Systematic definition and assent to eContracts for Web Services

David Portabella Clotet, Vincenzo Pallotta, Martin Rajman
EFPL, {david.portabella, vincenzo.pallotta, martin.rajman}@epfl.ch

Abstract

People are increasingly using provider services through the Internet. While a web site provides information about the contract terms and conditions that the clients have to assent to in order to use its services, in web services there is no such way for taking legal issues into account. There are some attempts to build machine readable eContract languages that can be used to express the contractual terms between the participants but they are mainly designed to govern the distribution and use of electronic content. We propose an architecture for the definition of and assent to eContracts for Web Services.

1. Introduction

We delegate increasingly sophisticated and complex tasks to computer programs, and we require them to act autonomously on our behalf in distributed environments (e.g. the web, the grid). Personal Assistants (PA) are computer programs that can be thought of a very natural metaphor for such systems. Web Services are self-contained, modular applications that can be semantically described and published over the Web by service providers. OWL-S is a Semantic Web Services description language that enriches Web Service descriptions with semantic information from OWL ontologies and the Semantic Web [OWLS]. This makes it possible for a PA to look for a particular type of service and invoke it "on the fly" in order to perform a task.

However, there is very little work on the automated use of a Web Service that requires assent to a contract (the manifestation by one party that accepts to be bound by the contract). We propose an extension of the OWL-S specification to include a *Master Contract* description that can be used by the client agent to assent to the contract and get access to the contracted service.

2. Contract Instantiation

When a party offers a good or service that another party is looking for, they need to negotiate to reach an agreement on the contract's terms. In the process of negotiation, their constraints and preferences are discussed and may need to be changed (typically relaxed) in order to successfully reach an agreement.

We are interested in the case where one party is a service provider offering a type of service to individual clients (e.g. a car insurance or booking a flight). In this situation, the terms of the service's contract are fully defined a-priori and the client is forced to accept all of the conditions in order to get the service, or reject them and do without. The client must also communicate a set of service parameters that the provider may specify (e.g. identity or departure date).

Contracts are the means by which we recognize and enforce agreements between parties. We define a Contract Instantiation (CI) as the information that should be available in a possible contract in order to reach an agreement for a service instantiation. A CI can be accepted by both parties (e.g. by signing it) and then the service can be executed. We define a Contract Instantiation Process (CI process) as the process of defining a CI. It typically requires a communication between both parties (e.g. a clerk and a consumer).

A provider that needs to run the CI process frequently (e.g. for a lot of customers) would eventually be interested in automating it[1]. The client typically needs to make the effort of dealing with the provider's CI process, such as filling in a Web Form or interacting with a Dialogue System through the phone. An exception to this practice is when the providers are really motivated in

[1] It might not seem cost-beneficial for a provider to automate the CI process if it needs to be run once or very few times. In contrast, once a CI has been agreed for a periodic service, it does make sense to automate its execution (e.g. sending the weather forecast every day by email).

selling a service to a particular class of customers so that they would make the effort to deal with the customers' CI process. This is typically the case for government institutions that make a public call for an expensive service, where the roles of provider and client are actually swapped.

While does not make sense for a provider to automate the CI process for a single, possibly infrequent, instantiation, from the customer perspective we think that it can be helpful to rely on a general purpose Personal Assistant (PA) that can help the user to deal with the CI process automation in a fairly high number of different service types. For instance, if a customer needs to book a flight, he/she can specify the trip requirements and leave the PA to look for various providers offering flight-booking services and automate the CI processes on behalf of the user.

The goal of this paper is to study the different options for the design of this type of Personal Assistant.

2.1 Information of a CI

In the spirit of keeping things simple, we identify three types of information that can be part of a service contract: obligations, rights and facts.

1. An obligation, for either party, is a requirement that compels the party to follow or avoid a particular course of action[2] (e.g. the customer is not allowed to cancel the flight or the provider is not allowed to overbook).

2. A right is the entitlement to do or refrain from doing something, or to obtain or refrain from obtaining an action, thing or recognition in civil society. The right can therefore be a faculty of doing something, of omitting or refusing to do something, or of claiming something (e.g. the customer is allowed to smoke during the flight).

3. A fact is a piece of genuine information that is required to be stated in the contract (e.g. the provider's name and contact information, or the name, address, nationality and a credit check for the customer)

Generally speaking, for each right pertaining to one party there exists an obligation for the other party about the same issue in one contract. In other words, if the customer has a right on an issue, simultaneously the provider has an obligation to do something (or to abstain from doing something) in order to respect that right or to give concrete execution to that right.

[2] This and the following definitions come from Wikipedia, which is now widely recognized as a reliable source of information.

Even if related obligations and rights could be stated as only obligations (or rights), it is still useful to formally distinguish between them. An important difference between rights and obligations is that a party can ignore a right of their own if it is considered non-relevant to their goals, while an obligation can never be ignored if it is part of a contract that has been agreed by both parties. This distinction will be used later for automation purposes.

2.2 CI Process

A CI is built by providing information from both parties. Note that only a part of this information is typically relevant for a specific CI. For example, the provider may want to enforce a rule stating that in order to qualify for a "Young discount" the consumer needs to be younger than 25 and needs to disclose their date of birth in the CI. In the case that the consumer does not qualify for the discount, their birthday is not relevant to this specific CI and does not need to be disclosed. In fact the rule is also not relevant when the customer simply states that he does not qualify for this option. We then define a "NORMALIZED CI" as a CI which only contains relevant information for a specific instantiation.

2.2.1 Master Contract. From our perspective, the provider is the main contributor to the CI. A client cannot modify the OBLIGATIONS and RIGHTS the provider defines. The provider defines some FACTS (such as their identity and contact info) and the FACTS TO BE INSTANTIATED BY THE CUSTOMER (such as their name and birthday). The provider may also define a set of PARAMETERS that the consumer needs to instantiate. For example, a flight-booking service would have, among others, parameters for stating the departure and arrival airports, and the date and time of the flight. Once they are instantiated they are added to the CI in the form of obligations and rights.

Typically, the provider can define CONSTRAINTS over the parameters, both unary and n-ary, and a set of "RELEVANCE RULES" (such as the "Young discount" one described above) to compute which parameters and requested facts are relevant in a specific contract.

We define a "MASTER CONTRACT" as the collection of all the types of information that contribute to the CI Process, namely: OBLIGATIONS, RIGHTS, FACTS, REQUESTED FACTS TO THE CONSUMER, PARAMETERS, PARAMETER CONSTRAINTS and RELEVANCE RULES.

3. Extending OWL-S for Service Contracting

We would like to automate the CI process within the framework of Web Services invocation. Essentially this CI process can be decomposed into three steps:

1. the provider implements a web service and provides a OWL-S description extended with the Master Contract information;

2. the client agent retrieves this description and, if the terms are compatible with the user's goals and constraints, it assents to the contract by invoking the web service and passing the requested information (facts and parameters);

3. finally the provider agent maps the given facts and parameters to the CI. It can either accept the CI and execute the service, or refuse it and send back a refusal notification.

A partial example of an OWL-S extended service description[3] is given in fig 1. Following the Semantic Web initiative, constraints and relevance rules could be modeled using SWRL[4]. From this description the client agent needs to evaluate whether the service fits the user's requirements. This evaluation has four tests that need to be passed.

First the client agent needs to test that the client's constraints can be satisfied by the provider. This information can be found in the provider's obligations, consumer's rights or the service parameters. In the latter case, the client could already determine the relevant parameters.

For instance, let's assume that the customer wishes a smoking seat. If the Master Contract states that the customer has the right of smoking then his constraint (the smoking seat) is validated. Alternatively, it could happen that the Master Contract defines the possibility of smoking as a service parameter (which could be restricted by the availability of smoking seats). In this case the customer agent would select this parameter as relevant in the contract.

The second test would be that the client had asserted all his obligations stated in the Master Contract as commitments. By differentiating between obligations and rights, we avoid the need for the client to be aware of the semantics of those rights that are not relevant (i.e. not stated in the user's constraints).

[3] We do not define in this document the complete CI ontology used, but we intend that it should be interoperable with the Business Collaboration Framework developed by the United Nations Centre for Trade Facilitation and Electronic Business [(UN/CEFACT].
[4] http://www.daml.org/2003/11/swrl/

```
<rdf:RDF
    xmlns:process=… xmlns:profile=…
xmlns:contract="…/TransportContract.owl#>

<profile:Profile rdf:ID="BookingFlightProfile">
   <contract:applyLawOf="Switzerland"/>
   <contract:hasParty rdf:ID="Offeror">
      <hasType rdf:about="#OfferorType"/>
      <hasActor rdf:about="#TheFlightCompany"/>
   </contract:hasParty>
   <contract:hasParty rdf:ID="Offeree">
      <hasType rdf:about="#OffereeType"/>
      <!-- the info about this actor has to be provided
by the customer-->
   </contract:hasParty>

   <contract:hasClause>
     <contract:Obligation>
       <clauseBy rdf:about="#Offeror"/>
       <clauseTo rdf:about="#Offeree"/>
       <contract:FlightTransportation>
          <hasOrigin rdf:about="#Geneva"/>
          <hasDestination rdf:about="#Paris"/>
          <!-- the time has to be provided by the
customer>
       </contract:FlightTransportation>
     </contract:Obligation>
   </contract:hasClause>

   <contract:hasClause>
     <contract:Right>
       <clauseBy rdf:about="#Offeree"/>
       <clauseTo rdf:about="#Offeror"/>
       <contract:SmokingAllowed/>
     </contract:Right>
   </contract:hasClause>
...
  <process:AtomicProcess rdf:ID="#PayAndSendTicket"/>
    <process:hasInput>
      <process:Input rdf:ID="theOfferee">

<process:parameterType>contract:Actor</process:parameter
Type>

<contract:map>contract/hasParty[@type="Offeree"]/hasActo
r</contract:map>
      </process:Input>
     </process:hasInput>

    <process:hasInput>
      <process:Input rdf:ID="flightTime">

<process:parameterType>xsd:timedate</process:parameterTy
```

Figure 1. A partial example of the extended OWL-S service description

Third, that the facts requested in the Master Contract are present in the facts provided (by both parties). The last test would be that no service parameters are left unspecified.

If the tests are successfully passed, the PA can now invoke the web service, passing the requested facts and service parameters. By doing so, the PA is providing the information for the final contract, giving its assent to the contract (the client wants to be bound by the contract) and asking the provider to execute the service specified by this final contract.

The provider agent will consequently use the Master Contract and the provided information to produce the final concrete normalized CI. If it accepts this CI (i.e. it may need to check its resources first), the provider executes the service. Otherwise it sends a refusal notification back to the client agent.

4. Related Work

There are some attempts to build machine readable deontic contract languages, mainly the Business Contract Language [BCL], the Contract Expression Language [CEL] compliant to the Business Collaboration Framework developed by the United Nations Centre for Trade Facilitation and Electronic Business [UN/CEFACT], and the OASIS eContracts specification[5]. They are XML-based and can be used to express the contractual terms between the participants, primarily to govern the distribution and use of electronic content. While these languages focus on how to describe the terms of the contract, we focus more on integrating the CI process with the Web Service framework. By doing so, we can take advantage of related work in the area of Web Services (e.g. signature, trust, composition, etc.).

5. Conclusions

This work is currently at an exploratory stage. The process flow has been now outlined and the next step is to look further into each stage of the proposed approach, by implementing the related supporting tools. We also plan to apply the proposed approach to real world case studies to identify the main challenges, such as the semantic interoperability of business terms.

6. Acknowledgments

David Portabella would like to thank Esteban Sánchez and Michael Schumacher for the stimulating discussions, as well as the anonymous reviewers and Ian Blacoe for correcting my English.

7. References

[OWLS] The OWL Services Coalition: Semantic Markup for Web Services (OWL-S), http://www.daml.org/services/owl-s/1.0/

[CEL] Contract Expression Language, revision 1, May 2, 2003, http://www.crforum.org/candidate/CELWG002_celspec.doc

[BCL] S.Neal et al, Identifying requirements for business contract language: A monitoring perspective,
Proceedings of the seventh International Enterprise Distributed Object Computng Conference, pages 50-61, Brisbane, Australia, September 2003. IEEE Computer Society.

[UN/CEFACT] Business Collaboration Framework (BCF), the Techniques and Methodologies Group (TMG), United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), http://www.unbcf.org

[5] Work in progress:
http://www.oasis -pen.org/apps/org/workgroup/legalxml-econtracts/documents.php