

The Second Generation Integrated Collaborative Decision Making (ICDM) Model: *A Three-Tier Approach to Agent-Based, Decision-Support Systems*

By

Kym J. Pohl
CAD Research Center
Cal Poly, San Luis Obispo, CA 93407

Abstract

Ten years ago the CAD Research Center at California Polytechnic University in San Luis Obispo, California identified a standard framework for agent-based, decision support systems. Employing inter-process and inference engine technologies of the time, the CAD Research Center termed this 'blueprint' the Integrated Collaborative Decision-Making (ICDM) framework. Over the past twelve years ICDM has been successfully used as a foundation in several systems. These systems focus on a wide range of domains of application including architectural design and ship cargo stowage. Success of the ICDM framework in conjunction with the availability of newer technologies has prompted an evolutionary leap in the ICDM architecture. Capitalizing on the recent introduction of technologies such as distributed object servers and web-based computing the second generation of ICDM promises to maintain its position on the technological cutting-edge. This paper describes this second stage in evolution of the ICDM framework

Keywords

Agent-based, decision-support, distributed systems, web-based computing, three-tier architecture, public subscription, inference

Introduction

Ten years ago the CAD Research Center identified a standard framework for agent-based, decision-support systems. Employing inter-process communication and inference engine technologies of the time, the CAD Research Center termed this

blueprint for agent systems the Integrated Collaborative Decision Making (ICDM) model. Since its inception ICDM has been successfully employed as a foundation in several systems dealing with a variety of domains. Some of the systems framed around ICDM include: the Integrated Computer-Assisted Design System (ICADS) focussing on architectural design (Pohl et al. 1997); the Collaborative Infrastructure Assessment Tool (CIAT) assisting in port management (Penmatcha et al. 1997); and the Integrated Marine Multi-Agent Command and Control System (IMMACCS) designed for situation awareness in hostile, military situations. Though focussing on distinctly different domains, each of these systems is based on a common ICDM framework. Success of the ICDM framework in conjunction with the availability of newer technologies has prompted an evolutionary leap in the ICDM architecture. As one of the main prompting forces, this evolution will capitalize on forefront technologies including distributed object servers and web-based computing. The incorporation of these technologies will essentially open the door for applications utilizing ICDM to employ enabling, three-tier architectures (Orfali and Edwards 1996, Gray and Lievano 1997).

Agent-based, decision-support systems provide human decision-makers with a means of solving complex problems through collaboration with collections of both human and computer-based agents. While supportable by a number of underlying architectures, employing a three-tier model to such systems offers numerous benefits. These advantages range from location transparency to automatic client notification. The following paper discusses a formalized three-tier architecture as the next logical evolutionary step for the Integrated Collaborative Decision-Making Model. As a second generation ICDM model, the abbreviation ICDM-G2 is used. This architecture together with a set of development and execution tools can be utilized to design, develop, and execute agent-based, decision-support applications. The presented model incorporates forefront technologies including distributed-object servers, object-based inference engines, and web-based presentation to provide a framework for collaborative, agent-based, decision making systems.

Second Generation ICDM

While supporting a similar agent-based, decision support environment, the next generation ICDM is based on a drastically different model than that employed by its predecessor (Pohl et al 1997).

The ICDM-G2 model is based on a three-tier architecture making clear and distinct separations between information, logic, and presentation (Gray and Lievano 1997). These tiers are represented by the three major components comprising the ICDM-G2 model; the Object Distribution Server (information tier), the agent community (logic tier), and the Object Management Component as a part of a client user interface (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 a collection of heterogeneous on-line agents as specified by an application.

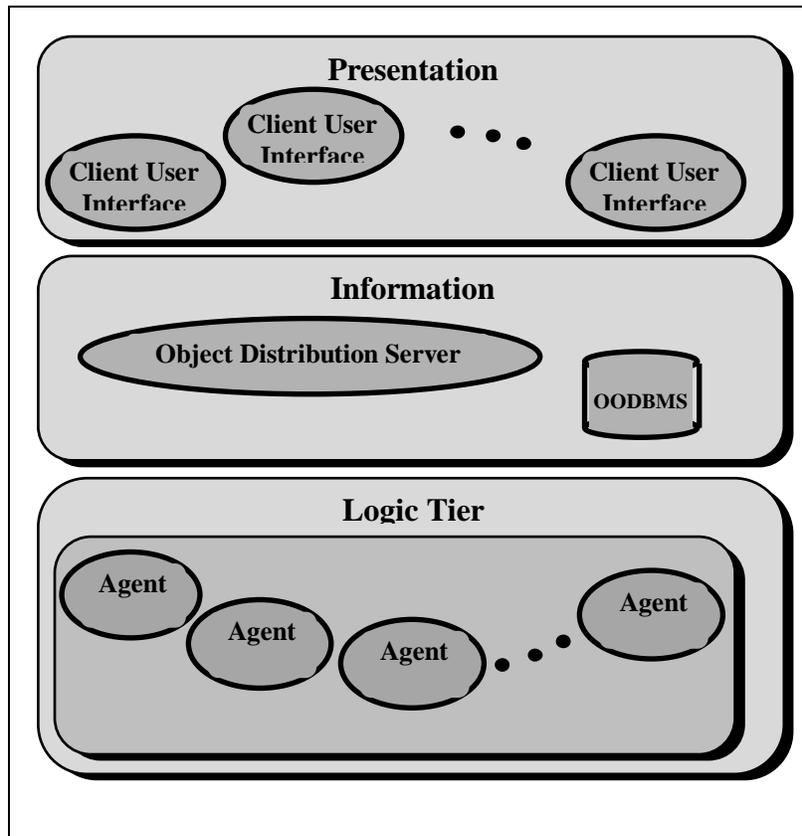


Figure 1 – Basic Three-Tier Architecture

Information Tier

Core to the ICDM-G2 model is the Object Distribution Server (ODS). Conceptually, the ODS represents a library of objectified information which clients utilize to both obtain and contribute knowledge. The only difference is that clients can obtain this information in, not only a *pull* fashion, but can also have the ODS *push* them information on a subscription basis. Physically, the ODS exists as a distributed object server based on the Common Object Request Broker Architecture (CORBA) (Mowbray and Zahavi 1995).

Being the basis for the ODS, distributed object servers are designed to service client requests for information. The knowledge of exactly where the information resides and how it can be retrieved is completely encapsulated inside the object server. 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 collaborative application components operate in an essentially de-coupled manner. 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 native objects.

The ICDM-G2 model takes full advantage of these object-oriented facilities by integrating an Object-Oriented DBMS (OODBMS) (Bancilhon et al. 1992). The OODBMS is the facility that the ODS 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 as it transitions into its persistent form. Whenever there is representational degradation there is potential for loss of informational content and meaning. By utilizing both transport and storage facilities which are capable of processing and manipulating information as native objects, degradation of representation is held to an absolute minimum as information flows throughout the application environment.

The second advantage of employing an OODBMS relates to the manner in which ODS clients request information. Whether mining for information or posting a standing subscription, clients formulate their information requests in terms of objects. More specifically, clients describe their queries and interests in terms of object attributes and inter-object relationships. These queries can range from simple existence criteria to the more complex requests incorporating both logical and relational operators.

Another method in which information can be obtained from the ODS deals with the notion of subscription. Clients can dynamically register standing subscriptions or interests with the ODS which are described in terms of the application's object model. Once registered, subscriptions are continually monitored by the ODS for satisfaction. When satisfied, the ODS essentially *pushes* the relative information to whomever has indicated an interest (i.e., registered an appropriate subscription). The most obvious alternative to this subscription mechanism would be to have interested clients perform the same query on an iterative basis until such a condition occurs. Each unsatisfied query may potentially decrease resources (i.e., computing cycles) available to other application components and would essentially prove to be wasteful. 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 a public subscription model becomes essential in providing decision-support applications with an efficient, up-to-date operating environment.

Logic Tier

As supported in ICDM-G2, the logic tier is represented by collections of agents or agent communities (Hayes-Roth et al. 1983). It is the intent of ICDM to provide mechanisms for housing application agents while limiting any constraints on their design. ICDM-G2 accomplishes this by providing agent objects the means to interact with the system in the same manner as any client would. That is, ICDM allows agents defined as objects to exist as clients to the ODS thus being able to both inject and obtain information into/from the system. Since the agents are represented as objects themselves they are managed by the ODS in the same manner as any other object. The only difference is that these agent objects are also clients to the ODS. This is a drastic departure from the original ICDM model where agents reside in a dedicated set of specialized componentry (Figure 2). Under the original ICDM model an additional set of components are required to provide agents with the ability to post subscriptions and collaborate among themselves

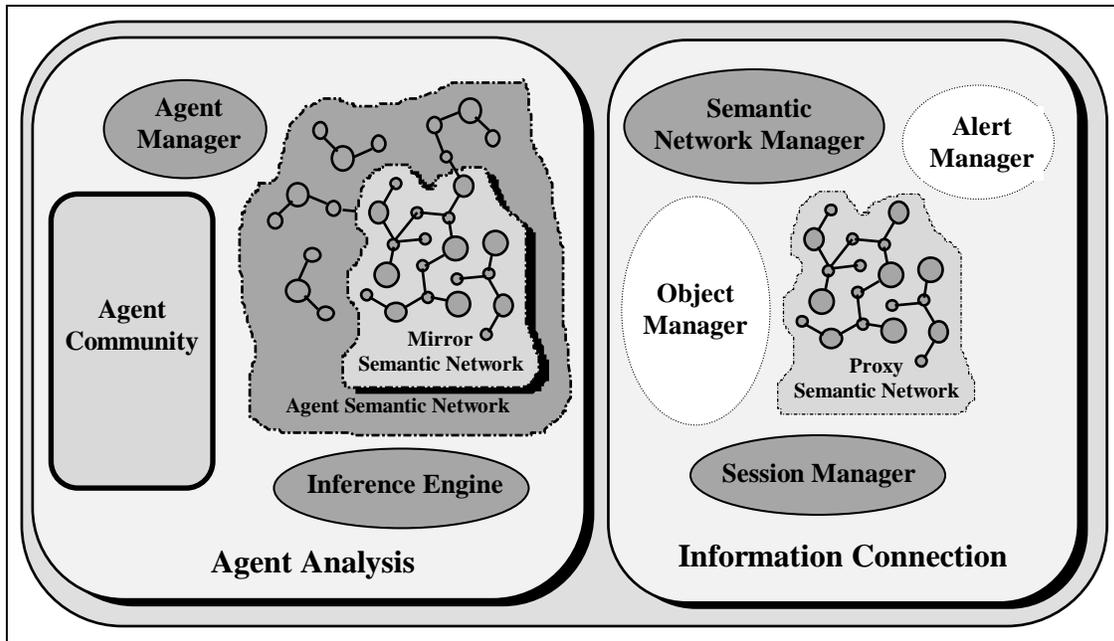


Figure 2 – Original ICDM Agent Session Architecture

based on changing information. In analyzing this original approach it became evident that the vast majority of the aforementioned componentry supports the same set of functionality provided by the ODS. That is, the same public subscription support for modeling agents as collections of autonomous interest/action pairs (i.e., rules (NASA 1992)) responding to changes in information and knowledge can also be provided by representing these agents as ODS clients. As ODS clients, a variety of additional functionality is now available. Being an object, an agent may be subscribed to by another ODS client which itself may be an agent. In addition, residing in a distributed environment, agent objects have the potential to be migrated to other locations throughout the system. This feature can be very helpful in managing and balancing dynamically changing resources as demands increase and decrease. Supporting the modeling of agents as objects again illustrates the degree to which an object representation is preserved as information and knowledge is processed throughout the application environment.

Multi-View Configuration

Considering the fact that the ICDM-G2 model is intended to support applications of a collaborative nature, it follows that such capability should be represented in the framework itself. In ICDM-G2 this is accomplished with the introduction of the notion of a view. A view can be thought of as a logical perception of the events and information described by an application object model. In some cases, a view may be a conceptual perspective of reality. 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 found in

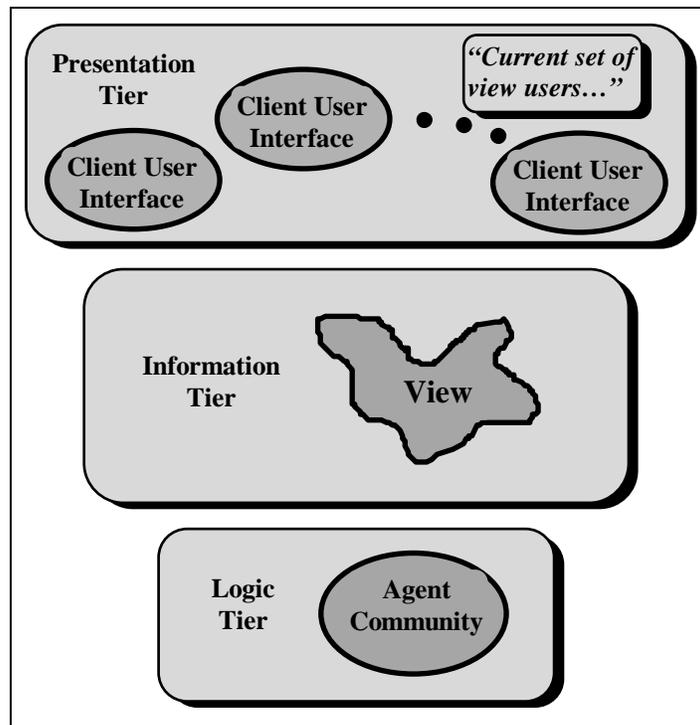


Figure 3 – Multiple Clients (e.g., Human User, Agent Object, etc.) Can Collaborate Within a Single View or Across Several Views

the IMMACCS application. IMMACCS uses a single view to represent the information and events occurring in the battlespace. 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. Clients may interact within single view or across multiple views (Figure 3). Organizing client interaction in this manner allows for an efficient and effective means of distinguishing activities relating to one view from activities pertaining to another. Unless prompted by user intervention, each set of information is completely separate from the other.

Presentation Tier

Representing the third and final tier of the three-tier ICDM-G2 architecture the User Interface (UI) can exist in a number of forms; (i.e., web-based, station-based, etc.). Regardless of form, the purpose of the client interface is to present the information and events stored in the ODS to some type of user. This user may be a human user or may take the form of another application. In either case, information and events are being presented to a user. The contribution made by ICDM takes the form of a standard Object Management Component (OMC). The OMC provides clients with both translatory and constraint management functionality. The OMC supports the translation of ODS objects to an equivalent string format. Such a format is readily useable by components wishing to see information or data as strings. Some examples of such components are Graphical

User Interfaces (GUI) and programmatic interfaces to an inference engine (i.e., CLIPS expert system shell), etc.). As the need to interface with additional components arises such translation could certainly be extended to include other formats.

The second purpose of the OMC is to provide some degree of client-based constraint management. Such constraint management would range from simple data type and range checking to complex association management. The motivation for placing such functionality with the client is centered around increasing performance. However, for more complex constraint management, such as is required to maintain logical consistency, a more appropriate location would be with the application-specific agents where the expertise to determine such consistency is readily available.

Future Research

As a further formalization of the ICDM-G2 approach to agent-based, decision-support applications, the creation of a robust collection of design and development tools would prove to automate a considerable amount of application development. Incorporating such automated development these tools would essentially combine the roles of application designer and application developer into a single effort. Along this path, decision-support applications could be designed and developed through a series of high-level models describing both information structure and analytical logic. High-level object classes can be identified through a series of Unified Modeling Language (UML) (Fowler and Scott 1997) class diagrams forming a comprehensive information object model. This model essentially describes the application-specific problem space as a collection of high-level objects complete with attributes and inter-object relationships.

By the same token, much of the analytical reasoning applied to this information can be described in terms of a methodology suitable for representing logic. Adopting a rule-based representation (Hayes-Roth et al. 1983) each item of logic is expressed as an interest/action pair. More specifically, each rule 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. 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 references to object attributes strung together with logical and relational operators. The corresponding action is itself described in terms of the object model (e.g., object instantiation or destruction and instance modification). 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 authors expectation that this approach can be successfully applied to a significant portion of analytical reasoning found in decision-support applications.

Once both the information and portions of the logic are described as high-level design models, much of the decision-support application can be automatically generated. The object model can be used as a basis for automatically generating any object-specific behavior required by the various ICDM-G2 components outlined in this paper including the ODS and OMC. In a similar manner, the logic model can be used to automatically

generate the condition and action components of rules that essentially form a significant portion of the application's agent community. Such automatic generation is possible because the information required to implement the application-specific portions of these components is present in a concise and unambiguous fashion within these two design perspectives.

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 traditional development. Further, this approach essentially eliminates the loss of intent that often occurs as application development moves from the knowledge engineers to the program developers. Utilizing the ICDM-G2 model together with its design and development tools, these roles become synonymous.

References

Bancilhon F, C Delobel and P Kanellakis (eds.); "*Building an Object-Oriented Database Systems*"; Morgan Kaufman, San Mateo, CA, 1992.

Fowler M and K Scott; "*UML Distilled: Applying the Standard Object Modeling Language*"; Addison-Wesley, Reading, Massachusetts, 1997.

Gray S and R Lievano; "*Microsoft Transaction Server 2.0*"; SAMS Publishing, Indianapolis, Indiana, 1997.

Hayes-Roth F, D Waterman and D Lenat (eds.); "*Building Expert Systems*"; Addison-Wesley, Reading, Massachusetts, 1983.

IONA; "*Orbix Web: Programming Guide*"; IONA Technologies Ltd., Dublin, Ireland, 1996.

Lewis B and D J Berg; "*Threads Primer: A Guide to Multithreaded Programming*"; SunSoft Press; Mountain View, CA, 1996.

Mowbray T and R Zahavi; "*The Essential CORBA: Systems Integration Using Distributed Objects*"; John Wiley and Sons, Inc., New York, New York, CA, 1995.

NASA; "*CLIPS 6.0 Reference Manual*"; Software Technologies Branch, Lyndon B Space Center, Houston, Texas, 1992.

Orfali R, D Harkey and J Edwards; "*The Essential Distributed Objects Survival Guide*"; John Wiley and Sons, Inc., New York, New York, CA, 1996.

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).

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.

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).

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).