Toward the Synthesis of Web Services and Agent Behaviors

Paul A. Buhler
College of Charleston
Dept. of Computer Science
66 George Street
Charleston, SC 29424, USA
pbuhler@cs.cofc.edu

José M.Vidal
University of South Carolina
Computer Science and Engineering
Columbia, SC 29208, USA
vidal@sc.edu

ABSTRACT

Today's software systems are becoming more net-centric, distributed, and heterogeneous. Hardware, software and networking technology will combine in a milieu in which they become ubiquitous and inseparable. The acceleration of technology and time-to-market pressures make it increasingly difficult to produce software. In order to achieve the promise of the information age, software developers will require new abstractions that will allow them to manage the overwhelming complexity of this digital landscape.

This short position paper describes a novel technique that will imbue agent software with dynamically configured capabilities. These capabilities, described with DAML-S, can represent atomic or orchestrated Web Services. The DAML-S specification will be transformed into an executable program written in a composition language named Piccola. When executed, the composite service will be available as a semantically described behavior within a FIPA compliant agent. The proposed architecture is designed for scalability, from mobile PDA devices with wireless connectivity to resource-rich server class systems.

1. INTRODUCTION

In 1999, NSF sponsored a workshop to discuss software engineering research strategies. The participants at the workshop drew several conclusions about software engineering research, one of which is particularly relevant to the vision of this thesis proposal. This conclusion is summarized by the phrase "skate to where the puck is going," meaning researchers need to be more forward thinking. "Heterogeneous distributed systems, dynamically changing software structures, and interactions among autonomous agents," were explicitly mentioned as requiring focused research [12].

Software development needs to progress from handcrafted, line-at-a-time techniques to methodologies that support reuse of existing software assets. In other words, software development needs to shift from paradigms that are purely creational to others that support compositional approaches. Traditional software engineering methodologies are giving way to new software development paradigms. Component-based software engineering and agent-oriented software engineering are two paradigms that are garnering attention. Although typically thought of as separate disciplines, it is likely that they are not only related, but also ultimately dependent upon one another. In the future, passive software components will be liberated by the proactive and social nature of agents. In effect agent-based technologies provide the mechanism for components to seek work, enter into cooperative agreements and thus otherwise address the requirements of dynamic, heterogeneous environments.

2. SOFTWARE COMPONENTS

The range of component-based software engineering practice can be constrained by the definition of a software component. The definition of a software component is hotly debated; a sampling of common definitions can be found in [21, 22, 34, 38]. For the purposes of this paper, the definition presented in [21] will be used. This definition was selected for several reasons: it has undergone extensive review and revision; the definition is architecturally neutral in that it does not favor any specific implementation language or component model; and it is abstract enough to be inclusive of the other commonly referenced definitions of a software component. The software component definition found in [21, pg 7] is:

> *A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard.*
>
> *A component model defines specific interaction and composition standards. A component model implementation is the dedicated set of executable software elements required to support the execution of components that conform to the model.*

2.1 Component Models

A component model provides standards that govern the interaction and composition of software components that conform to the model. Standards are essential to the concept of open systems. An open system is a collection of interacting software and hardware components. The interaction within the open system is defined by interface specifications that are complete, publicly available and non-proprietary [29].

2.1.1 Web Services as Components

Businesses are organizations whose participants collectively perform work. This work usually occurs in the form of a workflow. A workflow is a process "during which documents, information or tasks are passed from one participant to another in a way that is governed by rules or procedures”[5]. Businesses can achieve efficiencies by analyzing and redesigning their workflows; in fact this was a major focus in the early 1990’s when Business Process Reengineering (BPR) was commonly practiced [2]. From a workflow perspective, a composite software system can be viewed as a sequence of services operating upon data. Ideally these services should be language, platform and location independent [17]. Such services would then be interoperable, where interoperability is characterized by the “ability of two or more software components to cooperate despite differences in language, interface, and execution platform” [41].

A new class of interoperable, web-enabled software services is emerging. These services are known as Web Services, which are defined as:

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 artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols [40].

Several specifications have been developed that are forming the basis of a component model for Web Services; specifically SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery, and Integration) [16]. These specifications are used to describe, publish, discover and invoke Web Services. These Web Service specifications embrace an open systems viewpoint: XML (Extensible Markup Language) is utilized to exchange data in a neutral format and component communication occurs via a transport protocol like HTTP. A comprehensive overview of the industry players and their respective Web Service architectures is found in [30].

3. WORKFLOW SPECIFICATIONS

There are several ongoing initiatives that are defining compositional notations for Web Services. Large commercial entities like IBM, Microsoft, and Hewlett-Packard are actively participating because they believe that Web Service integration presents an enormous business opportunity. Likewise, consortium players like OASIS (Organization for the Advancement of Structured Information Standards), and BPMI (Business Process Management Initiative) have complementary efforts as well. Web Service architectures are stack based; the lower layers contain open networking and Web Service protocols, the higher layers are comprised of proprietary integration and orchestration protocols [30, 33]. The proprietary nature of the upper layers will allow corporations to differentiate their products and services from one another.

IBM has produced a markup language named Web Services Flow Language (WSFL) [14] that is designed for the specification of workflows that encompass multiple Web Services. Currently, IBM is integrating WSFL-based technologies into their WebSphere product. Microsoft’s XLANG [15] provides WSFL-like capabilities. Microsoft’s BizTalk Application Designer is a graphical tool that allows a business process to be flow-charted and output in XLANG, whereupon it can be executed with Microsoft’s BizTalk Server Orchestration Engine. Notably, it is anticipated that WSFL and XLANG will be merged and submitted to the W3C as a proposed web-standard. Hewlett-Packard has produced Web Services Conversation Language (WSCL) [13] that can be used to specify the conversation policies between Web Services. It remains to be seen how the marketplace will react to multiple, overlapped initiatives. The recently announced Web Service Interoperability Organization (WS-I) [6], includes both IBM and Microsoft as founding members. The goal of WS-I is to speed adoption of Web Service technologies by maintaining the interoperability of the lower layer protocols. A good overview of workflow description standards, including ebXML BPSS and BPML is found in [32].

3.1 Agent-based Workflow Approaches

As discussed in [36], it is anticipated that next generation workflow systems will employ agent-based technologies. Others share this view, specifically [18-20, 27]. To place this in perspective, an agent is a system that exhibits properties like: situatedness, autonomy, reactivity, pro-activeness, and social ability [42]. These properties allow an agent to “perceive, reason, and act in their environment, and communicate with other agents”[36].

If a collection of sociable agents, representing individual services, cooperate and coordinate they would have the capability to enact any workflow that is composed of the represented services. In other words, agents have the capability to dynamically form social structures through which they share commitments to the common goal of workflow enactment. The individual agents, through their coordinated interactions achieve globally coherent behavior; they act as a collective entity known as a multi-agent system.

The social metaphor gives power to the agent-oriented paradigm. It is one of the characteristics that makes the agent abstraction particularly suitability for developing complex, distributed systems [24, 25]. The fields of sociology and organizational theory provide valuable abstractions for multi-agent systems [23]. Recently the term socionics has been used to describe a new field of research that is a combination of sociology and distributed artificial intelligence [28]. The cross-pollination of these disciplines has created useful vocabulary and framing for describing the problems and issues that multi-agent systems encounter.

Workflow enactment by a multi-agent system is an example of cooperative problem solving. “Cooperative problem solving occurs when a group of autonomous agents choose to work together to achieve a common goal” [43]. For cooperative problem solving to occur, an agent in the multi-agent society must recognize that the best path to achieving a goal is to enlist the help of other agents. Social commitments arise when one agent makes a commitment to another. Typically a social commitment comes about due to a social dependency. As defined in [23, pg 113] a social dependence can be defined as:

$$(\mathit{SocialDependence}\ x\ y\ a\ p) \equiv (\mathit{Goal}\ x\ p) \wedge \neg(\mathit{CanDo}\ x\ a) \wedge (\mathit{CanDo}\ y\ a) \wedge ((\mathit{DoneBy}\ y\ a) \Rightarrow \mathit{Eventually}\ p)$$

[Meaning] agent x depends on agent y with regard to act a for realizing state p, when p is a goal of x and x is unable to realize p while y is able to do so.

As indicated, for such a social dependency to be established, agent x and agent y must be able to reason about their ability to perform act a, and have knowledge that the performance of a will establish state p. The concept of first-order ability, as introduced in [43, pg 150], states that for agent x to have first-order ability regarding the establishment of state p, it must know explicitly whether $\exists a((\text{CanDo}\ x, a) \wedge ((\text{DoneBy}\ x\ a) \Rightarrow \text{Eventually}\ p))$. If agent *x* desires to achieve state *p*, but knows ¬(FirstOrderAbility *x, p*), then it must solicit assistance in order to attain the goal.

4. CONFLUENCE

The views of [18, 27, 36] share a common theme. This theme is that cooperating agents, acting as workflow components, can self-assemble in order to enact business processes. If agents are to be treated as components, they should conform to a component model. Adherence to a component model is necessary to allow interoperability, but this is not a sufficient condition for interoperability in open environments. The work of the Foundation for Intelligent Physical Agents (FIPA) is essentially creating a component model that allows agents from heterogeneous origins to collaborate in open agent environments.

Open agent environments present many challenges. As previously demonstrated, if agents are to determine conditions of social dependency, they must have the ability to reason about their own capabilities. Likewise, in order to effectively negotiate for the services of another agent, an agent will need to reason about the abilities of potential partners. In open environments a common Agent Communication Language (ACL) facilitates message exchange between the agents; however, for true communication to occur the agents must understand the message contents. Ontologies help provide meaning to the contents of the messages; however, they are not a panacea for ontological mismatches can take place. A potential solution to this problem is to define a proprietary service description language as described in [37, chpt 3]; however, this solution has limited utility in open environments.

It is the belief of the authors that the semantic web and the emergence of a Web Services component model can facilitate agent-based workflow management in open environments. If agents are used to wrap semantically described Web Services, then the semantic service descriptions become the basis for determining the agent's first-order abilities. Likewise, a common semantic markup for Web Services will facilitate effective communication between agents. We intend to build an experimental system that will utilize DAML-S and a composition language named Piccola.

4.1 DAML-S

The semantic web initiative is developing technologies for locating web resources based upon their semantic content. Included in this vision is DAML-S, a specification for providing semantic markup for Web Services. DAML-S is being designed to support the following Web Service related tasks: discovery, invocation, composition and interoperation, and execution monitoring [4]. DAML-S provides a machine-interpretable, ontology-backed semantic description of both atomic and composite web-services. For a discussion of the relationship of DAML-S to other standards like UDDI, WSDL, and ebXML see [3].

As previously described WSFL, XLANG, et al. are designed to capture the flow of a composition of services. Likewise, DAML-S has the expressive power to encapsulate the composition of several services within a single service description. Likewise, DAML-S has the expressive power to encapsulate the composition of several services within a single service description. If an agent could enact a composite service as a behavior, it is intuitive that this will expand the agent's first-order abilities. Expanded first-order abilities will help the agent preserve its autonomy by reducing its social dependencies. As agents reduce their social dependencies, they create efficiencies across the operating environment. In effect, this approach is analogous to business process reengineering whose typical goal is to reduce transactional costs while providing the same or better service. Providing agents the capability to enact services described in DAML-S streamlines the workflow, thereby increasing the agent's goal-attaining efficiency by reducing the need for cooperative problem solving in multi-agent environments.

In DAML-S, a composite service can be recursively decomposed into a set of atomic services. Control constructs are provided by DAML-S to orchestrate the services that compose the workflow. These constructs are shown in Table 2.

Table 2. DAML-S Control Constructs [39].

Construct	Description
Sequence	Execute a list of processes in a sequential order
Concurrent	Execute elements of a bag of processes concurrently
Split	Invoke elements of a bag of processes
Split+Join	Invoke elements of a bag of processes and synchronize
Unordered	Execute all processes in a bag in any order
Choice	Choose between alternatives and execute one
If-Then-Else	*If* specified condition holds, execute *Then* else execute *Else*
Repeat-Until	Iterate execution of a bag of processes *Until* a condition holds
Repeat-While	Iterate execution of a bag of process *While* a condition holds

An examination of Table 2 hints at the potential complexity that a DAML-S described workflow might contain, including capabilities for concurrent execution and synchronization. Enactment of DAML-S described workflows is a difficult problem that has not been extensively studied; however, some initial work has been done. The DAML-S execution semantics presented in [10], were inspired by Milner's π–calculus. The π–calculus is useful for modeling systems of concurrent, communicating and mobile processes [35]. Fortunately, Milner's work has also been inspirational to the development of a composition language named Piccola.

4.2 Piccola

The development of specialized programming languages for expressing the composition of components is a recurring idea. Early evidence is provided by the utility attributed to UNIX shell scripting. The pipes and filters architecture of the UNIX shell in combination with a scripting language demonstrate the power of flexible composition via the pipelining of streams and commands.

In [31], the authors introduce the rationale and requirements for a general purpose composition language. The authors describe a composition language as providing the integration framework between the computational and compositional views of a system. The composition language requirements proposed by the authors are designed to support open systems development, where openness is characterized by the need for recomposability in the face of changing system requirements. The authors propose the development of a composition language using the π-Calculus as a theoretical foundation. This requirements groundwork ultimately results in the publication of [8, 9, 26], which describe the composition language named Piccola. A platform neutral implementation of the Piccola language exists, it is Java-based and is named JPiccola.

5. AGENTCITIES RELATED RESEARCH

The Agentcities Initiative intends to provide a platform to demonstrate the interoperation of independently authored agents that are geographically dispersed and executing within heterogeneous environments. The interoperation is accomplished through the use of open systems technologies and protocols. The protocols utilized in the Agentcities framework are those defined by the FIPA standards. It is the intent of the authors to use Agentcities as a research platform for the delivery of contextually appropriate Web Services, where the context is defined by geographic location. The proposed research overlaps well with the research goals of Agentcities as defined in [1]; specifically listed is the desire to investigate the "seamless interaction between wireless and wire line agents to dynamically compose services based on user location" [1, pg 14]. The proposed research also aligns with the objectives of three of the proposed Agentcities working groups. These three groups are: Engineering Self-Organizing Applications WG, Service Description and Composition in Agentcities WG, and Ontologies and Semantics WG.

The architecture for the proposed research is found in Figure 1. The architecture is designed for scalability, from mobile PDA devices with wireless connectivity to resource-rich server class systems. The architecture is designed to be compatible with existing and emerging open standards; as such interoperability within open agent societies and Web Services is maximized.

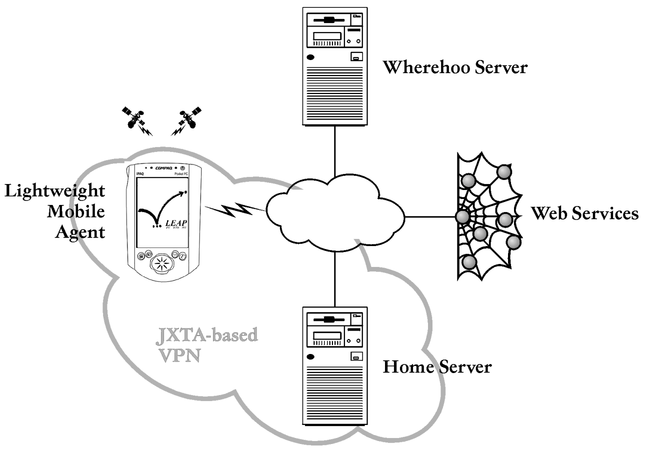

Figure 1. Proposed Architecture.

The major components of the architecture are:

- a Lightweight Mobile Agent implemented with LEAP [11]. The platform is an IPaq 3675 with a dual slot expansion pack; the expansion pack will hold an 802.11b wireless network card and a GPS receiver.
- the Wherehoo server [44] will store DAML-S descriptions of services associated with specific geographic locations. Wherehoo will return contextually appropriate DAML-S descriptions based upon the physical location of the mobile device.
- a Home Server will provide a Charleston, South Carolina, USA node to the Agentcities network, DAML-S to Piccola translation services, and a Piccola execution.
- the web of services will provide a dynamic set of behaviors for use by the mobile agent.

Operationally, the mobile agent will receive its absolute GPS position from the onboard GPS receiver. The location will be consumed by an internal behavior that will communicate with the Wherehoo server. The Wherehoo server will return a set of DAML-S descriptions for services that are appropriate within a physical region. The region is defined as a circle with selectable radius, whose center point is the current location. Each of the DAML-S descriptions will be passed to the Home Server where they will be transformed into Piccola programs. . It is anticipated that the transformation will leverage the Transformation API for XML (TrAX) [7]. A Piccola execution engine will execute the programs on the Home Server. The executing Piccola programs will communicate with the mobile agent via JXTA protocols and unidirectional pipes; JXTA provides the discovery services to allow the mobile agent to find the pipe end-points on the Home Server. The result is the delivery of contextually appropriate Web Services to the mobile agent who views them as semantically described behaviors.

Alternatively, the Piccola processes could be wrapped in an agent and registered with the Home Server's Agent Management System (AMS). The Home Server's Directory Facilitator (DF) could then be used to link the mobile agent with its agent-based behavior. However, when the behavior is no longer required, the mobile agent cannot teardown the remote agent without violating its autonomy. It is also intended that the enacted DAML-S descriptions are private, internal behaviors of the mobile agent. Advertising their existence via the DF defeats this intent. Although the mobile agent and Piccola processes communicate as peers, a master-slave relationship exists between them. When the mobile agent no longer requires a behavior, it will send a teardown request to the Piccola process, which will end execution.

5.1 Limitations

The centerpiece of the research is the DAML-S to Piccola transformation. A robust translation service would prove useful in numerous domains; however, this work is focused on a subset of DAML-S known as DAML-S Core [10]. It should also be noted that the proposed architecture leverages the Wherehoo Server for DAML-S discovery. Although this mechanism is suitable for the described system, its usefulness will be limited in other domains.

6. REFERENCES

[1] AgentLink. AgentLink News, Issue 8, November 2001, http://www.agentlink.org.

[2] searchEBusiness.com. Business Process Reengineering, http://searchebusiness.techtarget.com/sDefinition/0,290660,sid19_gci536451,00.html.

[3] The DAML Services Coalition. DAML-S Related Work, http://www.daml.org/services/daml-s/2001/10/survey-f-release.pdf.

[4] The DAML Services Coalition. DAML-S: A Semantic Markup For Web Services, http://www.daml.org/services/daml-s/2001/10/daml-s.pdf.

[5] Workflow Management Coalition. Introduction to the Workflow Management Coalition, http://www.wfmc.org/about.htm.

[6] Web Services Interoperability Organization. Introduction to WS-I, http://www.ws-i.org/AboutUS.aspx.

[7] The Apache XML Project. Transformation API for XML, http://xml.apache.org/xalan-j/trax.html.

[8] Achermann, F., Lumpe, M., Schneider, J.-G. and Nierstrasz, O. Piccola - A Small Composition Language. in Bowman, H. and Derrick, J. eds. *Formal Methods for Distributed Processing: A Survey of Object-Oriented Approaches*, Cambridge University Press, New York, NY, 2001, 403-426.

[9] Achermann, F. and Nierstrasz, O. Application = Components + Scripts - A tour of Piccola. in Aksit, M. ed. *Software Architectures and Component Technology*, Kluwer Academic Press, 2000.

[10] Ankolekar, A., Huch, F. and Sycara, K. Concurrent Execution Semantics for DAML-S with Subtypes. In *The First International Semantic Web Conference (ISWC)*, 2002.

[11] Bergenti, F. and Poggi, A. LEAP: A FIPA Platform for Handheld and Mobile Devices. In *Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, 2001.

[12] Boehm, B. and Basili, V. Gaining Intellectual Control of Software Development. *IEEE Computer*, *33*(5):27-33, 2000.

[13] The XML Cover Pages. Web Services Conversation Language (WSCL), http://xml.coverpages.org/wscl.html.

[14] The XML Cover Pages. Web Services Flow Language (WSFL), http://xml.coverpages.org/wsfl.html.

[15] The XML Cover Pages. XLANG, http://xml.coverpages.org/wscl.html.

[16] Curbera, F., Mukhi, N. and Weerawarana, S. On the Emergence of a Web Services Component Model. In *Workshop on Component-Oriented Programming 2001*, 2001.

[17] Glass, G. *Web Services, Building Blocks for Distributed Systems*. Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[18] Griss, M. Software Agents as Next Generation Software Components. in Heineman, G.T. and Councill, W.T. eds. *Component-based software engineering: putting the pieces together*, Addison-Wesley, Boston, 2001, 641-657.

[19] Griss, M.L. My Agent Will Call Your Agent. *Software Development Magazine*, *8*(2), 2000.

[20] Griss, M.L. and Pour, G. Accelerating Development with Agent Components. *IEEE Computer*, *34*(5):37-43, 2001.

[21] Heineman, G.T. and Councill, W.T. Definition of a Software Component and Its Elements. in Heineman, G.T. and Councill, W.T. eds. *Component-Based Software Engineering : Putting the Pieces Together*, Addison-Wesley, Boston, 2001, 5-19.

[22] Herzum, P. and Sims, O. *Business Component Factory : A Comprehensive Overview of Component-Based Development for the Enterprise*. John Wiley, New York, 2000.

[23] Huhns, M.N. and Stephens, L.M. Multiagent Systems and Societies of Agents. in Weiss, G. ed. *Multiagent Systems: A Modern Approach to Distributed Artifical Intelligence*, MIT Press, Cambridge, MA, 1999, 79-120.

[24] Jennings, N.R. An Agent-Based Approach for Building Complex Software Systems. *Communications of the ACM*, *44*(4):35-41, 2001.

[25] Jennings, N.R. On agent-based software engineering. *Artifical Intelligence*, *177*(2000):277-296, 2000.

[26] Lumpe, M., Achermann, F. and Nierstrasz, O. A Formal Language for Composition. in Leavens, G. and Sitaraman, M. eds. *Foundations of Component-Based Systems*, Cambridge University Press, New York, 2000.

[27] Maamar, Z. and Sutherland, J. Toward Intelligent Business Objects. *Communications of the ACM*, *43*(10):99-101, 2000.

[28] Malsch, T. Naming the Unnamable: Socionics or the Sociological Turn of/to Distributed Artificial Intelligence. *Autonomous Agents and Multi-Agent Systems*, *4*(3):155-186, 2001.

[29] Meyers, B.C. and Oberndorf, P. *Managing software acquisition : open sytems and COTS products*. Addison-Wesley, Boston, 2001.

[30] Judith M. Myerson. Web Service Architectures: How They Stack Up, http://www.webservicesarchitect.com/content/articles/myerson01.asp.

[31] Nierstrasz, O. and Meijler, T. Requirements for a Composition Language. in Ciancarini, P., Nierstrasz, O.M. and Yonezawa, A. eds. *Proceedings of the ECOOP '94 Workshop on Models and Languages for Coordination of Parallelism and Distribution, LNCS 924*, Springer, New York, 1995, 147-161.

[32] O'Riordan, D., Business Process Standards For Web Services, Tect,

[33] Dan Ruby. XML and Web Services Magazine. Speaking Up for Interop, http://www.fawcette.com/smlmag/2002_04/magazine/departments/ednote/.

[34] Sametinger, J. *Software Engineering with Reusable Components*. Springer-Verlag, New York, 1997.

[35] Sangiorgi, D. and Walker, D. *The pi -calculus : a theory of mobile processes*. Cambridge University Press, Cambridge, England ; New York, 2001.

[36] Singh, M.P. and Huhns, M.N. Multiagent Systems for Workflow. *International Journal of Intelligent Systems in Accounting, Finance and Management*, *8*:105-117, 1999.

[37] Subrahmanian, V.S., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F. and Ross, R. *Heterogeneous Agent Systems*. MIT Press, Cambridge, MA, 2000.

[38] Szyperski, C. *Component Software : Beyond Object-Oriented Programming*. ACM Press, New York, 1998.

[39] The DAML Services Coalition. DAML-S: Web Service Description for the Semantic Web. In *The First International Semantic Web Conference (ISWC)*, 2002.

[40] 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.

[41] Wegner, P. Interoperability. *ACM Computing Surveys*, *28*(1):285-287, 1996.

[42] Wooldridge, M. Agents and Software Engineering. *AI*IA Notizie*, *XI*(3):31-37, 1998.

[43] Wooldridge, M.J. *Reasoning about rational agents*. MIT Press, Cambridge, Mass., 2000.

[44] Jim Youll. Wherehoo Technical Overview, http://www.wherehoo.org/about/wherehoo_technical.html.