Dynamic Service Matchmaking Among Agents in Open Information Environments*

Katia Sycara, Matthias Klusch, Seth Widoff
The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA.
{katia, klusch, swidoff}@cs.cmu.edu

Jianguo Lu
Computer Science Department, University of Toronto, CA.
jglu@cs.toronto.edu

1 Introduction

The amount of services and deployed software agents in the most famous offspring of the Internet, the World Wide Web, is exponentially increasing. In addition, the Internet is an open environment, where information sources, communication links and agents themselves may appear and disappear unpredictably. Thus, an effective, automated search and selection of relevant services or agents is essential for human users and agents as well.

We distinguish three general agent categories in the Cyberspace, *service providers*, *service requester*, and *middle agents*. Service providers provide some type of service, such as finding information, or performing some particular domain specific problem solving. Requester agents need provider agents to perform some service for them. Agents that help locate others are called middle agents[2]. *Matchmaking* is the process of finding an appropriate provider for a requester through a middle agent, and has the following general form: (1) Provider agents advertise their capabilities to middle agents, (2) middle agents store these advertisements, (3) a requester asks some middle agent whether it knows of providers with desired capabilities, and (4) the middle agent matches the request against the stored advertisements and returns the result, a subset of the stored advertisements.

While this process at first glance seems very simple, it is complicated by the fact that not only local information sources but even providers and requesters in the Cyberspace are usually heterogeneous and incapable of understanding each other. This gives rise to the need for a common language for describing the capabilities and requests of software agents in a convenient way. Besides, one has to devise an efficient mechanism to determine a structural and semantic match of descriptions in that language. This means in particular to use methods for reconciling potentially semantic heterogeneous informations [14]. There is an obvious trade-off between the quality and efficiency of matchmaking on the Internet.

In the following, we briefly present the agent capability description language, LARKS, and then discuss the matchmaking process using LARKS. The paper concludes with a brief comparison with related works. We have implemented LARKS and the associated powerful matchmaking process, and are currently incorporating it within our RETSINA multi-agent infrastructure framework [29].

2 The Agent Capability Description Language LARKS

There is an obvious need to describe agent capabilities in a common language before any meaningful service matchmaking or brokering among the agents can take place. Some of the main desired features of such a language are the following:

- **Expressiveness** The language should be expressive enough to represent not only data and knowledge, but also the meaning of program code. Agent capabilities should be described at an abstract rather than implementation level.
- **Inferences**. Inferences on descriptions written in this language should be supported. Au-

*This research has been sponsored in part by Office of Naval Research grant N-00014-96-16-1-1222, and by DARPA grant F-30602-98-2-0138.

tomated reasoning and comparison on the descriptions should be possible and efficient.

- **Ease of Use**. Descriptions should not only be easy to read and understand, but also easy to write by the user. The language should support the use of domain or common ontologies for specifying agents capabilities.

Since none of the existing languages for program description such as Z [21], knowledge representation, like KIF [13], or description formats, like RDF [22], satisfies our requirements, we propose an agent capability description language, called LARKS (**L**anguage for **A**dvertisement and **R**equest for **K**nowledge **S**haring), that enables for advertising, requesting and matching agent capabilities. A specification in LARKS is a frame with the following slot structure.

`Context`	Context of specification
`Types`	Declaration of used variable types
`Input`	Declaration of input variables
`Output`	Declaration of output variables
`InConstraints`	Constraints on input variables
`OutConstraints`	Constraints on output variables
`ConcDescriptions`	Ontological descriptions of used words
`TextDescription`	Textual Description of specification

The frame slot types have the following meaning.

- `Context`: The context of the specification in the local domain of the agent.
- `Types`: Optional definition of the data types used in the specification.
- `Input` and `Output`: Input/output variable declarations for the specification. In addition to the usual type declarations, there may also be concept attachments to disambiguate types of the same name. The concepts itself are defined in the concept description slot `ConcDescriptions`.
- `InConstraints` and `OutConstraints`: Logical constraints on input/output variables that appear in the input/output declaration part. The constraints are described as Horn clauses [1].

[1] In future, we plan to allow for using ISO/IEC 13211-1 standard compliant Prolog programs to describe constraints and functional capabilities.

- `ConcDesriptions`: Optional description of the meaning of words used in the specification. The description relies on concepts in a given local domain ontology. Attachement of a concept C to a word w in any of the slots above is done in the form: w*C. That means that the concept C is the ontological description of the word w. The concept C is included in the slot `ConcDescriptions` if not already sent to the matchmaker.
- `TextDescription`: Optional, textual description of the meaning of the specification as a request for or advertisement of agent capabilities. In addition, the meaning of input and output declaration, type and context part of the specification may be described by attaching textual comments.

2.1 Using Local Domain Ontologies

As mentioned above LARKS offers the option to use application domain knowledge in any advertisement or request. This is done by using a local ontology for describing the meaning of a word in a LARKS specification. An ontology may be defined for example as a simple keyword hierarchy, a more sophisticated summary schema, or meta-data dictionary as well as a complex terminology written in a concept language, the unified modeling language UML, or the knowledge interchange format KIF.

In our implementation of the matchmaking process it is assumed that any local ontology is defined in the concept language ITL [28]. Conceptual knowledge about a given application domain, or even common-sense, is defined by a set of concepts and roles as terms in the given KL-ONE like concept language; each term as a definition of some concept C is a conjunction of logical constraints which are necessary for any object to be an instance of C. The set of terminological definitions forms a terminology. Any canonical definition of concepts relies in particular on a given basic vocabulary of words (primitive components) which are not defined in the terminology, i.e., their semantic is assumed to be known across boundaries.

The main benefits of providing a local ontology written in a concept language are twofold: (1) the user can specify in more detail what's being requested or advertised, and particularly (2) the matchmaker agent is able to make automated inferences on these additional semantic descriptions while matching LARKS specifications, thereby improving the overall quality of the matching process.

The matchmaker determines the relationship among two semantic descriptions written as concepts in ITL by computing the respective concept subsumption relation [27]: A concept C subsumes another concept C' if the extension of C' is a subset of that of C. This means, that the logical constraints defined in the term of the concept C' logically imply those of the more general concept C[2].

We assume that the subsumption relation among two concepts may be identified with a real world semantic relation among them. Like in [25], we utilize an injective, domain-independent mapping among primitive components which occur in the concept definitions based on given synonym relations[3].

The matchmaker computes the subsumption relations among the concepts included in any advertisement he receives from registered provider agents. This yields a (set of) subsumption hierarchies of available concepts from a variety of local domain ontologies. An extension of the partial global ontology of the matchmaker with additional types of relations is presented in section 3.5. Please note, that this ontology is not necessarily the union of all local domain ontologies of providers, and is dynamically built by the matchmaker while processing advertisements from registered provider agents. Any user or agent, requester or provider, may browse through the matchmaker's ontology and use the included concepts for describing the meaning of words in a specification of a request or advertisement in LARKS[4].

2.2 A Simple Example for Specification

Every specification in LARKS can be interpreted as an advertisement as well as a request; the specification's role depends on the agent's purpose for sending it to a matchmaker agent, and it is indicated in the wrapper language by an appropriate performative (advertise or request). Every LARKS specification must be wrapped by the sending agent in an appropriate message that indicates if the message content is to be treated as a request or an advertisement. The following is a simple example for a request and advertisement in LARKS in the air combat mission domain.

[2] Using the well-known tradeoff, we compromise expressiveness of the NP-complete decidable ITL for tractability in our subsumption algorithm, which is correct but incomplete.

[3] For a further discussion on possible loss of semantics due to mapping among multiple different ontologies we refer to e.g. [26].

[4] This is similar to the common use of domain namespaces in XML [32] for semantically tagging Web page contents.

Example 2.1: *A request and advertisement of agent capabilities*

We did apply the matchmaking process using LARKS in the application domain of air combat missions. As an example for specification consider the following request and advertisement, 'ReqAirMissions' and 'AWAC-AirMissions', respectively. The request is to find an agent which is capable to give information on deployed air combat missions launched in a given time interval. Some provider agent in this domain advertises his capability to provide information about a special kind of (AWAC) air combat missions. Both agents use the same domain ontology written in ITL.

ReqAirMissions	
`Context`	Attack, Mission*AirMission
`Types`	Date = (mm: Int, dd: Int, yy: Int), DeployedMission = ListOf(mType: String, mID:String‖Int)
`Input`	sd: Date, ed: Date
`Output`	missions: Mission
`InConstraints`	sd <= ed.
`OutConstraints`	deployed(mID), launchedAfter(mID,sd), launchedBefore(mID,ed).
`ConcDescriptions`	AirMission = (and Mission (atleast 1 has-airplane) (all has-airplane Airplane) (all has-MissionType aset(AWAC,CAP,DCA,HVAA)))
`TextDescription`	capable of providing information on deployed air combat missions launched in a given time interval

AWAC-AirMissions	
`Context`	Combat, Mission*AWAC-AirMission
`Types`	Date = (mm: Int, dd: Int, yy: Int) DeployedMission = ListOf(mt: String, mid:String‖Int, mStart: Date, mEnd: Date)
`Input`	start: Date, end: Date
`Output`	missions: DeployedMission;
`InConstraints`	start <= end.
`OutConstraints`	deployed(mID), mt = AWAC, launchedAfter(mid,mStart), launchedBefore(mID,mEnd).
`ConcDescriptions`	AWAC-AirMission = (and AirMission (atleast 1 has-airplane) (atmost 1 has-airplane) (all has-airplane aset(E-2)))
`TextDescription`	capable of providing information on deployed AWAC air combat missions launched in some given time interval

○

3 The Matchmaking Process Using LARKS

For the matchmaking process we adopt several different methods from the area of information retrieval, AI and software engineering for computing syntactical and semantic similarity among agent capability descriptions. These methods are particularly efficient in terms of performance as needed for dynamical matchmaking in the Internet.

The matching engine of the matchmaker agent contains five different filters: (1) *Context matching*, (2) *Profile comparison*, (3) *Similarity matching*, (4) *Signature matching*, and (5) *Constraint matching*. The computational costs of these filters are in increasing order. Users may select any combination of these filters on demand. For example, when efficiency is the major concern, a user might select only the context and profile filters (similar to most conventional SearchBots in the Internet). We will now briefly describe each filter.

3.1 Context Filter

Any matching of two specifications has to be in an appropriate context. In LARKS to deal with restricting the advertisement matching space to those in the same domain as the request, each specification supplies a list of keywords meant to describe the semantic domain of the service. When comparing two specifications it is assumed that their context or domains are the same (or at least sufficiently similar) as long as (1) the real-valued distances between the roots of considered words do not exceed a given threshold, and (2) subsumption relations between the attached concepts of the pairs of most similar words are the same, or the distance among these concepts does not exceed a threshold.

Word distance is computed using the trigger-pair model [23]. If two words are significantly co-related, then they are considered trigger-pairs, and the value of the co-relation is domain specific. In the current implementation we use the Wall Street Journal corpus of one million word pairs to compute the word distance. Computation of concept distance is discussed in section 3.5.

For example, both specifications 'ReqAirMissions' and 'AWACS-AirMissions' (see example 2.1) pass the context filter as to be in a sufficiently similar context. The most similar word pairs are (Attack, Combat), (Mission,Mission), and the concept AirMission subsumes the concept AWACS-AirMission.

3.2 Profile Filter

Although context matching is most efficient, it does not consider the whole specification itself. This is done with a profile filter that compares two LARKS specifications by using a variant of the known TF-IDF (term frequency-inverse document frequency) technique [24]. According to that, any specification is treated as a document. TF-IDF determines the degree of similarity between two documents based on frequency and relevance of words in a document considering a finite set of documents. (in our case the advertisement database of the matchmaker). If the computed similarity value exceeds a given threshold both documents pass the profile filter. For example, the profiles of both specifications in the example 2.1 are similar with degree 0.65.

3.3 Similarity Filter

The profile filter has two limitations. It does not consider the structure of the description. That means the filter, for example, is not able to differentiate among input and output declarations of a specification. Besides, profile comparison does not rely on the semantics of words themselves. Thus the filter is not able to recognize that the word pair (Computer, Notebook), for example, should have a closer distance than the pair (Computer, Book).

Computation of similarity relies on a combination of distance values as calculated for pairs of input and output declarations, and input and output constraints. Each of these distance values is computed in terms of the distance between concepts and words that occur in their respective specification section. The values are computed at the time of advertisement submittal and stored in the matchmaker database. The similarity among two specifications in LARKS is computed as the average of the sum of similarity computations among all pairs of declarations and constraints. Both specifications in example 2.1 pass the similarity filter with a similarity value of 0.83.

3.4 Signature and Constraint Filters

The similarity filter takes into consideration the semantics of individual words in the description. However, it does not take the meaning of the logical constraints in a LARKS specification into account. This is done in our matchmaking process by the signature and constraint filters. The two filters are designed to work together to look for a so-called semantic plug-in match known in the software engineering area ([11, 33, 8]).

Signature matching checks if the signatures of input and output declarations match. It is performed by a set of subtype inference rules as well as concept subsumption testing (see [28] for details). A software component description D_2 'semantically plug-in matches' another component description D_1 if (1) their signatures match, (2) the set of input constraints of D_1 logically implies that of D_2, and (3) set of output constraints of D_2 logically implies that of D_1. In our implmentation the logical implication among constraints is computed using polynomial θ-subsumption checking for Horn clauses[18].

Plug-in matching of LARKS specifications is valuable for selecting advertisements which are not as constrained in the input parameters as the considered request, but will return equal or greater number of more specific output parameters. For example, the advertisement 'AWAC-AirMission' plugs into the request 'ReqAirMissions' in example 2.1.

3.5 Computation of Distances Among Concepts

For matchmaking, the identification of relations between concepts other than subsumption is very useful because it promotes a deeper semantic understanding. Moreover, since we've restricted the expressiveness of the concept language ITL in order to boost performance, we need some way to express additional associations between concepts. For this purpose we use a weighted associative network (AN), a semantic network with directed edges between concept nodes. The type of edge between two concepts denotes their binary relation, and edges are labeled with a numerical weight (interpreted as a fuzzy number). The weight indicates the strength of belief in that relation, since its real world semantics may vary. The AN is the partial global ontology of the matchmaker dynamically built

In our implementation an associative network is created by the matchmaker by using the computed concept subsumption hierarchy and additional associations extracted from the WordNet ontology[4]. We assume that the terminological subsumption relation among two concepts in the partial global ontology of the matchmaker may be identified with a real world semantical relation among them. Distance between two concepts C, C' in an AN is computed as the strength of the shortest path between C and C' on the basis of triangular norms (see [15] for details). For performance reasons the matchmaker does not deal with dynamically resolving ambiguities due to potential genericity and polysemy in the AN (see e.g. [3]).

4 Related Works

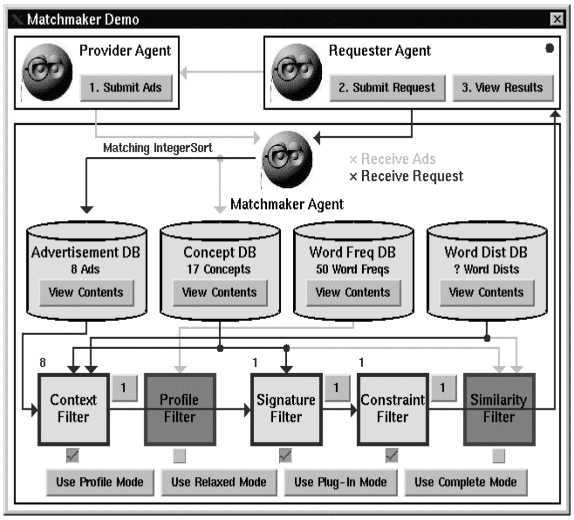
For dealing with semantic heterogeneity among distributed, autonomous information sources there exist solutions in the multidatabase and information systems area for years. Many of them are based on a database-style modeling of data, global schema, and use of meta-information such as provided by a common ontology or different domain ontologies for a content-based source selection [7, 1, 25, 6]. Others focus on information retrieval (IR) techniques for best-match queries, and relevance assessment. Alternative solutions towards an adaptive process for revealing semantic interdependencies among heterogeneous data objects is proposed, e.g., by SCOPES [20].

However, the main problem of dynamic matchmaking in the Internet is to deal with the trade-off between performance and quality of matching. Complex reasoning has to be restricted to allow meaningful semantic matches of requests and advertisements in a reasonable time. Unlike other approaches to matchmaking or brokering in multiagent systems [1, 16, 19], the presented matchmaking process using LARKS offers a flexible approach to satisfy both requirements. It does not deal with a global integration of heterogeneous source descriptions in terms of database schemas, but with comparing descriptions of functional capabilities such as constrained actions to provide services. For this purpose it combines techniques from IR, software engineering and description logics area in an appropriate way to perform such filtering efficiently. The matchmaker agent does not need to perform any complex query activities such as, e.g., by broker agents in InfoSleuth[19] or the mediator agent in SIMS[1]. In addition, we have developed protocols for efficient, distributed matchmaking among multiple matchmaker agents [10].

5 Implementation

We did implement the language LARKS and the matchmaking process using LARKS in Java. The following figure shows the user interface of the matchmaker agent.

To help visualize the matchmaking process, we devised a user interface that traces the path of the



advertisement result set for a request through the matchmaker's filters. The filters can be configured by selecting the checkboxes beneath the desired filters — disabled filters are darkened and bypassed. As the result set passes from one filter to the next, the filter's outline highlights, the number above the filter increments as it considers an advertisement, and the number above its output arrow increments as advertisements successfully pass through the filter. Pushing the buttons above each inter-filter arrow reveals the result advertisement set for the preceding filter.

6 Conclusion

Service matchmaking among heterogeneous software agents in the Internet is usually done dynamically and must be efficient. There is an obvious trade-off between the quality and efficiency of matchmaking on the Internet. We have defined and implemented a language called LARKS for agent advertisements and requests, and a matchmaking process that uses LARKS. LARKS judiciously balances language expressiveness and efficiency in matching. The LARKS matchmaking process performs both syntactic and semantic matching, and in addition allows the specification of concepts (local ontologies) via ITL, a concept language.

The matching process uses five different filters: context matching, profile comparison, similarity matching, signature matching and constraint matching. Different degrees of partial matching can result from utilizing different combinations of these filters. The selection of filters to apply is under the control of the user (or the requester agent). We did apply LARKS and the matchmaking process in several application domains for systems of information agents such as the air combat mission domain. Other application domains are currently under investigation.

References

[1] Y. Arens, C. A. Knoblock and C. Hsu. Query Processing in the SIMS Information Mediator. Austin Tate (Ed.), *Advanced Planning Technology*, AAAI Press, CA, 1996.

[2] K. Decker, K. Sycara, M. Williamson. Middle-Agents for the Internet. Proc. 15th IJCAI, pages 578-583, Nagoya, Japan, August 1997.

[3] P. Fankhauser, E.J. Neuhold. Knowledge based integration of heterogeneous databases. Proceedings of IFIP Conference DS-5 Semantics of Interoperable Database Systems, Lorne, Victoria, Australia, 1992.

[4] C. Fellbaum (Ed.). *WordNet: An Electronic Lexical Database*. MIT Press, 1998. http://www.cogsci.princeton.edu/ wn/

[5] T. Finin, R. Fritzson, D. McKay, R. McEntire. KQML as an Agent Communication Language. Proc. 3rd International Conference on Information and Knowledge Management CIKM-94, ACM Press, 1994.

[6] H. Garcia-Molina, et al.: The TSIMMIS Approach to Mediation: Data Models and Languages. *Proc. Workshop NGITS*, 1995. ftp://db.stanford.edu/pub/garcia/1995/tsimmis-models-languages.ps

[7] M.R. Genesereth, A.M. Keller, and O. Duschka. Infomaster: An Information Integration System. Proceedings of ACM SIGMOD Conference, May 1997.

[8] J. Goguen, D. Nguyen, J. Meseguer, Luqi, D. Zhang, V. Berzins. Software component search. Journal of Systems Integration, 6, pp. 93-134, 1996.

[9] N. Jacobs, and R. Shea. The role of Java in InfoSleuth: Agent-based exploitation of heterogeneous information ressources. Proc. of Intranet-96 Java Developers Conference, April 1996.

[10] S. Jha, P. Chalasani, O. Shehory and K. Sycara. A Formal Treatment of Distributed Matchmaking. In Proceedings of the Second International conference on Autonomous Agents (Agents 98), Minneapolis, MN, May 1998.

[11] J-J. Jeng, and B.H.C. Cheng. Specification matching for software reuse: a foundation. Proceedings of the ACM SIGSOFT Symposium on Software Reusability, ACM Software Engineering Note, Aug. 1995.

[12] V. Kashyap, and A. Sheth: Semantic heterogenity in global information systems: the role of metadata context and ontology. M.P. Papazoglou and G. Schlageter (Eds.), *Cooperative Information Systems: Trends and Directions*, Academic Press, 1998.

[13] KIF. Knowledge Interchange Format: http://logic.stanford.edu/kif/

[14] W. Kim, et al.: On resolving schematic heterogeneity in multidatabase systems. Intl. Journal on Distributed and Parallel Databases, Vol. 1:251-279, 1993.

[15] M. Kracker. A fuzzy concept network. Proc. IEEE International Conf. on Fuzzy Systems, 1992.

[16] D. Kuokka, L. Harrada, On using KQML for Matchmaking. Proc. 3rd Intl. Conf. on Information and Knowledge Management CIKM-95, pp. 239-45, AAAI/MIT Press, 1995.

[17] S.-H. Li, P. B. Danzig. Boolean Similarity Measures for Resource Discovery. IEEE Transactions on Knowledge and Data Engineering, Vol.9, No. 6, November/December, 1997.

[18] S. Muggleton, and L. De Raedt. Inductive logic programming: theory and methods. Journal of Logic Programming, Vol. 19(20):629–679, 1994.

[19] M. Nodine, J. Fowler. An overview of active information gathering in Infosleuth. Proceedings Intern. Conference on Autonomous Agents, USA, 1999. submitted.

[20] A. Ouksel. A framework for a scalable agent architecture of cooperating heterogeneous knowledge sources. M. Klusch (Ed.), *Intelligent Information Agents*, chapter 5, Springer, 1999.

[21] B. Potter, J. Sinclair, and D. Till. Introduction to Formal Specification and Z. Prentice-Hall International Series in Computer Science.

[22] Resource Description Framework (RDF) Schema Specification. http://www.w3.org/TR/WD-rdf-schema/.

[23] R. Rosenfield. Adaptive statistic language model. PhD thesis, Carnegie Mellon University, 1994.

[24] G. Salton, A. Wong. A vector space model for automatic indexing. Communications of the ACM, 18, 613-620, 1975.

[25] A. Sheth, E. Mena, A. Illaramendi, and V. Kashyap. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. Proc. of Intl. Conf. on Cooperative Information Systems CoopIS-96, IEEE Computer Soc. Press, 1996.

[26] Sheth, A., A. Illaramendi, V. Kashyap, and E. Mensa: Managing multiple information sources through ontologies: relationship between vocabulary heterogeneity and loss of information. Proc. ECAI-96, Budapest, 1996.

[27] G. Smolka, and M. Schmidt-Schauss. Attributive concept description with complements. AI, 48, 1991.

[28] K. Sycara, J. Lu, and M. Klusch. Interoperability among Heterogeneous Software Agents on the Internet. Carnegie Mellon University, PA (USA), Technical Report CMU-RI-TR-98-22.

[29] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed Intelligent Agents. IEEE Expert, pp. 36 - 46, December 1996.

[30] V. Vassalos, Y. Yapakonstantinou. Expressive Capabilities Description Languages and Query Rewriting Algorithms. available at http://www-cse.ucsd.edu/ yannis/papers/vpcap2.ps

[31] G. Wickler: Using Expressive and Flexible Action Representations to Reason about Capabilities for Intelligent Agent Cooperation. http://www.dai.ed.ac.uk/students/gw/phd/story.html

[32] XML. Extensible Markup Language. World Wide Web Consortium (W3C) Working Draft, 17 November 1997. http://www.w3.org/TR/WD-xml-link.

[33] A. M. Zaremski, J. M. Wing Specification matching of software components. Carnegie Mellon University, PA (USA), Technical Report CMU-CS-95-127, 1995.