The Free Speech Engine
Conversational web service compatibility for free

Ruud Stegers Frank van Harmelen Annette ten Teije

Vrije Universiteit Amsterdam, {rstegers}{frankh}{annette}@few.vu.nl

Full paper has been published at *The 2009 International Conference on Semantic Web and Web Services.*

1 Introduction

Generally accessible web services have been around now for a while, providing a variety of services to clients. Different technologies are being used to facilitate the communication and coordination between the peers involved. However, without exception these technologies require the publisher to define very specific interaction patterns between the peers involved. The result is a huge amount of clients and services which might be compatible from the perspective of the functional business requirements *(they do the same)*, but which are not due to message or call ordering constraints at the communication level. In the light of creating an open web of (compatible) services, this is an undesirable situation.

We position a service as an embodiment of a task (e.g. sending an email). All communication between a client and the service is about reaching agreement on the details of this task (e.g. recipient, subject, ...). Current web service implement custom operations (through the interface) to reach this state of agreement. Complexity varies from just passing a single data structure, to a full fledged negotiation scheme. Although there are many advantages in doing so, custom operations do impose strong restrictions on the interaction. Often these restrictions do not stem from the business requirements, and should therefore be avoided (e.g. the interface forces you to send *Name* before *Age* while it could as well have been the other way around).

We try to resolve this situation by introducing the Free Speech protocol: a generic web service interaction protocol which prevents restrictions on the interaction level that do not stem from business requirements. We do this by introducing an *explicit representation of the task description* as a key component in the protocol. During the interaction this task description is (gradually) filled with the appropriate information, by both peers in an effort the reach the required agreement. The protocol to make this work does not impose any particular ordering on this process, thereby maximising flexibility.

2 Service coordination by description molding

The task description has the form of a template. Free Speech defines a generic format to describe such a template. A description template contains slots for all the relevant details of a task. Refer to the rounded boxes in figure 1 for an example of a template. On top of this format, Free Speech defines a fixed set of operations that can be applied to parts of a template. This way the interaction between the peers entails a series of update operations, thereby molding the content of the template until the description is satisfactory for all peers involved.

To illustrate this idea consider two people negotiating the sale of a car by filling in an order template on paper. Also here the template contains slots for all the possible options. During the process the content of the template is changed by both the seller and the buyer until they agree on the specs of the car (and the price that comes with that). Simple verbal remarks are used in the process to guide the negotiation process (just like some of the operations in Free Speech).

While traditional services with different interfaces would be incompatible on the both the vocabulary and the interaction level, the sketched interaction model would reduce the problem to just an issue regarding the domain vocabulary (i.e. the domain specific concepts and relations used in the template).

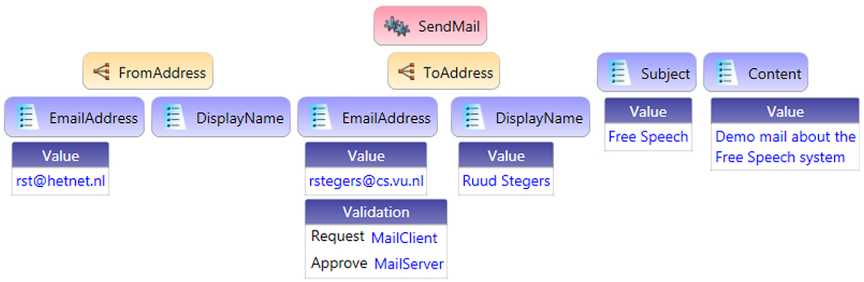

Figure 1: Final content of a communication session.

For the car sale a different domain vocabulary would mean to have a different template to fill in all the options. However, the process to fill it remains the same for both the buyer and seller. The real world equivalent of a different interface on the other hand, would be that the buyer is not only forced to go through a different set of questions and statements, but also in a very specific order.

The purpose of introducing Free Speech is to achieve interoperability between services as a side product of their implementation. Since Free Speech itself does not enforce any ordering, any constraint that might surface during the interaction stems immediately from the implementation which should be governed by the business requirements. Any conflicting constraint is a sign of incompatibility at the business level: something no protocol can (or should) solve.

3 Use case: A mail client and server

To explore the basic ideas behind FreeSpeech we implemented a prototype library, the Free Speech Engine, on top of which 2 mail clients and 2 mail servers were built. Both clients and both servers have different business requirements. The most notable business requirement for one of the mail clients is the need for explicit acceptance of the destination address by the server before any of the other mail fields are passed on. One server implementation only accepts mail for a specific domain.

Current operations supported by Free Speech are *Set value*, which allows a peer to fill a slot in the template with a specific value, and *Request Validation*, *Approve* and *Reject* which allow the peers to exchange information regarding the validity of pieces of data in the template.

Figure 1 shows the final content of the template after a communication session. Initially, the client applied *Set value* to the 'ToAddress.EmailAddress' slot, followed by a *Validation request* on that same slot. Once the service had responded with *Approve*, the client applied *Set value* to the rest of the slots. The semantics attached to this template dictate that once both peers agree on the content, the server will forward the message to the appropriate destination. As can be seen from this example, not only *Set value* makes a change to the template, both all operations do so: the template contains all the required state.

4 Conclusion

With Free Speech, a new communication and interaction protocol for web services was introduced. It is designed to overcome incompatibilities that do not stem from business requirements as is often the case with many contemporary protocols as they are defined in for example WSDL.

An initial prototype of a library has been presented – the Free Speech Engine – on top of which a small set of clients and servers was build. While the WSDL implementation required either changes to the interface, or a very good estimation beforehand of the intended future use to fit in all the business requirements of the peers, the Free Speech protocol allowed seamless interaction between all peers, regardless of their ignorance of each others business requirements and without the protocol enforcing specific ordering requirements.

An issue that does not surface in the current use case, is the point of convergence. If the interaction is left unspecified, there is no guarantee that there will be convergence towards agreement. This will require our attention. At least should the library be able to spot repetitions, and possibly support termination decision making for the peers. Work also needs to be done to mature the set of operations in such a way that also services with a rich set of business requirements are able to take full advantage of the system. The initial effort looks promising and has offered useful insights and will work be continued.