

# Artificial Intelligence: a Promised Land for Web Services

D. Richards<sup>1</sup>, M. Sabou<sup>2</sup>, S. van Splunter<sup>2</sup>, F.M.T. Brazier<sup>2</sup>,

<sup>1</sup> Department of Computing  
Macquarie University  
Sydney, Australia  
richards@ics.mq.edu.au

<sup>2</sup> Department of Computer Science  
Vrije Universiteit Amsterdam  
Amsterdam, The Netherlands  
< marta, sander, frances >@cs.vu.nl

## Abstract

*This paper considers the possibilities offered by the application of techniques from the field of artificial intelligence (AI) to the newer field of web services (WS). Current commercial and research-based efforts are reviewed and positioned within the two fields. Particular attention is given to the application of AI techniques to the important issue of WS composition. Within the range of AI technologies considered, we focus on the work of the Semantic Web and Agent-based communities to provide WSs with semantic descriptions and intelligent behaviour and reasoning capabilities. Re-composition of web services is also considered and a number of adaptive agent approaches are introduced, including our own, to achieve this.*

## 1. Introduction

Research in the web service (WS) and artificial intelligence (AI) communities is coming together to develop solutions that will take us to the next and more mature generation of the WWW. The composition of web services to create a value-chain greater than the sum of the parts is a key part of what can be expected. The fulfillment of the vision of the web as an *information-providing* and *world-altering* provider of services [29] is not far away. More futuristic is the notion of *serendipitous interoperability* which Lassila [27] defines as “the ability of software systems to discover and utilize services they have not seen before, and that were not considered when the systems were designed”. In both visions the services and outcomes may be the same. However, the difference between the two visions is that the first can be achieved through static and manual solutions and the second requires dynamic and automated solutions. While helpful for the first, the addition of semantic content on the web is essential to enable automatic discovery and composition of multiple services. It is natural that earlier work in the field of AI will assist in realization of the (artificially) intelligent web.

The work on the Web Services Modeling Framework<sup>1</sup> (WSMF) is an example of AI being applied to this new field. WSMF offers the combined use of ontologies, goal (problem-type) repositories, web service descriptions and mediators to handle interoperability issues. The agent community, which is primarily AI-based, have also been actively conducting WS related research. Examples are the FIPA-led work that allows agents to use web service

infrastructure and the inclusion of a web services track at the recent AgentCities [26].

Our own distributed agent-based work and the Agent Factory [6], originates from our earlier AI research into complex knowledge based systems and generic task based configuration. On the one hand, our work on planning and automated configuration offers a way of composing web services. On the other hand, WSs potentially provide us with components needed to achieve an implementation of our design. Through the addition of techniques from the Semantic Web community, the benefits of combining our agent technology with WSs has been mutual.

This paper offers a review of research that overlaps the fields of WS and AI. In the following section we describe web services and the need for semantics to be added. In section 3 we look at how the Semantic Web community, within the field of AI, are offering semantics. In section 4 we present AI-based research to address the discovery of WSs. In section 5 we consider both commercial and AI-based techniques for WS composition. In section 6, the notion of re-composition of WS is considered and how adaptive agent technology, including our own, can address this problem. We conclude with future directions for the role of AI in the web services field.

## 2. Web Services

As is typical in new fields, there are a number of definitions of a web service. According to the W3C, “a web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artefacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols” [46]. We find this definition overly-focused on the technical aspects of a web service. A more business-oriented definition which supports the idea of services as providing a value-chain is “a self-contained, internet-enabled application capable not only of performing business activities on its own, but also possessing the ability to engage other web services in order to complete higher-order business transactions” [48]. The definition that most fits with our intended use of WSs as components in the Agent Factory is given by the Stencil<sup>2</sup> group who define a WS as “loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard

internet protocols". The three definitions offered differ in their emphasis on technology, business and software engineering but all encapsulate the self-contained, modular,

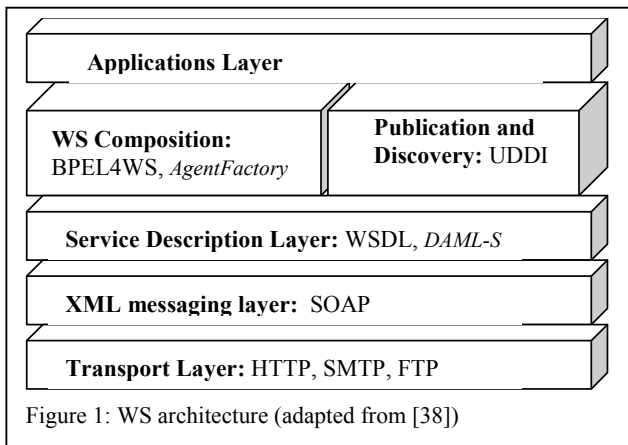


Figure 1: WS architecture (adapted from [38])

composable and distributed nature of WS.

These four characteristics of WS are well supported by a layered-architecture where the base is a well-established transport layer, as shown in Figure 1. In each layer we give an example of a major standard. In italics we position the work reported in this paper. The next layer up uses the Simple Object Access Protocol (SOAP) which is an XML-based communication protocol for exchanging data in decentralized and distributed environments via typed message exchange and remote calls. The service description layer includes the XML-based Web Service Description Language (WSDL). The next layer is split into two main types of WS technologies: ones that support single service advertising and discovery and ones that support service composition. For service registration and discovery there is the Universal Description, Discovery and Integration (UDDI) (by IBM, Microsoft and Ariba) standard service repository. To provide some very basic semantics (such as identification via a product classification code) one or more tModel descriptions may be attached to a service. For service composition there are a myriad of possible solutions. Figure 1 just includes the Business Process Execution Language for Web Services (BPEL4WS)<sup>3</sup> which has grown out of the early offerings WS Flow Language (WSFL) (IBM) [28] and XLANG (Microsoft) [41] (an extension of the W3C's Web Services Description Language (WSDL)).

### 3. Semantic description of Web Services

WSDL, SOAP and UDDI are seen as steps in the right direction but ones that will fail to achieve the goals of improved automation and interoperability, because they rely on *a priori* standardisation and require humans in the loop [27]. To support automated reasoning, knowledge representations (such as markup languages) will be needed that express both data and rules for reasoning. The ability to dynamically locate and compose web services based on

their semantic description will rely on the richness of the description and the robustness of the matching techniques used. Ontologies will be used to enable definition and comprehension of meaningful content. These are the concerns of the Semantic Web community. Additionally, agents will be needed to interpret the content and transform user requests into optimized delivered solutions.

The Intelligent Brokering Service for Knowledge-Component Reuse on the WWW (IBROW)<sup>4</sup> can be seen as a forerunner of the Semantic Web. In IBROW problem solving methods (PSMs) and ontologies were the components being configured, the current focus is on WS configuration. PSMs and ontologies when used together are also capable of delivering services.

The most significant work that has been done to describe web services has been conducted by the DAML-S coalition [40]. The matching of service providers and service requesters via semantic descriptions of the services are key goals of this work. DAML-S uses the DAML+OIL specification language (which extends the weak semantics of RDF(S)) to define a number of ontologies that can be specifically used to describe web services. DAML-S is built on the AI-based action metaphor where each service is either an atomic/primitive or composite/complex action. Knowledge preconditions and knowledge effects are handled via the inputs and outputs of the web service [29]. The DAML-S coalition are providing solutions to work with current WS standards. For example, a DAML-S service grounding definition can be mapped to a WSDL definition of the service. A number of approaches to service discovery and composition that we discuss in the following sections use or extend the DAML-S web service ontologies.

### 4. Discovering Web Services

Discovery involves locating and/or matchmaking against some selection criteria. An earlier AI system, Lark [39], which involved annotation of agent capabilities to enable them to be located and brokered, clearly solved a problem similar to the discovery of WS by a middle agent. This work has developed into the DAML-S Matchmaker<sup>5</sup>. To support matchmaking a number of filters may be configured by the user to achieve the desired tradeoff between performance and matching quality. These filters include: word frequency comparison, ontology similarity matching, ontology subsumption matching, and constraint matching.

[11] offer an alternative to sequential searching when matchmaking an agent with a service request. They point out that finding possible partners via matching of service advertisements with requests is not enough. To support runtime interactions we need smarter behaviour to handle components that are not quite what was requested and combining several partial components to meet the original request. The solution to overcome sequential searching is the conversion of the concepts into number intervals and

the use inheritance hierarchies to determine subclass and equality relations. A generalised search tree is used to handle partial matches.

The feasibility of matchmaking largely depends on the annotation of web services. AI can also be applied to this problem. A number of markup tools have been developed for document markup and these could be applied to the semantic description of WSs. The SHOE Knowledge Annotator [19] uses ontologies to guide knowledge annotation. To produce RDF-based markup, COHSE [1] or AeroDAML [24] can be used. These approaches start with descriptions in DAML+OIL and DAML, respectively. These approaches support automatic conversion of markup languages but do not support information extraction or automatic markup. OntoMat [18] does support some form of automated extraction of semantics. OntoMat combines the resource with its DAML-S markup. The MnM [44] approach additionally stores the annotations in a knowledge base. Automated markup in MnM is achieved using techniques from knowledge engineering, machine learning and natural language processing.

[22] have developed a query language that is used to find services. The solution to finding services is to first describe the service using the process ontology with the assistance of the MIT Process Handbook. The Handbook is large and allows reuse to assist in ontology definition. Next, the ontology is indexed by breaking it down into its components such as attributes, ports, dependencies, subtasks and exceptions. The requester can form a query in the query language that will use the index to find matches.

Clearly AI is already contributing solutions for locating, matchmaking, querying and annotation of WS to facilitate their discovery. Discovery of web services is an important issue as it is a prerequisite to their use. However, the real value of web services lies in their composition.

## 5. Composing Web Services

Web service composition can be simply defined as: “the problem of composing autonomous services to achieve new functionality” [33]. WS composition is not just an alternative to application development but a means of reducing the application backlog problem because: many services are moving online; integration is easier since WSs conform to the HTTP protocol and many independent providers have related services that need to be combined to satisfy user requirements. The rigidity and lack of intelligence of current solutions has spawned a number of research projects from a number of other research fields.

The work by [42] has arisen from experience in the distributed systems and networking fields. They have developed the Infrastructure for Composability at Runtime of Internet Services (ICARIS). They have extended WSDL to develop the Web Services Offerings Language (WSOL). They offer flexibility and adaptability but their approach is very alternative. Instead of trying to solve the problem of how to find services dynamically and combine them, they

focus on the situation where providers and requestors are already matched but will at times either make changes to their services or requests. A service is seen to have numerous offerings. The functionality will be the same but the constraints will differ such as authorisation rights and QoS. They suggest that a limited number of classes of services be offered and described. Then using WSOL they are able to specialise the classes into offerings. Their solution offers dynamic switching between offerings. From a commercial point of view the notion of offerings makes sense as customers probably prefer to do business with companies they already know and businesses want to maintain their existing client base.

The work at Hewlett Packard laboratories on *eFlow* [9] is similar in that dynamic composition involves automatic adaptation of the configuration at runtime according to the requests of the individual customer. The approach is driven by the view that composition adds value but to stay competitive, composition needs to be dynamic as services offered need to adapt to stay competitive. Their goal is to allow dynamic change in service processes with no or minimal human intervention. While they take a business process perspective they point out that web services are less static, predictable or repetitive compared to “traditional” business processes. Similar to most current commercial solutions, dynamic composition is made possible due to the use of a central repository that has clients and providers already attached to it.

The notion of generic solutions that are customized according to user constraints is a recurring theme in much of the literature. [37] also look at composition as the selection of possible services based on user specified criteria. They offer a centralized, pipes and filters architecture with two main components: a composer (user interface) and an inference engine (IE) component (which includes a knowledge base of known services). The inference engine is an OWL reasoner and includes axioms to find all relevant entailments, such as the inheritance relation between two classes which may not have been made explicit. The user identifies some criteria that the service must satisfy. The matchmaker (IE) selects services that might be suitable based on those criteria and the composer shows them to the user. Suitable services for composition are ones whose output can be an input to a selected service. While execution of WS may be performed automatically, the actual task of composition is performed by a human using the services suggested by the system.

Model-based reasoning is a common technique employed in AI approaches. In SWORD [33] entity-relationship modeling of services is performed by “base service modelers” to produce a “world model”. After building a world model for each service, a composition model is developed that models each service as an action. An expert system is used to automatically determine if the composite service can be created with existing services and if so a plan of execution is generated.

In summary, a number of solutions are offered to provide web service composition. The approaches described in this section show that composition can be assisted through the use of class definitions, inheritance hierarchies and model and rule-based reasoning. In many cases, decision making is left to humans. The only automated composition offered is in limited situations where a central repository is used and the requestor and provider are part of the same system. However, the web is distributed in nature. Intelligent reasoning and collaboration between services is needed to handle this complexity. Agents are capable of both.

## 6. Agents and Web Services

The autonomous and reasoning capabilities of agents make them well suited for handling cross-organisational decision making. For example, agents can be used to (re)negotiate contracts which would then require: determination of which processes are needed to fulfil the contract; creation of new business processes; and adaptation of existing business processes. Two main agent-oriented approaches exist: use wrappers to make WS behave like agents and; using agents to orchestrate WS.

### 6.1 Adding Behaviour to WS via Agents Wrappers

WS are componential, independent, software applications similar to agents. However, agents are also reactive, social and capable of reasoning [47]. If we wish web services to work together, we need to give them social and reasoning capabilities. This can be achieved by wrapping a service in an agent. In the work of [8], a composition language is used to create an agent wrapper which allows services to collaborate. The created agent has first-order reasoning abilities that have been derived from the DAML-S description of the service. This then allows one agent-wrapped service to know what other agent-wrapped services are capable of doing and whether they can assist in the service/agent meeting its goals. [23] also offer an agent-based wrapper approach to web services. They have developed a tool for creating wrappers so that web sources can be queried in a similar manner to databases. They then use an interactive, hierarchical constraint propagation system to perform integration. As in [37], the end-user interacts via a GUI to manage the orchestration. The Racing project<sup>6</sup> offers a mediator architecture also using agent wrappers that are structured into a hierarchy. A number of different agent wrappers are supported: user, query translation, query planning, resource wrapper, ontology, matchmaking, cloning and coordination agents. The use of agent wrappers is a way of allowing multi-agent system technology to be applied to web services.

### 6.2. Composing Web Services using Agents

The work of [29] combines ideas from the Semantic Web, Knowledge Representation and Agent communities to allow WSs to be composed. Their goal is to “construct reusable, high-level generic procedures, and to archive them in shareable (DAML-S) generic-procedures ontologies so that multiple users can access them”. In the approach, WSs and user constraints are marked up in DAML-S. A generic task procedure is selected by the user and given to the DAML(-S) enabled agent, who customises the procedure according to the user specific constraints. The generic procedures are written in an extended version of ConGolog, a situation calculus agent programming language, and executed using a Prolog inference engine. Others provide agent-oriented languages for web service description. [10] propose an Agent

Service Description Language (ASDL) and Agent Service Composition Language (ASCL). ASDL is an extension to WSDL and captures external behaviour via a finite state machine. Their work is based on the argument that composition requires more than description of the data, but also requires a strong representation of actions and processes. A number of approaches are focused on the design of agent systems with web services as the components. [3] has developed WARP (Workflow Automation through Agent-based Reflective Processes) that uses the XML and WSDL standards. The goal is automatic configuration and management of low-level services (components). The software engineering development process that has been developed is semi-automatic involving multiple software agents and a human workflow designer. They support visualisation of the process based on activity diagrams in UML.

Niersatz [31] argues that (re-)composability is a distinguishing feature of open systems. We have considered some approaches which support the use of agents to reason about and coordinate services over the ultimate open system, the web. In some of these approaches, composition involves reuse and specialization of generic components. As [29] point out, services often provide multiple outputs, only some of which may be needed as inputs to a subsequent service. Sometimes additional services will be needed to overcome a mismatch in inputs and/or outputs. Such shortcomings in the original configuration may require recomposition of WSs. To achieve this may involve adaptation of the agent's behaviour, since web services are by nature closed and immutable.

### 6.3 (Re-)composition and Adaptable Agents

The ability of agents to adapt according to changes in system requirements and the environment is important to enable dynamic and reactive behaviour.

Agents may be adapted in a number of different ways. The knowledge and facts that an agent uses may be adapted for example the agent may use a client profile that changes according to the clients activities (e.g. [45]). This type of adaptation typically involves machine learning, e.g. [25]. An agent may also adapt its interface according to the platform on which it is being used (e.g. [brand]). A third type of adaptation, and the type of adaptation we are concerned with, is adaptation of the agent's functionality. There is limited work in this area. Semi-automatic agent creation tools such as AGENTBUILDER [34], D'AGENTS/ AGENT/TCL [17], ZEUS [32] and PARADE [2] could possibly be extended to support agent adaptation.

Following the use of compositionality in the major software engineering paradigms (e.g. functional programming [21], object-oriented programming [4], component-based programming [20] and the Factory design pattern [16]), we have developed an Agent Factory [6]. The approach is based on the use of components, the general agent model (GAM) and the DESIRE formal knowledge-level modelling and specification framework for multi-agent systems [7]. Our agent (re-)structuring approach allows an agent to automatically adapt by reusing existing components. Our approach is a combination of process-oriented and object-oriented approaches by treating processes as the 'active' parts of our agent, which are our agent components, and classes as the 'passive' part of our agent, which are the data types used in the agent components. We are currently exploring whether DAML-S descriptions of web services are adequate for automated configuration of web services by the Agent Factory. Our initial report on how the Agent Factory can be used to perform this task

is found in [43]. Our observations and recommendations regarding DAML-S are given in [35].

The work by [12], which is also called the Agent Factory and based on the notion of design patterns, assists human designers in functional design, and the configuration of software components to fulfil the conceptual design specified by the designers, depending on the agent platform that is to be used. Our approach does more: it automates the creation and redesign of both the conceptual and operational design based on the requirements on function, behaviour and state of an agent. Our use of web services as components is a further distinguishing feature.

While not currently working in the WS area, the AdaptAgent [30] approach, bring together adaptive workflow and agent research. They consider how agents can be used to collaborate to perform a workflow and make workflow more intelligent and how workflow can be used to organise a set of agents and coordinate interaction between people and agents.

The reuse of knowledge has also been a widely researched topic and the creation of libraries of problem solving methods [36] and generic task models [7] offer a similar idea to the functional components in our agent factory. The IBROW project, mentioned earlier, has even more in common with our approach by semi-automatically configuring intelligent problem solvers using problem solving methods as building blocks. They use mappings to act as glue between the components which are modeled as CORBA objects. Unlike our approach, their architecture is restricted to specific languages and architectures, they only support semi-automation and they do not distinguish between conceptual and implementation level designs.

## 7. Discussion and Future Directions

An interesting phenomena of AI research is that when a problem becomes solved it no longer holds any mystery and moves from being called AI to being just another part of information processing. This phenomena was first noted by Donald Michie and is thus known as *the Michie effect* [15]. Examples of the assimilation of AI concepts into mainstream data processing are the use of machine learning techniques in knowledge discovery from databases, the inclusion of business rules in database technologies and the use of ontologies for information systems development.

Similarly, AI-based research will benefit B2B, e-commerce and internet applications requiring knowledge-level interoperability of information systems and intelligent processing by agents. Advances in natural language technology research will assist discovery of web services and agents will play an important role in using web services to satisfy user requests.

The current trend towards interoperability of systems and integration of technologies will continue and increase the need for standards. Standards, as mentioned in section 2, are emerging for web services. As the roles of agents on the web increases, further work is required in the area of communication standards between agents and web services. For invocation, the Java Agent Services (JAS) project is developing an industry standard and API for network agent and service architectures. JAS does not, however, specify how an ACL message can be translated into the format needed by the web service. HP BlueJade also does not describe how agents can use SOAP, UDDI, WSDL, etc, or say how services and agents can communicate [26].

Existing agent platforms may need to be adapted to handle the specific requirements of web services. In this direction, CMU

have proposed the RETSINA<sup>7</sup> agent architecture for web-based agents. The RETSINA functional architecture includes four basic types of agents: interface, task, information and middle agents who communicate via a special agent communication language. Each of these agents includes four reusable modules: communication and coordination, planning, scheduling and monitoring. The middle agent plays a critical role in matching providers with requesters and is offered as a solution to the heterogeneous nature of agents over the web.

The work of the Semantic Web community to provide semantic description of web services will play a key role in enabling agents to automatically compose web services. We are more than interested onlookers in these developments. While it is still early days, the incorporation of ideas from AI is already proving to be mutually beneficial.

## 10. References

- [1] Bechhofer, S. and Goble, C. Toward Annotation Using DAML+OIL, 1st Int. Conf. on Knowledge Capture (K-CAP'2001), W'shop on Semantic Markup and Annotation, Victoria, BC, Canada, Oct. 2001
- [2] Bergenti, F., Poggi A.: A Development Toolkit to Realize Autonomous and Inter-Operable Agents. In: Proc. of Fifth Int. Conf. of Autonomous Agents (Agents 2001), Montreal (2001) 632-639
- [3] Blake, M.B. An Agent-Based Cross-Organizational Workflow Architecture in Support of Web Services, WETICE 2002: 176-181
- [4] Booch, G.: Object oriented design with applications. Benjamins Cummins Publishing Company, Redwood City, 1991
- [5] Brandt, R., Hörtnagel, C., Reiser, H.: Dynamically Adaptable Mobile Agents for Scaleable Software and Service Management. Journal of Communications and Networks 3:4 (2001) 307-316
- [6] Brazier, F.M.T., Wijngaards, N.J.E.: Automated Servicing of Agents. AISB Journal 1 (1), Spec. Issue on Agent Tech. (2001) 5-20
- [7] Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering 41 (2002) 1-28
- [8] Buhler, P. A. and Vidal, J. M. (b) Semantic Web Services as Agent Behaviors. In B. Burg, J. Dale, T. Finin, H. Nakashima, L. Padgham, C. Sierra, and S. Willmott, editors, *Agentcities: Challenges in Open Agent Environments*, pages 25-31. Springer-Verlag, 2003
- [9] Casati, F., Ilnicki, S. and Jin, L. Adaptive and Dynamic Service Composition in eFlow. HP Technical Report, HPL-2000-39, March, 2000, [www.hpl.hp.com/techreports/2000/HPL-2000-39.pdf](http://www.hpl.hp.com/techreports/2000/HPL-2000-39.pdf)
- [10] Cheng, Z., Singh, M.P. and Vouk, M.A., Composition Constraints for Semantic Web Services. WWW2002 Workshop on Real World RDF and Semantic Web Applications, to appear in the proceedings, May 7, 2002
- [11] Constantinescu, I. and Faltings, B. Efficient Matchmaking and Directory Services. Technical Report No IC/2002/77, 18 Nov 2002, Laboratoire d'Intelligence Artificielle, Department Informatique, Swiss Federal Institute of Technology, IN (Eculbens), 2002
- [12] Cossentino, M. Burrafato, P., Lombardo, S. and Sabatucci, L. Introducing Pattern Reuse in the Design of Multi-Agent Systems. AITA'02 workshop at NODe02 - 8-9 October 2002 - Erfurt, Germany
- [13] Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D. and ter Hofstede, A. Towards a semantic framework for service description In Proc. of the IFIP Conference on Database Semantics, Hong Kong, Kluwer Academic Pub., April 2001
- [14] The DAML Services Coalition (alphabetically Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne and Katia Sycara), DAML-S: Web Service Description for the Semantic Web, The First International Semantic Web Conference (ISWC), Sardinia (Italy), June, 2002

- [15] Gaines, B. (2000) Knowledge Science and Technology: Operationalizing the Enlightenment In P. Compton, A. Hoffmann, H. Motoda and T. Yamaguchi (eds) Proceedings of the 6<sup>th</sup> Pacific Knowledge Acquisition Workshop, Sydney Dec. 11-13,2000, 97-124
- [16] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of reusable object-oriented software. Addison Wesley Longman, Reading, Massachusetts, 1994
- [17] Gray, R.S., Kotz, D., Cybenko, G., Rus, D.: Agent Tcl. In: Cockayne, W., Zypa, M. (eds.): Itinerant Agents: Explanations and Examples with CD-ROM. Manning Pub. (1997) 58–95
- [18] Handschuh, S., Staab, S. and Maedche, A. CREAM- Creating Relational Metadata with a Component-Based, Ontology-Driven Annotation Framework, 1st Int. Conf. on Knowledge Capture (K-CAP'2001), Workshop on Semantic Markup and Annotation, Victoria, BC, Canada, October 2001
- [19] Heflin, J. and Hendler, J. A Portrait of the Semantic Web in Action, IEEE Intelligent Systems, 16(2), 2001
- [20] Hopkins, J.: Component primer. Communications of the ACM 43:10 (2000) 27–30
- [21] Kernighan, B.W., Ritchie, D.M.: The C Programming Language. 2nd edn. Prentice Hall Software Series, 1988
- [22] Klein, M and Bernstein, A. Searching for Services on the Semantic Web using Process Ontologies. In The Emerging Semantic Web - Selected papers from the first Semantic Web Working Symposium, Isabel Cruz, S. Decker, J. Euzenat, and D. McGuinness, Eds. Amsterdam: IOS press, 2002, pp. 159-172
- [23] Knoblock, C., Minton, S., Ambite, J.L., Muslea, M., Oh, J. and Frank, M. Mixed-initiative, multi-source information assistants, In Proceedings of the World Wide Web Conference, pages 697--707, ACM Press, New York, NY, May 2001
- [24] Kogut, P. and Holmes, W. AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages, 1st Int. Conf. on Knowledge Capture (K-CAP'2001), Workshop on Semantic Markup and Annotation, Victoria, BC, Canada, October 2001
- [25] Kudenko, D., Kazakov, D., Alonso, E.: Machine Learning for Multi-Agent Systems. In: V. Plekhanova, V.(ed.): Intelligent Agents Software Engineering, Idea Group Publishing, 2002
- [26] Kuno, H. and Sahai, A. My Agent Wants to Talk to Your Service: Personalizing WSs through Agents. HP Tech. Rep., HPL-2002-114, www.hpl.hp.com/techreports/2002/HPL-2002-114.html
- [27] Lassila, O. Serendipitous Interoperability, in Eero Hyvönen (ed.): "The Semantic Web Kick-Off in Finland - Vision, Technologies, Research, and Applications", HIIT Publications 2002-001, University of Helsinki, 2002
- [28] Leyman, F. Web Service Flow Language (WSFL) 1.0, IBM, Armonk, NY, www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, 2001
- [29] McIlraith, S. and Son, T., Adapting Golog for Composition of Semantic Web Services, Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), Toulouse, France, April, 2002
- [30] Narendra, N. C. AdaptAgent: Integrating Adaptive Workflows and Multi-Agent Conversations for B2B E-Commerce. Proceedings of International Conference on Artificial Intelligence, Special Session on Agent-Oriented Workflow Architecture for B2B, 2001
- [31] Nierstrasz, O. and Meijler, T. D. Requirements for a Composition Language. Object-Based Models and Languages for Concurrent Systems, Paolo Ciancarini, Oscar Nierstrasz and Akinori Yonezawa (Eds.), 147—161, Springer-Verlag, 1995
- [32] Nwana, H.S., Ndumu, D.T., Lee, L.C.: ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems. Applied AI 13:1/2 (1998) 129-185
- [33] Ponnekanti, S.H. and Fox, A. SWORD: A Developer Toolkit for Web Service Composition. In The Eleventh WWW Conference (Web Engineering Track), Honolulu, Hawaii, May 7-11, 2002
- [34] Reticular: AgentBuilder: An Integrated Toolkit for Constructing Intelligent Software Agents. Reticular Systems Inc, white paper edition. <http://www.agentbuilder.com>, 1999
- [35] Sabou, M., Richards, D. and van splunter, S. An experience report on using DAML-S , Workshop on E-Services and the Semantic Web, Budapest, Hungary, May, 2003 (submitted)
- [36] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., Wielinga, B.: Knowledge Eng. and Management, the CommonKADS Methodology. MIT Press, 2000
- [37] Sirin, E., Hendler, J. and Parsia, B. Semi-automatic Composition of Web Services using Semantic Descriptions. Accepted to "Web Services: Modeling, Architecture and Infrastructure" workshop in conjunction with ICEIS2003, 2002
- [38] Staab, S., Benjamins, R., Bussler, C., Gannon, D., Sheth, A. and van der Aalst, W., Web services: Been there, Done that? IEEE Intelligent Systems, Trends & Controversies, 18(1), Jan/Feb 2003
- [39] Sycara, K., Widoff, S., Klusch, M. and Lu, J., LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace, Autonomous Agents and MAS, 5 (2): 173-203, June, 2002
- [40] The DAML Services Coalition (alphabetically A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), "DAML-S: Semantic Markup for Web Services", in Proceedings of the International Semantic Web Working Symposium (SWWS), July 30-Aug. 1, 2001
- [41] Thatte, S. XLANG: Web Services for Business Process Design, [www.gotdotnet.com/team/xml\\_wsspecs/xlang-c/default.htm](http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm), 2001
- [42] Tasic, V., Pagurek, B. Esfandiari, B. and Patel, K. :On the Management of Composition of Web Services position paper at the Workshop on Object-Oriented Web Services - OOWS (at OOPSLA 2001), Tampa, USA, October 15, 2001. Also published as: Technical Report OCIECE-01-10, Ottawa-Carleton Institute for Electrical and Computer Engineering - OCIECE, October, 2001
- [43] van splunter, S., Sabou, M., F.M.T. Brazier and Richards, D. Configuring Web Services, using Structuring and Techniques from Agent Configuration, EEE/WIC International Conference on Intelligent Agent Technology (IAT 2003) (submitted)
- [44] Vargas-Verá, M, Motta, E., Domingue, J, Lanzoni, M., Stutt, A. and Ciravegna, F. MnM: Ontology Driven Tool for Semantic Markup. European Conference on Artificial Intelligence (ECAI 2002). In proceedings of the Workshop Semantic Authoring, Annotation & Knowledge Markup (SAKM 2002). Lyon France, July 22-23, 2002
- [45] Wells, N., Wolfers, J.: Finance with a personalized touch. Comms of the ACM, Issue on Personalization 43:8 (2000) 31–34
- [46] W3C Web Services Architecture Working Group. Web Services Architecture Requirements, Working Draft 29 April 2002, <http://www.w3.org/TR/2002/WD-wsa-reqs-20020429>
- [47] Wooldridge, M. and Jennings, N. (1995), Agent Theories, Architectures, and Languages: a Survey, in Wooldridge and Jennings Eds., Intelligent Agents, Berlin, Springer-Verlag, 1-22.
- [48] Yang, J and Papazoglou, M. Web Component: A Substrate for Web Service Reuse and Composition. in Procs of the 14th International Conference on Advanced Information Systems Engineering (CAISE02), May, Toronto, Lecture Notes in Computer Science, Vol. 2348, p21-36, Springer, 2002

<sup>1</sup> <http://informatik.uibk.ac.at/c70385/wese>.

<sup>2</sup> [www.stencilgroup.com/ideas\\_scope\\_200106wsdefined.html](http://www.stencilgroup.com/ideas_scope_200106wsdefined.html)

<sup>3</sup> <http://xml.coverpages.org/bpel4ws.html>

<sup>4</sup> [www.ibrow.org](http://www.ibrow.org)

<sup>5</sup> [http://www-2.cs.cmu.edu/~softagents/daml\\_Mmaker/daml-s\\_matchmaker.htm](http://www-2.cs.cmu.edu/~softagents/daml_Mmaker/daml-s_matchmaker.htm)

<sup>6</sup> <http://www.zsu.zp.ua/racing/>

<sup>7</sup> [http://www-2.cs.cmu.edu/~softagents/retsina\\_agent\\_arch.html](http://www-2.cs.cmu.edu/~softagents/retsina_agent_arch.html)