTransferMoney”), its non-functional properties, as well as imported ontologies. The service includes the precondition for transfer money. The precondition requires the existence of an instance of the “requestTransfer” concept in the ontology. This instance contains all the necessary information about transfer money. Where information such as customer ID, sender account number, receiver account number, and transfer amount are controlled to support liability about customer transactions. The web service includes the assumption for the transfer money web service. It states that a customer must have an account at the bank whose customer instance is the same as the one in the request, and who has depositor at the time for which the transfer money is applied. Also it checks the sender balance for which it should be greater than or equal to transfer amount. The postcondition states the existence of an instance of the money transfered “transferAcknowledgment” concept in the ontology after the web service has finished its execution. This instance contains information about the customer ID, sender account number, receiver account number, transfer amount, and sender balance which transfer was executed. The web service depictions the definition of the effect element in the web service specification where, as a consequence of the execution of the web service, the sender and receiver balances are updated after transactions.

4. E-BANKING GOAL: “FINDTRANSFERMONEY”

The last goal is money transfer request. Figure 14 depicts the complete definition of the money transfer goal where in precondition ?requestTransfer variable denotes “?accountNumberFrom”, “?accountNumberTo”, and some other attributes in which that is member of “TransferRequest” concept. The “?accountNumberFrom” and “?accountNumberTo” specify two accounts to transfer money from a sender account to a receiver account. In postcondition ?transferAcknowledgment variable denotes a customer balance and some other attributes that were considered to refresh sender balance after the transaction has finished in which it belongs to “TransferAcknowledgment” concept.

5. CONCLUSION AND FUTURE WORK

We presented a novel application of semantic web services to the area of banking. We showed how a web service for making money transfers between two accounts can be semantically described so that the job of discovery and invocation can be automated. In the future we are planning to extend our work to include all kinds of banking transactions, and test our system in a real life on-line banking environment.

6. REFERENCES

webService ServiceTransferMoney
importsOntology "$http://cmpe.emu.edu.tr/omid/services#"
capability ServiceTransferMoneyCapability
   nonFunctionalProperties
discovery#discoveryStrategy hasValue discovery#HeavyweightDiscoveryendNonFunctionalProperties
sharedVariables {?balance ,?transferAmount ,?accountNumberTo ,?accountNumberFrom ,
?senderBalance ,?newSenderBalance ,?receiverBalance ,?customerId}
precondition
   nonFunctionalProperties
dc#description hasValue "request transfer money from customer A account to customer B account"
edefinedBy
   ?requestTransfer [omid#customerId hasValue ?customerId ,
   omid#senderAccountNumber hasValue ?accountNumberFrom ,
   omid#receiverAccountNumber hasValue ?accountNumberTo ,
   omid#transferAmount hasValue ?transferAmount ] memberOf omid#TransferRequest.
assumption
   nonFunctionalProperties
dc#description hasValue "The balance of customer A should be more than the transfer amount"
edefinedBy
   ?customer [omid#customerId hasValue ?customerId]
   and depositor(?customer , ?accountFrom , ?date) and
   ?accountFrom [omid#AccountNumber hasValue ?accountNumberFrom ,
   omid#balance hasValue ?senderBalance ] memberOf omid#Account and
   ?accountTo [omid#AccountNumber hasValue ?accountNumberTo ,
   omid#balance hasValue ?receiverBalance ] memberOf omid#Account and
   wsml#greaterEqual(?senderBalance ,?transferAmount ).
postcondition
   nonFunctionalProperties
dc#description hasValue "transfer acknowledgment"
edefinedBy
   ?transferAcknowledgment [omid#customerId hasValue ?customerId ,
   omid#senderAccountNumber hasValue ?accountNumberFrom ,
   omid#receiverAccountNumber hasValue ?accountNumberTo ,
   omid#transferAmount hasValue ?transferAmount ,
   omid#balance hasValue ?newSenderBalance ] memberOf omid#TransferAcknowledgment.
effect
   nonFunctionalProperties
dc#description hasValue "update receiver balance"
edefinedBy
   wsml#numericSubtract(?newSenderBalance ,?senderBalance ,?transferAmount ) and
   wsml#numericAdd(?newReceiverBalance ,?receiverBalance ,?transferAmount ) and
   ?accountFrom [omid#AccountNumber hasValue ?accountNumberFrom ,
   omid#balance hasValue ?newSenderBalance ] memberOf omid#Account and
   ?accountTo [omid#AccountNumber hasValue ?accountNumberTo ,
   omid#balance hasValue ?newReceiverBalance ] memberOf omid#Account.

Figure 13: WSMO specification of ServiceTransferMoney web service
<pre>wsmlVariant "http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace {
  "http://cmpe.emu.edu.tr/omid/goals#",
  "http://cmpe.emu.edu.tr/omid#",
  "http://purl.org/dc/elements/1.1#",
  "http://www.wsmo.org/wsml-syntax#",
}
goal FindTransferMoney
importsOntology
  "http://cmpe.emu.edu.tr/omid#Banking-Ontology"
capability FindTransferMoneyCapability
  nonFunctionalProperties
discovery#discoveryStrategy hasValue { discovery#HeavyweightDiscovery , discovery#NoPreFilter }
endNonFunctionalProperties
sharedVariables {?customerId ,?accountNumberFrom ,?accountNumberTo ,?transferAmount}
precondition
  nonFunctionalProperties
dc #description hasValue "request transfer money from customer A account to customer B account"
endNonFunctionalProperties
definedBy
  ?requestTransfer [omid#customerId hasValue ?customerId ,
  omid#senderAccountNumber hasValue ?accountNumberFrom ,
  omid#receiverAccountNumber hasValue ?accountNumberTo ,
  omid#transferAmount hasValue ?transferAmount ] memberOf omid#TransferRequest.
postcondition
  nonFunctionalProperties
dc #description hasValue "transfer acknowledgment"
endNonFunctionalProperties
definedBy
  ?transferAcknowledgment [omid#customerId hasValue ?customerId ,
  omid#senderAccountNumber hasValue ?accountNumberFrom ,
  omid#receiverAccountNumber hasValue ?accountNumberTo ,
  omid#transferAmount hasValue ?transferAmount ,
  omid#balance hasValue ?newSenderBalance ] memberOf omid#TransferAcknowledgment.

Figure 14: WSMO specification of FindTransferMoney goal

</pre>