The Representation of Context in Computer Software

Hisham Assal[*], Kym Pohl[**], and Jens Pohl[*]

[*]Collaborative Agent Design Research Center, California Polytechnic State University
[**]CDM Technologies, Inc.
San Luis Obispo, California, USA

Abstract

Computers do not have the equivalent of a human cognitive system and therefore store data simply as the numbers and words that are entered into the computer. For a computer to interpret data it requires an information structure that provides at least some level of *context*. This can be accomplished utilizing an *ontology* of objects with characteristics, semantic behavior, and a rich set of relationships to create a virtual version of real world situations and provide the *context* within which *intelligent* logic (e.g., *agents*) can automatically operate.

This paper discusses the process of developing ontologies that serve to provide context for agents to interpret and reason about data changes in decision-support software tools, services and systems.

Keywords: agents, CMAP Tools, context, semantic model, data, information, information-centric, knowledge, ontology, OWL, Protégé, use-case, UML, WordNet.

The Need for Context

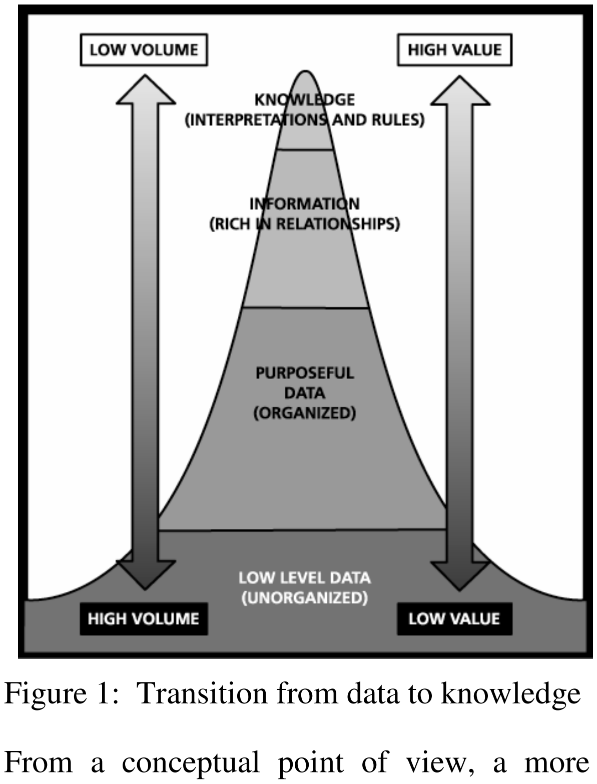
The design of any information system architecture must be based on the obvious truth that the only meaningful reason for capturing and storing data is to utilize them in some planning or decision-making process. However for data to be useful for planners and decision makers they have to be understood in context. In other words, data are just numbers and words that become meaningful only when they are viewed within a situational framework. This framework is typically defined by an expressive *blueprint* rich in associations that relate data items to each other and peripheral factors that influence the meaning of the data in a particular situation. Succinctly stated, numbers and words (i.e., data) found within a rich, structured set of relationships become information, which provides the necessary context for interpreting the meaning of the data, the recognition of patterns, and the formulation of rules, commonly referred to as knowledge (Pohl 2003, 1-3)

The larger an organization the more data it generates, both by itself as well as captured from external sources. With the availability of powerful computer hardware and database management systems the ability of organizations to store and order these data in some purposeful manner has dramatically increased. However, at the same time, the expectations and need to utilize the stored data in monitoring, planning and time-critical decision-making tasks has become a major human resource intensive preoccupation. In many respects this data-centric focus has become a bottleneck that inhibits the ability of the organization to efficiently and effectively accomplish its mission.

The reasons for this bottleneck are twofold. First, large organizations are forced to focus their attention and efforts on the almost overwhelming tasks involved in converting unordered data into purposefully ordered data (Figure 1). This involves, in particular, the establishment of gateways to a large number of heterogeneous data sources, the validation and integration of these sources, the standardization of nomenclatures, and the collection of data elements into logical data models.

Second, with the almost exclusive emphasis on the slicing and dicing of data, rather than the capture and preservation of relationships, the interpretation of the massive and continuously increasing volume of data is left to the users of the data (Figure 2). The experience and knowledge stored in the human cognitive system serves as the necessary context for the interpretation and utilization of the ordered data in monitoring, planning and decision-making processes. However, the burden imposed on the human user of having to interpret large amounts of data at the lowest levels of context has resulted in a wasteful and often ineffective application of valuable and scarce human resources. In particular, it often leads to late or non-recognition of patterns, overlooked consequences, missed opportunities, incomplete and inaccurate assessments, inability to respond in a timely manner, marginal decisions, and unnecessary human burn-out.

These are symptoms of an incomplete information management environment. An environment that relies entirely on the capture of data and the ability of its human users to add the relationships to convert the data into information and thereby provide the context that is required for all effective planning and decision-making endeavors.


LOW VOLUME
HIGH VALUE
KNOWLEDGE (INTERPRETATIONS AND RULES)
INFORMATION (RICH IN RELATIONSHIPS)
PURPOSEFUL DATA (ORGANIZED)
LOW LEVEL DATA (UNORGANIZED)
HIGH VOLUME
LOW VALUE

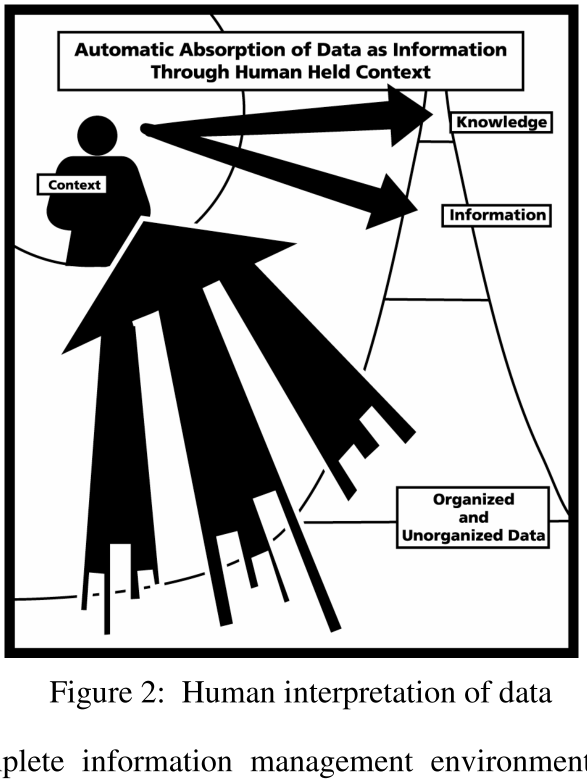

Automatic Absorption of Data as Information Through Human Held Context
Knowledge
Context
Information
Organized and Unorganized Data

Figure 1: Transition from data to knowledge

Figure 2: Human interpretation of data

From a conceptual point of view, a more complete information management environment considers data to be the bottom layer of a three-layer architecture, namely: a data layer; a mediation layer; and, an information (i.e., semantic) layer. The *Data Layer* is responsible for integrating heterogeneous data sources into accessible and purposefully ordered data. It typically includes a wide variety of repositories ranging from simple textual files to databases, Data Portals, Data Warehouses, and Data Marts.

The *Mediation Layer* defines the structure of the data sources (i.e., logical data models), data transfer formats, and data transformation rules. The two principal purposes of the Mediation Layer are to facilitate the automated discovery of data, the reconciliation (i.e., merging) of data, and finally the mapping of such unified data to information. In other words, the Mediation Layer serves as a registry for all definitions, schemas, protocols, conventions, and rules that are required to recognize data within the appropriate *context*. The Mediation Layer also serves as a translation facility for bridging between data with structural relationships (e.g., based on a logical data model) and information that is rich in contextual relationships.

The *Information Layer* consists of many functionally oriented planning and decision-assistance software applications and services. Typically, these semantic capabilities are based on internal information models (i.e., object models or ontologies) that are virtual representations of particular portions of the real world context. By providing context, the internal information model of each application is able to support the automated reasoning capabilities of rule-based software agents.

In such a three-layered information management environment the Mediation Layer continuously populates the information models of the applications in the Information Layer with the data changes that are fed to it by the Data Layer. This in turn automatically triggers the reasoning capabilities of the software agents. The collaboration of these agents with each other and the human users contributes a powerful, near real-time, adaptive decision-support environment. The agents can be looked upon as intelligent, dynamic tools that continuously monitor changes in the real world. They utilize their reasoning and computational capabilities to generate and evaluate courses of action in response to both real world events and user interactions. As a result the human user is empowered by agent-based assistance while at the same time relieved of many of the lower level filtering, analysis, and reasoning tasks that are a necessary part of any useful planning and problem solving process. A vital enabler of these benefits is the ability of the software agents to continuously and tirelessly monitor the real world execution environment for changes and events that may impact current or projected plans.

Definition of Terms

The following brief explanation of key terms and concepts referred to in this paper is provided as an introduction for clarification purposes.

Ontology: The term ontology is loosely used to describe an information structure, rich in relationships that provides a virtual representation of some real world environment (e.g., the context of a problem situation such as the management of a transport corridor, the loading of a cargo ship, the coordination of a military theater, the design of a building, and so on). The elements of an ontology include objects and their characteristics, different kinds of relationships among objects, and the concept of inheritance. Ontologies are also commonly referred to as *object models*. However, strictly speaking the term ontology has a much broader definition. It actually refers to the entire knowledge in a particular field. In this sense an ontology would include both an object model and the software agents that are capable of reasoning about information within the *context* provided by the object model (since the agents utilize business rules that constitute some of the knowledge within a particular domain).

Information and context: Information refers to the combination of data with relationships to provide adequate *context* for the interpretation of the data. The richer the relationships the greater the *context* (i.e., meaning conveyed by the combination of data with relationships), and the more opportunity for automatic reasoning by software agents.

Information-centric: Software that incorporates an internal information model, such as an ontology, is often referred to as *information-centric* software. The information model is a virtual representation of the real world domain under consideration and is designed to provide adequate *context* for software agents (typically rule-based) to reason about the current state of the virtual environment. Since information-centric software has some understanding of what it is processing it normally contains tools rather than predefined solutions to predetermined problems. These tools are commonly software agents that collaborate with each other and the human user(s) to develop solutions to problems in near real-time, as they occur. Communication between information-centric applications is greatly facilitated since only the changes in information need to be transmitted. This is made possible by the fact that the object, its characteristics and its relationships are already known by the receiving application.

Agents: This term has been applied very loosely in recent years. There are several different kinds of agents. Symbolic reasoning agents are most commonly associated with knowledge management systems. These agents may be described as software modules that are capable of reasoning about events (i.e., changes in data received from external sources or as the result of internal activities) within the context of the information contained in an internal information model (i.e., ontology). The agents collaborate with each other and the human users as they monitor, interpret, analyze, evaluate, and inform users of emerging issues or plan alternative courses of action.

Identifying the Purpose of the Ontology

The objective(s), or intended purpose, of the ontology must be defined in some formal manner in order to facilitate the development process. Without a well-defined purpose of the ontology, development can continue with no apparent end-state, and the ontology can grow in different directions beyond the control of system developers. Some common purposes of ontologies include the representation of knowledge in a given domain of interest, facilitating communication among system components, re-use by other applications, or as a common language for multiple systems within the same domain.

A good way for defining the purpose of the ontology is by means of use-cases[1]. The utility an ontology must provide can be broken down into specific, well-defined use-cases, in which actors and actions are identified, as well as the perceived components that will be involved in each

[1] In software development a use-case describes a process or scenario from the point of view of the actors involved in the process (Cockburn A. (2001); 'Writing Effective Use Cases'; Addison-Wesley, Reading, Massachusetts.)

action. Another tool for identifying the purpose of an ontology is a set of questions, which the ontology should be equipped to answer.

The ontology is complete, in the context of a given set of requirements, when all the use-cases are supported by ontology concepts and all the questions needed to be asked can be answered by the ontology.

Building the Ontology

Once the domain knowledge has been captured in free form, it is time to start building the ontology. This is the step, where the captured knowledge is formalized and concepts are given specific descriptive names to allow the communication with other stakeholders. The process of building the ontology can be described in the following steps:

1. Capture the knowledge in the domain of interest. Many knowledge acquisition techniques can be applied in this step, including textbooks, interviewing subject matter experts (SME), databases of case studies, analysis reports, and so on. One of the primary methods of capturing knowledge in this domain is utilizing a subject matter expert to formalize the concepts and produce the model. Other sources of knowledge assist the expert in this task, such as books, military manuals, past plan analyses, and training material.
2. Identify the key concepts and relationships in the domain of interest. The key concepts are the ones that relate to the identified purpose of the ontology. They typically answer critical questions or contribute to the communication among system components and concepts that are involved in actions of use-cases. Other concepts that help to relate key concepts to each other or add details to key concepts, are considered supporting concepts. For example, the key concepts in a human factors ontology are likely to be: Person, Organization, Communication, Personal Traits, and Behavioral Traits.
3. Produce precise textual definitions of such concepts and relationships. The textual definitions help disambiguate the concepts and define their role in the ontology. Existing textual definitions in standard lexicons can help in this step. For example, the WordNet[2] database offers a good electronic resource for common definitions of English language terms. The use of a lexicon like WordNet also facilitates the search for terms and their synonyms, for the purpose of analyzing free text. Other specialized lexicons, such as military manuals, can also be a good source for accepted definitions for common terms.
4. Identify terms to refer to such concepts and relationships. The selection of meaningful ontology terms helps developers understand the role of each concept and possibly the common uses of it. Also system developers do not need to go back to the formal definition of each term every time they need to

[2] WordNet is an electronic lexical database developed by researchers at Princeton University (Fellbaum C. (ed.); 'WordNet: An Electronic Lexical Database', Bradford Book Series, MIT Press, Cambridge, Massachusetts.) and http://wordnet.princeton.edu/perl/webwn3.0?s=word-you-want

use it. The selected terms should be expressive of the concept and close to its natural language description.

5. Obtain agreement on all of the above. It is important for all stakeholders to agree on the selection of concepts and the terms used to refer to them in the ontology. Ontology-based systems are typically a collaborative effort, often among multiple organizations. To facilitate communication among all participants there has to be agreement on the ontology.
6. Select a representational methodology (e.g., Protégé[3], UML[4], etc.). Modeling of ontologies is a step to formalizing the captured knowledge and producing an artifact, which can communicate that knowledge to other stakeholders. Most modeling methods have a graphical notation to easily connect concepts and navigate through the ontology. The criteria for selection of a modeling method are:
    - *Coverage:* Does the crafted model provide enough elements to represent all of the captured concepts and the types of relationships that exist among them?
    - *Granularity:* How much detail can the modeler represent in a concept?
    - *Learning curve:* Is this modeling method a standard method, which modelers are already familiar with? Or is it a new method that requires investment of time and effort to learn to use efficiently?

    Protégé is the modeler of choice for OWL[5]-based ontologies. There are other tools that support OWL development, such as Concept Maps, but the support that is offered by Protégé is stronger in visualization and ontology navigation.

Coding the Ontology

The implementation of ontology-based systems requires translating the ontology model into an implementation language. The language chosen for coding an ontology (e.g., formal logic, UML, OWL, etc.) has to provide the following characteristics:

- *Conceptual distance:* The ability of the language to represent abstract concepts at multiple levels of abstraction.

[3] Protégé is an *Open Source* ontology editor and knowledge-base framework that is extensible and based on Java (see: http://protege.stanford.edu/users.html)

[4] UML (Unified Modeling Language) provides a standard notation for modeling systems and context utilizing object-oriented concepts and principles (Booch G., J. Rumbaugh and I. Jacobson (1999); 'The Unified Modeling Language User Guide'; Addison-Wesley, New York, New York.)

[5] OWL (Web Ontology Language) facilitates the machine processing of the content of data through software (McGuinness D. and van Harmelen F. (2004); 'OWL Web Ontology Language Overview'; MIT Press, Cambridge, Massachusetts.)

- *Expressive power:* The ability to represent complex concepts with consistent language constructs.

- *Standards compliant:* The language should follow accepted standards and notations to allow for better communication among development team members.
- *Translatability:* The language constructs have formal structures that can be converted to forms in other languages, without ambiguity.
- *Guidelines:* The model development process is supported by a set of guidelines and best practices.
- *Formal semantics:* The intended meaning of each language construct is unambiguous and well-defined.
- *Flexibility:* The ability to represent concepts in different ways, using different constructs.
- *User base:* The availability of user groups provides support for ontology development, through the exchange of experiences and best practices.
- *Availability:* The language has to be available, preferably, in the public domain, along with tools to support its use.

For example, the selected coding language may be OWL to facilitate communication with other system developers, especially in the case of a multi-organization effort. OWL satisfies many of the selection criteria mentioned above.

- *Conceptual distance:* OWL allows the representation of abstract concepts, maintaining its level of abstraction and allowing for details as needed.
- *Expressive power:* OWL employs description logic in a dynamic environment utilizing the open world assumption. Description logic is a powerful mechanism for stating concepts.
- *Standards:* OWL is based on RDF, which is a standard that is becoming more popular with many tools for processing formats.
- *Translatability:* As a formal language with well-defined semantics, OWL can be translated into other implementation languages, especially RDF-based languages. The degree to which the translation preserves all of the ontology features depends on the target language and its supported features.
- *Formal semantics:* OWL has well-defined semantics for language constructs. The semantics capability is supported by *Reasoner* specifications that describe what a valid structure should be.
- *Flexibility:* OWL offers a wide range of constructs to model concepts and relationships. In most cases, the modeler provides multiple choices to model any concept. The selection of a particular construct is usually determined by the use-cases for the concept and the relationships to other concepts.

- *User base:* OWL enjoys a strong user base for OWL, especially using the modeler Protégé. There are also many conferences and user group meetings and on-line forums supporting the development of ontologies in OWL.
- *Availability:* The OWL specification is published in the public domain and tools for modeling in OWL are available for free (e.g., Protégé and CMAP Tools[6]).

The next step is to translate the model into actual system implementation. Two aspects need to be addressed in this step, namely: verification tools; and, code generation. Checking tools are needed to make sure that the ontology structure is consistent and remains consistent during system operation, after changes have been made. Code generation tools assist in taking a formal ontology consistently and repeatedly from a formal language to an implementation language. System implementation typically goes through multiple iterations that may require re-writing the basic model, or large sections of it. Utilizing code generation tools makes this task easier.

Integration of the Ontology with Existing Ontologies

It is often the case that an ontology is being developed as an extension of an existing ontology or to connect with an existing ontology. In such cases, integration with the existing ontology must be carefully considered.

- Existence of other ontologies that are relevant to this ontology.
- All assumptions have to be made explicit.
- Agreement has to be achieved regarding concepts and relationships.

Analysis and Evaluation

The ontology must be examined from a technical perspective, along with the associated software environment, and the documentation with respect to a frame of reference, which includes:

- Requirements specifications.
- Competency questions.
- Real world appropriateness.

The selection of the frame of reference and the evaluation criteria have to align with the purpose and requirements of the ontology. The semantic correctness of an ontology is crucial for the proper functioning of applications. In order to evaluate an ontology it is useful to employ a methodology that has two main components: structural analysis; and, domain knowledge analysis.

[6] IHMC CMAP Tools facilitates the construction, navigation, and sharing of knowledge models represented as concept maps (see: http://cmap.ihmc.us/download/)

Structural Analysis: This involves the analysis of the structure of concepts in terms of hierarchy (taxonomy) and in terms of the relationships among concepts. The main criteria for this analysis are:

- *Uniqueness of concepts (no redundancy):* Every relevant concept in the domain should be represented in a clear and concise manner within the model. Concepts that are similar or have some common properties with other concepts should be represented in relationship to the existing concepts, either in a class hierarchy or through other types of relationships such as "part-of". The ease with which existing concepts co-exist and new ones can be added is a key indicator of the model's elegance and sophistication.
- *No circular reference should exist at any level:* Circular references can occur when a parent class in a class hierarchy inherits from a child class at any level down the hierarchy. This circular reference may not be obvious if the child class is more than two levels down from the parent class. Circular references are problematic because they confuse the semantics of the two concepts (e.g., "… a jet plane is a kind-of aircraft" and "… an aircraft is a kind-of jet plane").
- *Levels of Abstraction:* Class hierarchies can have any number of levels, where every level introduces more details to the classes at that level. The choice to add many attributes to a class in one level of the hierarchy or to create many levels with few attributes at each level has implications on the semantics of the model and on the operational aspects of applications that use this model.
- *Complexity (number of concepts + number of relationships for each concept):* The complexity of an ontology plays an important role in its usability. Applications typically traverse a collection of related concepts to form a context for reasoning or decision making. The more complex the ontology, the more involved it becomes for the application (and for the application developer) to form the proper context.

Domain Knowledge Analysis: Focuses on the *purpose* of the ontology. Use-cases for the application identify its information needs and form the basis for assessing the ontology's completeness. The criteria for this analysis are:

- *Coverage of use-cases (completeness):* Application use-cases define the different ways the ontology will be used. All concepts that are referenced in the target use-cases must exist in the ontology in some form (either directly or inferred). Other concepts not explicitly mentioned in any use-case may exist in the ontology serving as extended specifications for further reasoning or increased scope.
- *Partitioning:* The arrangement of classes in a hierarchy, where features of subclasses do not overlap, forms a *disjoint decomposition* of classes. When subclasses represent all the possible classifications of a super class, then this is called *exhaustive decomposition*. In this case, any instance of the super class is also an instance of one of the subclasses. These two properties of

ontology partitioning (i.e., disjoint decomposition and exhaustive decomposition) place integrity constraints on the ontology and provide for tighter semantics, as well as a more powerfully expressive ontology that in turn leads to more straightforward reasoning capabilities.

- o *Extensibility:* When the incorporation of additional concepts is required, perhaps due to the need to support additional use-cases, it should be possible to add these concepts without the need to re-structure the entire ontology. If engineered correctly, the incorporation of extended or entirely new concepts can be achieved in a fairly isolated manner without unduly impacting unrelated areas of the model or actual model users.
- o *Documentation:* The intended meaning and the usage of each concept must be clearly documented, so that reasoning facilities can effectively and appropriately employ them.

Documentation of the Ontology

The development of software components that are based on an ontology relies on good documentation of the ontology and the availability of the documentation to all developers. The documentation must include:

- Purpose and intended use of the ontology.
- Assumptions made at every level about concepts and their relationships.
- Primitives used to express the definitions (i.e., meta-ontology).
- Relationship to existing ontologies.

Using a lexicon such as WordNet standardizes the definitions across multiple developer teams and across organizations, and reduces the chances for ambiguity in dealing with concepts that may have multiple word-senses. The choice of WordNet also offers the opportunity for other ontologies to integrate with the ontology under consideration, by examining the standard definition of its concepts and deciding on concept compatibility.

References

Barber K., A. Goel, D. Han, J. Kim, D. Lam, T. Liu, M. MacMahon, C. Martin and R. McKay (2003); 'Infrastructure for Design, Deployment and *Experimentation* of Distributed Agent-based Systems: The Requirements'; The Technologies, and an Example, Autonomous Agents and Multi-Agent Systems. Volume 7, No. 1-2 (pp 49-69).

Brown P. (2008); 'Implementing SOA: Total Architecture in Practice'; Addison-Wesley.

Erl T. (2008); 'SOA: Principles of Service Design'; Prentice Hall.

Fahad M. and M. Abdul Qadir (2008); 'A Framework for Ontology Evaluation'; Proceedings International Conference on Conceptual Structures (ICCS), Toulouse, France, July 7-11 (pp. 149-158).

Forgy C. (1982); 'Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem'; Artificial Intelligence, 19 (pp. 17–37).

Jennings N., K. Sycara and M. Wooldridge (1998); 'A Roadmap of Agent Research and Development'; Autonomous Agents and Multi-Agent Systems, Vol. 1 (pp. 7-38).

Lu Q. and V. Haarslev (2006); 'OntoKBEval: A Support Tool for DL-based Evaluation of OWL Ontologies'; Proceedings OWL: Experiences and Directions 2006, Athens, Georgia, November 10-11.

Pohl J. (2003); 'USTRANSCOM: Corporate Information-Centric Environment (CICE)'; Report prepared for USTRANSCOM TCJ6-A (available from: CDM Technologies Inc, 2975 McMillan Avenue (Suite 272), San Luis Obispo, California 93401), October.

Supekar K. (2005); 'A Peer-Review Approach for Ontology Evaluation'; Proceedings 8th International Protégé Conference, Madrid, Spain, July 18-21.

Wooldridge M. and N. Jennings (1995); 'Intelligent Agents: Theory and Practice'; The Knowledge Engineering Review, Vol. 10(2) (pp. 115-152).

Wooldridge M., N. Jennings and D. Kinny (1999); 'A Methodology for Agent-Oriented Aanalysis and Design'; Proceedings Third International Conference on Autonomous Agents (Agents-99), Seattle, Washington.