A Dialogue-based Grounding mechanism and new Service Description Features for adapting Semantic (Web) Services to Personal Assistants

(EPFL Technical Report IC/2004/85)

David Portabella and Martin Rajman
Artificial Intelligence Laboratory
Swiss Federal Institute of Technology (EPFL)
{david.portabella, martin.rajman}@epfl.ch

November 3, 2004

Abstract

We delegate increasingly sophisticated and complex tasks to computer programs, and we require them to act autonomously on our behalf in distributed environments (eg the web, the grid). Personal Assistants (PA) are computer programs that can be thought as a very natural metaphor for such systems. Semantic Web Services are self-contained, modular applications that can be semantically described and published over the Web by service providers. This makes it possible for a PA to look for a particular type of service in a directory and invoke it on the fly. However, current service description languages such as OWL-S do not help much the PA to determine the correct input arguments in order to invoke the service and so user intervention is required. We are aimed at minimizing the necessary user intervention by adding new service description features and the use of a dialogue-based grounding mechanism that can be exploited by the procedural semantics capabilities of the provider and the reasoning capabilities of the PA. As a validation we use the proposed approach in an e-service example to compute an insurance premium based on the requirements of the user.

Keywords: interoperability, semantic web, service invocation, personal assistant, dialogue

1. Introduction

Distributed on-line services become more and more available for ordinary computer and internet users. Moreover, computing and communication capabilities are pervasive in consumer appliances. We can imagine a new society where people, computer devices and varieties of consumer appliances will be identifiable on the internet. Interaction between humans and network devices will benefit of technology advancement in sensory systems and user will be able to communicate with them by means of less intrusive and natural interaction.

This scenario offers a series of new challenges for researchers in distributed systems and human-computer interaction. On the one hand, the massive presence of networked devices in everyday life will extend the possibilities of interaction in augmented reality environment. On the other it will introduce a higher level of complexity in operating such devices. However, in many cases the smart design of the software infrastructure for the coordination of these devices may help in solving this problem. If users are not required to dramatically change their behaviors in everyday life they will eventually accept new technologies and benefit of their real power.

There are a lot of situations where the above principles apply ranging from information seeking tasks to remote control of consumer appliances and e-commerce services. These tasks can be made easier to perform from the user perspective in at least three types of situations:

1. the user is accessing the same service several times
2. the user is trying to achieve the same goal by contacting different services
3. the user is learning how to use a (complex) service.

Semantic Web Services are becoming very popular. They are self-contained, modular applications that can be semantically described and published by service providers. This makes it possible for a PA to look for a particular type of service in a directory and invoke it on the fly. However, current service description languages such as OWL-S [3c] do not help much the PA to determine the correct input arguments in order to invoke the service and so user intervention is required.

We can imagine that structurally complex services are activated by complex forms of interactions such as dialogues. This is often the case also in Web transaction where the users need to fill several forms and undergo several stages before actually activating the service (eg a purchase). In these cases, the user might benefit of an intelligent mediation by delegating a Personal Assistant (PA) who knows the "typical" user behavior and is able to simulate it [1]. The PA takes care of specifying all the necessary details required for the activation of the service, thus enabling the user to specify only just the strict necessary information.

It is apparent that complex services will still require the user intervention. We are aimed at minimizing the necessary user intervention by adding new service description features and the use of a dialogue-based service grounding mechanism that can be exploited by the procedural semantics capabilities of the provider and the reasoning capabilities of the PA.

We present now a (1) state-of-the-art, (2) followed by a quick overview which explains with the use of an example the problems that we try to solve, (3) the proposed approach and (4) finally a discussion, conclusion and future work.

In this paper, for the sake of comprehension we will refer to the provider system as "it", to the PA as "he" and to the user as "she".

2. State-of-the-art

2.1. Dialogue systems

Spoken dialogue systems have been defined as computer systems with which humans interact on a turn-by-turn basis and in which spoken natural language plays an important part in the communication [2].

The main purpose of a spoken Dialogue System (DS) is to provide an interface between a user and a computer-based application such as a database or expert system. DS enable casual and naive users to interact with complex computer applications in a friendly way [2b]. Because of this, DS should accept several ways of providing the same relevant information, which is the process of "accommodation". The features of these systems show its adaptivity, robustness and flexibility.

DS can be classified into several types, according the methods used to control the flow of the dialogue with the user: structural dialogue models, form filling or frame-based, and reasoning and planning models. In frame-based dialogue models the user is asked questions that enable the system to fill slots in a template. The dialogue flow is not predetermined but depends on the content of the user's input and the information that the system has to elicit. They are appropriate for well-defined tasks in which the system takes the initiative in the dialogue and elicits information from the user to complete a task [2b].

An example of frame-based DS which implements also mixed-initiative is the RDP [2c]. Its dialogue flow strategy is hand craft in the system and it deals with the branching logic, the request for help, request for repetition, error handling because of no match or incoherence, and confirmation. This flow strategy is task-independent.

2.2. The Semantic Web and Semantic Web Services

"The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming." [3]

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [3b]

A Web Service is a self-contained, modular application that can be described, published, located, and invoked over the Web. OWL-S is a Semantic Web Services description language that enriches Web Services descriptions with semantic information from OWL ontologies and the Semantic Web [3c]. OWL-S is organized in three modules: (1) a Profile that describes capabilities of Web Services, (2) a Process Model that provides a description of the activity of the Web Service provider, and (3) a Grounding that is a description of how abstract information exchanges described in the

Process Model are mapped onto actual messages that the provider and the requester exchange.

OWL-S distinguishes between two types of processes: composite processes and atomic processes. Atomic processes correspond to operations that the provider can perform directly, while composite processes are used to describe collections of processes organized with some control flow structure. For the sake of simplicity we will presuppose only atomic processes.

3. Quick overview

We introduce the following example that will be used during this paper and afterwards we identify some problems that appear when using current services grounding mechanisms. Treating these problems is the purpose of this paper.

3.1. Example

Let's suppose a user is interested in getting the optimal insurance offer for her car based on her needs. Nowadays some car insurance companies have an e-service to compute the insurance premium with an idiosyncratic algorithm which needs of a set of parameters. Among these parameters, the ones shared by the majority of the providers are stated in a generic car insurance description while the rest are in other more specific ontologies. Given this situation, the way in which the PA of the user will proceed is by asking her all the information defined in the generic car insurance ontology and by looking for all the providers offering this type of service. For each provider found, the PA will look at its service description to see all the input parameters needed and if he does not foresee some of them (not the ones defined in the generic car insurance description but the more specific ones), he asks them to the user. Only then the PA invokes the service.

```
Semantic service name:
ComputeCarInsurancePremium

Input parameters ontology:
CarModel type String
MainDriverDateOfDriverLicense type Date
MainDriverAge type Age
PartialCoverage type Boolean
CollisionCoverage type Boolean
   # CollisionCoverage being true implies PartialCoverage also to be
true
Retention type Price restriced to 0 CHF, 500 CHF or 1000 CHF.

Outputs parameters ontology:
Premium type Price;

Effects:
OfferComputed
CannotMakeAnOffer
InvalidInputArguments
```

Fig 1. Generic car insurance service description

It defines in addition to the generic car insurance service description, the following inputs parameters:

```
MainDriverNationality type Nationality
MainDriverSwissResidencePermitType type SwissResidencePermitType
     restricted to “B”, “C”, “L”, “none” or “I am Swiss, no need for Swiss Residence permit”
MainDriverIsUsedToDrugs type Boolean
  # True if the main driver is used to drugs
GarageAtHome type Boolean
  # True if the car is protected from bad weather and from human malintentional purposes
```

Implemented algorithm to compute the premium:

```
{
      if (mainDriverNationality is not Swiss and mainDriverSwissResidencePermitType is not C nor B)
            return Effect CannotMakeAnOffer;

      Price premium = 500 CHF;
      premium += getPlusAccordingToCarModel(carModel);

      if (mainDriverAge < 25 && expensiveSportiveCar(carModel) is true && mainDriverIsUsedToDrugs is true)
            return Effect CannotMakeAnOffer;

      if (mainDriverAge < 25 or (CurrentDate - mainDriverDateOfDriverLicense) < 2 years)
            premium += 300 CHF;

      premium += getPlusAccordingToMainDriverNationality(mainDriverNationality);

      if (partialCoverage is true)
            premium += 200 CHF;

      if (collisionCoverage is true and partialCoverage is false)
            return Effect InvalidInputArguments;

      if (collisionCoverage is true)
            premium += 400 CHF;

      switch (retention) {
            case 0 CHF:
                  if (mainDriverAge < 25)
                        return Effect CannotMakeAnOffer;
                  premium += 400 CHF;

            case 500 CHF:
                  premium += 200 CHF;
      }

      if (garageAtHome is true)
            premium -= 20 CHF; //discount

      return premium and Effect OfferComputed;
}
```

Fig 2. Specific provider implementing the same type of service

3.2. Problems identified

We are now going to identify several types of interrelated problems with current services grounding mechanisms that we try to treat in this paper.

- The PA will ask to the user whether she has a garage at home, even in the situation that the user is not Swiss and has not a residence permit in which case the company does not need the former information to inform that it cannot make an offer anyway.
- The PA will ask to the user her type of residence permit even if she is Swiss, which discards the former information.
- The provider possibly would like to avoid reveling that it uses the information about drugs unless really necessary, ie when the user is younger than 25 and has an expensive sportive car.
- The user has to answer whether she has a garage at home even if she just wants a first approximation and she does not care about small discounts.
- If the objective of the user is just to know whether the company can make an insurance offer for her in the situation that she is younger than 25 and wants an insurance with no retention, she still will need to answer non relevant questions such as the garage.
- It would be quite difficult that the PA can determine the actual argument for the concept "MainDriverNationality" even if the main driver is the user herself and the PA knows her nationality.

4. Proposed approach

4.1 Dialogue

Services can be described using a service description language such OWL-S [3c]. The description includes, among other features, a semantic name for the service, a set of input and output formal parameters defined in an ontology and a set of possible effects. Let's say that there is a generic service description for a particular type of service. A specific provider implement this type of service but it needs some more formal inputs parameter, so it extends the generic service description with the extra input formal parameters. Now there is a PA coded to use this type of service based on the generic service description. The OWL-S Grounding force that before the service can be activated the PA needs to pass to the provider all the actual input arguments in one-shot. As the specific provider needs some input parameters not foreseen by the PA (the ones that are not in the generic service description), the PA will need to ask their value to the user before invoking the service. However, for a particular situation not all the input arguments are usually used nor they all have the same degree of importance for the activation of the service.

In a dialogue approach, the PA invokes the service of the provider without passing any actual input argument. The provider has a private frame that is associated with formal input parameters which must be filled with the actual input arguments in no particular order during the course of the dialogue. (Note that because of the analogy with dialogue, from now on we may use "question" to refer to "formal input parameter" and "answer" to mean "actual input argument"). The provider asks questions one by one on the fly, as

needed, so the PA will only have to determine the answers required for a particular situation, thus reducing the user intervention, and the provider can hide the insights of how its service works by not revealing the questions used for all possible situations. Moreover, questions can be "discussed" between the provider system and the PA (as explained in the following sections) which will be of crucial importance to minimize the user intervention. Finally, the formal input parameters defined in the generic service description allows for a mixed-initiative dialogue approach, so that the PA can modify the course of the dialogue based on the importance that her user gives to each of these questions.

4.2 Roles

Identifying the roles for a service in the generic service description can help reducing the user's intervention, let's see how. Note that it would be quite difficult for the PA to determine the actual argument for the concept "MainDriverNationality" even if the main driver is the user herself and the PA knows her nationality. On the other hand, associating the questions of the provider to the PA's model of the user could be much more useful.

This can be achieved in the following way. As far as the provider is concerned, the service description should define in addition to the input parameters, several roles associated with the service. Besides, the input parameters of the service should be concepts defined in a public ontology which refer to these roles. And as regards to the PA, he should have a model of the user which should be organized in entities, each of which could be represented by several (maybe overlapping) public ontologies describing the type of entity. When invoking the service, the user can associate the roles of the service to the pertinent entities she has in her PA.

In the example of the insurance, the generic car insurance service description could define the role "Main driver". The concept "Date of driver's license" could be defined in a public ontology for "Driver", and the concept "Age" in a public ontology for "Person". Now let's say that main driver is the sister of the user. The PA of the user has an entity called "my sister", which is instantiated using the ontology "Person" cited previously, and contains the information of her age.

Before invoking the service, the user would associate the role "Main driver" with the entity "my sister". The PA would then create a new instantiation of the ontology "Driver", associate it to the entity "my sister" and ask to the user her sister's date of driver's license. The PA would not need to ask the age of her sister because it is already defined.

After invoking the service, the provider could ask the "Date of Driver's license" and "Age" for the role "Main driver", which the PA could easily answer.
The specific provided needs also the nationality of the main driver. We can imagine that in the "Person" ontology there is also the concept "Nationality". Then the provider could ask in the same way the "Nationality" for the role "Main driver". If it turns out that the PA has this information about the sister of the user, he can answer on behalf of

the user. Otherwise he can ask this information to the user and memorize it for further interactions.

Let's compare both approaches using the example of the nationality of the main driver. Memorizing the concept "MainDriverNationality" could only be useful when invoking similar types of services, while memorizing the nationality of "my sister" could be more useful because this information could be applied again in a booking flight service for instance.

4.3. Structure of a question

We said before that the private frame of the provider system has associated several questions. A question is a concept defined in a public ontology and so there is an ontological commitment by the people who uses the ontology about the semantics of the concept. A concept has associated a type of value, eg a number (integer, decimal...), a text string, a time value (time, date...), and a multimedia value (the provider system may ask a photo of the user)...

In dialogue systems the user can typically ask for help for a question. This behavior is still useful for the PA. It could ask the provider system the constraints for a valid answer. Independently of the type, the answer could be restricted to a list of possible values. There can be other type of constraints based on the type of the answer. For instance, if it is a number, it can have a minimum and/or a maximum.

This needs to be integrated in a more complete private service description which contains the private frame cited above. A question is associated a list of properties. One of these properties refers to the constraints for the valid answers to the questions. This "constraints" property would contain the sub-properties "list of possible values", "min", "max" and so on if applicable. More types of properties will be proposed during this paper.

Furthermore a question could be a collection of sub-questions. Indeed, having a hierarchy of questions can help organizing the data. In the example of the insurance, there could be four main questions: the vehicle, the main driver and the desired insurance. Then the questions "date of driver's license" and "age" would go under "main driver". Note that instead of making a hierarchy of concepts based on generalization-specification as in common ontologies such as WordNet and CYC ("car" goes under "vehicle" which goes under "machine"), we make a hierarchy based on the functionality of the concepts as in FrameNet. This hierarchy would be valid only for a particular type of service and can help understand the meaning of the concepts.

We said in the previous sections that during the dialogue the questions can be discussed between the provider system and the PA. The PA can do so by asking to the provider for the different types of questions properties. At the moment we have only described the "constraints" property by which the PA can ask for example for the list of possible values. More interesting properties are defined in the following sections.

4.4. Intentions of the questions, implications of the answers and user goals

An e-service is built with clear objectives in mind. In the insurance example, the goal of the service is that the company makes an insurance offer to the user based on his requirements. This is the general objective. In fact, the user could instruct the PA to just find all the companies that can assure her for liability with no retention being less than 25 years old, and then the user could just personally contact them.

So there can be situations where the goal of the PA (on behalf of the user) is not exactly the one for which the e-service was designed for. This should not be a problem if the user goal can be satisfied just by answering the questions defined in the generic service description. In the example, the PA just needs to answer the questions for main driver's age, liability coverage and retention zero. If the provider system success in making an insurance offer, the PA just adds this company to the interesting companies list.

The problem arises when a specific provider needs more information to compute an offer than the questions described in the generic service description, ie questions not foreseen by the PA. In order to compute the premium of the insurance offer, the provider may need to know whether the user has a garage at home in which case the user would benefit from a premium discount. Asking this to the user would annoy her, as she is not interested in the exact premium, only wants to know if the company can make her an offer with his restrictions.

One approach to solve the problem above would be that the PA could explain somehow the user goal to the provider system. An opposite way would be that the PA could ask to the provider system the reason why it is doing such a question, and then act in consequence based on the final goal of the user.

In any of both approaches before, again using a kind of logic formalism to autonomously reason about in unforeseen situations seems quite unrealistic.

We propose to add, in the generic service description, a hierarchy of basic **intentions** for possible unforeseen questions, **implications** for its answers and **user goals** of the user. We defended that, when designing a generic service description, it is quite unrealistic to think about all the possible specific questions that each provider could need. Our assumption is that thinking about a set of reasons why specific unforeseen questions would be needed for a particular type of service is much more feasible.

Very often, the activation of a service requires a great amount of information that the user is asked to provide. Not all of this information is strictly required to describe the parameter settings for the service. In the example, there could be defined two main intentions in the generic service description: "We need to ask the question in order to determine the price of the premium", and "We need to ask the question in order to determine whether we can make an insurance offer to the user". Such a distinction can help the PA decide whether the question can be ignored.

Sometimes may be worth to give some precise information along with the intention, using templates: "we need to ask the question because there could be a discount of up to

X CHF / year", where X = 5. The PA could decide whether it is worth to intruder the user for such a small benefit.

Defining a set of implications for answers to the questions in the generic service description can also be exploited by the PA. If the implication of choosing "yes" to the unforeseen question "Coverage option for theft of personal belongings?" is that "The premium will be X CHF more per year", where X=300, and the user instructed her PA to limit the total offer to 250 CHF, the PA can autonomously decide on behalf of the user.

Sometimes services could be better requested by stating goals and not by describing the operational details for achieving those goals. Again, defining a set of user goals in the generic service description can improve the performance of setting a default value. Based on the user goal selected by the PA for a particular question (eg, "Select the cheapest option"), the provider system can better select a default answer for the user.

We integrate all this information by firstly defining in the generic service description a hierarchy of template intentions for possible unforeseen questions, implications for its answers and user goals of the user, and secondly by adding in the private service description a property to the questions named "intention", a property to the answers named “implication” and a property to the questions named “help”, where there is a list of user goal-default value pairs. We believe that this information could be extensively exploited by the PA for reducing the user intervention.

4.5. Procedural Semantics

For the approach to be useful, it should be able to resolve situations in which the PA does not understand the question of the provider system (or there are privacy issues which do not allow the PA to answer), but could be easily inferred from other questions. An example could be that the provider system asks whether the user is underage, but instead the PA only knows the concept of birthdate. An example related to the insurances could be that the PA does not know the concept “Garage at home” defined in the ontology of the specific provider, but there is a “Home place” role defined in the generic service description and the PA of the user has an entity called “my home place” associated with the cited role and instantiated with a “Home Place Ontology” public ontology which describes, in addition to other features, the type of parking place.

One possible approach would be that the provider system describes to the PA the concept (eg "Garage at home") using some kind of formal logic and then the PA could compute its value. CYC has carried out a lot of work defining a common top ontology using formal logics. New concepts could then be defined using this top ontology and formalism. Unfortunately, automatically computing similarity of two concepts seems quite unrealistic and so it makes this approach impracticable.

We propose instead to use procedure semantics to just infer the answer needed, without the need to explain the question. For this to be possible, the provider system needs to be prepared to accept alternatives for a question. If the PA does not understand a question or does not know its answer, it needs to first look whether there are alternative questions that he can autonomously answer before asking the user intervention.

4.5.1. Relations between questions

In order to achieve the goal defined previously, the PA needs to know what the relations between the questions are. We identify four types:

- The simplest relation is the answer for a (set of) question is **equivalent to** the answer for another (set of) question. The provider system could be prepared to accept either an answer for "Garage at home?" or an answer for the three questions “Parking place?”, “Is the parking place covered?” and “Is the access to the parking place restricted?” from the “Home place” role. In this case, if the provider system asks to the PA for the question "Garage at home?" and he does not know it, the PA can ask to the provider system for the alternative questions. The provider system would inform him that it can also accept the three questions from the “Home place” ontology as an alternative. If this time the PA knows the answers for them, he would have succeeded in avoiding the user intervention. Note that this is a symmetric relation.

- Another useful relation would be: the answer for a (set of) question is **determined by** the answer for another (set of) question. It is a less strong relation than the previous one as it is not symmetric. Let’s suppose that the provider system ask if any of the usual passengers is a “baby”, defined as person who is less than 3 years old. An alternative question which determines this concept is “age”. If the PA only knows the answer for the latter question, the provider system can apply procedural semantics to infer the former one. In this case, "baby" and "age" are not equivalent questions, but answering the latter determines the answer for the former one. However, the other way around it is not true.

- In the previous case, by answering with "age" as an alternative question for "baby", the PA is giving more information than necessary. For this reason we introduce another kind of relation less strong: the possible answers for a (set of) question are **restricted by** the current answer for another (set of) question. This means that answering one question does not fully determine the other one. In the previous example, answering the related question "underage" as true does not determine the question "baby". On the other hand answering the first question as false clearly determines "baby" also to false. In the example, when the provider system asks for the question "baby" to the PA, he can request all the related questions, getting "age" and "underage". Maybe the user did not authorize the PA to answer the questions "age" because of privacy issues, but she allowed her PA to answer the question "underage". Once the provider system has the answer for “underage”, it can apply procedural semantics and attempt to determine the answer for "baby". If the passenger was not underage, the provider system would successfully determine "baby" to false and the PA would have avoided the user intervention. If the passenger was underage, unfortunately the PA would need to ask the original question to the user, or ask permission to give the "age".

- There is another useful relation similar to the "determined by" one which is "discarded by": a (set of) question is **discarded by** a condition (eg the answer to (a set of) question). In the example of insurances, asking for the type of residence permit is not pertinent if the main driver is already Swiss. Then, answering the nationality question with "Swiss", the condition, discards the question about the type of residence permit.

When the provider system asks the PA for a question that the he does not know (either the questions itself or its answer), the PA may ask also for conditions that discard this question and avoid the user intervention.

We integrate all this information in the private service description by adding a new property to the questions called "relations". For each relation it should be stated whether inferring the answer from the related ones depends on external knowledge of the provider (eg inferring “age” from “birthdate” depends on the external knowledge of the current date) or would be stable (eg, inferring “underage” from “age”).

4.5.2. During the dialogue

Up to now, all the properties for a question we have introduced are fixed, ie they do not change during the interaction with the PA. We introduce now two alive properties which reflect the state of the interaction: the "status" and the "current constraints".

The "status" property is meant to explain to the PA whether he still needs to answer the question or if it can be ignored. Initially all the questions have an "unset" status. This status can be changed directly by answering the question, in which case it would get the "set" status, or indirectly by answering related questions. In this last case we can find four different situations.

An "inferred" status would be reached by answering an equivalent question.

Similarly, a question would have the "discarded" status if a condition which discards the question has become true.

However, the case where a question is determined by another one is trickier. We need to differentiate whether the nature of this relation is service dependent or independent. An example of a service independent relation is the implication that not being "underage" determines not being a "baby". In this case the question "baby" needs no further confirmation from the PA because this statement is true independently of the service and so the provider system would set the status to "inferred".

On the other hand, an example of a service dependent relation is that asking for "collision coverage" determines accepting also "partial coverage". This inferred value needs to be confirmed by the PA because this implication depends on the service (accepting this option will make the insurance premium more expensive). In this case the provider system would set the status just to "constrained".

When the question is "restricted by" another one, if it remains only one possible answer the case is completely analogue to the "determined by" situation. On the other hand, if there is still more than one possible answer, the status will be set to "constrained" regardless of the case because the PA will still need to choose one of the remaining answers.

The second alive property is "current constraints" which is similar to the property "constraints" explained in the previous sections. When the status of a question is "constrained", this property indicates the possible remaining values.

4.6. Mobile code

We said in the previous section that an answer for a (set of) question can be inferred from a (set of) alternative question. The PA gives the latter one to the provider system and this one applies procedural semantics to infer the former one.

In some situation it would be better if this procedural semantics could be applied in the side of the PA.

For instance it can be used for privacy issues. Taking a previous example, the provider is only interested to know whether the user has a garage at home but it could happen that the PA does not understand the concept. The PA could instead have the entity "my house" described with an instantiation of the "Home Place Ontology" public ontology but he is not allowed to reveal so much information. In this case the PA could download the mobile code to apply this procedural semantics on his side to infer the former question using the questions "Parking place?", "Is the parking place covered?" and "Is the access to the parking place restricted?" from the "my house" entity, and send it to the provider system.

Another case where this approach could beneficial is due performances issues. We can imagine that there is a telephony company that offers a service to compute how much it would cost the calls of your last month using their services. To do so, the PA could invoke the provider service and ask questions such as "how many minutes did you call from 8am to 8pm during the week", "how many minutes on the Sundays" and so on, but this would be too annoying. Let's say that there is a public ontology named "Call Records" which your telephone instantiates and automatically fills with the details of every call. The provider system could accept an alternative question whose value is this ontology instantiation. The PA could send all this information and the provider system applies procedural semantics to compute the previous questions. This approach has also a performance problem as it would waste network bandwidth unnecessarily. As the previous case, the PA could make use of mobile code to infer the former questions and send them to the provider.

The process of using the mobile code could be as follow: the PA asks the provider system whether he can download procedure semantics mobile code for a particular question. If so, when downloaded it has to be executed in a controlled environment so that it cannot be used for malintentional purposes, typically in a virtual machine. This mobile code can communicate with the PA to request the answers for the alternative questions and after applying the procedural semantics it has to communicate to the PA the result for the particular question. Depending on the trust of the provider, the PA could first present the answer that he is about to send to the provider for confirmation.

4.7. Learning a user model

Kobsa [4] has found that in order to be capable of exhibiting a wide range of cooperative behavior, a computer based dialog system must have available assumptions about the current user's goals, plans, background knowledge and (false) beliefs, ie

maintain a so-called “user model”. Translating this to our framework, it seems obvious that the user model should reside in the PA as he can make use of this knowledge during the interactions with the different provider systems.

The more complete is the user model, the more autonomous the PA will be able to act on her behalf. Whenever the provider system asks a question that the PA does not know, either the semantics of the question itself or the answer for it, and all the previous strategies to avoid the question failed, he needs to ask it to the user.

The PA needs to take care about which information he asks to the user. In order to minimize her intervention in the future the questions have to be as much recyclable as possible. For instance, if the provider system can accept the answer for any of the questions “underage”, “age” or “birthdate” it is clear that it is more recyclable to ask her birthdate since the two former questions can be inferred with the latter one and some background knowledge, ie the current date. Moreover, it is in a normalized form which means that the value does not change. However, if an answer is more recyclable it may also mean that we are giving more information to the provider than necessary. So there is a trade-off between minimizing the user intervention and giving the minimum private information to the provider.

One way to know which question is the most useful is to look at their relations. When a question is determined by another one it is clear that the latter is more recyclable. On the other hand, given a question, any of its alternatives that just restrict or are equivalent to it are not more recyclable.

If there is mobile code available to interrelate questions (eg computing underage and age from the birthdate), either from the provider system or from a third-party, the optimal proceeding would be to ask the more recyclable question to the user and get the mobile code to infer the alternative ones. In this case, the user is intruded as little as possible and the provider gets only the strictly necessary information.

Another dimension that must be considered by the PA is the temporal validity of the answers. In the cases where the user model is learned by the provider system as described by Kobsa, the provider system can benefit of the information about the nature of the question. To simulate this behavior in our framework, the PA should also be aware of this information. When the PA does not know the semantics of a question, it would be then useful if he could ask the provider system the temporal validity of its answers. Let’s see an example. Suppose a sport center is affiliated with a university and its customers get a discount if they have a valid student card from this university. The sport center knows that these student cards expire at the end of each academic year, eg end of July. When the provider system of the sport center asks the PA whether the user has this student card, if he does not know the meaning of the question and finally needs to ask it to the user, he could also ask the provider system the temporal validity related to this question. The provider system would inform him that this card expires on 31st July 2005 and so he could presuppose that in future interactions until this date he can reuse the information given by the user.

For this purpose we add a new property to the questions named “answer validity”. If the question is in a normalized form such as “birthdate”, the answer validity would be to

"always true". Using some predefined structures, the provider system could state that the answer validity depends on the time (eg expires in two days, valid only on Wednesdays…), the place (the answer can only be used while in the same country) or other situations. If the provider system cannot know the temporal validity, eg for how much time the residence permit type will be valid, the provider system has to state so by letting the PA ask it to the user if pertinent.

Finally, advanced user model learning techniques could benefit of this information.

4.8. Interaction between the PA and the user

As said in the previous section, the PA has more chances to know better the user than several provider systems. If seems obvious that the PA should be in charge to decide the type of input and output multimodality to use with the user, eg it would make sense that the provider systems asks the PA to play a sound if the PA knows that the user has a hearing disability.

This could be achieved by adding a list of sub-properties named "representation" under the "help" property of the questions. Each of the "representation" sub-property could associate the type of output modality and its resource, eg "text string" and "what is the age of the main driver", or "sound file" and the associated resource. For the answer could work the same way with input modalities.

The PA could select the input and output multimodality resources present in the provider system according to the user's preferences.

5. Discussion, Conclusion and Future Work

In this paper we have proposed new service description features and the use of a dialogue-based grounding mechanism that can be exploited by the procedural semantics capabilities of the provider and the reasoning capabilities of the Personal Assistant to minimize the user intervention to active an e-service.

We make use of the accommodation feature inherent to dialogue systems for accepting several ways of providing the same relevant information by using alternative questions combined with procedural semantics. The feature of these systems show its adaptivity, robustness and flexibility, which should be present in any system working in open environments.

We defended that when designing a generic service description, it is quite unrealistic to foresee all the possible specific questions that each provider could need and we base our assumption in that foreseeing a set of roles, intentions of the questions, implications of the answers, objectives of the user for a particular type of service is much more feasible.

Contrary to other service activation approaches which use natural language processing techniques, we avoid instead semantic problems by exploiting the power of the Semantic Web where an ontological commitment has already been reached with the use of public shared ontologies.

An important issue concerns how much private information we are ready to give to our PA. The more relevant information the PA has about us, the more autonomous he will be able to act on our behalf. On the other hand there are also privacy issues: how the PA can know how much private information he should give for each type of service? Also we will still be quite worried against hacker attacks that could steal our private information. We said that interacting with a provider system can help inferring related information that our PA can use in further interactions with other services. How much we can rely on the correctness (either for mistakes, either for malintentioned purposes) of this inferred information? The answers to these questions will hopefully come from more proven secure systems and the use of repudiation mechanisms.

Future work includes carrying out evaluation experiments and generalizing the proposed approach to benefit from areas other than B2C applications.

6. Acknowledgments

This project was funded by the Swiss National Science Foundation. David Portabella would like to thank Núria Sánchez, Radu Jurca, Steven Willmot and especially to Vincenzo Pallota for our stimulating discussions.

7. References

[1] Maes, P., *Agents That Reduce Work and Information Overload*, Communications of the ACM, 37 (7), 31-40, 1994

[2] Norman M. Fraser, *Spoken dialogue system evaluation: A first framework for reporting results*, In Proceedings of the 5th European Conference on Speech Communication and Technology, pages 1907-1910, 1997

[2b] Michael F McTear, *Spoken dialogue technology: enabling the conversational interface*, ACM Computing Surveys, Volume 34 , Issue 1, March 2002, pp. 90 - 169.

[2c] M. Rajman, A. Rajman, F. Seydoux, and A. Trutnev. Assessing the usability of a dialogue management system designed in the framework of a rapid dialogue prototyping methodology. First ISCA Tutorial Research Workshop on Auditory Quality of Systems, Akademie Mont-Cenis, April 2003.

[3] The W3C Semantic Web Activity Colalition: *The Semantic Web*, http://www.w3.org/2001/sw/

[3b] Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, Scientific American, May 2001

[3c] The OWL Services Coalition: *Semantic Markup for Web Services (OWL-S)*: http://www.daml.org/services/owl-s/1.0/

[4] Kobsa, A., *User Modeling in Dialog Systems: Potentials and Hazards*, AI & Society 4(3), 214-240, 1990