

ICDM: An Architecture and Toolkit in Support of Agent-Based, Decision-Support Applications¹

Kym Jason Pohl

Collaborative Agent Design (CAD) Research Center
California Polytechnic, San Luis Obispo, CA 93407
805-541-3750 x233
kpohl@cadrc.calpoly.edu

Abstract-- Agent-based, decision-support systems provide human decision-makers with a means of solving complex problems through collaboration with heterogeneous collections of both human and computer-based expert agents. Over the past decade the Collaborative Agent Design (CAD) Research Center has developed several proof-of-concept and production-oriented agent-based, decision-support systems for both commercial and Department of Defense applications. These applications range in domain from engineering design to tactical command and control. While diverse in application, each of these systems is predicated on the same set of fundamental principles derived from years of experience in this area. Primary among these principles are the notions of high-level representation, human/computer collaborative partnership, and the development of tools as opposed to predefined solutions. The CAD Research Center has formalized this philosophy into an architectural framework together with a suite of development and execution tools. Collectively, these components are known as the ICDM (Integrated Cooperative Decision Model) framework.

TABLE OF CONTENTS

- INTRODUCTION
- THE ICDM FRAMEWORK
- THE ICDM DEVELOPMENT TOOLKIT
- REFERENCES
- BIOGRAPHY

INTRODUCTION

In modern-day society, the need for an effective means of engaging in collaborative decision making is more prevalent than ever. With the development of newer, agent-

based technologies, this need is only recently beginning to be successfully addressed.

Throughout the past decade the Collaborative Agent Design (CAD) Research Center (CADRC) at California Polytechnic, San Luis Obispo, California has been intensely involved in the design and development of agent-based, decision-support systems from a practical standpoint [11]. As a result of these efforts, the CADRC has developed a manifesto of sorts describing a collection of criteria fundamental to the development of agent-based, decision-support systems [12].

First and foremost on this list is the need for an object-based representation of information. That is, information processed within the system should be described as objects having attributes, behavior, and most importantly relationships. Collectively, these descriptions form an application's information object model or ontology [2]. This requirement not only applies to the modeling of information but at times is even portrayed in the representation of the expert agents themselves. It has been the experience of the CADRC that without such a rich, objectified representation, where critical informational relationships can be captured, determination of information meaning and implication becomes extremely difficult if not impossible.

After numerous implementations it became clear to the members of the CADRC that to take full advantage of such objectified representation, a supportive framework needed to be established. A framework that centered around objects.

THE ICDM FRAMEWORK

The ICDM framework exists as an architecture, together with a set of development and execution tools which can be used to design, implement, and execute agent-based,

¹ 0-7803-6599-2/01/\$10.00 @ 2001 IEEE

decision-support applications. Conceptually, ICDM incorporates all of the core concepts and philosophies developed or adopted by the CADRC over its 15 year involvement in developing decision-support systems.

The ICDM model is based on a three-tier architecture making clear and distinct separations between information, logic, and presentation [3].

These tiers are represented by the three major components comprising the ICDM architecture; a collection of information management servers (i.e., Information Server, Subscription Server, etc. representing the information tier), the Agent Engine representing the logic tier, and the Client User Interface representing the presentation tier (Figure 1). Each of these components functions in an integrated fashion to form a comprehensive agent-based decision-support execution framework. This framework allows multiple human decision-makers to solve complex problems in a collaborative fashion obtaining decision-support assistance from collections of heterogeneous on-line agents.

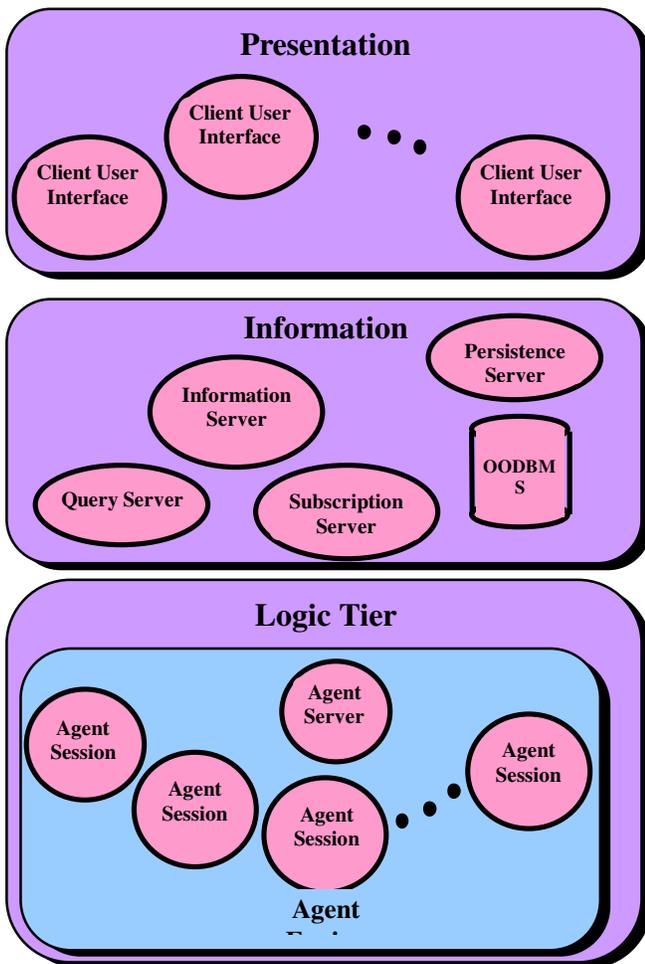


Figure 1 – 3-Tier Architecture

Information Server

Core of the information tier is the Information Server (IS). Conceptually, the IS represents a library of objectified information which clients utilize to both obtain and contribute information. The only difference is that clients can obtain this information in, not only a *pull* fashion (query service), but can also have the IS *push* them information on an interest basis (subscription service). Physically, the IS exists as a collection of distributed object servers based on the Common Object Request Broker Architecture (CORBA) [7].

At the application level, distributed object servers are designed to manage distributed information at the object level. Such management functions have traditionally included object creation and access, modification, and destruction. The knowledge of exactly where the information resides and how it can be retrieved is completely encapsulated inside the object server facility. This means that clients need not be concerned with who has what information and in what form that information exists. This feature becomes instrumental in providing an environment where application components collaborate with each other in an essentially de-coupled manner. Inter-component collaboration occurs via the object-based services offered by the IS.

The post-representation objectification paradox-- Regardless of the information's native representation, distributed object servers can be used to present information to clients in the form of objects. However, this does not discount the need for information to be modeled as high-level objects in its native form portraying behavior and conveying relationships. While on the surface this representational morphing capability of object servers seems promising, in practice this feature proves to be quite misleading. If the information is not represented at a high level upon its conception, such objectification amounts to little more than wrapping data in communicable object shells. For the most part these shells fail to convey any additional insight into the meaning or implication of the information than was present to begin with in its original form. Although in the future there may be potential for successful research efforts in this area, at present, unless information is originally modeled as objects, knowledge-oriented applications prove to gain little from this distributed object server feature. In other words, expecting to elevate the richness of poorly described data or even information solely through the employment of powerful object-based technologies is simply not realistic.

However, applications that do, in fact, model information as high-level objects stand to gain considerably from employing object-based collaborative technologies such as CORBA-based object servers. Distributed object servers preserve purely objectified representations of information as

it moves throughout the system. This is due to the fact that the internal mechanisms of distributed object servers process information as objects themselves.

The ICDM model takes full advantage of these object-oriented facilities by integrating an Object-Oriented DBMS (OODBMS) [1] into its information environment. The OODBMS is the facility that the IS uses to store the application's objects. Employing an OODBMS to store the information objects has two significant advantages.

First, an OODBMS retains the object-oriented representational nature of the information when transferred into a persistent form. Whenever there is representational degradation there is potential for loss of informational content and meaning. By utilizing both access and storage facilities which are capable of processing and manipulating information as collections of objects, there is no degradation of representation as information flows throughout, and is processed by the application environment.

The second advantage of employing an OODBMS relates to the manner in which IS clients request information. Whether mining for information through pointed queries or maintaining dynamic subscription profiles, clients specify their information needs in terms of objects and object characteristics. More specifically these requests are described in terms of object attributes and object relationships. Such queries and subscriptions can range from simple existence criteria to more complex criteria incorporating both logical and relational operators.

As briefly mentioned above, another method in which clients can collaborate with each other is through interaction with the subscription service. Clients can dynamically register a set of standing interests with the Subscription Server. Like queries, these interests are described in terms of the application's object-based ontological system. For example, a client may request to be notified whenever the company InfoTech hires a new employee. Once registered, the Subscription Server continually monitors the truth level of this condition against the application's ever-changing information pool. When satisfied, the Subscription Server essentially *pushes* the results to whomever has indicated interest (i.e., registered an appropriate subscription). To illustrate the efficiency of this approach one only needs to investigate the alternatives. One alternative to this subscription mechanism would be to have interested clients perform the same query on an iterative basis until either such a condition occurs or interest in the condition no longer exists. Each unsatisfied query may potentially decrease resources (i.e., computing cycles) available to other application components and would essentially prove to be a waste of time. On the other hand, if a client takes a more conservative approach where the repeated query is made on a less frequent basis, the

client risks being out of date with the current state of affairs until the next iteration is performed. With this in mind, the incorporation of an asynchronous *push information to interested clients* mechanism becomes essential in providing decision-support applications with an efficient operating environment without jeopardizing information currency.

Agent Engine

The Agent Engine represents the logic-tier of ICDM's underlying three-tier architecture. Existing as a client to the Information Tier services (i.e., access, query, subscription, and persistence) the Agent Engine is capable of both obtaining and injecting information from and into the common information pool. Architecturally, the Agent Engine consists of an agent server capable of serving collections of expert agents (Figure 1). These collections, or Agent Sessions, exist as self-contained, self-managing agent communities capable of interacting with other collaborators via the same set of services offered to all clients (i.e., object, query, subscription, and persistence). For the most part, the exact nature of agents and collaborative model employed in an ICDM-based decision-support system is left to the application specification. However, regardless of the types of agents contained in an Agent Session, agent activity is triggered by changes in application information. This information may take the form of global objects managed by the Information Tier or local objects utilized in agent collaboration which are managed by the Agent Session itself. Regardless of whether agents are interacting with the Information Tier or each other, interaction takes place in terms of objects. This again illustrates the degree to which an object representation is preserved as information is processed throughout the application environment.

Agent Session Configuration-- Breaking agent analysis into heterogeneous collections of agents allows for a number of interesting configurations. These configurations determine the size, number, and individual scope of the agent sessions. While a wide variety of Agent Session configurations is possible, the CADRC has found considerable success in formulating session configuration based on the following two criteria.

The first criterion introduces the notion of a *view*. A view is a conceptual perspective of reality. In a problem-space sense, a view can be thought of as a single investigation into solving a problem whether it is based on fact or speculation. For example, a view may describe events and information relating to what is actually occurring in reality. Yet, another view may describe an alternative or desired reality. An illustration of this approach can be observed in

the Integrated Marine Multi-Agent Command and Control System (IMMACCS) developed by the CADRC for the Marine Corps. IMMACCS uses a single view to represent the information and events occurring in the battle-space. In a similar manner, IMMACCS employs any number of additional views to represent hypothetical investigations to determine suitable strategies for dealing with potential events or circumstances. Regardless of use, however, there is a one-to-one correspondence between a conceptual view and a decision-support Agent Session (Figure 2). This means that independent of exactly which version of reality a view represents, there exists a dedicated Agent Session providing view-users with agent-based analysis and decision-support. Each agent of a particular Agent Session deals only with the section of the information pool relating to the associated view. Partitioning information analysis in this manner allows for an efficient and effective means of distinguishing information and activities relating to one view from those pertaining to another. Unless prompted by user intervention (i.e., user-directed movement of information between views), each view is completely disjoint from the other.

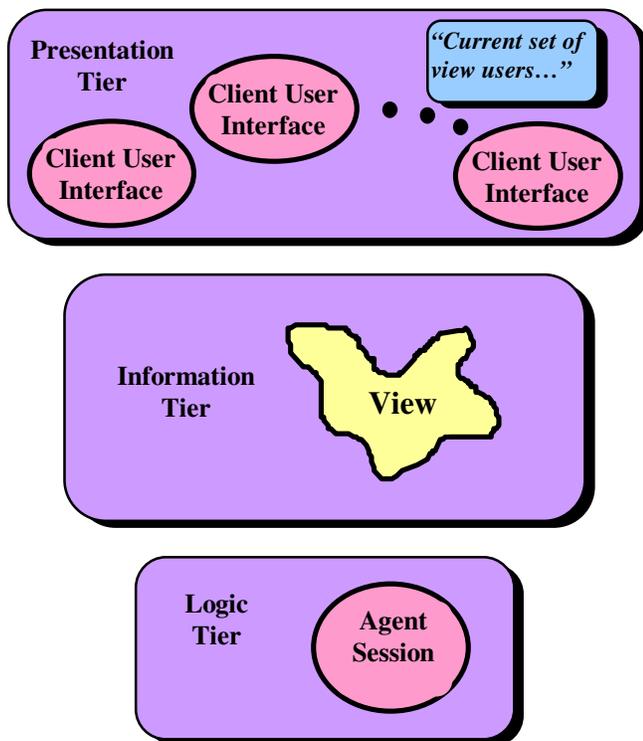


Figure 2 - Multiple users can interact with a view that in turn is analyzed by a single Agent Session

The second configuration criterion employed by the CADRC determines the quantity and nature of agents contained in an Agent Session at any point in time. As mentioned above, the decision-support applications developed by the CADRC utilize a variety of agent types.

Three of these agent types include Domain Agents, Object Agents, and Mediator Agents [13]. Service-oriented Domain Agents embody expertise in various application-relevant domains (i.e., structural systems and thermal dynamics for architectural design, tidal dynamics and trim and stability for ship stow planning, etc.). The collection of Domain Agents populating an Agent Session determines the degree and variety of domain-specific, agent-based decision support available to view users at any given point in time. Under the configuration scheme utilized by the CADRC, users can add or remove these domain perspectives in a dynamic fashion as needs arise or change.

Object Agents, on-the-other-hand extend the notion of high-level informational representation by essentially *agentifying* information. This agentification of information takes place through empowering information objects with the ability to act on their own behalf. Either human users or other agents can initiate agentification.

As one would suspect, in such a dynamically interactive environment there is potential for collaborative conflict and non-convergence. In an attempt to address such occurrences the agent taxonomy employed by the CADRC includes Mediator Agents. As their name suggests Mediator Agents attempt to bring about consensus among agents that may have reached an impasse on a particular issue. As with real world human-based interactions (especially in regard to business-oriented activities) mediators are often employed to assist in clarifying issues as well as perhaps even exposing personal agendas.

Under the ICDM model each of these agent types is dynamically configurable by either the user(s) or the system itself. This approach to Agent Session configuration promotes the notion of offering assistance in the form of dynamically configurable tools rather than predefined solutions [12].

Agent Session Architecture-- Architecturally, an Agent Session consists of several components including the Semantic Network and Semantic Network Manager, Session Manager, Inference Engine, and Agent Manager (Figure 3). These components operate in an integrated fashion to maintain a current information connection between the agents residing in the Agent Session and the associated view described in the Information Tier.

Semantic Network-- The Semantic Network consists of a collection of two sets of application specific information objects. The first set is used for local collaboration among agents. Depending on the specific collaborative model employed, agents may use this local Semantic Network to propose recommendations to each other or request various services. This information is produced and modified by the

session agents and remains local to the housing Agent Session. The second set of information is a sort of duplicate, mirror image of the view information stored in the Information Tier. In actuality, this information exists as a collection of object-based interfaces allowing access to view-related information stored in the information pool. Such interfaces are directly related to the application's

ontological system. Through these interfaces, IS clients have the ability to access and modify objects contained in the Information Tier as though they are local to the client's environment. All communications between the object interfaces and their remote object counterparts are encapsulated and managed by the IS and completely

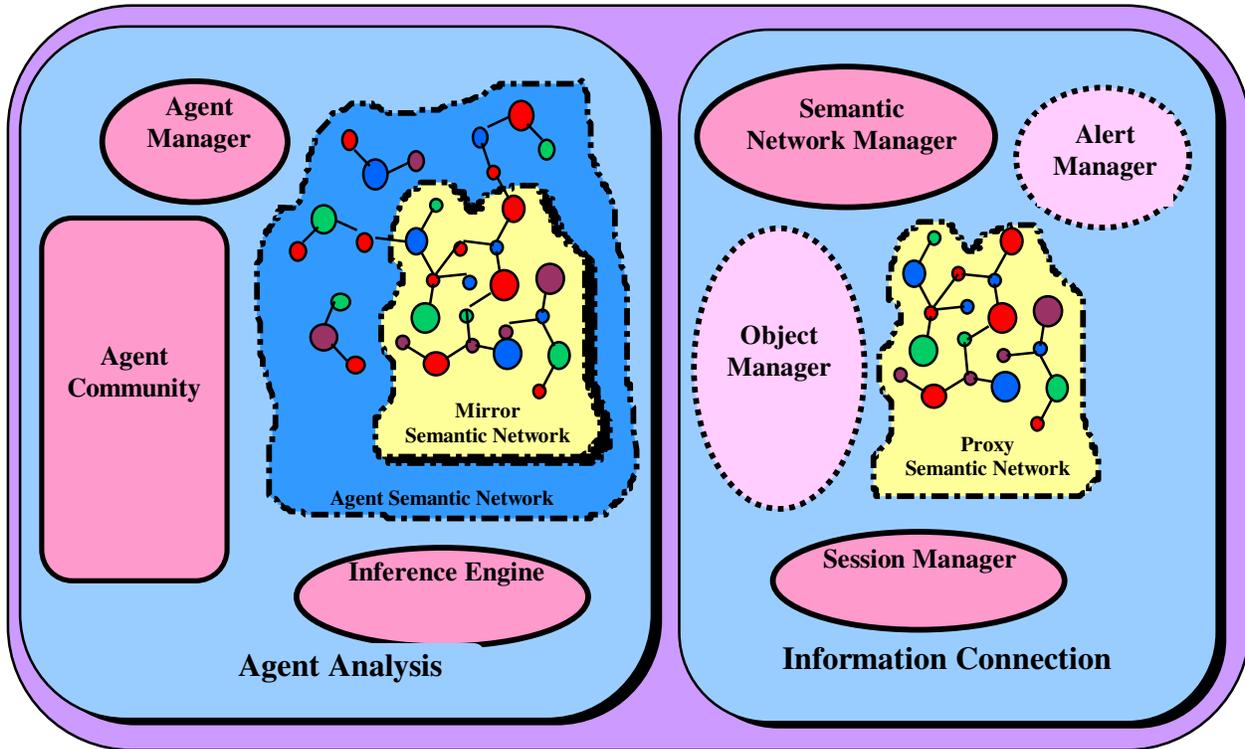


Figure 3 - Agent Session Architecture

transparent to the clients. This is a fundamental feature of distributed object servers on which the IS is based [9]. *Semantic Network Manager*-- As the primary manager of the two sets of information described above, the Semantic Network (SN) Manager focuses the majority of its efforts on the management of the bi-directional propagation of information between IS proxies and an equivalent representation understandable by the Inference Engine. Such propagation is accomplished through employing an Object Manager. The purpose of the Object Manager is to essentially maintain mappings between the object proxies and their corresponding Inference Engine counterparts. The necessity of this mapping reveals a limitation inherent in most distributed object server and inference engine facilities. Most facilities supporting one of these two services require significant control over either the way client information is represented or the manner in which it is generated. This is due to the fact that both facilities require specific behavior to be present in each object they process. Examples of such facilities include IONA's ORBIX distributed object server [5] and NASA's CLIPS

inference engine [8]. Both of these facilities suffer from this limitation. One solution to this dilemma is the use of multiple inheritance allowing the semantic network objects to be derived from multiple sources. Unfortunately, many popular programming languages in use today do not provide support for this feature and is therefore not employed by ICDM. Nonetheless, this dilemma can also be solved through the use of an intermediate object manager who maintains mappings between the two sets of objects. Granted this method incurs some overhead, however, in practice the CADRC has found this associated cost to be quite acceptable.

An additional responsibility of the SN Manager deals with the subscriptions, or interests held on behalf of the agent community housed within the particular Agent Session. That is, the SN Manager is responsible for maintaining the registration of a dynamically changing set of information interests held on behalf of the Agent Session agents. In addition, the SN Manager is responsible for processing notification(s) when these interests are subsequently

satisfied. Such processing includes the propagation of information changes to the agent community that may in turn trigger agent activity. To perform these two interest-related tasks the SN Manager employs the services of the Alert Manager. The Alert Manager provides an interface to the Subscription Server facility and is available to any Information Tier client wishing to maintain a set of information interests. Employment of the Alert Manager by subscribers has two distinct advantages. First, clients are effectively de-coupled from the specifics of the subscription service interface. This allows the same application client to be compatible with a variety of subscription server implementations. Second, the Alert Manager interface allows subscribers to effectively decompose themselves into a dynamic collection of thread-based interest clients [6]. That is, the Alert Manager extends the monolithic one-to-one relationship between the Subscription Server and its clients into one that supports a one-to-many relationship. Such decomposition of functionally related behavior into lightweight processes promotes the concepts of multi-processing in conjunction with resource conservation.

Inference Engine-- The Inference Engine provides the link between changes occurring in the Semantic Network and agent activation. Recall that agent activation can occur when a change in the Semantic Network is of interest to a particular agent. In such a case, the Inference Engine, having knowledge of specific agent rule interests in addition to changes occurring in the Semantic Network is responsible for activating, or scheduling the rule-based action(s) the agent wishes to execute. This activation list forms the basis for the Agent Manager to determine which agent actions to execute on behalf of the currently scheduled agent.

Agent Manager-- The Agent Manager is responsible for the management of the agent community housed in an Agent Session. This management includes the instantiation and destruction of agent identities in the Information Tier as agents are dynamically allocated and unallocated to and from the agent community. In addition, the Agent Manager is responsible for managing the distribution of execution cycles allowing each agent to perform operations in a balanced fashion. Disbursement of execution cycles occurs in a round-robin manner allowing agent analysis to be evenly distributed among relevant agents. Whether or not an agent utilizes its allotted cycles depends on whether it has any tasks or actions to perform. Being a *plug-in* component the Agent Manager can easily be replaced with one offering an alternative scheduling scheme.

Session Manager-- As the overall manager of the Agent Session environment the Session Manager has two main responsibilities. The first of these responsibilities focuses on

the initialization of each of the other Agent Session components upon session creation. When an Agent Session is created as a response to the creation of a view, the Session Manager is the first component to be activated. Once initialized, the Session Manager activates the SN Manager and Inference Engine. Continuing its efforts, the Session Manager then activates the Agent Manager. Upon startup, the Agent Manager initializes itself by allocating an application-specified initial set of agents. Depending on the application specifics, these agents may in turn perform a series of initial queries and subscriptions that will eventually propagate to the Information Tier via the SN Manager.

Client User Interface

Representing the third and final tier of the three-tier architecture supported by ICDM the Client User Interface (CUI) exists as either a station-based or web-based application that interfaces with some type of client. This client may take several forms including a human user or an external system. The CUI essentially provides system users with a means of accessing and manipulating the information and analysis provided by the other two tiers of the agent-based, decision-support application.

As clients of the IS, CUI users have the ability to interact with each other in a collaborative fashion. That is, by virtue of either injecting or obtaining information from the IS, CUI users working on the same view have the potential of exchanging relevant information in a collaborative manner. All information and analysis remains localized within its particular view unless explicitly copied into another view. In this manner, no informational or analytical collisions occur between conceptual views without the potential for either system-level or human user supervision and subsequent reconciliation.

THE ICDM DEVELOPMENT TOOLKIT

As a further formalization of the ICDM approach to agent-based, decision-support applications, the CADRC is continuing its development of a collection of design and development tools to accompany the execution framework discussed above. These tools essentially combine the roles of application designer and application developer into a single effort. Decision-support applications can be designed and developed through a series of high-level models describing information structure and analytical logic. High-level objects can be identified through a series of Unified Modeling Language (UML) [2] class diagrams forming a comprehensive information object model or ontological

system. This model essentially describes the application specific design and problem space as a collection of high-level objects complete with attributes and relationships. This is the same high-level description of application information identified earlier as being crucial to agent-based, decision-support applications.

By the same token, it is the belief of the CADRC that much of the analytical reasoning applied to this information can be described in terms of a methodology suitable for representing object-based propositional logic. Though currently at the theoretical stage, the methodology intended to be employed to serve this purpose attempts to represent propositional logic as a series of rules [4]. Each of these rules identifies both a condition and a corresponding action to take upon the satisfaction of that condition. This is where the advantages of using a high-level, object-based representation again become apparent. It is the belief of the CADRC that both the condition and action components of these rules can be described in terms of the application's information object model. That is, conditions can be represented as a series of constrained references to object attributes strung together with both logical and relational operators. The corresponding rule action is itself described in terms of a series of basic manipulation functions (i.e., create, modify, delete) executed against the information pool housed within the Information Tier. When the informational state described in the condition section of the rule occurs, the corresponding action component will modify or produce information thus creating an entirely new informational state. This new state may in turn trigger other rules to execute in a similar fashion. Although not all logic can be represented in this manner, it is the expectation of the CADRC that this approach can be applied to a significant portion of analytical reasoning found in decision-support applications.

Using high-level design models that describe both domain information and domain logic as input, the ICDM Toolkit allows for the automatic generation of a significant portion of the supporting decision-support system in the form of an executable framework. In other words, using the ICDM Toolkit the information object model can be used as a basis for automatically generating basic (i.e., construction, attribute-level modification, and destruction) object-specific behavior and management. In a similar manner, the logic model can be used to automatically generate the condition and action components of the agent rules representing the logic-tier of a decision-support application.

By elevating the vast majority of agent-based, decision-support application development to the level of conceptual design, such applications can be developed, maintained, and modified in a considerably more efficient and proficient manner as compared to a more manual, implementation-specific method. Further, this approach essentially eliminates the loss of intent that often occurs as

application development moves from the designers to the program developers. Utilizing the ICDM model together with its design and development tools these roles essentially become synonymous.

REFERENCES

- [1] Bancilhon F, C Delobel and P Kanellakis (eds.); *"Building an Object-Oriented Database Systems"*; Morgan Kaufman, San Mateo, CA, 1992.
- [2] Fowler M and K Scott; *"UML Distilled: Applying the Standard Object Modeling Language"*; Addison-Wesley, Reading, Massachusetts, 1997.
- [3] Gray S and R Lievano; *"Microsoft Transaction Server 2.0"*; SAMS Publishing, Indianapolis, Indiana, 1997.
- [4] Hayes-Roth F, D Waterman and D Lenat (eds.); *"Building Expert Systems"*; Addison-Wesley, Reading, Massachusetts, 1983.
- [5] IONA; *"Orbix Web: Programming Guide"*; IONA Technologies Ltd., Dublin, Ireland, 1996.
- [6] Lewis B and D J Berg; *"Threads Primer: A Guide to Multithreaded Programming"*; SunSoft Press; Mountain View, CA, 1996.
- [7] Mowbray T and R Zahavi; *"The Essential CORBA: Systems Integration Using Distributed Objects"*; John Wiley and Sons, Inc., New York, New York, CA, 1995.
- [8] NASA; *"CLIPS 6.0 Reference Manual"*; Software Technologies Branch, Lyndon B Space Center, Houston, Texas, 1992.
- [9] Orfali R, D Harkey and J Edwards; *"The Essential Distributed Objects Survival Guide"*; John Wiley and Sons, Inc., New York, New York, CA, 1996.
- [10] Penmetcha K, A Chapman and A Antelman; *"CIAT: Collaborative Infrastructure Assessment Tool"*; in Pohl J (ed.) *Advances in Collaborative Design and Decision-Support Systems*, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22, 1997 (pp. 83-90).
- [11] Pohl J, A Chapman, K Pohl, J Primrose and A Wozniak; *"Decision-Support Systems: Notions, Prototypes, and In-Use Applications"*; Technical Report, CADRU-11-97, CAD Research Center, Design Institute, College of

Architecture and Environmental Design, Cal Poly, San Luis Obispo, CA, January, 1997.

[12] Pohl J; "*Human-Computer Partnership in Decision-Support Systems: Some Design Guidelines*"; in Pohl J (ed.) *Advances in Collaborative Design and Decision-Support Systems*, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22, 1997 (pp. 71-82).

[13] Pohl K; "*KOALA: An Object-Agent Design System*"; in Pohl J (ed.) *Advances in Cooperative Environmental Decision Systems*, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 14-18, 1995 (pp. 81-92).



BIOGRAPHY

Kym Jason Pohl is a senior software engineer in the Collaborative Agent Design (CAD) Research Center at the California Polytechnic State University, San Luis Obispo. His current focus is on agent-based, collaborative decision-support systems with particular interest in representation and collaboration architectures. Following an undergraduate degree in Computer Science he earned Master degrees in Computer Science and Architecture. Over the past decade he has provided technical leadership in the design and development of a number of multi-agent decision-support systems for the US Department of Defense, including the Integrated Marine Multi-Agent Command and Control System (IMMACCS) for tactical command and control and the SEAWAY system for the coordination of logistical sea-based sustainment operations.