SUMMARY

The Web and application systems like BroadVision One-To-One™ remove the technical barriers to direct consumer interaction and for the first time empowers business managers to utilize the Internet as a two-way medium for brand promotion, content distribution and transactions. By deploying innovative one-to-one marketing principles online, businesses will redefine the concept of niche markets and ultimately be capable of targeting single individuals and their unique needs. This will enable businesses to extend the highest levels of personalized attention and service to all of their customers and to realize new opportunities to serve customers and generate profits.

SERVICE TRADING IN ELECTRONIC MARKETS

BY KURT GEIHS AND REZA FARSI, GOETHE-UNIVERSITY FRANKFURT AM MAIN, GERMANY*

INTRODUCTION

Electronic markets are characterized by a wide variety of service types and service providers. The service supply is subject to permanent changes. One of the basic problems that users, i.e. service clients, must deal with in such an environment is finding a suitable service provider for a certain kind of desired service. Generally, a service requester (i.e. client) can only have a restricted view on the offered services due to the dynamic fluctuations of the market. These fluctuations are caused by, e.g. service providers joining and leaving the market, and offering the same type of service by different, competing providers.

The World Wide Web (WWW) of today may be considered as an early example for such a dynamic electronic market. It is impossible to have a complete overview of the services that are provided in the WWW. Nobody can determine such a knowledge since the WWW is evolving in an inherently decentralized fashion. Therefore, it is a requirement that clients and servers may establish a service binding "just in time" before the service usage. Such a "just in time"-binding is also called "late binding", as opposed to a static binding at the compile-time of the application.

The client-server model in such complex distributed systems needs to be expanded to a three-parties-model, namely the so-called *client-server-mediator* model. The mediator is a matchmaker that matches clients' service demand with the available service supply. A trader is an instance which facilitates the advertising and discovery of services in an open distributed environment. Such a mediation component is called *service trader,* and is the target of several ongoing standardization efforts (ISO13235, Object Management Group 1996). A trader facilitates the advertising and discovery of services in an open distributed environment. It enables clients to deal with the dynamic characteristic of the market.

This paper presents a novel approach to service trading based on advanced knowledge representation techniques. First we describe briefly the role of a trader in an open distributed system and the conventional techniques for service specifications. Then we present our approach for knowledge-based trading that provides more expressiveness and more flexibility for service specifications and service matching.

SERVICE TRADING

The variety of offered services makes it impossible for a service requester to have a global view on the market. The *trader* has a partially global view on the market and the offered services. It interacts with the service providers, service requesters and other traders. Primarily, it offers at its interface two operations, *export* and *import.* A *service provider* advertises its service offer by an export operation. An export operation includes at least the specification of the service type and a service interface identifier which is used by clients to access the service provider. Optionally, the export may contain additional information on other service attributes. The trader maintains a service repository in which it stores the available service exports. A *service requester* calls the import operation of the trader. It specifies the desired service and optionally required service attribute values. Figure 1 shows the participants of an electronic market.

SERVICE SPECIFICATION TECHNIQUES

There are two common techniques to specify service types. The first one is based on syntactical information on the programmatic interface of the service. It uses a programming language independent Interface Definition Language (IDL). This technique has been used in distributed system platforms such as CORBA (Object Management Group 1995) and DCE (Opens Software Foundation 1991) and was carried over into the service trading standards proposed by ISO (ISO13235) and OMG (Object Management Group 1996). The second technique is based on plain text descriptions and text comparison. The well-known Internet search engines basically compare keywords and substrings provided by the search client. Both techniques have their disadvantages and limitations.

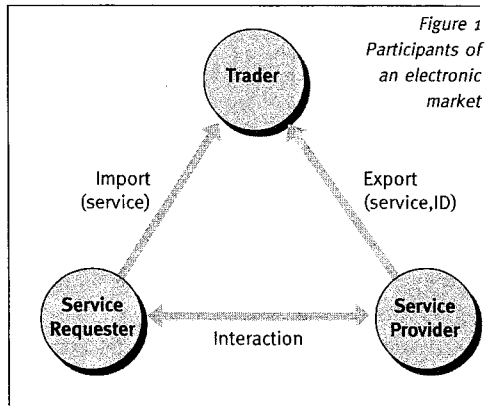

Figure 1 Participants of an electronic market

IDL-based service specification

Formulating a service in IDL only maintains the signature of the operations that the service offers, i.e. it defines the operational interface. Knowing the IDL of an object, a service requester can call the specified operations and provide the required parameters. Because of its static nature, an IDL-based service specification is not flexible enough. Services must be known at compile time. Thus, services must be standardized, which contradicts the idea of an *open* market. New service types have to be propagated to all participants of the market. An IDL is a low-level formulation of a service. It is therefore not suitable for application users, since it does not provide suitable information in order to comprehend the semantics of a service.

Service specification based on text comparison

This is the common technique used by Internet search engines. The engines extract keywords out of documents offered in the WWW. This is done by robots "crawling" through the web, indexing the pages, and storing the indexed information together with the location of the document (URL) in their service data base. A service requester who searches for a particular document sends a query to a search engine. The query typically contains several keywords, that maybe connected by simple boolean expressions.

This technique has some disadvantages. The service specification based on string comparison can be used only for text documents. The multimedia services and Java applets that occur in the electronic markets are not formulated by strings. This technique only deals with the documents on a syntactical basis only. Semantical relationships between concepts are not exploited.

AI-TRADER

With the emergence of service markets where service users deal with service types in order to look up service providers, more adequate and more powerful mechanisms for service specification are needed in order to avoid the above mentioned disadvantages of IDL-based and text-oriented methods. Our approach is based on techniques which are derived from the field of machine learning. We have decided to use a knowledge representation method called *conceptual graphs*. This technique has been built into a service trader that is called AI-Trader.

Puder, A. and Geihs, K. "Meta-Level Service Type Specifications", Proceedings of the International Conference on Open Distributed Processing 1997 (ICODP'97), Toronto, Canada, Chapman and Hall, 1997.

** Kurt Geihs (geihs@vsb.informatik.uni-frankfurt.de) is professor for computer science at the University of Frankfurt a.M., Germany. His research and teaching focuses on distributed systems and operating systems.*

Reza Farsi (farsi@vsb.informatik.uni-frankfurt.de) is a PhD student working on the AI Trader project.

Both authors gratefully acknowledge the contributions of Arno Puder and Kay Römer.

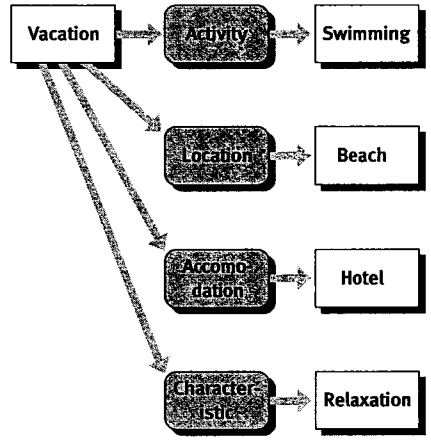

Figure 2
Conceptual representation of a vacation

Originally, conceptual graphs have been developed to model the semantics of natural language. From a formal point of view a conceptual graph is a finite, connected, directed, bipartite graph. The nodes of the graph are either *concept nodes* or *relation nodes*. Two concepts may be connected by a relation.

A concept node may represent concrete objects (e.g. hotel) with one or more specific instances as well as abstract concepts with no physical representation (e.g. relaxation). Whereas concepts model objects of our perception, a relation node expresses a specific relationship between concept nodes. Figure 2 shows a conceptual graph that describes a vacation service offer. Obviously, the conceptual graph representation is quite intuitive and readable. It is close to the cognitive domain of the users, contrary to e.g. low-level IDL-based service specifications that were made for application programmers.

Conceptual graphs can be represented in graphical and in textual form (*linear notation*). In the linear notation a concept node is surrounded by square brackets and relations by round brackets. The simplest conceptual graph in the linear notation would be [Something]. A more complex would have the form [C1]->(R)->[C2]. The linear notation facilitates the machine processing of conceptual graphs.

The root concept of the conceptual graph denotes the service. If this graph is regarded as a service which is offered by some provider, then the conceptual graph shown in Figure 3 represents a query which matches the previous service description.

With our AI-Trader service exports and imports are specified using conceptual graph specifications for services. In an export a provider would specify its service providing a conceptual graph that describes his viewpoint of the service offer. Clients would independently formulate their service requirements in form of a conceptual graph and send it to the trader

Figure 3
A query graph

in the import operation. The trader has to match imports with available exports by comparing conceptual graphs that may be rather different.

How can one decide whether a given graph matches another one? Finding the answer to this question consists of two steps. The first step is comparing their root concepts. If they do not match, the compare operation is finished and the graph is rejected. If they match, a recursive process will be started to see *how well* they match.

Another question must be answered: When do two nodes match? The matching is decided using a set of Prolog rules that was given to the trader in order to control its operation. This rule base for matching can be adjusted flexibly to different application domains. It enables a partial, i.e. *fuzzy* matching. The following example explains this fact.

The integration of a dictionary enables the AI-Trader to use concepts that are a specialization, synonym, antonym etc. of a given concept. Two nodes match if there is a chain of "is a" and "synonym" relations between them. The following example demonstrates how two graphs are matched and how their similarity is determined.

Let TG be a previously exported service offer and QG a query graph.

TG: [vacation] -
-› (activity) -› [swimming],
-› (location) -› [beach],
-› (accommodation) -› [hotel],
-› (characteristic) -› [relaxation].

QG: [something] -
-› (location) -› [city],
-› (accommodation) -› [hotel],
-› (characteristic) -› [relaxation].

The common subgraphs are computed by mutually eliminating concept and relation nodes which are not contained in the other graph.

[something] -
-› (accommodation) -› [hotel],
-› (characteristic) -› [relaxation].

The similarity of the QG and TG can be computed comparing their sizes in terms of the number of common nodes. Employing this method, the example shows that there is a 66% match between TG and QG.

PROTOTYPE

In the Distributed Systems and Operating Systems research group, Computer Science Department, Goethe-University Frankfurt am Main, we have developed a prototype of a knowledge-based trading system over the last three years. The AI-Trader is completely implemented in C++.

The overall architecture of the AI-Trader is shown in Figure 4. It consists of three main components: the *service data base* storing conceptual graphs for services, the *matching rule base* for matching the graphs, and the *lexicographical data base* for handling concept relations such as synonym, specialisation, etc.

The lexicographical database is used to find out relations between compared nodes. It is based on the freely available dictionary WordNet and contains over 80000 nouns and 90000 relations.

The AI-Trader complies to the above mentioned requirements. It uses conceptual graphs as the service specification language. The graphical user interfaces written in Tcl/Tk and Java allows users of the trader (providers and requesters) to specify their services simply by conceptual graphs using point-and-click techniques.

The separate Prolog engine allows the specification of matching rules for conceptual graphs. It supports fuzzy and weighted matching of graphs.

The AI-Trader is available for Sun Solaris, Linux, HP-UX and AIX. A demo of the AI Trader is available via the WWW (adress given at the end of the paper). In this demo a Java applet is provided that allows to build conceptual graphs and to test the trading interactions.

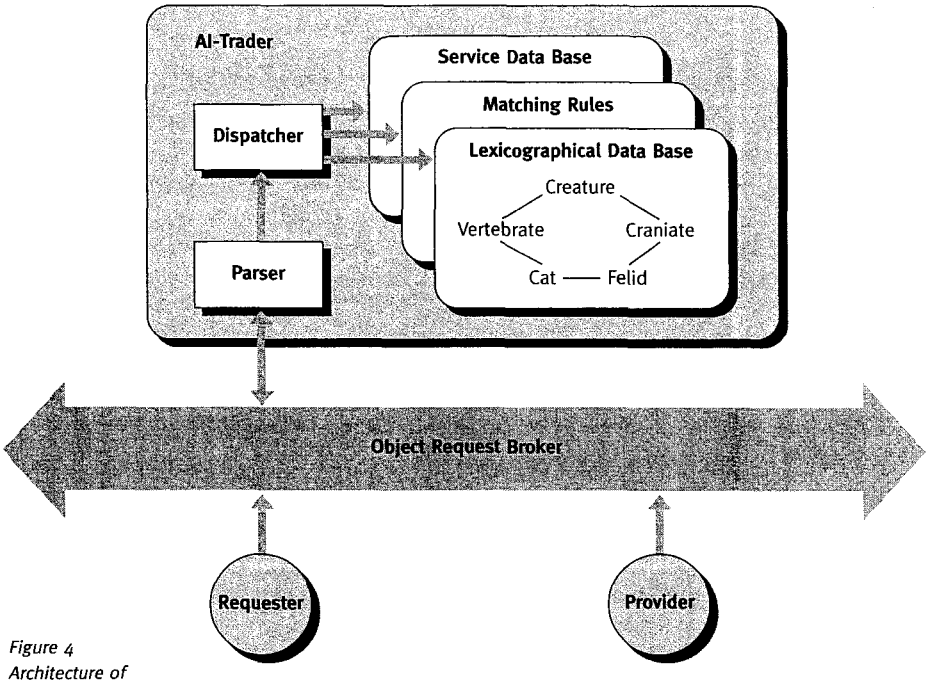

Figure 4
Architecture of AI-Trader

More information on the approach can be found in the literature, e.g. in Puder and Geihs (1997).

OUTLOOK

The demonstration of the AI-Trader at the CeBIT '97 in Hannover successfully showed the potential of our approach. We presented a travel office scenario where providers and clients can specify their vacation offers and demands, respectively, in a rather unrestricted way. The power and elegance of the approach were commonly acknowledged.

REFERENCES

Object Management Group "The Common Object Request Broker: Architecture and Specification", Version 2.0, 1995.

Object Management Group "Trading Object Service", Version 1.0.0, 1996.

Open Software Foundation "Introduction to OSF DCE", Cambridge, USA, 1991.

The AI-Trader can be used as a rather general search tool for all kinds of application scenarios. Specialised traders can be built for particular market sectors by adjusting the matching rule base and perhaps the dictionary. The collaboration of such specialised traders would enable access to even larger amounts of service offers. We are currently working on such enhancements.

For more information please take a look at our WWW pages: *http://www.vsb.cs.uni-frankfurt.de/projects/aitrader*

TOWARDS A METHODICAL DEVELOPMENT OF ELECTRONIC CATALOGUES

BY NORA KOCH, UNIVERSITY OF MUNICH, GERMANY*, AND ANDREAS TURK, FORWISS, GERMANY**

INTRODUCTION

The technologies currently used to develop and deliver multimedia systems like electronic catalogues are still far from being elaborate and easy to apply. Due to the distinct difficulties arising during development, production, and maintenance of sophisticated multimedia software, it is necessary to get the job done by a multi-disciplinary team of programmers, graphic designers, media-experts, and quality control specialists. The results of their work could be significantly improved by easy-to-use tools which achieve the following goals:

- help to determine the requirements,
- reduce the design efforts,
- increment the quality testing speed,
- reduce the costs of producing and updating multimedia systems, and
- simplify the system maintenance.

Electronic product catalogues (EPCs) are information systems, that put emphasis on the multimedia presentation of products (or services) and which contain some standard functionality of searching, selection, and ordering of products. In this paper we will concentrate on this particular sort of information systems. However, we expect most of the presented topics to be useful for the development of information systems in general.

We present some of the results of the EPK-fix project, which deal with the systematic construction of EPCs. Within the scope of this project a methodology of EPC-engineering has been developed and an an integrated set of powerful software tools has been designed and implemented to cover the whole life cycle of electronic product catalogues of CD-ROM. A SGML-based specification language permits a declarative description of catalogues, enabling the tools to support the requirements analysis and the specification of EPCs. The essential EPC-functionality is provided in advance as predefined services. An exhaustive automated validation of the resulting catalogue becomes feasible.

The project EPK-fix is supported by the german Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF). The partners are the Bavarian Research Center for Knowledge-Based Systems (FORWISS) in Erlangen, the Ludwig-Maximilians-University of Munich, the Technical University of Darmstadt, the Technical University of Dresden, and the mediatec GmbH in Nuremberg.

In the first section we describe the software development process for EPCs. The second section presents the architecture of the EPK-fix system including brief descriptions of the specification language EPKML and the four assistance systems. In the last section some conclusions are delineated.

* *Nora Koch (kochn@informatik.uni-muenchen.de) is a research assistant of the Institute of Computer Science at the University of Munich. Her main interests are in the fields of Multimedia and Electronic Publishing with a special focus on user modeling.*

** *Andreas Turk (turk@forwiss.de) works at the Bavarian Research Center for Knowledge-Based Systems on strategies of system development. In the EPK-fix project he focuses on requirements and system analysis for EPCs.*