In: Proceedings Third International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2001), Munich Germany, Sept. 2001, pp. 186-194, LNCS 2114.

Matchmaking for Structured Objects

Thomas Eiter[1], Daniel Veit[2], Jörg P. Müller[3], and Martin Schneider[3]

[1] TU Vienna, Institute of Information Systems, Knowledge Based Systems Group,
Favoritenstrasse 11, A-1040 Vienna, Austria
eiter@kr.tuwien.ac.at
http://www.kr.tuwien.ac.at

[2] Universität Karlsruhe (TH), Information Management and Systems,
Englerstrasse 14, D-76131 Karlsruhe, Germany
veit@iw.uni-karlsruhe.de
http://www.iw.uni-karlsruhe.de

[3] Siemens AG, Corporate Technology, Intelligent Autonomous Systems,
Otto-Hahn-Ring 6, D-81739 Munich, Germany
joerg.mueller, martin.schneider@mchp.siemens.de
http://www.siemens.de

Abstract A fundamental task in multi-agent systems is matchmaking, which is to retrieve and classify service descriptions of agents that (best) match a given service request. Several approaches to matchmaking have been proposed so far, which involve computation of distances between service offers and service requests that are both provided as aggregates of the same set of attributes which have atomic values. In this paper, we consider the problem of matchmaking in the setting where both service offers and requests are described in a richer language, which has complex types built from basic types using constructors such as sets, lists, or record aggregation. We investigate methods for computing distance values of complex objects, based on a generic combination of distance values of the object components, as well as domain-dependent distance functions. The methods have been implemented in GRAPPA, the Generic Request Architecture for Passive Provider Agents, which is a framework for developing open matchmaking facilities that can handle complex objects described in XML. Using GRAPPA, a large scale application has been built in the Human Resource Network project of the Office for Labor Exchange of the German government, in which job offerings have to be matched against a large database of unemployed persons and qualified candidates should be retrieved.

Areas: Emerging trends; data warehouses; systems and applications.

1 Introduction

Today, distributed and heterogeneous information systems which are connected via open networks such as the Internet provide a huge, wide spread wealth of information. The vast amount of information so accessible has created a strong need for powerful methods and techniques that help in ranking the information retrieved in answer to a given query.

In multi-agent systems, this problem instantiates to the fundamental issue of classifying and ranking agents in a system by their service descriptions, given that a particular service is requested. For this task, special kinds of middle agents have been proposed, among them matchmaking and brokering agents (see [12,9,11]), following the mediator approach [17].

The key task in matchmaking is to compute the similarity of a given service request to the service description of a given agent. This is usually done by computing a function measuring the distance between the service request and the service description. Current approaches to matchmaking assume that these descriptions are in a flat format, which essentially is an aggregation of attributes over elementary domains. Distances are computed using well-known methods for computing the distance between values for a single attribute.

However, no methods for matchmaking of complex, structured service descriptions have been provided so far. Such methods are needed, though, for emerging applications that desire services descriptions in a data format which is structurally rich, and, moreover, obeys to some acknowledged standard (e.g., XML) such that multiple, application-independent use of this information is supported. In this paper, we address this issue and investigate methods for matchmaking of service descriptions which are provided as complex objects, built over possibly heterogeneous data types such as text, intervals, or time data, using forms of aggregation such as sets, lists, or records. We pursue a bottom up approach of combining distance values of components of complex objects into a single distance value, which may use generic combination functions as well as customized domain-dependent distance functions.

The main contributions of this paper can be summarized as follows:

- We provide methods for calculating the relevance of a service offer for a requested service, both given as structured complex objects, through distance functions for complex objects which combine distance values of their components. The latter may be complex objects as well, built using common forms of aggregations such as lists, sets, and records. Different from previous matchmaking approaches, ours is not based on a fixed scheme but works for *generic types* of service descriptions. Furthermore, service requests and offered service offers may be of different (yet fixed) type.
- We present the GRAPPA framework, which facilitates the development of matchmaking applications involving complex service and request descriptions. At the generic level, the schemes of the descriptions are stored as *XML document type definitions* (*XML-DTDs*). GRAPPA provides a number of predefined generic functions for combining distance values of description components, and furthermore certain domain-specific distance functions.
- We report on the Human Resource Network (HRNET), which is a large-scale application for employment relaying that has been implemented for the German Office for Labor Exchange on top of GRAPPA. In this application, hiring requests of employers have to be matched against the database of persons in Germany seeking a job (currently, about 3.9 millions), in order to single out best-qualified candidates. Experiments have shown that HRNET performs well, and a full-fledged system is planned for the future.

Our results, and in particular the GRAPPA framework, can be readily applied in building matchmaking facilities for agent systems. However, since the methods have been developed at a generic level, they can be used for matching complex objects against a database of complex objects in general, and in engineering facilities for this task.
In this paper, we focus on the task of matchmaking per se. We pursue an approach in which the generic structure of offers and requests is given by offer and request class descriptions, respectively.

2 Matchmaking Facilities for Complex Objects

We contract here matchmaking to the problem of performing multidimensional distance computation on structured objects, which are composed bottom up from basic and complex values. We next discuss the components which a generic facility for performing this task should have, and after that in Section 2.2 the matchmaking process.

2.1 Components

As discussed in [12,2,9], a matchmaking facility needs certain components. In this spirit, we propose that a matchmaker for complex objects has the following four components: Data Types, Distance Functions, Service Scheme, and Service Repository. They have the following roles.
Data Types: This component includes a kernel set of basic types $\mathcal{B} = \{\tau_1, \ldots, \tau_n\}$ which are supported by the matchmaking facility. This kernel may be extended by customized types in applications. Complex types can be inductively constructed by applying one of the following constructors. Let τ and $\tau_1, \ldots, \tau_k$ be any already defined types: Set of τ, denoted $\{\, \tau \,\}$; Multiset of τ, denoted $\{\, \tau \,\}_m$; List of τ, denoted τ^+; Array of dimension n of τ, denoted $\tau[1 : n]$; Record of $\tau_1, \ldots, \tau_k$, denoted $(\tau_1, \ldots, \tau_k)$.

Each basic type τ has an associated domain $D(\tau)$ of values. For a complex type τ, its domain of values $D(\tau)$ is defined by recursion to subtypes as usual, where sets, multisets, and lists are defined as finite aggregations. Note that different types may have overlapping or even the same sets of values; thus, subranges and synonym types may be defined.
Distance Functions: A *distance function* on a domain $D(\tau)$ is a map $d : D(\tau) \times D(\tau) \rightarrow \mathbb{R}_0^+$ which assigns each pair (x, y) of values for τ a unique nonnegative real number. It is desired that d enjoys certain properties, which guarantees a meaningful behavior. The following are some well-known axioms for distance functions:

(i) $d(x, y) = 0 \iff x = y$ for all $x, y \in D(\tau)$. (Zero-Distance)

(ii) $d(x, y) = d(y, x)$ for all $x, y \in D(\tau)$. (Symmetry)

(iii) $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in D(\tau)$. (Triangle Inequality)

Any d which satisfies (i)–(iii) is a *metric distance function*. Its properties, in particular (iii), may be exploited for pruning the search space in matchmaking. However, not all meaningful distance functions in practice are metric. We postulate, though, that each distance function must satisfy (ii) and the if-direction of (i) (i.e., $d(x, x) = 0$ for all $x \in D(\tau)$).

The matchmaking facility must support for each type τ at least one distance function d. In particular, it must contain at least one *atomic distance function* for each basic type. Besides tailored distance functions, a complex type τ may have, as in GRAPPA, generic distance functions $d = f(d_1, \dots, d_m)$ which combine, by functions f, the distance values at the top-level components $\tau_1, \dots, \tau_m$ of τ, computed using respective distance functions $d_1, \dots, d_m$, into a single distance value. For example, in case of a record $\tau = (\tau_1, \tau_2)$ the function f may be the average of the distances at the components τ_1 and τ_2 (see [14] for further discussion).

Service Description Scheme (SDS): This component contains generic descriptions of the service offers and requests, given as types, that the matchmaking facility can handle. Particular service descriptions are instances of these generic descriptions (i.e., complex values of the types). The **SDS** consists of the following three parts:

1. OSDS: A scheme (type) for the definition of a service offer.
2. RSDS: A scheme (type) for the definition of a service request.
3. MAP: A mapping which assigns each component $\mathtt{R_i}$ of the scheme $\mathtt{RSDS} = (\mathtt{R_1}, \dots, \mathtt{R_m})$ a function
$$\mathtt{MAP}(\mathtt{R_i}) : D(\mathtt{OSDS}) \rightarrow D(\mathtt{R_i}),$$
such that $\mathtt{MAP}(\mathtt{R_i}) = f_{\mathtt{R_i}}(\mathtt{S_1}, \dots, \mathtt{S_k})$, where $f_{\mathtt{R_i}}$ is a function on functions and $\mathtt{S_1}, \dots, \mathtt{S_k}$ are subschemes of OSDS, viewed as deconstructor functions on the instances of OSDS. Informally, $\mathtt{MAP}(\mathtt{R_i})(o)$ constructs for the service request component $\mathtt{R_i}$ a value, given any service offer o. We thus can construct a request object $\mathtt{MAP}(o) = (\mathtt{MAP}(\mathtt{R_1})(o), \dots, \mathtt{MAP}(\mathtt{R_m})(o))$ from o, which can be used for computing the distance between o and a given request object r. Any non-record type RSDS is viewed as record type (RSDS), and MAP defined for it this way.

Notice that in current matchmaking systems, OSDS and RSDS coincide, and MAP is identity, i.e., $\mathtt{MAP}(\mathtt{R_i}) = \mathtt{R_i}$. There, MAP is identity for several components (e.g., MAP(Workstatus) = Workstatus, where we use Workstatus to name the component of this type), while it assigns to $\mathtt{Profession}^+$ the list obtained by concatenating DesiredJob and $\mathtt{Job}^+$, i.e., $\mathtt{MAP}(\mathtt{Profession}^+) = \mathtt{DesiredJob@Job}^+$, where "@" denotes concatenation of lists (as previously, components are named here by their types).

Special cases are that RSDS is a subscheme of OSDS and vice versa. Here MAP is straightforward: in the former case, it projects out components of OSDS, while in the latter, $f_{\mathtt{R_i}}(\mathtt{S_1}, \dots, \mathtt{S_k})$ may add missing attributes to a service offer and assign dummy values to them. In practice, $f_{\mathtt{R_i}}$ may perform various complex operations such as merging lists, taking the union of sets, combining values into a complex value (e.g., assemble dates) etc.

Service Repository: The matchmaking facility maintains an up-to-date repository of service descriptions for all services offers which are advertised to it.

2.2 Matchmaking process

When the matchmaker receives a *query*, which consists of an instance r of $\mathtt{RSDS}$, and an (optional) query requirement (best match, k-nearest neighbors, etc), then it computes the answer to the query and sends the result back to the querying requester agent. Basically, the matchmaker must compute the distance between r and each service offer o in the service repository, and then select those o which qualify for the answer. The distance between r and o is measured by $d(r, \mathtt{MAP}(o))$, where d is the distance function for $\mathtt{RSDS}$ and $\mathtt{MAP}(o)$ is the conversion of o into the request object.

This process can be implemented in many ways. For details concerning this issue see [14,5,15].

2.3 The GRAPPA Matchmaking Framework

GRAPPA, the **G**eneric **R**equest **A**rchitecture for **P**assive **P**rovider **A**gents, is a generic framework that instantiates the general matchmaking facility for complex objects described in the previous section. It is designed for computing k-nearest-neighbor matchings of multidimensional requests against multidimensional offers.
Generic algorithms to incorporate Data Types, Distance Functions and Service Description Schemes are implemented. The main Data Types considered are Numbers, Intervals, Time and Time Intervals as well as Free Text. For the latter one distance functions from the Information Retrieval domain (see [10]) are implemented. For further details see [14,5].
As complex distance functions Minimum Link Distance (see [4]), Hausdorff Distance as well as a Weighted Average Distance are incorporated.

Both the $\mathtt{OSDS}$ and the $\mathtt{RSDS}$ are mapped to XML-DTDs. Service offers and requests, respectively, are instantiated XML documents. An example is shown in Figure 1. A requester can query the **ServiceRepository**, which contains XML documents instantiating $\mathtt{OSDS}$, by sending an XML document which instantiates the $\mathtt{RSDS}$ to the GRAPPA matchmaker.

3 Application

Because of its genericity, our approach and the GRAPPA framework is not restricted to agent systems and can be applied in different domains.

This is exemplified by two projects in which GRAPPA has been applied so far: The Human Resource Network (HRNET), a large-scale application for the mediation of jobs described in Section 3.1, and the Cooperation Market (COMA) of the Siemens AG. In the following we will only report on the HRNET project.

3.1 HRNET for the German Office for Labor Exchange

The Human Resource Network (HRNET) is an application of GRAPPA for matching open jobs in companies, which are defined by an employer, to profiles of job applicants (i.e., unemployed persons), stored in various data bases. The current version of HRNET

XML-DTD for RSDS:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT RSDS (Profession*, Experience, CarRequired,
                Location, Requirements, WorkStatus,
                WorkMode, WorkType, Salary)>
<!ELEMENT Experience (ExperienceLevel, Profession*)>
<!ELEMENT Location (ZipCode, Town, Country)>
<!ELEMENT Requirements (GeneralRequirement,
          SpecialRequirement, Language)>
<!ELEMENT GeneralRequirement (Description, Level)>
<!ELEMENT SpecialRequirements (Description, Level)>
<!ELEMENT Languages (Description, Level)>
    <!ELEMENT DesiredJob (\#PCDATA)>
    <!ELEMENT Description (\#PCDATA)>
    <!ELEMENT Duration (\#PCDATA)>
    <!ELEMENT Profession (\#PCDATA)>
    <!ELEMENT Car (\#PCDATA)>
    <!ELEMENT ZipCode (\#PCDATA)>
    <!ELEMENT Town (\#PCDATA)>
    <!ELEMENT Country (\#PCDATA)>
    <!ELEMENT RegionalPreference (\#PCDATA)>
    <!ELEMENT Level (\#PCDATA)>
    <!ELEMENT WorkStatus (\#PCDATA)>
    <!ELEMENT WorkMode (\#PCDATA)>
    <!ELEMENT WorkType (\#PCDATA)>
    <!ELEMENT Salary (\#PCDATA)>
    <!ELEMENT ExperienceLevel (\#PCDATA)>
    <!ELEMENT CarRequired (\#PCDATA)>
```

XML instance for computer scientist:

```
<RSDS>
<Profession>Computer Scientist</profession>
<Profession>Mathematician</profession>
<Experience>
  <ExperienceLevel>expert</ExperienceLevel>
  <Profession>Computer Scientist</profession>
  <Profession>Mathematician</profession>
</Experience>
<CarRequired>false</CarRequired>
<Location>
  <ZipCode>81541</ZipCode>
  <Town>Munich</Town>
  <Country>Germany</Country>
</Location>
<Requirements>
  <GeneralRequirement>
    <Description>soft skills</Description>
    <Level>very good</level>
  </GeneralRequirement>
 ...
</Requirements>
<Description>We are looking for...
</Description>
<WorkStatus>employed</WorkStatus>
<WorkMode>in office</WorkMode>
<WorkType>fulltime</WorkType>
<Salary>40.000 Euro</Salary>
</RSDS>
```

Table 1. RSDS of HRNET as XML-DTD and an instance (computer scientist)

is a prototype system that has been developed for the Office for Labor Exchange of the German government, and demonstrates the feasibility of a partially automated approach to employment relaying. Based on its success, a full-fledged system is planned for the near future. Note that it promises a high return of investment: reducing the relaying time of unemployed persons (currently, about 3.9 millions) just by one day on average will save the German government more than a hundred million dollars a year.

In the HRNET system architecture each company supplies its open positions to a designated GUI-Agent, which has the role of a requester agent in the system. The GUI-Agent queries the matchmaker by sending to HRNET the description of the open position which should be filled.

The service repository of HRNET consists of a collection of data sources. Most of them are databases wrapped by a database wrapper agent. One of them is the central database of the Office for Labor Exchange of the German government, in which all currently unemployed persons in Germany are stored. Another one we used is the Siemens AG internal employee database. Further databases can be easily integrated.

If the number of applicants exceeds a certain limit in a database, the database wrapper agents supplies only a *preselection* of profiles to the matchmaker. This preselection eliminates all profiles which do not match any of the values in $\texttt{Profession}^+$ of a given job offer (RSDS instance). In HRNET, the RSDS and OSDS schemes are, as required by GRAPPA, converted to XML-DTDs which are considered as the document classes of these types. The XML-DTD of RSDS, together with an instance, is shown in Table 1.

Besides the generic basic types and distance functions, HRNET uses customized basic types including RegionalPreference, Level and WorkStatus. The default distance functions for these types are defined using distance matrices and exact match-

job offering (request) / quality of k-th best match	$k =$ 1	5	10	20	50	100
r_1 : Computer scientist	61 %	45 %	39 %	33 %	23 %	13 %
r_2 : Bus driver	72 %	68 %	61 %	52 %	43 %	30 %
r_3 : Anesthesia male nurse	40 %	23 %	14 %	12 %	10 %	8 %
r_4 : Clerk	36 %	35 %	31 %	26 %	22 %	12 %
r_5 : Truck driver	79 %	70 %	65 %	60 %	55 %	40 %
r_6 : Haircutter	48 %	46 %	39 %	36 %	25 %	12 %
r_7 : Taxi driver	81 %	79 %	75 %	60 %	43 %	20 %
r_8 : Interpreter	52 %	47 %	38 %	34 %	28 %	12 %
r_9 : Children nurse	68 %	67 %	57 %	51 %	29 %	14 %
r_{10} : Engine fitter	59 %	40 %	36 %	29 %	21 %	15 %

Table 2. HRNET matchmaker results

ing functions. The distances for complex types are computed by using the Hausdorff distance and weighted averages.

3.2 Experiments

In this section, we give a sample of the set of experiments that we have conducted with the HRNET system. It appeared that in these experiments, the HRNET matchmaker performed quite well and was ranking the job applicants (i.e., `OSDS` instances) realistically.

Precison. In the first experiment, we considered ten different job offerings (requests) $r_1, \ldots, r_{10}$, which were supplied to a GUI-Agent for querying the matchmaker. Table 2 shows the results for a k best matches (i.e., nearest neighbors) query, where for each k the quality of the last (worst) among the k matches is reported. The quality is the similarity between the request r_i and the applicant profile (offer) o measured by $1 - d(r_i, o)$, where $d(r_i, o)$ is the (normalized) distance value. Request r_1 is the computer scientist instance of `RSDS` shown in Table 1; the other requests instantiated `RSDS` to jobs in different areas.

It is, of course, difficult to judge the quality of matchings computed by the HRNET matchmaker to the one of a human matchmaker, and in particular whether it computes "human like" rankings. Rankings compiled by a human matchmaker may be subjective, and different human experts may come up with different rankings. However, inspection has shown that among a larger set of profiles, the best candidate singled out by the matchmaker is the same one would manually select.

Recall. In a further experiment, we modified the data repository by adding two further `OSDS` instances that should match the request r_1 intuitively high. One of them was intuitively an exact match (quality 100%), and the other one was a profile which intuitively supported more desired properties than the previous best match, which was 61%. As expected, the exact match was the new best best and had a score of 100%. The second best match was the other addition to the data repository. It obtained a significantly better score (85%) than the previous best match.

4 Related Work

[8] considered matchmaking in the context of emerging information integration technologies, where potential providers and requesters send messages describing their capabilities and needs of information (or goods). They presented two matchmakers: COINS (COmmon INterest Seeker), which is based on free text matchmaking using a distance measure from information retrieval [10], and SHADE (SHared DEpendency Engineering), which uses a subset of KIF [6] and a structured logic text representation called MAX [7]. While COINS aimed at e-commerce, SHADE aimed at the engineering domain.

Complementing the theoretical work in ([2,3]), Sycara and coworkers addressed the matchmaking problem in practice. They developed and implemented the LARKS matchmaker (LAnguage for Advertisement and Request for Knowledge Sharing) described in [13,12]. In LARKS, the matchmaking process runs through three major steps: (1) Context matching, (2) syntactical matching, and (3) semantical matching. Step 2 is divided into a comparison of profiles, a similarity matching, and a signature matching. Compared to previous approaches, LARKS provides higher expressiveness for service descriptions. Like those, however, LARKS has a static scheme for service descriptions, which restricts its application to agents that comply with this fixed description format.

In the context of electronic auctions, [16] introduce a service classification agent which has meta knowledge and access to nested ontologies. This agent dynamically generates unique agent and auction descriptions which classify an agent's services and auction subjects, respectively. A requester obtains from it the name of the best auction to its needs.

In IMPACT [1,11], so called Yellow Pages Servers play the role of matchmaker agents. Offers and requests are described in a simple data structure which represents a service by a verb and one or two nouns (e.g., *sell:car*, *create:plan(flight)*). The matchmaking process computes the similarity of descriptions from shortest paths in directed acyclic graphs that are built over the sets of verbs and nouns, respectively, where edges have weights reflecting their distance.

5 Conclusion

In this paper, we have considered the problem of matchmaking given that service descriptions are complex objects, formulated in a rich language. Furthermore, we have presented various methods for computing distance values between complex objects and the GRAPPA framework, which can be used for building matchmaking facilities. As the HRNET application has shown, GRAPPA is an attractive tool for developing application-specific matchmakers.

Our ongoing and future work comprises several issues. One is to exploit the properties of metric distance functions and to design algorithms which avoid scan the entire service repository. The development of specific distance functions is another issue. Last, but not least, an important issue is to improve the efficiency of the access to the service repository, which currently is a bottleneck of the system. In our future work we also intend to transfer matchmaking algorithms into the multi-attribute auction domain.

An interesting issue is the use of self-trained neural networks in the design of customized distance functions which reflect the judgment of a human expert as close as possible. Hence important future work will be to implement neural networks computing distance functions for information classification via matchmaking. Finally, a full scale implementation of the HRNET prototype and the development of further applications of GRAPPA complement our research.

References

1. K. Arisha, T. Eiter, S. Kraus, F. Ozcan, R. Ross, and V.S Subrahmanian. IMPACT: A Platform for Collaborating Agents. *IEEE Intelligent Systems*, 14(2):64–72, March/April 1999.
2. K. Decker, K. Sycara, and M. Williamson. Matchmaking and brokering. In *International Conference on Multi-Agent Systems (ICMAS96)*, December 1996.
3. K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 578–583, August 1997.
4. T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34:109–133, 1997.
5. T. Eiter, D. Veit, J.P. Müller, and M. Schneider. Matchmaking for Structured Objects. Extended version, manuscript, 2000.
6. M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, June 1992. `http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps`.
7. D. Kuokka. *The Deliberative Integration of Planning, Execution, and Learning*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1990.
8. D. Kuokka and L. Harada. Integrating information via matchmaking. *Journal of Intelligent Information Systems*, 6(2/3):261–279, 1996.
9. M. Nodine, W. Bohrer, and A. H. H. Ngu. Semantic brokering over dynamic heterogenous data sources in InfoSleuth. In *Proceedings of the Fifteenth International Conference on Data Engeneering*, pages 358–365, August 1999.
10. G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.
11. V.S Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Ozcan, and R. Ross. *Heterogenous Agent Systems*. MIT Press, June 2000. (ISBN: 0262194368).
12. K. Sycara, J. Lu, and M. Klusch. Interoperability among heterogenous software agents on the internet. Technical Report CMU-RI-TR-98-22, The Robotics Institute Carnegie Mellon University, Pittsburgh, October 1998.
13. K. Sycara, J. Lu, M. Klusch, and S. Widoff. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOD Record 28 (1), Special Issue on Semantic Interoperability in Global Information Systems*, pages 47–53, 1999.
14. D. Veit. Matchmaking algorithms for autonomous agent systems. Master's thesis, Institute of Computer Science, University of Gießen, Germany, 1999.
15. D. Veit, J.P. Müller, M. Schneider, and B. Fiehn. Spt: Matchmaking for autonomous agents in electronic marketplaces. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.
16. P. Weinstein and W. Birmingham. Service classification in a proto-organic society of agent. In *Proceedings of the IJCAI-97 Workshop on Artificial Intelligence in Digital Libraries*, 1997.
17. G. Wiederhold. Intelligent Integration of Information. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 434–437, Washington, DC, 1993.