A Service Oriented Architecture for Advertising Games

P. Avesani, M. Cova, R. Tiella
ITC-irst
via Sommarive 18
38050 Povo, Italy
{avesani,cova,tiella}@itc.it

A. Sharma
Birla Institute of Technology
Department of Computer Science
835215 Ranchi, India
arun_bit123@rediffmail.com

ABSTRACT

A critical issue of distributed systems is concerned with the advertising task. Current solutions require an ex-ante agreement on a common shared language. Although such an approach is feasible from the technological point of view, it is not effective in practice. The process of managing this agreement may present social implications that make the solution difficult to achieve. Recent trends in research propose a new approach based on advertising games where the agreement on a common language is produced at run time. Nevertheless up to now such a model has been studied only through simulations with standalone platforms. Our contribution is the design and the development of the first web services oriented architecture for advertising games. Therefore we approached all the issues typical of distributed systems neglected by the simulators like asynchronous communications, denial of services, and so on. Finally we present a real world application where the architecture has been deployed to support the advertising task using an advertising game model.

1. INTRODUCTION

The development of distributed systems is a complex task that becomes even more complex if one adopts the open world assumption where single peers can autonomously design and deploy their own services. In such a scenario interoperability arises as a critical issue because of the heterogeneity of services specifications.

Advertising is one of the steps that is affected by heterogeneity. Usually advertising is concerned with two tasks: service discovering and service semantics specification. In the following we will focus on service semantics specifications.

When new services are published there is the need to notify the specification of service contents. The lack of a shared representation language makes this step harder. The semantic web scientific community is currently working on this issue. Mainly its research effort is pursuing two alternative strategies: promoting a reference ontology [5, 9] or supporting the reconciliation of different ontologies [3, 7, 11]. The first approach, simple from the technological point of view, is not feasible in practice due to sociological implications. The second approach requires a significant manual effort because a full automated reconciliation of ontologies is too complex.

More recently a new research trend proposes an alternative approach to overcome the need of ex-ante agreement in favor of an ex-post agreement. The basic idea is to support a run time process of negotiation that enables the convergence to a common lexicon [15]. Differently from semantic web, the advertising game model aims to shift the problem from matching of representations to negotiation of lexicon. Lexicon can be conceived as an association list that maps labels to meanings, i.e. the service content specifications. Of course a shared lexicon is not equivalent to an expressive knowledge representation language. In fact, a lexicon supports only a common set of symbols to refer to a collection of objects or categories.

There are other research initiatives to support the automated negotiation of mapping between two information sources but these approaches don't scale because they build pairwise mapping [4, 12].

Naming games [14, 16], and more specifically advertising game [1], have been proposed to support the negotiation of a common lexicon independently from the number of players. Up to now the studies on advertising games have been proved to be effective in dealing with the critical issues of service advertising [2]. Simulations have been developed to test hypothetical scenarios of distributed recommendation systems.

The main drawback of these preliminary results is that they are achieved through simulations on standalone platform. No one has approached the issue of porting the model based on advertising game into a full service oriented architecture.

The main contribution of this work is the design and the implementation of a fully distributed, service oriented architecture to enable a real world deployment of an advertising game model. With this approach we dealt with many issues neglected by standalone platforms, like asynchronous communications, denial of services and many others.

The proposed architecture has been deployed to support a sharing annotation system among distributed blog servers. In this specific application the advertising game is used to align the references of distributed ski route catalogs. A first trial is currently under test in the domain of ski mountaineering. Nevertheless, it is not a goal of this paper a

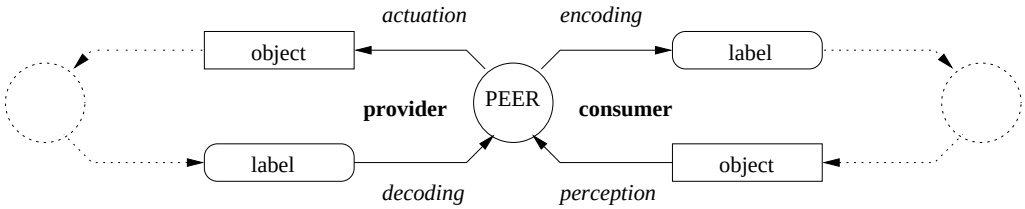

Figure 1: The schema shows the four basic actions performed by a peer during the advertising game, according to its role in the game (provider or consumer).

detailed presentation of such an application.

In the next section we briefly summarize the basic concepts of advertising games. After that we illustrate the service oriented architecture that has been designed to support the deployment of the advertising game model in practice. Finally we show a real world application where such an architecture has been deployed to support the advertising task in a fully distributed systems.

2. GAMING APPROACH TO ADVERTISING PROBLEM

An advertising game involves two or more peers. The basic interaction involves two peers with different roles: consumer (or speaker) and provider (or hearer), therefore a session of communication is not symmetric. Nevertheless each peer can play different roles in different sessions.

More formally an advertising game is defined by a set of peers $\mathcal{P}$, of size $\mathcal{N}_\mathcal{P}$ where each peer $p \in \mathcal{P}$ has a set of concepts $\mathcal{C}_p = \{c_1, \ldots, c_n\}$ of size $\mathcal{N}_\mathcal{C}$. A set of objects is defined, $\mathcal{O} = \{o_1, \ldots, o_m\}$, such that a subset of them can be conceived as representative of a given concept. The objects are shared among the peers while concepts are private of each peer. A lexicon $\mathcal{L}$ is a relation between concepts and words, where it is assumed that words are composed using a shared and finite alphabet. Lexicon is extended with a couple of additional information: the number of times the relation has been used and the number of times the relation was in successful use. Each peer $p \in \mathcal{P}$ has its own lexicon drawn from the Cartesian product $\mathcal{L}_p = \mathcal{C}_p \times \mathcal{W} \times \mathcal{N} \times \mathcal{N}$, where $\mathcal{W}$ is a set of words and $\mathcal{N}$ the natural numbers to represent the peers' preferences. The lexicon may include synonymous words, two words associated to the same concepts, and homonymous words, the same word can be associated to two different concepts. A peer $p \in \mathcal{P}$ is then defined as a pair $p =< \mathcal{L}_p, \mathcal{C}_p >$.

An advertising game is an iterative process where at each step two peers are selected to interact together. The interaction proceeds as follows (see Figure 1). First the speaker p_s randomly selects a concept from its set of concepts, then it encodes the concept $c_s \in \mathcal{C}_s$ through a word w_j. The word is chosen accordingly to the current version of the local lexicon $\mathcal{L}_s$ (local to speaker p_s). The denotation of concept c_s is obtained looking at the most successful word. A word w_j is more successful than a word w_k iff $< c_s, w_j, u_j, a_j > \in \mathcal{L}_s$, $< c_s, w_k, u_k, a_k > \in \mathcal{L}_s$, $u_j \geq u_k$ and either $a_j/u_j > a_k/u_k$ or $a_j/u_j = a_k/u_k$ and $u_j > u_k$, where u_j represents how many times the word w_j has been used and a_j represents how many times there was an agreement on word w_j with other peers. In case of a tie, a random choice is performed. The hearer p_h decodes the word w_j retrieving the associated concept $c_h \in \mathcal{C}_h$ looking at its own lexicon $\mathcal{L}_h$.

The next step is concerned with the actuation task. Actuation can be modeled as a function $f_a : \mathcal{C} \longrightarrow 2^\mathcal{O}$ that takes in input a concept and gives in output a subsample of objects. Actuation function has a stochastic component, therefore two subsequent invocations of $f_a(c_h)$ do not necessarily produce the same outcome. The outcome of actuation is sent back to the speaker. Once received a sample of objects, the speaker has to deal with the perception task. Perception can be modeled as a function $f_p : 2^\mathcal{O} \longrightarrow \mathcal{C}$ that takes in input a sample of objects and gives in output an hypothesis of concept, namely $\hat{c}_h$, that may subsume such a sample. Of course the hypothesis formulated by the perception function is sensitive to the size of the sample.

The last step is concerned with the assessment. The speaker has to verify whether the concept perceived from the hearer's objects is the same selected at the beginning of the communication session. The assessment process can now be carried on easily checking the condition $c_s = \hat{c}_h$.

If the concept referred by the hearer is the same selected by the speaker, both of them give a positive reinforcement to their lexica updating the corresponding word-concept association as follows: $< c_s, w_j, u_j + 1, a_j + 1 > \in \mathcal{L}_s$ and $< c_h, w_j, u_j + 1, a_j + 1 > \in \mathcal{L}_h$. If the hearer replies with a different concept $c_s \neq \hat{c}_h$, it means that the communication failed, the peers' lexicon is updated with a negative reinforcement increasing only the counters of lexical relation (while the counters of agreements on the lexical relation remain the same): $< c_s, w_j, u_j + 1, a_j > \in \mathcal{L}_s$ and $< c_h, w_j, u_j + 1, a_j > \in \mathcal{L}_h$.

3. AN ARCHITECTURE FOR ADVERTISING GAME

Now that we have laid out the theoretical basics of the advertising game technique, it is time to discuss the architecture we propose for its concrete realization. While some implementations already exist, e.g. [13], they only consist of stand-alone simulators that lack the distributed nature of the model. The architecture presented here, by contrast, brings the model into a fully distributed environment, where its strength and weak points can thoroughly and more realistically be evaluated.

In fact, the main goal that guided the design of our architecture was to support an open-ended distributed system. The architecture promotes heterogeneity (by only specifying the minimal set of requirements needed for interoperability and allowing for alternative solutions to be adopted in many parts of the system), autonomy (by avoiding strong

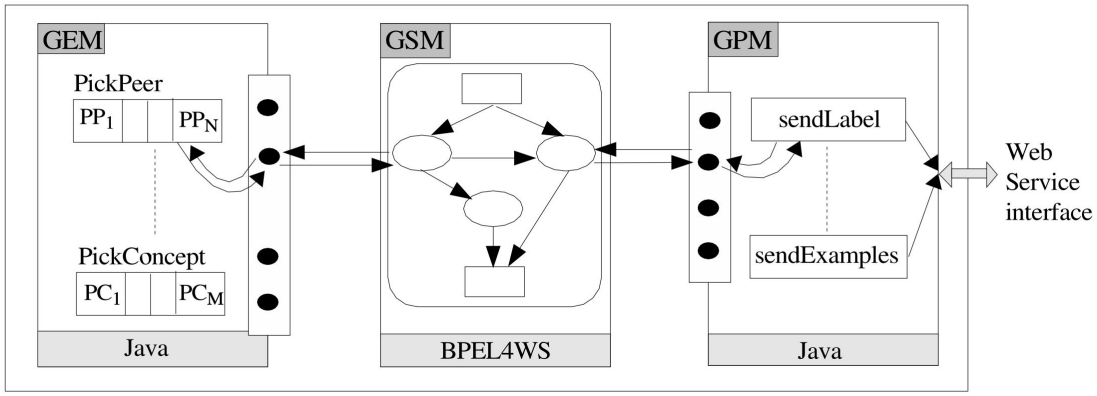

Figure 2: The architecture of the component that implements the advertising game technique.

or centralized coordination among parts of the system), and robustness to evolution (by taking into account that the system is inherently dynamic and subject to change).

By providing an implementation of the advertising game technique, we target legacy distributed systems, where peers are interconnected and cooperating, but have no or little ability to interoperate in terms of lexicon alignment, as explained in the introduction. Therefore, from the software engineering point of view, the main challenge of our research is to realize a component that can be transparently plugged into an existing distributed system, requiring no or as few as possible modifications to its legacy parts. We decided to focus especially on web services based systems, because they represent the current solution of choice for distributed systems interoperability.

The component that implements and encapsulates the advertising game technique is made up of three modules: the *Game Engine Module* (GEM), the *Game Strategy Module* (GSM), and the *Game Protocol Module* (GPM). The GEM module provides the set of primitives needed to perform the advertising game, the GSM module models and realizes the strategy used during a game, the GPM module provides communication primitives. GEM and GPM are providers of mechanisms: they implement and export the building blocks of the advertising game technique. The GSM module, by contrast, realizes policies: it specifies which of these building blocks should be used and how they should be combined to perform a game. The key advantage obtained by this division of the system is that it allows to clearly separate communication protocols from gaming strategies. As a consequence, only protocols need to be defined at system level, while strategies can be autonomously decided by each peer. Furthermore, changes in one module do not affect other modules as long as the common interface remains constant. A graphical description of the architecture is given in Figure 2.

We start describing the component architecture from the Game Strategy Module. The GSM module represents the core part of this architecture. It encompasses the "intelligence" of the peer and informs its acts during the game. It is in charge of a number of activities: deciding when to start a new game and how long it is to last, what are the concepts to play with, which peers are to be taken as opponents, how to evaluate other peers' performance, what type of feedback to provide on current lexicon on the ground of past game results.

At this point of our research, different choices for each of these activities seem to be reasonable and worth more experimentation. It is therefore critical being able to easily implement, test, and compare different alternatives. While one can think of many technologies to model a strategy and implement its runtime, our favor went to BPEL4WS. BPEL4WS is a composition language normally used to perform web services orchestration. It fits our requirements with its built-in coordination features, relatively high-level flow control constructs, and the ability to use services offered by external components through standard invocation interfaces. Other appealing features of BPEL4WS are its relatively high level of abstraction, the possibility to do quick, graphical programming, and its flexibility.

```
<sequence>
  <invoke name="choosePeer" partner="GEM"
    portType="GemPT" operation="pickPeerAtRandom"
    outputContainer="PeerInfo" />
  <assign><copy>
    <from variable="PeerInfo" part="PeerURL" />
    <to variable="Message" part="PeerURL" />
  </copy></assign>
  <invoke name="chooseConcept" partner="GEM"
    portType="GemPT" outputContainer="ConceptInfo"
    operation="pickConceptAtRandom" />
  <invoke name="denote" partner="GEM"
    portType="GemPT" operation="denote"
    outputContainer="LabelInfo" />
  <assign><copy>
    <from variable="LabelInfo" part="Label" />
    <to variable="Message" part="Label" />
  </copy></assign>
  <invoke name="sendLabel" partner="GPM"
    portType="GpmPT" operation="sendLabel"
    inputContainer="Message"
    outputContainer="Examples" />
</sequence>
```

Figure 3: BPEL4WS code that implements a step of the strategy.

BPEL4WS processes realize strategies selecting and orchestrating primitive services offered by the GEM and the GPM modules [1]. In fact, a typical step in a strategy requires the following operations: choose from the primitives offered by GEM one that provides the desired functionality, e.g., choose the peer to be challenged next according to a particular selection method, for instance, randomly. Then, invoke the primitive and collect the result, e.g., get information about the peer, in particular its address. Use this information as intended, e.g., leverage the information about the chosen peer to guide the selection of the concept to play with. Lastly, communicate, if needed, to a remote peer via one of the primitives defined in GPM.

This step of the strategy may be described by the following pseudo-code snippet:

```
Peer p := GEM.pickPeerAtRandom();
Concept c := GEM.pickConceptAtRandom();
Label l := GEM.denote(c);
sendLabel(p, l);
```

Its BPEL4WS implementation is shown in Figure 3.

The GPM module embeds the *choreography* [2] of the distributed system: it defines the protocols available for inter-peer collaboration in terms of sequences of communication primitives.

We identified two fundamental protocol variants: pull-based and push-based protocols. In the pull-based protocol, the speaker chooses the label to play with, sends it to the hearer and waits for a set of examples. Finally, the perception step allows the speaker to align its lexicon to the hearer's lexicon. By contrast, in the push-based protocol, the speaker chooses the label to play with and a set of examples and sends them to the hearer. The hearer then performs the perception step and updates its lexicon to align it with the speaker's. Symmetrical feedback protocols are also possible: in this case, the variants discussed above are extended with a final feedback message, from the speaker to the hearer in the pull-based version, from the hearer to the speaker in the push-based version.

Accordingly with this sketch of the protocols, the GPM module provides primitives

- To send a peer a label;
- To send a label and a set of examples;
- To send a set of examples;
- To send a feedback message.

Higher level characteristics of the communication protocol, such as timeout settings and exception conditions handling, are left to be decided to the strategy module. For full details on the communication protocol, refer to [6].

[1]Some BPEL4WS engines adopt the WSIF [8] framework which allows to define Java bindings to invoke services. When available, we leveraged this technology to minimize inter-module communication cost.

[2]We use the term "choreography" as defined in the Web Services Choreography Description Language draft [10]:

> [Choreography] defines from a global viewpoint [web services'] common and complementary observable behavior, where message exchanges occur, when the jointly agreed ordering rules are satisfied.

Support and enforcement of choreography represent the minimal requirement to be satisfied for interoperability. In other words, as long as an implementation of the GPM primitives is provided, alternative realizations of the advertising game technique can be developed, even adopting different design or technologies, and still maintain interoperability with the architecture we propose.

The last module of the advertising game component is the Game Engine Module. The game engine has to provide an implementation of the operations that are used internally by a peer during a game. In particular, it offers methods

- To select concepts to play with;
- To select peers to be challenged;
- To perform the reification step;
- To perform the denotation step;
- To perform the perception step;
- To update the lexicon on the basis of game results;
- To extend the lexicon if a new label or concept are learnt during the game.

These basic activities can be performed in many different ways. As an example, let consider the activity of selecting the peer to play with the next game. One can imagine alternative criteria to guide this process: e.g., random selection among the set of known peers, selection on the basis of the number of past contacts or past performances, or more sophisticated selection techniques such as active sampling. The GEM module collects and makes available the implementations of each different criterion to the strategy module.

Consequently, it is clear that the GEM module has no requirement of minimality, as it was for the communication primitives. Instead, it provides an open-ended collection of activity realizations that can grow with the necessity of devising more complex strategies. At first glance, for each basic activity there seems to be a limitless number of alternative realizations. However, we expect that further research on advertising game strategy will help single out those criteria that really are helpful for building a shared lexicon and, thus, that the number of implementations needed will be quite limited.

To get a feeling of the primitives that the GEM module might make available, Table 1 shows some variants for the activities of perception, concept and peer selection.

It should be further noted that different peers can have completely disjointed sets of gaming primitives, e.g., one adopts the random peer selection mechanism, the other opts for a selection based on the history of past games. The only requirement is that every peer has at least one realization for each activity, i.e., each peer must be provided with an operation to choose the next game opponent.

While minimality and standardization of communication primitives guarantee interoperability of peers, extendability and specialization of game engines promote differences in peers' personalities and lexicon building behaviors.

Differences in the purpose of modules is reflected in the technology used to implement them. To encode the business logic, we use Java, because it offers a convenient environment to implement computationally demanding activities required by some game primitives. We have already

Activity Family	*Variants*	*Input*	*Output*
perceive	perceivePairwiseMatching	Concept, ExampleSet	int
	perceiveMachineLearning	Concept, ExampleSet	int
pickConcept	pickConceptAtRandom	Peer	Concept
	pickConceptAfterPeer		Concept
	pickMostUsedConcept		Concept
	pickLeastUsedConcept		Concept
pickPeer	pickPeerAtRandom	Concept	Peer
	pickPeerActiveSampling		Peer
	pickPeerAfterConcept		Peer
	pickMostContactedPeer		Peer
	pickMostFriendlyPeer		Peer

Table 1: An excerpt of the primitives made available by the GEM module.

discussed the advantages of using BPEL4WS to define the games strategies. Lastly, for the communication protocol, we turned to web services technology to benefit from its interoperability, reusability and deployability characteristics.

4. A REAL WORLD APPLICATION

The technique and the architecture described in the previous sections are currently under test in a real-world application in the domain of ski mountaineering.

It is common for ski mountaineers to form on-line communities. A community is aggregated around a web site that generally offers two kinds of services: a ski route catalog and a ski trip annotation list. Ski routes are mainly concerned with persistent, static and validated information while ski trips are usually volatile, fresh and not certified.

The ski routes catalog provides information about ski routes. For each ski route it generally gives geographical information, e.g., starting and ending point of the route, a measure of the its difficulty, and other details that might be useful, e.g., an estimate of the time needed to complete it. Members of the community are encouraged to write comments about their excursions on routes contained in the site catalog. These annotations take the form of a report of a trip and provide information about dynamic aspects of the route, e.g., snow conditions, presence of dangers, etc. Annotations are collected in the ski trip annotation list and made available to all members of the community.

The typical use case for the services offered by a similar site is as follows. A ski mountaineer browses the ski route catalog to identify a number of candidate routes for her next trip. Her choice among all possible destinations is guided by her reading of other users' annotations. For example, she might decide not take a trip if another ski mountaineer signaled in his trip annotation the danger of avalanches on that route.

In the scenario described so far, ski mountaineers have access only to the annotation list provided by their own community. However, many ski mountaineering communities exist and their catalogs present large overlapping sections, i.e., the same routes are recorded on multiple sites. Therefore, annotations about trips done along the same routes are produced in different communities. Nevertheless, it is important to maximize the sharing of past trip experiences in order to achieve better trip planning and safer ski trips. Thus, there is a need for inter-community annotations exchanging and sharing.

Aggregation services are designed to solve this problem. Essentially, an aggregation service collects annotations from different sources and builds a reverse mapping between annotations and annotated items. That is, given a certain item, it allows to access all known annotations referring to it.

It is clear that to make this approach effective the aggregation service must have a way to understand that different annotations, generally originating from different sources, refer to the same item. In other words, items must be globally identified. As an example, let consider the case of an aggregator serving a certain number of readers communities, where annotations consist of comments about books. In this domain, the ISBN number represents a global identifier of a book. Therefore, as long as every annotation uses the ISBN number to identify the book it refers to, the aggregator can easily associate a book to its comments.

Unfortunately, a unique global identifier is not available for most domains. In particular, not only there is no global reference catalog for ski routes, but also an effort in this direction is not foreseeable in the future. What is needed then, in order to make possible the sharing and aggregation of ski trip annotations, is a method to dynamically build a common ski route catalog that does not require a standardization or agreement effort.

The model based on the advertising game technique and the service oriented architecture that we illustrated in previous sections satisfy this need. In particular, the advertising game technique can be leveraged to originate a common reference system for ski routes without requiring any kind of ex-ante agreement among different ski mountaineering systems.

This common reference system allows single communities to preserve their autonomy in the design of ski route schemas, while, at the same time, permits ski mountaineers to effectively access trip annotations independently from the specific catalog they refer to.

Let examine how the advertising model maps to the ski mountaineering domain. Ski mountaineering web sites play the role of peers. Ski routes map to concepts. They are private to each peer, in the sense that a peer is free to model a route adopting the schema it prefers. As a consequence, generally, different sites represent the same routes in different ways. The role of objects is played by concrete representations of ski route models. A common choice to represent a ski route is to provide an XML linearization of the information available for the route. For example, Figure 4 shows such linearization for the same route as modeled by

two different ski mountaineering sites.

```
<item>
  <id>5947</id>
  <top>Monte Cevedale (Zufallspitze)</top>
  <region>Ortles</region>
  <title>Dalla Vedretta di Solda</title>
  <global_difficulty>PD+</global_difficulty>
  <ski_difficulty>S3</ski_difficulty>
  <base_height>2600</base_height>
  <top_height>3757</top_height>
  <gap>1300</gap>
  <exposure>NW</exposure>
</item>

<route>
  <trip_id>2109</trip_id>
  <end_p>Cevedale (Monte)</end_p>
  <start_p>da Solda</start_p>
  <area>Alto Adige</area>
  <district>null</district>
  <valley>Valle di Solda - Suden tal</valley>
  <difficolty>BSA</difficolty>
  <exposure>N</exposure>
  <start_h>2610</start_h>
  <end_h>3769</end_h>
  <gap>1159</gap>
  <start_place>Solda, Funivia di Solda</start_place>
</route>
```

Figure 4: Representation of the same route on two different ski mountaineering web sites.

A critical step in the advertising game is represented by the assessment task. It is in charge of evaluating whether two route linearizations represent the same ski route, possibly modeled using different schemas. At the moment the assessment task is performed using a bipartite matching algorithm. The linearizations are divided in tokens and schema information is dropped. This leaves with two sets of tokens, in our example {*5947, Monte Cevedale (Zufallspitze), Ortles, Dalla Vedretta di Solda, PD+, S3, 2600, 3757, 1300, NW*} and {*2109, Cevedale (Monte), da Solda, Alto Adige, null, Valle di Solda - Suden tal, BSA, N, 2610, 3769, 1159, Solda, Funivia di Solda*}. A bipartite matching algorithm is then used, given a distance function, to find the optimal matching of tokens. The assessment ends successfully if the evaluation of the matching is above a given threshold. More experimentation is underway to improve and tune the algorithm.

It should be clear, then, that the advertising game is completely independent from ski route schemas and dependent only on route information. The underlying assumption is that while the modeling of a ski route can be done in many different ways, i.e., there is high variance on schema models, the information that describes a route is rather homogeneous.

5. CONCLUSIONS AND FUTURE WORK

The paper focuses the attention on a recent approach to advertising based on the notion of advertising games. After a brief summary of the basic concepts of advertising games, we have introduced a service oriented architecture to support a real distributed implementation of such a model. We argued how the design choices are compliant with the requirements of distributed systems. More in detail, we aimed to minimize the global assumption enabling autonomous local design choices.

The next step will be concerned with the evaluation of the architecture in a real world setting that has been recently deployed in the domain of ski mountaineering. We are currently testing it in a scenario that involves three web sites: www.moleskiing.it, www.skirando.ch and www.gulliver.it.

From the technological point of view we are particularly interested in assessing whether BPEL4WS is effective in developing flexible alternative strategies for evolutionary advertising games.

6. ACKNOWLEDGEMENTS

We would like to thank Paolo Busetta for his suggestion on BPEL4WS as a powerful tool for game strategy encoding. Alessandro Agostini and Paolo Busetta helped us to revise this paper.

This work was partially funded by the DiCA project, thanks to a grant of INRM (Istituto Nazionale Ricerca Montagna) and PAT (Provincia Autonoma di Trento).

7. REFERENCES

[1] A. Agostini and P. Avesani. Advertising games for web services. In R. Meersman, Z. Tari, and D. Schmit, editors, *Eleventh International Conference on Cooperative Information Systems (CoopIS-03)*, Berlin Heidelberg, 2003. Springer-Verlag LNCS.

[2] P. Avesani and A. Agostini. A peer-to-peer advertising game. In M. Orlowksa, M. Papazoglou, S. Weerawarana, and J. Yang, editors, *First International Conference on Service Oriented Computing (ICSOC-03)*, pages 28–42, Berlin Heidelberg, 2003. Springer-Verlag LNCS 2910.

[3] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[4] P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In *Second International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 130–145. Springer Verlag, September 2003.

[5] J. Broekstra, M. C. A. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the web by extending RDF schema. In *World Wide Web*, pages 467–478, 2001.

[6] M. Cova, A. Sharma, and R. Tiella. Specification of a service oriented architecture for advertising games. Technical Report T04-06-01, ITC-irst, June 2004.

[7] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

[8] M. J. Duftler, N. K. Mukhi, A. Slominski, and S. W. ana. Web Services Invocation Framework (WSIF). In *OOPSLA 2001 Workshop on Object-Oriented Web Services*, 2001.

[9] J. Hendler and D. McGuinness. Darpa agent markup language. *IEEE Intelligent Systems*, 15(6), 2000.

[10] N. Kavantzas, D. Burdett, and G. Ritzinger. Web services choreography description language version 1.0. `http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/`, 27 April 2004. W3C Working Draft.

[11] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[12] B. Magnini, L. Serafini, and M. Speranza. Linguistic based matching of local ontologies. In *Workshop on Meaning Negotiation (MeaN-02)*, Edmonton, Alberta, Canada, July 2002.

[13] A. McIntyre. Babel: A testbed for research in origins of language. In *Proceedings of COLING-ACL 98*, Montreal, 1998. ACL.

[14] L. Steels. Grounding symbols through evolutionary language games. In A. Cangelosi and D. Parisi, editors, *Simulating the evolution of language*, pages 211–226. Springer Verlag, London, 2001.

[15] L. Steels and F. Kaplan. Bootstrapping grounded word semantics. In T. Briscoe, editor, *Linguistic evolution through language acquisition: formal and computational models*, chapter 3, pages 53–73. Cambridge University Press, Cambridge, 2002.

[16] L. Steels and A. McIntyre. Spatially distributed naming games. *Advances in complex systems*, 1(4), January 1999.