Sponsored by the Office of Naval Research (ONR)

ONR Decision-Support Workshop Series

A Decision-Making Tools Workshop

hosted by the

Collaborative Agent Design Research Center (CADRC)
Cal Poly State University, San Luis Obispo, CA

in conjunction with

CDM Technologies, Inc.
San Luis Obispo, CA

Proceedings of Workshop held on April 20-22, 1999

at the
Embassy Suites Hotel
San Luis Obispo, CA

August, 1999

Foreword

The Decision-Making Tools Workshop had its genesis in a series of discussions between myself and Profs. J. Pohl and A. Chapman of the CAD Research Center at Cal Poly. Subsequently, Col Anthony Wood, USMC (ret.) joined us, playing a major role as dynamic organizer and speaker. His efforts led to a significant scientific event.

The rationale of the Workshop rests on some very simple observations. It is well known that every situation encountered by individuals and organizations demands an appropriate response. The choice of such a response from among a multitude of options is the decision-making process. Simple situations that give rise to a small number of options allow the decision maker to decide on a course of action without great effort. This state of affairs changes dramatically when the situations are complex and involve a large number of factors. In this case, an extensive field of options is engendered that makes it impossible for the human decision maker to explore (especially in real time), without the proper support tools. To this, one should add that each decision option has an associated cost (e.g., time, money, human and other assets) that without support becomes difficult, if not impossible, to determine.

These considerations and an apparent dearth of applicable support tools led us to the idea of convening a workshop in which active developers and proponents of such tools will present their approaches to an audience of potential users.

In view of the involvement of the CAD Research Center in developing and applying very successfully decision-making support tools in various military and civilian projects, including several funded by the ONR Logistics Program, it seemed both logical and beneficial to hold this workshop at Cal Poly. It is our intention to convene such workshops in the future and also enlarge both the number of presenters and the participating audience.

In conclusion, it is a pleasure to thank, on behalf of ONR, our hosts, the distinguished speakers and demonstrators, and the attendees who, through their incisive questions, contributed greatly to the success of the Workshop.

Phillip B. Abraham
Logistics Program Officer
Office of Naval Research

August 1999

Table of Contents

A Decision-Making Tools Workshop

April 20–22, 1999

The Office of Naval Research and the CAD Research Center, California Polytechnic State University, San Luis Obispo

On behalf of the Office of Naval Research, Washington, D.C., and the CAD Research Center (CADRC), California Polytechnic State University, San Luis Obispo, Welcome! While "Decision Support" emerges as a mainstream direction for research and development, we hope this conference represents the first in a series of annual events here in San Luis Obispo.

I REGISTRATION

The Conference Registration Desk is located in the main lobby of the Embassy Suites Hotel. The Registration Desk is open from 7:30 AM until 5:00 PM on the 20th of April, and again on the 21st from 7:30 AM until noon. If you are unable to register during these hours, please see Colonel Anthony Wood or Professor Art Chapman of the CADRC, and they will assist you. Your registration information is the basis for the post-conference roster of attendees that will be mailed to all participants as a part of the formal *Report of Proceedings*.

II ACTIVITIES AND PRESENTATIONS

All formal presentations and application "round-robins" are designed with time for questions. As the schedule in Section V indicates, a round-robin composed of three related events is scheduled for the afternoons of the 20th and 21st. In each case, we will be divided into small groups to allow for greater discussion and participation. You will receive a color code on your badge at registration indicating which group you should join for the afternoon portions of the schedule. Please observe these codes, so groups remain roughly equal in size.

III MEALS & SNACKS

Breakfast is an individual matter. Guests staying at the Embassy Suites have a complimentary breakfast included in their room rate. Coffee and rolls will be served outside the plenary session each morning during scheduled breaks.

Luncheons each day are provided at no cost as a formal part of the conference and will be accompanied by formal presentations. Please make every effort to attend these sessions.

Like breakfast, dinners are an individual matter. The hotel has a fine restaurant, and the City of San Luis Obispo boasts a wide selection of fine dining opportunities. Please consult the hotel and the city map included in your welcome packet to locate these establishments.

IV DOCUMENTATION

Formal versions of all presentations will be included in the post conference *Report of Proceedings*. Speakers are reminded of the requirement to submit these as soon as possible. Please submit electronically to Ms. Jan Barrett by email at *jbarrett@cdmtech.com*. The *Report of Proceedings,* including short biographies for all speakers, will be mailed within 30 days. Presentations not received by May 5th cannot be included.

V CONFERENCE SCHEDULE

Tuesday, April 20: "Collaborative Decision Support Today"

TIME	ACTIVITY	LOCATION
8:00–8:30	***"Introduction and Welcome,"*** Dr. Philip Abraham, Office of Naval Research.	San Luis Obispo Room North
8:40–9:30	***"A Personal Perspective on Decision Makers and Their Decisions,"*** Vadm (ret.) Jerry Tuttle, ManTech International Corporation.	San Luis Obispo Room North
9:40–10:30	***"The Decision Industry: Makers, Processes, Support, and Communication Links,"*** Col (ret.) Anthony A. Wood, Cal Poly CADRC.	San Luis Obispo Room North
10:40–11:30	***"Collaborative Decision Support and the Man-Machine Relationship,"*** Dr. Jens Pohl, Cal Poly CADRC.	San Luis Obispo Room North
12:00	**Lunch:** ***"The Technical Horizon,"*** Radm (ret.) L. S. Kollmorgen.	Atrium
1:30–2:30	**Round Robin Cycle 1:** A series of applications and demonstrations focusing on transforming data to provide information and implications using collaborative decision-support tool kits.	
	Group Red: ***ICODES:*** Mr. Steve Goodman, MTMC (ICODES PM), with Mr. Fred Abler, Mr. Matt Parrott, and Mr. Nick Kephalos, Cal Poly CADRC. The *Integrated Computerized Deployment System* provides a set of collaborative decision-support tools to assist complex stow planning in the maritime industry.	Edna East
	Group Blue: ***IMMACCS:*** LtCol David Durham, C4I Officer in Charge, ECOC, MCWL; Mr. Mark Porczak, Mr. Kym Pohl, and Mr. Russ Leighton, Cal Poly CADRC. The experimental *Integrated Marine Multi-Agent Command and Control System* provides collaborative decision support to the dynamic urban combat environment.	Edna West

Tuesday, April 20: Continued

TIME	ACTIVITY	LOCATION
	Group Green: ***CIAT:*** Mr. Al Antelman, NFESC (CIAT PM) with Mr. Jonathan Lee and Mr. Mike Zang, Cal Poly CADRC. The *Collaborative Infrastructure Assessment Tool* provides collaborative decision support for waterfront services and to future planning at the U.S. Naval Station, San Diego.	Del Monte
2:40–3:30	**Round Robin Cycle 2:**	
	Group Red: *IMMACCS.*	Edna West
	Group Blue: *CIAT.*	Del Monte
	Group Green: *ICODES.*	Edna East
3:30–3:45	**Break.**	
3:45–4:45	**Round Robin Cycle 3:**	
	Group Red: *CIAT.*	Del Monte
	Group Blue: *ICODES.*	Edna East
	Group Green: *IMMACCS.*	Edna West

Wednesday, April 21: "Decision Making Under Stress"

TIME	ACTIVITY	LOCATION
8:30–9:45	***"Decision Making in Practice,"*** Dr. Gary Klein, Chairman, Klein Associates.	Edna Room
9:45–10:00	**Break.**	
10:00–11:00	***"Facing Uncertainty,"*** LtGen (ret.) Paul Van Riper, former Commanding General of the USMC Combat Development Command.	Edna Room
11:10–12:00	***"Intelligent Agents Infrastructure for Information Gathering and Decision Support in an Open Environment,"*** Dr. Katia Sycara, Carnegie Mellon University.	Edna Room
12:00	**Lunch:** ***"Emerging Decision-Support Requirements in the USAF,"*** Mr. Eric Werkowitz, the USAF Command and Control Battle Lab.	Atrium
1:30–2:30	**Round Robin Cycle 1:** A series of applications and presentations highlighting the use of decision tools.	
	Group Red: ***"The Combat Decision Range,"*** MajGen (ret.) Ord Steele, former Commanding General of the 2d Marine Division.	Edna East

Wednesday, April 21: Continued

TIME	ACTIVITY	LOCATION
	Group Blue: ***"Conversational Case-Based Plan Authoring for Interactive Decision Support,"*** Dr. David Aha, Naval Research Laboratory.	Edna West
	Group Green: ***"A Case Study,"*** Dr. Gary Klein, Chairman, Klein Associates.	Del Monte
2:40–3:30	**Round Robin Cycle 2:**	
	Group Red: Dr. David Aha, Naval Research Laboratory.	Edna West
	Group Blue: Dr. Gary Klein, Klein Associates.	Del Monte
	Group Green: MajGen (ret.) Ord Steele, former Commanding General of the 2d Marine Division.	Edna East
3:30–3:45	**Break.**	
3:45–4:45	**Round Robin Cycle 3:**	
	Group Red: Dr. Gary Klein, Klein Associates.	Del Monte
	Group Blue: MajGen (ret.) Ord Steele, former Commanding General of the 2d Marine Division.	Edna East
	Group Green: Dr. David Aha, Naval Research Laboratory.	Edna West
5:00–7:00	**Workshop Reception.**	Atrium

Thursday, April 22: "Communication and Data Feeds"

TIME	ACTIVITY	LOCATION
8:00–8:30	***"The Shared Net,"*** Dr. Thomas McVittie, NASA Jet Propulsion Laboratory.	San Luis Obispo Room North
8:40-9:30	***"Information Assurance,"*** Ms. Kathy McCollum, Defense Advanced Research Project Agency, Program Manager, Cyber Command and Control Program.	San Luis Obispo Room North
9:30–9:45	**Break.**	
9:45–11:00	***"Probabilistic Models for Decision Making,"*** Prof. Daphne Koller, Stanford University.	San Luis Obispo Room North
11:15	**Concluding Luncheon.**	Atrium

About the Speakers

Dr. David Aha, Naval Research Laboratory

David W. Aha (UCI, 1990) leads research projects on machine learning and case-based reasoning at NCARAI/NRL with an emphasis on developing decision-support tools. The focus of this presentation will concern an ONR 6.2 project for the C2 & Combat Systems Program. He serves as an Editor for Machine Learning, guest-edited the first special issue/book on Lazy Learning, and is a member of Inference Corporation's Technical Advisory Board and the AI Research in Environmental Science committee. He has (co-)chaired several workshops on machine learning and case-based reasoning, and frequently serves on program committees related to these areas. His current interests include: interactive case-based planning, case-based reasoning for knowledge management applications, and the development of more useful lessons learned systems.

Dr. Gary Klein, Chairman, Klein Associates

Gary Klein, Ph.D. is Chairman and Chief Scientist of Klein Associates. He has performed research on naturalistic decision making in a wide variety of task domains and settings including: firefighting, aviation, command and control, market research, and software troubleshooting. Based on these and related projects, he has developed significant new models of proficient decision making. His research interests include the study of individual and team decision making under conditions of stress, time pressure, and uncertainty. Dr. Klein has furthered the development and application of a decision-centered approach to system design and training programs. He has edited two books on naturalistic decision making and authored *Sources of Power: How People Make Decisions* (1998, MIT Press). Currently, he is extending the naturalistic decision making framework to cover planning and problem solving. He received his Ph.D. in Experimental Psychology from the University of Pittsburgh in 1969.

Dr. Daphne Koller, Stanford University

Daphne Koller received her Ph.D. from Stanford University in 1994. After a two-year postdoc at Berkeley, she returned to Stanford, where she is now an Assistant Professor in the Computer Science Department. She has a broad range of interests: artificial intelligence, economics, and theoretical computer science. Her main research interest is in creating large-scale systems that reason and act under uncertainty. The theme underlying her work is the integration of ideas from decision theory and economics into these systems. This task raises the need for compact and natural knowledge representation schemes and efficient inference and learning algorithms that utilize these schemes. Daphne Koller is the author of over 50 refereed publications, which have appeared in AI, theoretical computer science, and economics venues. She has served on numerous program committees, on the editorial board of the Journal of Artificial Intelligence Research, and on the editorial board of the Machine Learning Journal. She was awarded the Arthur Samuel Thesis Award in 1994, the Sloan Foundation Faculty Fellowship in 1996, the Stanford University Terman Award in 1998, and the ONR Young Investigator Award in 1999.

Radm (ret.) Leland S. Kollmorgen

Rear Admiral Leland S. Kollmorgen ("Lee") was born May 20, 1927 in Los Angeles, CA. He graduated from the U.S. Naval Academy in 1951. He was awarded a B.S. in Aeronautical Engineering from the U.S. Naval Postgraduate School in 1960, and in 1966, he earned a M.S. in International Affairs from George Washington University. He continued his studies in Technology Assessment from 1971-1972.

Between his various scholarly endeavors, he was given various shipboard assignments such as Attack Squadron Pilot and Heavy Photographic Squadron Pilot, which included tours in Laos and Vietnam. From 1966 to 1971, he was Staff, Commander Fleet Air Whidbey, Whidbey Island, Washington. From 1971 to 1974, Kollmorgen was assigned to the Office of the Chief of Naval Operations as the Assistant to DCNO (AIR) for Aviation R&D and Nuclear Matters, and then later as the Special Assistant for Acquisition to Director, Navy Program

Planning. In 1974, he was reassigned to NAS Cecil Field, Jacksonville, Florida as Commanding Officer. In 1975, he was again reassigned to the Office of the President of the United States as Military Assistant to the President. Following his position as Military Assistant to the President, he was reassigned to the Office of the Secretary of Defense as the Assistant Director, Test and Evaluation. In 1978, Kollmorgen was assigned to the Office of Chief of Naval Operations as the Director, Systems Analysis Division.

From 1981 to 1983, he served in the Office of Naval Research; Headquarters, Naval Material Command. His positions here included: Chief of Naval Research, Chief of Naval Development, and Deputy chief of Naval Material for Technology. During this period, he conceived and implemented new management procedures to more effectively align Navy technology base investment with the needs and requirements of the service.

Upon retiring from the Navy in 1983, he moved to the private sector where he became President of TLK, Inc. Currently, he also sits on the board of a number of companies including Xyvision, Information Presentation Technologies, Inc., and Racal Communications.

His personal decorations and awards include: the Navy Distinguished Service Medal, a Silver Star, three Legions of Merit, the Distinguished Flying Cross, two Air Medals, seventeen Air Medal Strike Flight, and a Battle Efficiency "E" for training and operational Excellence.

Ms. Catherine D. McCollum, Defense Advanced Research Project Agency, Program Manager, Cyber Command and Control Program

Catherine D. McCollum is a Principal Engineer with the MITRE Corporation, working closely with DARPA in the area of information assurance and cyber command and control. Ms. McCollum has been an active researcher in information systems security for over a decade. She has performed research in secure database management algorithms, security policies, secure systems architectures, and information survivability. Her work has been presented widely in security research conferences. Ms. McCollum recently headed a research project in database information warfare defense, investigating application-level isolation techniques for containing the spread of suspect data while operations and repairs proceed. She led a previous research project in secure federated systems, investigating capabilities for controlled cooperation among autonomous systems. Working with DARPA, Ms. McCollum has been involved in the development of the Common Intrusion Detection Framework and the Information Assurance program's security architecture and vision. Ms. McCollum received a B.S. degree in Applied Mathematics from Carnegie Mellon University. Prior to MITRE, Ms. McCollum was a researcher and systems engineer with Unisys Defense Systems, System Development Corporation, and BDM.

Dr. Thomas McVittie, Jet Propulsion Laboratory

Thomas McVittie is a principal software engineer at NASA's Jet Propulsion Laboratory. He is the principle engineer for DISA's DII COE kernel. He has a Ph.D. in Electrical & Computer Engineering from UCSB. His research interests are in Reliable Distributed Systems and Formal Specifications.

Dr. Jens Pohl, Cal Poly CADRC

Dr. Jens Pohl holds the positions of Professor of Architecture, Executive Director of the CAD Research Center, and Post-Graduate Studies Coordinator in the College of Architecture and Environmental Design, California Polytechnic State University, San Luis Obispo, California, USA.

Professor Pohl received his formal education in Australia with degrees in Architecture and Architectural Science: B.Arch. (1965), M.Bdg.Sc. (l967), and Ph.D. (1970). He taught in the School of Building at the University of New South Wales in Sydney, Australia, until the end of 1972 and then left for the USA where he was appointed to the position of Professor of Architecture. Following several years of research and consulting activities in the areas of building support services and information systems, Dr. Pohl's research focus today lies in the application

of distributed artificial intelligence methodologies to decision-support systems in engineering design, logistical planning, and military command and control.

Under his direction, the Cal Poly CAD Research Center has developed and implemented a number of distributed computing applications in which multiple computer-based and human agents collaborate in the solution of complex problems. Foremost among these is the ICDM (Integrated Cooperative Decision Model) framework which has been applied to engineering design (industry sponsorship: ICADS, 1986–1991), energy conservation (US Dept. of Energy sponsorship: AEDOT, 1992–1993), logistical planning (US Army MTMC sponsorship: ICODES, 1993–present), military mission planning (US Marine Corps MCWL sponsorship: FEAT, FEAT4, and IMMACCS; 1994–present), and facilities management (US Navy ONR sponsorship: CIAT, 1996–present).

The Integrated Marine Multi-Agent Command and Control System (IMMACCS) was successfully field-tested as the command and control system of record during the Urban Warrior Advanced Warfighting Exercise (AWE) conducted by the Marine Corps Warfighting Laboratory (MCWL) in Central California (Monterey and Oakland) during the period March 11 to 18, 1999.

Dr. Pohl is the author of two patents (USA), several books, and more than 80 research papers. He is a Fellow of the International Institute for Advanced Studies in Systems Research and Cybernetics, and was awarded on honorary doctorate by the Institute in August, 1998, during the InterSymp-98 conference held in Baden-Baden, Germany.

MajGen (ret.) Orlo Steele, former Commanding General of the 2d Marine Division

Major General Orlo Keith Steele ("O.K. Steele") has been named the Federal Aviation Administration's Assistant Administrator for Civil Aviation Security. He joined the FAA following a 35-year career with the U.S. Marine Corps. His latest assignment was Deputy Naval Inspector General for Marine Corps Matters.

General Steele graduated from Stanford University in 1955 with a degree in Political Science, and during that same year, he enlisted in the Marine Corps. After his commissioning as an infantry officer in 1956, General Steele served in many command and staff assignments during his Marine career, which took him from the Far East to the Mediterranean Sea, with many stops in between.

General Steele is also a graduate of the Amphibious Warfare School at Quantico, the Marine Corps Command and Staff College, and the National War College in Washington, D.C. He was an instructor at the Mountain Warfare Training Center in Bridgeport, CA, and with the NROTC Unit at Dartmouth College in Hanover, NH.

Early command assignments include: rifle company commander; Commanding Officer, Marine Detachment, USS America; Commanding Officer/Executive Officer, Second Battalion, 5th Marines; Commanding Officer, Second Battalion, 1st Marines; and Executive Officer, Marine Ground Defense Force, U.S. Naval Base, Cuba.

From 1980 to 1983, General Steele held assignments in Marine Headquarters Plan Division and as Commanding Officer Marine Barracks, 8th & I. In 1983, he was promoted to Brigadier General and Assigned Commanding General, 1st Marine Brigade FMF, Pacific, Kaneohe Bay, Hawaii. In June 1985, he became the Legislative Assistant to the Commandant of the Marine Corps. He was promoted to Major General in 1987 and assigned as Commanding General, 2d Marine Division/Deputy Commander, II Marine Expeditionary Force, FMF, Atlantic, Camp Lejune, NC. Prior to retirement, General Steele was assigned as Deputy Naval Inspector General for Marine Corps Matters.

General Steele retired from the United States Marine Corps in October 1990. In November of that same year, he returned to federal service as head of Civil Aviation Security, FAA. He served there until December 1993. He and his wife, the former Catharine H. LeBaron of Honolulu, Hawaii, with his two children, Colin and Wendy, retired to Grass Valley California in 1994. He currently acts as a consultant in DOD and Civil Aviation Security Matters.

General Steele's personal decorations and awards include: The Distinguished Service Medal, The Bronze Star Medal with Combat "V," Combat Action Ribbon, Presidential Unit Citation with bronze star, Meritorious Unit Citation with bronze star, National Defense Service Medal, Vietnam Service Medal with four bronze stars, Sea Service Deployment Ribbon, Overseas Service Ribbon, Republic of Vietnam Cross of Gallantry with gold star, Republic of Vietnam Meritorious Unit Citation Gallantry Cross with Palm, and the Republic of Vietnam Campaign Medal with device.

Dr. Katia Sycara, Carnegie Mellon University

Dr. Sycara is a Senior Research Scientist in the Robotics Institute in the School of Computer Science, Carnegie Mellon University. She is also the Director of the Enterprise Integration Laboratory. She is directing/conducting research and developing technology for integrating organizational decision making. She holds a B.S. in Applied Mathematics from Brown University, M.S. in Electrical Engineering from the University of Wisconsin, and a Ph.D. in Computer Science from the Georgia Institute of Technology. Her research has contributed to the definition of case-based reasoning and the development of computational negotiation models as a means of resolving goal conflicts and inconsistencies in assumptions and viewpoints of heterogeneous agents in distributed problem solving. She has applied her research to concurrent engineering design, crisis action planning, and manufacturing scheduling. She has published extensively in these areas. Currently, she is engaged in the development of intelligent agents that interact with their users, other agents, and distributed information resources in order to assist their users in the planning and execution of various tasks. In the course of performing their tasks, the agents (1) retrieve, route, and integrate information; (2) negotiate to resolve conflicts; and (3) learn from their users and other agents.

Dr. Sycara is Area Editor for AI and Management Science for the journal "Group Decision and Negotiation" and on the editorial board of "IEEE Expert," "AI in Engineering," and "Concurrent Engineering, Research and Applications". She is a member of AAAI, ACM, Cognitive Science Society, IEEE, and the Institute of Management Science (TIMS).

Vadm (ret.) Jerry Tuttle, ManTech International Corporation

Jerry joined ManTech International Corporation's executive management team October 18, 1996, as Senior Vice President of ManTech International Corporation and President of ManTech's largest subsidiary, ManTech Systems Engineering Corporation. He is responsible for strategic planning at the international level and total operating responsibility at the subsidiary level of this 4,500 person management and technology firm.

Previously, Jerry was with Oracle Government for 33 months as Vice President, Business Development and Chief Staff Officer. During this period, Oracle Government quadrupled in size and in revenue.

Admiral Tuttle retired from the United States Navy, following a blissful 39 year career. From seaman recruit to Vice Admiral his career included assignments to numerous attack and fighter squadrons. He served as Aide and Flag Lieutenant to the Commander in Chief, U.S. Pacific Fleet. He commanded an attack squadron, an air wing, a replenishment ship, the aircraft carrier USS John F. Kennedy, and two Battle Groups/Forces. He served as Special Assistant to the Chief of Naval Operations and as Deputy Director for Intelligence, Defense Intelligence Agency. He flew over 220 combat missions over North Vietnam and has more than 1,025 carrier arrested landings. At the time of his retirement, he was Navy's "Grey Eagle" signifying the earliest designated Naval Aviator on active duty.

Jerry is widely regarded as an information technology strategist, having created Navy's C4I Joint Operations Tactical System (JOTS). In 1989, he became Director, Space and Electronic Warfare, an assignment he held until retirement. During this tour he crafted Navy's C4I architecture, *Copernicus*, and Information Warfare architecture, *Sonata*. Prior to that he was Director, Command, Control and Communications (C3) Systems, the Joint Staff. From 1985 to 1987, he was Deputy and Chief of Staff for the Com-

mander in Chief, U.S. Atlantic Fleet, following a tour as Naval Inspector General. He is a member of the Defense Science Board, a member on the Board of Directors for the USO-Metro, a member of the Board of Advisors to the Superintendent of the Naval Postgraduate School, a member of the Board of Advisors to the Georgia Tech Research Institute, and an Advisory Board Member to the Software Engineering Institute. He is also a Navy Aviator Gold Eagle.

His personal decorations include: the Defense Distinguished Service Medal (3), Defense Superior Service Medal; Legion of Merit (4), Distinguished Flying Cross (3), Meritorious Service Medal (2), Air Medal (23), Navy Commendation Medal (4), Letter of Commendation from the Japan Defense Agency, and numerous campaign awards. He received the 1978 Navy League's John Paul Jones Award for inspirational leadership, the 1983 Association of Old Crow's Award for his contributions to electronic warfare, and the 1984 Annual Tailhook Award for his contributions to Naval Aviation. He was 1989's AFCEAN of the Year for his contributions to the Armed Forces Communications and Electronics Association and received the 1991 American Institute of Aeronautics and Astronautics (AIAA) Command, Control, Communications and Intelligence Award for his contribution to the overall effectiveness to the C3I Systems. He received the AFCEA 1992 Jon L. Boyes award for major contributions to that organization. He was chosen as one of Federal Computer Week's 1991 and 1992 Federal 100 for his impact on government computer systems. He received the Washington Space Business Roundtable 1993 Excellence in Government Award. He was inducted into the Government Computer News Information Resource Management Hall of Fame in 1993 and received the 1994 American Astronautical Society Military Astronauts Award. In 1995, he was awarded the French "Commandeur de l'Order National du Merite" medal by the President of the Republic of France for his efforts in promoting greater interoperability between the U.S. and French Navies.

Admiral Tuttle received a Communications Engineering Degree from the Naval Postgraduate School in 1962, having attended the undergraduate and postgraduate schools simultaneously. He graduated with honors from the Naval War College, concurrently receiving a master's degree in International Relations from George Washington University in 1969. He has authored a myriad of articles and speeches.

LtGen (ret.) Paul Van Riper, former Commanding General of the USMC Combat Development Command

Lieutenant General Van Riper retired from the United States Marine Corps on 1 October 1997, after more than 41 years of commissioned and enlisted service. He currently resides in Williamsburg, Virginia. A Senior Fellow with the Center for Naval Analyses, he is also a member of several defense and service advisory boards and panels including: the Defense Science Board's 1998 Summer Task Force on Joint Operations; the National Research Council's Naval Studies Board, the Army Science Board; the Defense Advanced Research Projects Agency's Information Science and Technology Study Group; the Institute for Defense Analyses' Joint Advanced Warfighting Program Senior Advisory Group; and the National Reconnaissance Office's Operational Support Office Gold Team. Lieutenant General Van Riper continues to participate in various defense and security related seminars and conferences, both in the United States and overseas. Recently, he has been a speaker and panelist at sessions held by the Military Operations Research Society; the Chief of Naval Operations' Strategic Studies Group; the Naval War College's Center for Naval Warfare Studies; the Department of Defense's Office of Net Assessment; the National Security Program at the John F. Kennedy School of Government, Harvard University; the Association of the United States Army; the Defense Technology Seminar '98, Lincoln Laboratory, Massachusetts Institute of Technology; the Securities Studies Program, Center for International Studies, Massachusetts Institute of Technology; the Center for Strategic and International Studies; and the Center for Strategic Education, the Paul Nitze School of Advanced International Studies, John Hopkins University. Lieutenant General Van Riper lectures frequently at the National Defense University and other professional military education institutions. He also consults part time for a number of firms

on defense and operational matters. A student of military history, Lieutenant General Van Riper spends his leisure hours reading and visiting battlefields. In addition, he writes for pleasure and publication.

Mr. Eric Werkowitz, the USAF Command and Control Battle Lab

Mr. Werkowitz is the Air Force Research Laboratory (AFRL) representative to the USAF Command and Control Battle Lab at Hurlburt Field, Florida. His primary responsibility is coordinating AFRL support to the battle lab. Being an on-site representative, he also serves as a battle lab initiative researcher, initiative manager, technology adviser, and staff engineer. Prior to this assignment, Mr. Werkowitz served as an action officer with the Plans Directorate of the AFRL. In this capacity, he performed many long-range planning functions related to the Air Force Modernization Planning Process and the development of the Air Force Strategic Plan. In partnership with the Air Staff, he also planned and directed two future-based technology war games known as Vulcan's Forge.

Before coming to the AFRL, Mr. Werkowitz served as a long-range planner with the Aeronautical System Center in the Development Planning Directorate. There he performed air-to-surface operations modeling at the campaign level. He used the results of the modeling efforts to develop and justify a 20-year research and development roadmap.

Mr. Werkowitz's previous assignment was as a human factors engineer in the Crew System Development Branch of the USAF Fight Dynamics Engineer. In this capacity, he built a speech technology research program that led to the first flight test of automatic speech recognition in fighter aircraft.

Mr. Werkowitz has a Bachelor of Science degree in Human Factors Engineering from Wright State University and a Master of Science degree in Computer Science from the Air Force Institute Technology. He served with the US Marine Corps and is a Vietnam veteran.

Col (ret.) Anthony A. Wood, Cal Poly CADRC

Colonel Anthony A. Wood joined the California Polytechnic State University CAD Research Center in 1998 following his retirement from the United States Marine Corps. In his last assignment as a Marine Officer, Colonel Wood founded the Marine Corps Warfighting Laboratory and served as its first Commander for three years. In that capacity, he directed the Sea Dragon series of experiments including Hunter Warrior and the first stage of Urban Warrior. Designed to identify future warfighting enhancements, these pioneering experiments blazed the way for the current joint experimentation effort in the Department of Defense. In previous tours, he led the team that developed the Marine Corps' first Master Plan, drafted the Marine Corps Maritime Prepositioning Doctrine, and designed the Maritime Prepositioning Decision Support System, the Corps' first formal decision-support system. Other significant accomplishments include his contributions as a principal author of the Operational Maneuver from the Sea doctrine and his work as principal designer of the Joint Conflict Model while serving with the U.S. Pacific Command. At the time of his retirement, Colonel Wood was the only colonel on active duty twice decorated with the Distinguished Service Medal.

A Personal Perspective on Decision Makers and Their Decisions

Vadm (ret.) Jerry Tuttle
President, ManTech Systems Engineering Corporation

Good morning, cyberspace caters, consiglieres, pundits, you who reside on the ethereal side of Pluto. I am delighted to appear before this panoply of geniuses. Addressing this learned group makes me feel akin to an astronaut suffering from acrophobia. You shock me with awe.

What I might contribute to this brilliant group escapes me. You, who have masterfully created primordial decision support systems for solving complex problems by exploiting, distributed collaborative computing and utilizing artificial intelligence methodologies. Although I understand the concepts of your rule based system, I have no idea how to add value. I realize that your decision support system defines, with great precision, the relationships between all known and related variables of a manageable and bound problem set, and then integrates the solutions to these problems with other related and associated sub-problems through service, object and human agents. But, that is the limit of my knowledge of your marvelous work.

Nevertheless, I will reach deep into my right cerebral hemisphere and attempt to bring something to the altar. Oliver Holmes opined that "Man's mind, once stretched by a new idea, never regains its original dimensions", Unquote. We should all learn as if we are going to live forever and live like we are going to die tomorrow.

These decision support systems that you are designing will have as an abysmal affect on the tiered hierarchical command and control architecture of our warfighting forces. The results will be more profound than the introduction of information management systems in the private sector that threatened middle managers with becoming an endangered species.

For today, I have opted to look beyond current conventional wisdom and this year's intellectual fantasies, because my contribution would be miniscule in areas which you are indubitably more knowledgeable. My nirvana is to take us on an exciting journey into the future and stretch metaphysics to the limits of our imagination and predict the Information Infrastructure technologies that your "Decision Support" systems will reside. This adventure is taken with great trepidation, as I am mindful that good speeches are not written, but re-written. My remarks today are in Alpha test.

I will confidently predict where the Information revolution will take us, who will be the benefactors, and who the new users will be. The fulcrum about which future

information systems will pivot will not be technology, computers, procedures/processes, software and speed, but seminal concepts, derived from creative operators, whether in the boardroom, the wardroom, in war rooms, in the foxhole, on the bridge, or in the cockpit.

Typical expectations and fundamental requirements for future information systems will still consist of traditional features, as the crucible of information technology churns out one astounding product after another to perpetuity. The purpose of these information systems will remain to collect, process, and disseminate a secure uninterrupted flow of information and knowledge. A malleable and scaleable architecture, sensor management and distribution system and the necessary equipment, technology, software and speed necessary for collecting, storage, retrieval, transmission, analysis, and depiction of information in a manner that can be readily understandable and assimilated will remain in fashion and essential.

Future information systems must eliminate, or greatly ameliorate the first law of information theory that every relay doubles the noise and cuts the message in half. Our legacy information systems were designed to support management decisions, not leadership strategic decisions. They focus internal to the enterprise, not externally into the environment where information resides that can have a profound effect on an organization and its decision-makers' objectives. What the operator needs to access, which in large part is outside of his organic information universe, is effectively hidden by the overburden of what he will never need.

We in the Information Industry grouse about interoperability and stovepipe information systems. We will always be confronted with these challenges and one need only to look in the wake, observe the heritage of information systems and how they evolved to understand why.

At first there was the stand alone computer which for all practical purposes simply digitized our analog information, then computer peer-to-peer information exchanges, followed by clusters, client/servers architectures, both two and three tier, thence networks of networks and the latest elixir of network centric information exchanges. But, these information domains are still for all practical purpose unique to a specific discipline, or organization i.e. intelligence, logistics, etc. and there is no appetite to share information across organizational boundaries. The biggest reason for stovepipe systems had nothing to do with technologies, but had every thing to do with those who had sway over them. They had no intentions of sharing information that resided on their systems with others, even public information. This situation flies in the face of your model that depends on shared distributed knowledge bases.

We have moved into an age whereby the equities of the byte age can be combined with that of the divinely given brain, to ameliorate ignorance and biases, and permit a

more holistic, safer and peaceful world. These exciting capabilities will enable us to combat disease, enhance the quality of life, promote commerce and conduct and win wars in a more benevolent manner. Information will be used as an instrument to preserve the peace and failing that as a weapon in war. Information of all descriptions and the electronic and photonic arteries to carry it will ever increasingly become global utilities and commodities. Shared information will no longer be an oxymoron and be the alchemy for the decision support systems that you are creating.

Dimensions will expand to unfathomable proportions. Measurements will span from the molecular to the vastness of our galaxy and beyond. Our vocabulary will change and we will have a new lexicon. There will be a merging at the boundaries between the traditional and basic information systems of logistics, maintenance, weather, administration, personnel management, etc. These boundaries will erode away and all of their functions and disciplines will reside on the same scaleable infrastructure, resulting in far greater utility of resources. Such an all encompassing information system will require a doctrinal hierarchical set of priorities that will be premeditatedly determined but alterable and automated by problem solving tools. This hierarchical architecture eventually will succumb to your cooperative and distributed decision support systems.

The Information Age revolution will cause a seismic shift in distribution of wealth and power and will transform business, communications, societies, cultures, lifestyles, how we conduct wars, work, play, live and in fact every facet of our lives. Life in 2030 will be very different from what we know today. We are in the infancy of mankind and will make a quantum jump in quality of life and soar to new celestial heights in lifestyles. Whereas the laws of gravity may be repealed, mortality cannot be rescinded. Nevertheless, individuals born three decades from now will live routinely to 120.

Not since that certain stroll across the Sea of Galilee has the world witnessed a miracle as that which biotechnology will bring to the altar. Advanced technologies are greatly accelerating the pace of discovery in biology. Scientists are unlocking bio-chemical mysteries in cancer, clogged arteries, and Alzheimer's disease. Biotechnology will permit us to grow our own organs and kidney transplants will become as common as an oil change today. It is the biotechnology discipline where I would prefer to see you adroitly exploit your sublime decision support systems

Human knowledge is doubling every decade, ergo there will be 8 times the amount of knowledge in the world in 2030 than exist today. In the last 10 years, more knowledge has been created than in the history of mankind. Clearly, we must assimilate knowledge differently and faster. Learning has never been easier. Information has never been more plentiful. Knowledge has never been as accessible or as crucial. We need to release not limit the power of human ingenuity. The cost of education is dear but the expense of ignorance can be staggering.

Our daily lives will be profoundly different. Business and work will be conducted from our homes, in airports, in hotels, etc, and span every aspect of our lives from banking, to shopping, to entertainment, to the elastic limits of the most sagacious. In three decades our homes will have more bandwidth available than exists in the world today.

Network computers will become public and as pervasive as the telephone, enabling anyone, anywhere to communicate with anyone in the world. Your wristwatch will also be a cellular telephone and polyglot computer terminal with color display and a built-in real-time language translator to communicate with any nationality in the world.

The volcanic eruption in Medium and Low Earth Orbiting satellite constellations will serve as the loom with which to weave a global communications network to carry the torrential flow of information. New compression algorithms will permit interactive multimedia communications over cellular bandwidth. Technology is moving so rapidly that if you get a busy signal on your cellular telephone you will merely order a new one.

Within a decade, the Internet will force a new common global monetary system, something that the European statesmen have taken decades to accomplish via political means. The PC and the workstation will have become an endangered species in 30 years with microprocessors being ubiquitous.

Weather data will be assimilated in meteorological models in near real-time and used to predict weather further into the future than anyone can now envision. Far greater numbers of sensors with galactic sensitivity will permit predictions of weather years in advance and with regression analysis achieve the granularity and fidelity of measurements to permit us to influence and change the weather. Your decision support systems could be the enabler.

The multi-tiered cube will replace the transistor, providing unfathomable computing permutations. We will no longer be restricted to binary choices. Boolean algebra will necessarily surrender to the inexorable passage of time and a higher order of mathematics. We will eventually be able to use the speed of electrons first, then the speed of photons, to add other computational dimensions. Silicon will yield to erbium and electrons to photons. Power requirements will plummet and a wide variety of alternate power sources will be available. Our body temperature will power electronic devices worn in our garments. Microelectronics will have transitioned to nano-electronics.

We have progressed from the 8-bit chip architecture with 256 addresses are about 10 times larger than our alphabet, to a 16-bit chip with 65,536 addresses) approximately the size of the population of a midsize city, that must be ceremoniously retired to

achieve Y2K compliance. The current predominately 32-bit architecture provides 4.3 billion addresses or about the total adult population of the world. In two years, we will have a 64-bit chip, with its inherit 18 quintillion addresses or the equivalence of the grains of sand at a large beach.

Computer memories and databases now measured in terabytes (10 to the 12th) will blossom to petabytes (10 to the 15th), then exabytes (10 to the 18th) and thence zettabytes (10 to the 21st).

Bandwidth will become a commodity and rates insensitive to time and distance. Today 3.5 terabits per second can be sent over a single strand of typical optical fiber, which is greater than the capacity of the entire world's Internet. The available bandwidth will double every two years. By 2004 there will be 80 zettabytes (10 to the 21st) of available bandwidth.

Chip speeds will continue to increase at an exponential rate to gigabytes within a decade and the measure of how many transistors, now about 100 million, are on a chip will give way to the measure of how many Central Processing Units (CPUs) can be placed on the same real-estate. The price of microchips will drop to a penny. We will drop by a convenience store and pick up a six-pack of microchips and Bud Lite, with the latter costing more.

Computer power has increased by a factor of ten billion in the past thirty years and there is no reason to believe that future performance increases will not at least keep pace.

There will be a Proteus global information systems forged on the anvil of technology that will liberate us from the dependence on people, geography and time and be a global utility.

Voice activated, controlled and operated computers will become common place, permitting the operator to focus on the tasks at hand and decision making, instead of data reduction and manual information retrieval. Using genetic algorithms and agents that can communicate between each other, software will adapt and evolve to solve problems, without new programming. Ever increasing number of attributes and/or dimensions assigned to objects will permit unfathomable precision.

There will be human-like reasoning machines that will emulate and eventually supersede the capability of man's brain. Computers will surpass the raw computing capability of the human brain within the next decade. There will be neural networks that will emulate our brain's parallel processing structure and derive inductive conclusions. These thinking machines will discover new knowledge, actually create knowledge, through digital insight, solve problems by silicon intuition and be adaptive by learning from

mistakes or adjusting to a changing environment. The veritable issue is not that computers will begin to think like humans, but that humans begin to think like computers. We want to release not limit the power of human ingenuity. Computers will facilitate this.

In biotechnology, silicon chips are being used to interface directly with human nerves. These neurotransducers will allow one to interface with, activate and control a computer with one's thoughts by wearing a headband. The Biotechnology Age will dwarf the Information Technology Age and turn hospitals into museums. Highly sensitive and inexpensive sensors, particularly biotechnical ones will be combined with cheap lasers and powerful microprocessors to perform medical miracles. Smart bathrooms will monitor people's health, i.e. urine can be checked for diabetes, etc. Biochips coated with millions of DNA probes in microscopic checkerboard patterns will be scanned to make medical diagnoses that would be otherwise prohibitively expensive and time-consuming. Algorithms used to detect, locate and identify enemy submarines from acoustical sensors are now being used to detect blood flow disturbances caused by artery restrictions in humans, reducing and eventually eliminating the requirement for angiography.

The complete human genome will be decoded within five years, providing us a human blueprint. We will be able to analyze the pattern of x-rays scattered off a DNA molecule and reconstruct the atomic structure of DNA and grow replacement organs. There will be undetectable hearing aids, inplantable cardiac pacemakers and heart bypasses will be conducted without open-heart surgery. We will move from curing diseases to preventing them and migrate from chemical means of treatment to molecular biological ones.

Soon 25 nations will have earth observation satellites, many with one meter resolution, making the expression "nation state privacy" an oxymoron. Soon, they will be able to determine from space when you need a haircut. Seemingly, nobody of consequence accepts these obvious diagnoses. A current Defense Science Board study will give DOD a wake-up call.

There are two gapping technologies necessary for us to achieve Information Dominance. One is multi-level security, which could be and should be solved instantaneously, by accepting a risk management vis-à-vis a risk avoidance policy. The other area that inflicts an incredible penalty in maneuverability, stealth, etc. is our antediluvian antenna, the spinal cord for weapons and communications systems. These extremely vulnerable, bulky, mechanical and predominately single frequency and narrow-bandwidth antennae must be replaced with shared aperture, conformal, wideband, multiple-frequencies, electronically steered and high gain ones.

Am I quixotic? Certainly not! Our future is as bright as that bolt from the heavens that Saul witnessed on the road to Damascus. We are indeed surrounded with fantastic opportunities, brilliantly disguised as unsolvable problems.

You have honored me by your gracious attention. Thank you! A “thank you” filled with more genuine emotion than the words were ever intended to convey. May the most that you wish for be the least that you receive and may your worst tomorrow be better than your best yesterday. I will now attempt to answer any of your questions.

The Components of the "Decision Industry"

Col Anthony Wood
USMC (ret.)
Director of Applied Research, CAD Research Center

A young soldier raises his rifle as he confronts rioters during a peace-keeping operation. A junior broker in Jakarta clicks his mouse initiating an international currency transfer. A sailor manning the air defense missile system identifies what appears to be an attacking aircraft on his air defense radar screen. A rail traffic controller moves a giant freight train to an alternate route. Unable to communicate with their parent unit, a Marine squad confidently continues on patrol in an alien urban metropolis. A corporate staff meets "on the net" across the country and an hour later orders an abrupt change in product mix affecting millions of dollars in sales.

Will the soldier shoot? What happens if he is filmed by CNN as he squeezes the trigger? How does the junior broker know the conditions are ripe for an international funds transfer, and by what authority is he executing it? How many lives will be affected by the radar operator's decision — and how will he make it? With hundreds of trains moving on dozens of tracks at varying speeds, where does the rail traffic controller gain his confidence? Faced with communications blackouts and uncertainty, why do the Marines proceed into the urban dark instead of freezing? What happened to three inch thick corporate plans supporting day long conferences and months-long product change cycles?

If you answered "computers" or "training," or perhaps even "decision support," congratulations! You can join a very small group of men and women who are sensitive to the enormous changes underway in how we make complex decisions. But don't pat yourself on the back too quickly; you didn't get it right, only "close." And that isn't enough. Whether as individuals, as members of a commercial firm, or as military leaders, we are increasingly facing difficult problems which must be solved "in stride." Frequently the problems are complex. They usually involve great uncertainty and often carry high risk in outcomes affecting lives, fortunes, or even national policy. Further, the tempo, whether of combat or competition, may well preclude an approach involving "sequentially staffed changes to a three inch thick plan." Contemporary problems are far more likely to demand a collapsed planning-execution cycle whose only link to the initiating formal plan was to use it as the "stepping-off point." We are in the midst of rapid change, change which has altered and continues to alter our traditional approach to corporate and military decision making.

Global Trends and Their Implications

The on-going digital revolution and the evolution of a speedy worldwide communication network are the dominant technological trends of our time. These two trends have enabled military and corporate decision making by vastly increasing the data which could be manipulated while dramatically decreasing the time required for that manipulation. Software developments in conjunction with quantum jumps in memory capacity and processing speed have permitted simultaneous exploitation of large multiple databases. Meanwhile, the advent of reliable commercially-affordable satellite-based communication links and the explosive growth of networks such as the world wide web offer new options for real-time and near real-time links between systems.

However, the same trends that fostered these benefits also introduced new complexities and complications for decision makers. "Data overload" can drown the decision maker in a sea of "facts" without providing him the means to sift for accuracy or currency, much less convert the data into meaningful information. The variables in the corporate and military decision processes have grown far more complex as worldwide connectivity allows bull market and battlefield to intrude directly in near real time. As technology has advanced, we have developed faster more complicated and inter-related processes. But the fact that the processes are more complicated and interconnected also opens up the potential for having to deal with multiple simultaneous impacts vertically and horizontally across the organization. As if this weren't challenge enough, global connectivity has reduced time for reflection and reaction to the vanishing point. In many cases we will have little or no warning, a situation which argues strongly for formal decision making training for "front line leaders." To complicate the situation further, countless others viewing the same phenomena may take actions of their own. The problem sets are not only complex, they are constantly changing.

The increasingly public nature of events is influencing every aspect of military and corporate decision making. The detailed "rules of engagement" prescribed for contemporary military operations have a long pedigree, but the impetus for their current prominence is due in no small part to the likelihood that the international media will "film" the war and transmit it to a global audience as it occurs. "Life as digital photo journalism" also means that military leaders, corporate executives, and policy makers must be ready to act before others leverage the media weapon.

The most gifted leaders have responded with a variety of actions. Sensing the need for faster reaction to developments, some have proposed collapsing the planning and execution cycles into a single continuous feedback-linked process. Recognizing that only through the exercise of initiative at every level can simultaneous impacts be dealt with, these leaders have sought to replace hierarchies with decentralized organizational structures fed by open access to previously restricted information. Knowing that complex problems will require inputs from many systems and sources, they have

sought to develop data mining and data fusion capabilities. A few, recognizing that filtering data flows for currency and reliability is not enough, have sought decision support systems that can transform the data into implications and inferences tailored to the needs of the decision maker. Still others have recognized that only through a degree of continuous "self adjustment" can an organization adjust its actions to this dynamic frame. In the military, this recognition finds its clearest expression in the use of concepts such as "commander's intent" campaign "end state", and a spirit of opportunism as the binder helping to achieve unity and coordination among decentralized elements. There are many more examples, but in each case these reform initiatives focus on the key elements of the collaborative decision process portrayed in Figure 1 below.

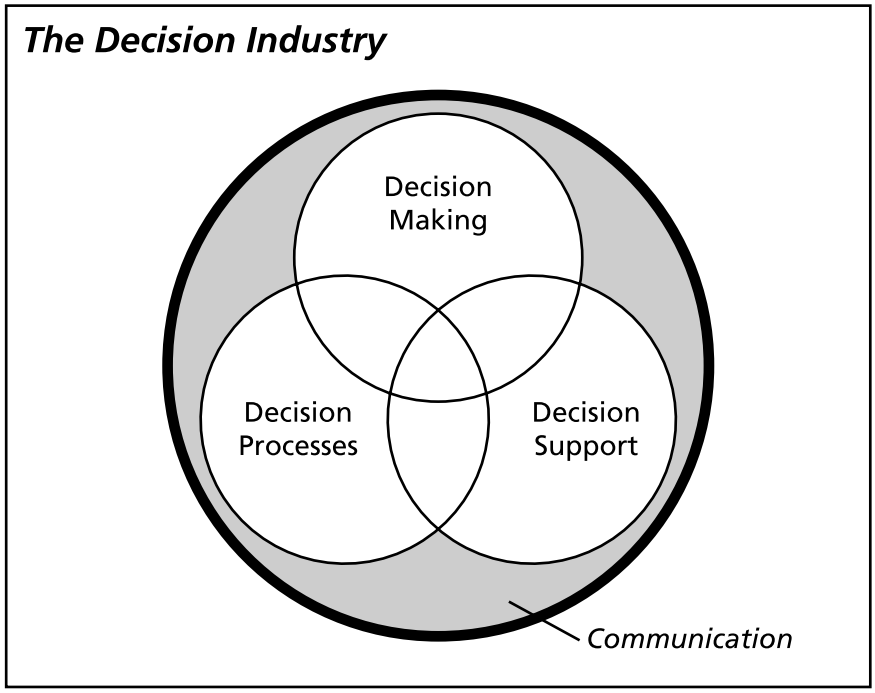
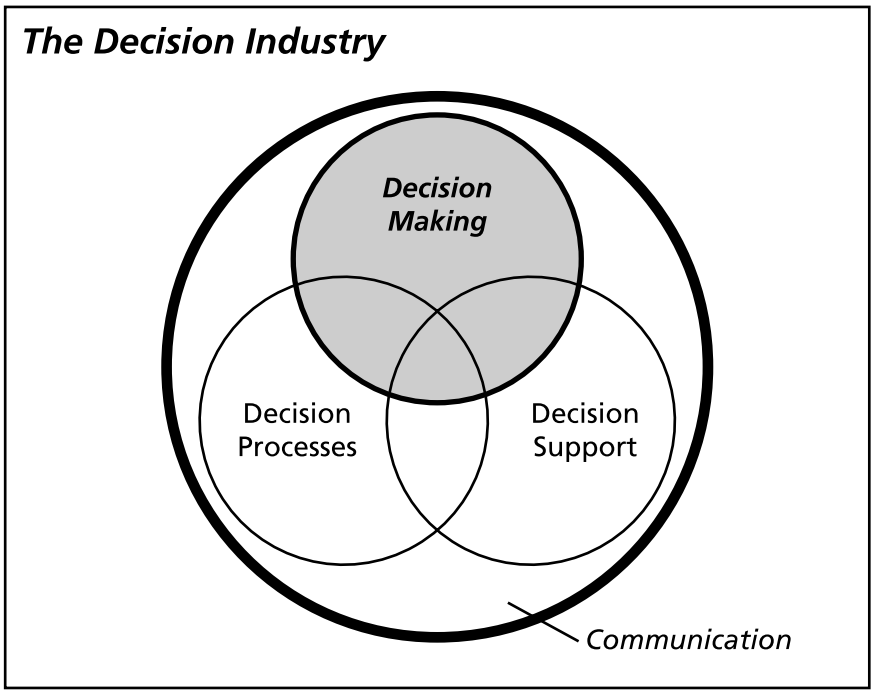


Figure 1: Key elements of the collaborative decision process.

When organizations sense that their key reactions are too slow or ineffective, then the question is, "What is the problem?" Is it poor decision making skills ? Is it the result of flaws in the decision process, the sometimes tortuous network of information and data flows which feed key decisions? Is it the absence of effective collaborative decision-support systems designed to transform data into more useful inference and implication? Or, is it the reliability and responsiveness of the communication linkages which bind these three closely related spheres of activity?

There must be a degree of balance and compatibility between these spheres of activity which make up the "decision industry." A good collaborative decision support system cannot overcome the weaknesses of an indecisive leader. In the same way, a good

collaborative decision support system will never realize its full potential to assist even a decisive leader if data flows are corrupted, incomplete, or untimely. But even a functional collaborative decision support system fed by timely accurate data flows can be crippled by hierarchical organization, restrictions on access to key information, or a slavish commitment to a balky process. In short, providing effective decision support demands a tailored decision support system fitted to the user and a clear understanding of the purposes of and the inter-relationships which exist between the three spheres of activity which makeup the "decision industry" in Figure 1. The first of these spheres is decision making and the individual skills which it demands.



Decision Making

Most of us are now familiar with the image of the "strategic corporal," the young fighter whose publicized actions could change the course of a war. With the cameras rolling in the background, we can only hope that he is prepared for decisions under stress with life and death — and perhaps national policy — in the balance. He may be a NATO pilot over Yugoslavia, a U. S. Marine Corporal in Mogadishu, the rail dispatcher in Houston, or one of hundreds of young brokers manipulating international capital flows. Regardless, individual decision-making skills play a central role in this fast breaking, public, complex, and risky future. Can he or she be trained to make better decisions, to handle stress, and to proceed in the face of uncertainty?

The answer is an unequivocal "yes." A recent case study involving the Marine Corps Warfighting Laboratory and its Sea Dragon program of experiments illustrates what can be done.

In early 1995, the Commanding Officer of the Marine Corps Warfighting Laboratory met in New York with Dr. Gary Klein, Chief Scientist and CEO of Klein Associates. The Marine made a strange request: help the Lab improve the individual decision skills of sixteen young Marine Corporals and Sergeants. It was a strange request in several respects. Although he had extensively examined decision processes in the Corps, Klein had never tried to "teach" decision making under stress and doubted that it was practical. And why focus on Corporals and Sergeants? In the course of a day-long discussion, some things were clarified. In seven weeks time, these young men were to lead their infantry squads in simulated combat in a dramatic experiment called Hunter Warrior. They would be part of a small experimental Marine Air Ground Task Force that would employ a new decision process, a new model command element, and new tactics enabled by technology in an eleven day force-on-force instrumented series of clashes. Their "foe" would be a larger traditional mechanized ground force.

Traditional tactics do not involve widely dispersing small units such as squads other than in patrolling activity. Even then distances are relatively close to the parent headquarters. In Hunter Warrior, the Marines envisioned squads operating in a widely dispersed fashion as far as one hundred kilometers forward of their parent unit. Further, they would be operating independently within the thirty four hundred square kilometers that made up the instrumented "battle box." It was hoped that, in combination with a new decision process, the eyes, ears, opportunism, and knowledge of these young combat leaders would result in a stream of information that would enable their parent task force to operate inside the opposing commander's "OODA" loop. In doing this, they were to be equipped with new technology in the form of precision GPS location beacons, palm top computers, and access to a wide range of information supplied through a new decision-support system. If the new tactics were to get a fair evaluation, the squad leaders — and their squads — would have to be thoroughly trained in both tactics and technology, embody great self confidence, and exhibit a strong spirit of opportunism. These young men would carry far more than the weight of their packs and rifles on their shoulders. They recognized this and they were worried.

With seven weeks to go, both the squad leaders and Klein continued to harbor deep reservations. Klein continued to harbor doubts that decision making could be "taught." Nevertheless, he reluctantly agreed to provide an experimental program of "individual decision skills training" for the squad leaders. The rest, as they say, is history. In the short six weeks that remained before Hunter Warrior commenced, Klein and his team led the Marine squad leaders and their men through a participatory program that transformed their attitudes and mental capabilities. In the final analysis, the performance of this group of confident aggressive and opportunistic young leaders astounded Marine observers and subsequently resulted in a whole series of new training initiatives for the preparation of young Marine battle leaders.

Klein Associates has since built on the Hunter Warrior experience and developed a series of programs designed to enhance individual decision skills. This recognitive and intuitive approach accepts uncertainty as a constant in situations of great stress. Klein's most recent book, Sources of Power, provides among other things a detailed discussion of individual decision skills.

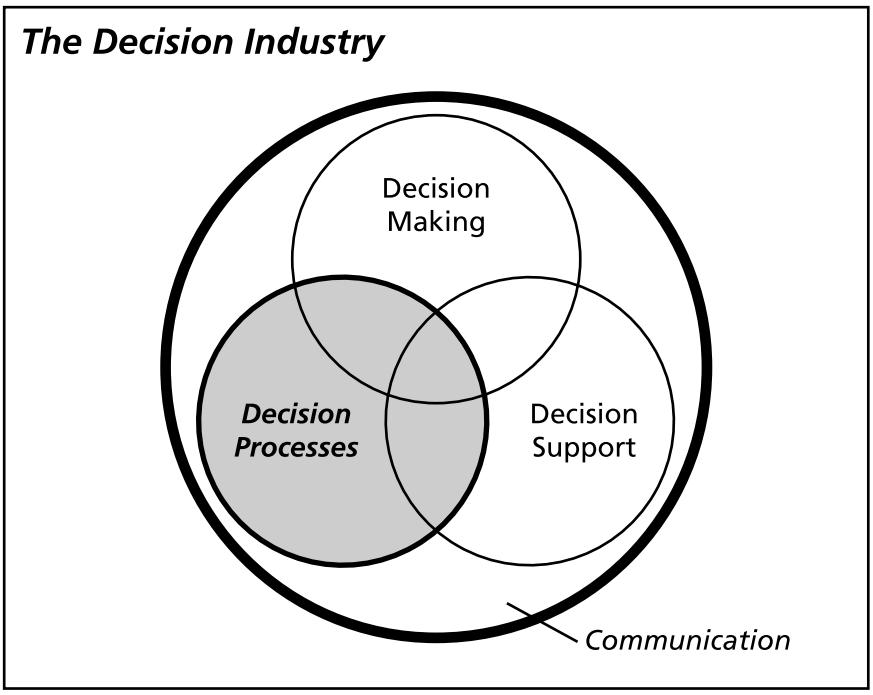
From all of this, several things stand out. First, if you want to become a better decision maker then you must make decisions often. You must make them in a stressful realistic environment tailored to your responsibilities. And, you should listen to the qualified observers who witness your decisions and consider their critiques before resuming "training". In short, the implication is that whether formally or informally, only through continuous stressfull decision training will you maintain your edge.

Among the upshots of Hunter Warrior is that Klein Associates now has a very successful individual decision skills training business. A second upshot has been the Marines' adoption of GAMA Corporation's "Collins Combat Decision Range" sponsored and developed by the Marine Corps Warfighting Lab. This PC-based "range" provides a portable, immersive, stressful, realistic, multimedia decision environment which can be tailored to the needs of the decision trainees. Critically, it fulfills one of the more difficult requirements necessary to improving individual decision skills: it allows Marines to make decisions frequently in a simulated stressful combat environment. Its early success with the Marine Corps Warfighting Laboratory has led to plans for widespread application among the Corps' operating forces.

The Decision Process

Enhancing individual decision skills is one key to improving decision making. As figure 1 indicates, a second key is insuring an effective decision process. Organizational structures and their connecting information and data flows come in every shape and variety. However, as implied earlier in this article, excessively hierarchical organizations operating in the "information hoarding" mode are likely to find themselves increasingly less effective in dealing with the pace and scope of contemporary change. On the other hand, those dealing with this issue most effectively have combined a "commander-led process", flatter structure, and more open access to information with the adoption of tailored collaborative decision support. When data is supplanted by information ... when information access is opened... when authority to act on the newly available information is pushed down...when everyone understands intent… and when collaborative decision support is offered to all from top to bottom... when these circumstances combine, the result can be revolutionary.

The concept of "end state" and "intent" are fundamental to process improvement. End state is the terminal set of conditions to be achieved at the culmination of the campaign. Intent is far more transitory, usually prescribing the result that a commander would like to see achieved in the next period of action. Intent provides the glue that brings unity of

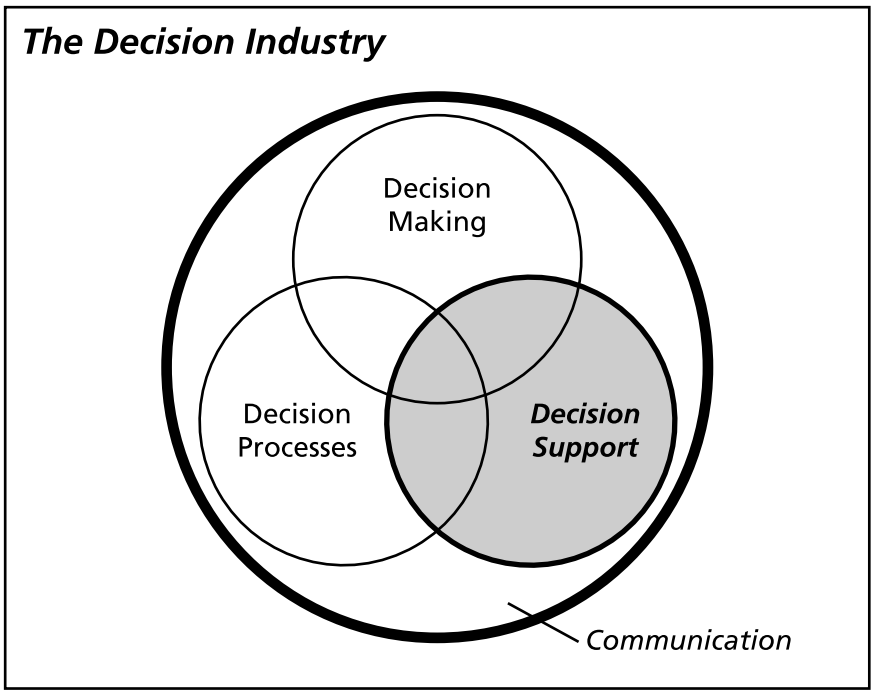


action. Whether in or out of communication with the parent unit, a small unit can proceed so long as there is clear understanding of intent. The same is true in corporations — intent lays out the result to be achieved in the next performance period within the larger corporate campaign. Intent is always related to end state, representing the commander's view of the results needed now in order to continue to make progress toward the final goal. But these two closely related concepts contribute more than unity, important as that is. Together, they impart a sense of opportunism to the "field units" whether military or commercial. In place of step-by-step central direction, it is this all important spirit of opportunism focused by intent which enables the organization to effectively respond at multiple levels across its breadth.

The Role of Decision Support

Collaborative decision support systems supporting an effective decision process can be vital aids to exploiting intent and opportunism. It is fair to say ***that the purpose of the decision process and supporting collaborative decision support systems is to discipline the decision making environment and tailor it for the decision maker.***

But doesn't visualization accomplish this "disciplining?" Yes... and no. It depends on what is being "visualized." A whole literature has grown up around the term. It is often held up as fundamental to effective decisions making. However, contrary to popular literature, there is no "information overload" threatening to bury leaders in either military command posts or in corporate suites. In fact the popular fascination with visualization obscures the real problem. That problem is "data overload." The drive to visualize data often acts to conceal a more fundamental challenge: ***data must***



be transformed into inference and implication as it enters the decision environment. Collaborative decision-support systems are the major players in executing this vital transformation.

Agent-based decision-support systems get mixed reviews depending on the user's experience. However, no matter their specific design these systems must meet four criteria if they are to discipline the dynamic problem sets facing the key decision maker.

First, these systems should satisfy a compelling need in a complex decision situation. In any case, the decision-support system should aid in solving difficult and vital problems which would otherwise prove intractable.

The second criteria is that these systems should provide useful assistance in the user's context. The decision support tool should be shaped and its technical workings subjugated to the requirement to be an intuitive and natural helper to the decision maker. Arcane screens populated with numbers and alien symbols often "illuminated" by tiny boxes filled with stilted technical terminology are too often the norm. In the final analysis the user interface ***is*** the system. If the user interface is an intuitive design and reflects the special characteristics of the decision domain, then the odds are that it will prove genuinely useful and will be used.

Developers should never be left to design this interface. Why not? The mental strengths which make a good programmer/developer are also associated with skills such as classical music composition or complex electrical circuitry design. These

conceptual strengths enable these men and women to design wonderfully complex systems, and there is a natural desire on the part of these artisans to have this artistry recognized. Why hide genius behind a mundane but functional GUI? The answer is simple: because this isn't a fine arts competition. The decision process and the decision support systems which aid it have one goal: ***to provide knowledge at a glance***. While complexity and elegance certainly have their place in our lives, these qualities are hostile to this all important goal. GUIs must be clear, functional, and intuitive.

The third requirement for an effective decision support system may appear to be a "blinding flash of the obvious:" these systems should be collaborative. The most effective decision-support systems embody a fundamental system design which leverages the unique capabilities which both sides contribute when the computer and the experienced operator assume complementary roles in the decision process. Humans should do what we do well, conceptualize, use emotion, employ intuition, and engage in complex communication. Computers should contribute the special strengths which digital technology offers: huge memory, rapid processing, attention to detail, etc. Far too many systems are being designed absent this all important partnership.

Finally, modern decision support systems should be adaptive. The system should adapt to the user's needs by permitting a high degree of user-defined functionality. Examples? Opportunity cost should be capable of being measured in terms set by the user (e.g., "missions delayed", etc.). The system should provide for "views" set by the user which are then continuously maintained by agents actively searching for the latest information and then incorporating it. A good decision support system should adapt to circumstances identifying the commander's critical information requirements and then establishing agent-maintained "views" which display the latest information affecting these requirements. Because the use of intelligent agents allows the system to actively seek data, convert it to information, and then post it in a user-defined view, the potential for adaptation is enormous. Clearly, this will be a major area for development in the years ahead.

Collaborative decision-support tools can perform a wide range of tasks. A few examples may serve to illustrate this wide-ranging utility. A function which can be especially useful in our technical society is to provide expert assistance to the individual whose training, background, or experience isn't up to the task he has been assigned. ICODES, the Integrated Computerized Deployment System, was specifically designed as a collaborative tool to assist stowplanners in the complex task of loading the Military Transportation Command's fleet of ships. Now being installed in 55 ports worldwide, ICODES is an alert "intelligent assistant" for stowplanners from Alexandria, Egypt to Okinawa.

Another system which bolsters individual expertise is COACH. COACH is being developed for the Office of Naval Research (ONR) to provide a full range of technical

assistance to the individual sailor repairman. COACH will figuratively sit on the technician's shoulder and, using a variety of intelligent agents, will assist the technician in locating appropriate diagrams, performing diagnosis, etc. COACH will deploy with the Navy's ships to help offset the absence of experienced maintenance personnel in the modem fleet

Another tool designed to provide skilled assistance to Navy personnel is CIAT, the Collaborative Infrastructure Assessment Tool. In addition to assisting the water front operations officer is selecting an appropriate berth and range of port services for the Navy's ships in San Diego, CIAT will provide the capability to recall patterns from previous berthing actions and to review histories. Thus CIAT offers the capability to rapidly recall the past, examine and game it, and then apply it to the present and the future.

Among the most challenging tasks which face collaborative decision support systems is to assist the user in monitoring, identifying conflicts, and coordinating actions in near real time in complex dynamic situations. Few situations are more dynamic and uncertain than the tactical battlefield.

IMMACCS, the Integrated Marine Multi-Agent Command and Control System, is an experimental system designed to provide a high level of situation awareness from the Commander to the Squad Leader. A first generation example of adaptive command and control, IMMACCS provides agents which leaders at different levels can direct to perform specific services. Aspects such as potential ROE violations in fires planning, "blue-on-blue" situations, enemy activity and status in a given area, etc. are examples of the functions in which agents assist in IMMACCS.

FEAT, the Force Evaluation and Assessment Tool, provides a set of intelligent agent tools for rapidly developing force options in crisis response. Using lift agents, readiness agents, availability agents, and capability selection tools, FEAT identifies conflicts and assists the planner in rapidly assembling a force list that meets mission demands while identifying qualifying conditions which must be included in deployment and employment plans.

SEAWAY, a system now under development, combines aspects of all of the systems discussed thus far. SEAWAY will provide end-to-end visibility for all maritime logistic support during contingencies. SEAWAY is designed to satisfy sea basing demands for JV 2010. As such, it is focused on supporting OMFTS, STOM, and other joint force deep maneuver concepts. Among many capabilities, SEAWAY will track supplies, project availability, and coordinate and control unopposed ship-to-shore and ship-to-objective delivery of supplies to the forces operating ashore. As it performs these functions it will also provide a range of functions vital to sea basing including the capability to locate and project timelines to access specific cargo items embarked aboard the sea base.

The intelligent agent technology which makes these systems "ever alert assistants" to the user can perform many functions in the background allowing the user to concentrate his or her attention on the decision at hand. Among such background functions is an agent's ability to send data "out" to a simulation or technical model, receive the results, and then post them to a continuously maintained "operational view" set by the user. In FIRE AIDE, a collaborative decision-support system now under design, agents will automatically perform this function each time that they sense a significant change in weather conditions on the fire ground. As a result, the Incident Commander on the scene will be alerted that there has been a change, and at his convenience, will switch to the agent-maintained "view" of the latest possible projection plotting the likely spread of the fire.

The absence of activity may be as significant as the incoming reports of incidents. Agent-based systems can report what is NOT happening which might reasonably be expected to be occurring. Why is there an information void? Agents, embodying an object model of the domain, can alert users at many levels to what might be expected under existing circumstances. The user can then contrast that with incoming information for decision. This function is critical to disciplining the decision environment for the commander. It is not enough to insure currency, accuracy and relevance. It is not enough to replace visualized data with information. The absence of activity is itself information..

Finally, collaborative decision support systems are "meta systems" in that they accept the data outputs from existing systems, convert these into objects, and then provide information and inference for decision makers. Not only does this filter and transform multiple data streams into usable decision support, but it provides a new lease on life for expensive already installed legacy systems and allows their use for an extended period. It also avoids creating yet another unique data base. ***As "meta systems" then, collaborative decision support can not only enhance decision making but at the same time revalue the sunk investment already made in large legacy systems.***

Conclusion

Collaborative decision support systems can be valuable assistants in disciplining the decision environment. However, to realize their full potential, such systems need to be carefully tailored to the needs of the user. At the same time, the two other aspects of the "decision industry" must achieve a rough equilibrium. The individual decision skills of leaders must be constantly honed under pressure in a realistic setting. Assuming the presence of experienced decision makers assisted by a capable decision support system, then the decision process remains. Information should flow horizontally and vertically throughout the organization. Guided by commander's intent, and understanding the goal or end state to be achieved, the organization can mount an agile decentralized response to dynamic change.

Finally, there is an element of risk in seeking to improve institutional decision making capability. Resistance to change is a very real obstacle, especially when change may require major adjustments by those long accustomed to business as usual. For example, we can all agree that the goal is an organization capable of simultaneous opportune decision-action sequences on many levels at many locations. But, as adoption of commander's intent and opportunism as guiding principles illustrates, the price to be paid for that more nimble organization is sharing information and granting increased authority downward. This price can be too high for some. Recognizing this is important in crafting a successful strategy to introduce improvements in the way organizations make decisions and prepare their decision makers. Unfortunately, even well considered strategies are sometimes met by a decision to silence the spokesmen of reform, "...to shoot the messenger." Don't despair. We live in an era of streaming change, and time is not on the side of the status quo. Like overweight executives whose blood oozes through plaque constricted arteries, layered hierarchical organizations with balky decision processes are inviting crisis. Its not a question of "if," only of "when."

Collaborative Decision-Support and the Human-Machine Relationship

Jens Pohl
Executive Director, CAD Research Center

Some Underlying Human Realities.

Human beings are inquisitive creatures who seek explanations for all that they observe and experience in their living environment. While this quest for understanding is central to our success in adapting to a changing environment, it is also a major cause of our willingness to accept partial understandings and superficial explanations when the degree of complexity of the problem situation confounds our current cognitive capabilities. In other words, a superficial or partial explanation is considered better than no explanation at all. As flawed as this approach may be, it has helped us to solve difficult problems in stages. By first oversimplifying a problem we are able to develop an initial solution that is later refined as a better understanding of the nature of the problem evolves.

Unfortunately, now we have to contend with another characteristic of human beings, our inherent resistance to change and aversion to risk taking. Once we have found an apparently reasonable and workable explanation or solution we tend to lose interest in pursuing its intrinsic shortcomings and increasingly believe in its validity. Whether driven by complacency or lack of confidence, this state of affairs leads to many surprises. We are continuously discovering that what we believed to be true is only partly true or not true at all, because the problem is more complicated than we had previously assumed.

At times a particular set of explanations, or school of thought, becomes entrenched as a paradigm that is not easily broken. Kuhn (1977) has drawn attention to the stagnating influence on progress of scientific paradigms, the resistance experienced by individuals or small groups that wish to correct flaws in a paradigm, and the resurgence of innovative activity after the paradigm has been broken. If experts in science will succumb to this weakness in human nature then how much more difficult will it be for a layperson to maintain a discerning mind?

Throughout modern history these intrinsic human characteristics of resisting change, avoiding risks, and endeavoring to maintain status quo have created a tension in society. A prominent example is, of course, the Information Revolution driven by the rapid development of computers and communication systems and their potential assistance in human decision making endeavors.

The Increasing Complexity of Problems in a Global Community.

The complexity of problems faced by human society in areas such as management, economics, marketing, engineering design, military operations, and environmental preservation, is increasing for several reasons. First, computer-driven information systems have expanded these areas from a local to an increasingly global focus. Even small manufacturers are no longer confined to a regionally localized market for selling their products. The marketing decisions that they have to make must take into account a wide range of factors (e.g., international currency rates, political alliances, and climatic conditions) and a great deal of knowledge (e.g., language, conventions, and cultural beliefs) that is far removed from the local environment.

Second, as the scope of the problem system increases so do the relationships among the various factors. These relationships are difficult to deal with, because they require the decision maker to consider many factors concurrently. Although the biological operation of the human brain is massively parallel, our conscious reasoning processes are sequential. Simply stated, we have difficulty reasoning about more than two or three variables at any one time.

Third, as the scope of problems increases decision makers suffer simultaneously from two diametrically opposed but related conditions. They tend to be overwhelmed by the shear volume of information that they have to consider, and yet they lack information in many specific areas. To make matters worse, the information tends to change dynamically in largely unpredictable ways.

It is therefore not surprising that governments, corporations, businesses, down to the individual person, are increasingly looking to computer-based decision-support systems for assistance. This has placed a great deal of pressure on software developers to rapidly produce applications that will overcome the apparent failings of the human decision maker. While the expectations have been very high, the delivery has been much more modest. The expectations were simply unrealistic.

It was assumed that advances in technology will be simultaneously accompanied by an understanding of how these advances should be applied optimally to assist human endeavors. History suggests that such an a priori assumption is not justified. There have been countless experiences in the past that would suggest the contrary. For example, the invention of new materials (e.g., plastics) have inevitably been followed by a period of misuse. Whether based on a misunderstanding or lack of knowledge of its intrinsic properties, the new material was typically initially applied in a manner that emulated the material(s) it replaced. In other words, it took some time for the users of the new material to break away from the existing paradigm. A similar situation currently exists in the area of computer-based decision-support systems.

The Rationalistic Problem Solving Tradition.

To understand current trends in the evolution of progressively more sophisticated decision-support systems it is important to briefly review the foundations of problem solving methodology from an historical perspective. Epistemology is the study or theory of the origin, nature, methods and limits of knowledge. The dominant epistemology of Western Society has been technical rationalism (i.e., the systematic application of scientific principles to the definition and solution of problems).

The rationalistic approach to a problem situation is to proceed in well defined and largely sequential steps (Fig.1): define the problem; establish general rules that describe the relationships that exist in the problem system; apply the rules to develop a solution; test the validity of the solution; and, repeat all steps until an acceptable solution has been found. This simple view of problem solving suggested a model of sequential decision making that has retained a dominant position to the present day. With the advent of computers it was readily embraced by 1st Wave software (Fig.2) because of the ease with which it could be translated into packaged, automated solutions utilizing the procedural computer languages that were available at the time (Pohl 1996).

1st Wave software assumes that problem solving is essentially a sequential process in which every subsequent step depends on the completion of the preceding step. This view of problem solving is far removed from real world experience, where project teams solve problems collaboratively and contribute to the decision making process whenever they have something useful to share with the other team members. Seldom, if ever, is a team member prevented from contributing information until a certain stage or milestone has been reached. On the contrary, team members are encouraged to exchange information freely in the hope that their contributions will accelerate the solution process and increase the quality of the solution.

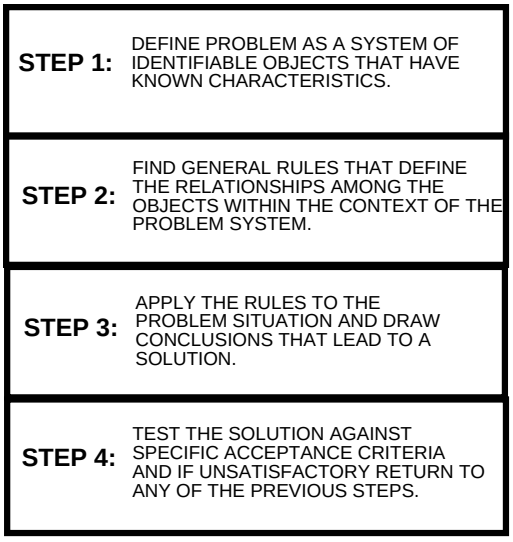


Fig.1: Solution of simple problems

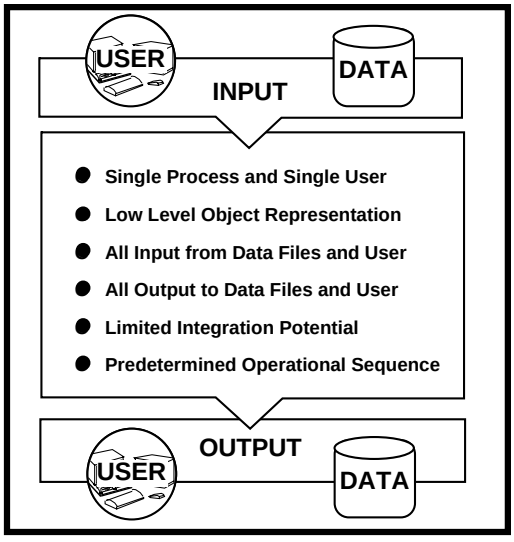


Fig.2: '1st Wave' computer applications

Over the past 50 years with the availability of more and more convenient and effective communication facilities, government and industry have been increasingly challenged by real world problems that are often very complex involving many related variables. Neither the relationships among the variables nor the variables themselves are normally sufficiently well understood to provide the basis for clear and comprehensive definitions. In other words, problem situations are often too complex to be amenable to an entirely logical and predefined solution approach. Under these conditions the analytical strategy has been to decompose the whole into component parts, as follows:

- Decompose the problem system into sub-problems.
- Study each sub-problem in relative isolation, using the rationalistic approach (Fig.1), and if the relationships within the sub-problem domain cannot be clearly defined then decompose the sub-problem further.
- Combine the solutions of the sub-problems into a solution of the whole.

Underlying this problem solving strategy is the implicit assumption that an understanding of parts leads to an understanding of the whole. Under certain conditions this assumption may be valid. However, in many complex problem situations the parts are tightly coupled so that the behavior of the whole depends on the interactions among the parts rather than the internal characteristics of the parts themselves (Bohm 1983, Senge 1993). An analogy can be drawn with the behavior of ants. Each ant has only primitive skills, such as the ability to interpret the scent of another ant and the instinctive drive to search for food, but little if any notion of the purpose or objectives of the ant colony as a whole. Therefore, an understanding of the behavior of an individual ant does not necessarily lead to an understanding of the community behavior of the ant colony of which the ant is a part.

Decomposition is a natural extension of the scientific approach to problem solving and has become an integral and essential component of rationalistic methodologies. Nevertheless, it has serious limitations. First, the behavior of the whole usually depends more on the interactions of its parts and less on the intrinsic behavior of each part. Second, the whole is typically a part of a greater whole and to understand the former we have to also understand how it interacts with the greater whole. Third, the definition of what constitutes a part is subject to viewpoint and purpose, and not intrinsic in the nature of the whole. For example, from one perspective a coffee maker may be considered to comprise a bowl, a hotplate, and a percolator. From another perspective it consists of electrical and constructional components, and so on.

Rationalism and decomposition are certainly useful decision making tools in complex problem situations. However, care must be taken in their application. At the outset it must be recognized that the reflective sense (Schön 1983) and the intuitive capabili-

ties of the decision maker are at least equally important tools. Second, decomposition must be practiced with restraint so that the complexity of the interactions among parts is not overshadowed by the much simpler behavior of each of the individual parts. Third, it must be understood that the definition of the parts is largely dependent on the objectives and knowledge about the problem that is currently available to the decision maker. Even relatively minor discoveries about the greater whole, of which the given problem situation forms a part, are likely to have significant impact on the purpose and the objectives of the problem situation itself.

Decision Making in Complex Problem Situations.

In several previous CAD Research Center publications we have drawn attention to the importance of internal and external relationships in complex problem situations (Pohl et al. 1997 (48-62), Pohl and Myers 1994). As shown in Fig.3, there are several characteristics that distinguish a complex problem from a simple problem. First, the problem is likely to involve many related issues or variables. As discussed earlier the relationships among the variables often have more bearing on the problem situation than the variables themselves. Under such tightly coupled conditions it is usually not particularly helpful, and may even be misleading, to consider issues in isolation. Second, to confound matters some of the variables may be only partially defined and some may yet to be discovered. In any case, not all of the information that is required for formulating and evaluating alternatives is available. Decisions have to be made on the basis of incomplete information.

Third, complex problem situations are pervaded with dynamic information changes. These changes are related not only to the nature of an individual issue, but also to the context of the problem situation. For example, a change in location of an enemy force (even within the same sector of the battlefield) could easily have a major impact on the entire nature of the combat situation facing the commander. Apart from the disposition of friendly forces under these changed conditions, the influence on target priorities, and the effectiveness of available weapons, such a relocation could call into question the very feasibility of the existing battle plan. Even under less critical conditions it is not uncommon for the solution objectives to change several times during the decision making process. This fourth characteristic of complex problem situations is of particular interest. It exemplifies the tight coupling that can exist among certain problem issues, and the degree to which decision makers must be willing to accommodate fundamental changes in the information that drives the problem situation.

Fifth, complex problems typically have more than one solution (Archea 1987). It is normally unproductive to look for an optimum solution, because there are no static benchmarks available for evaluating optimality. A solution is found to be acceptable if it satisfies certain performance requirements and if it has been determined that the search for alternatives is no longer warranted. Such a determination is often the result of

resource constraints (e.g., availability of time, penalty of non-action, or financial resources) rather than a high level of satisfaction with the quality of the proposed solution.

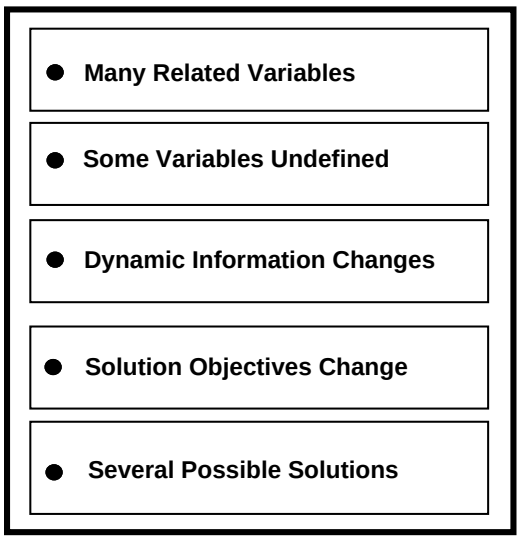


Fig.3: Character of complex problems

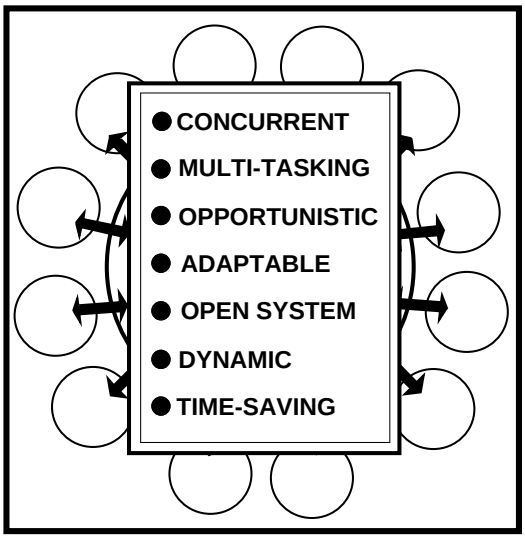


Fig.4: Parallel decision support

While human decision making in complex problem situations has so far defied rigorous scientific explanation, we do have knowledge of at least some of the characteristics of the decision making activity.

- Decision makers typically define the problem situation in terms of issues that are known to impact the desired outcome. The relative importance of these issues and their relationships to each other change dynamically during the decision making process. So also do the boundaries of the problem space and the goals and objectives of the desired outcome. In other words, under these circumstances decision making is an altogether dynamic process in which both the rules that govern the process and the required properties of the end result are subject to continuous review, refinement and amendment. ***Accordingly, the borderline between planning and execution is blurred by the constant need for replanning.***

- The complexity of the decision making activity does not appear to be due to a high level of difficulty in any one area but the multiple relationships that exist among the many issues that impact the desired outcome. Since a decision in one area will tend to influence several other areas there is a need to consider many factors at the same time. This places a severe burden on the human cognitive system. Although the neurological mechanisms that support conscious thought processes are massively parallel, the conscious operation of these reasoning capabilities is largely sequential. Under these conditions the

individual human decision maker is very much in need of assistance. ***The availability of computers would appear to offer welcomed support through parallelism (Fig.4), connectivity, and information access, as long as the human decision makers are able to effectively communicate their assistance needs to the computer.***

- Observation of decision makers in action has drawn attention to the important role played by experience gained in past similar situations, knowledge acquired in the general course of decision making practice, and expertise contributed by persons who have detailed specialist knowledge in particular problem areas (Mackinder and Marvin 1982, Mallen and Goumain 1973). The dominant emphasis on experience is confirmation of another fundamental aspect of the decision making activity. Problem solvers seldomly start from first principles. In most cases, the decision maker intuitively builds on existing solutions from previous situations that are in some way related to the problem under consideration. ***Again, computers should be potentially useful through their ability to store not only vast amounts of data but also higher level information and knowledge. It is not unreasonable to expect knowledge-based computer systems (i.e., software applications) to alert the user to past solutions and suggest how these might relate to the current problem.***

- Finally, there is a distinctly irrational aspect to decision making in complex problem situations. Schön (1983) refers to a *"...reflective conversation with the situation..."*. He argues that decision makers frequently make value judgments for which they cannot rationally account. Yet, these intuitive judgments often result in conclusions that lead to superior solutions. It would appear that such intuitive capabilities are based on a conceptual understanding of the situation, which allows the problem solver to make knowledge associations at a highly abstract level. ***This strongly suggests that a collaborative human-computer partnership is essential. Both must contribute their respective strengths and assist each other to overcome their respective weaknesses. Any attempt to automate the decision making process to the exclusion of the human element is not only likely to be counterproductive, but dangerous as well.***

Based on these characteristics the solution of complex problems can be categorized as an information intensive activity that depends for its success largely on the availability of information resources and, in particular, the experience and reasoning skills of the decision makers. It follows that the quality of the solutions will vary significantly as a function of the problem solving skills, knowledge, and information resources that can be brought to bear on the solution process. This clearly presents an opportunity for the useful employment of computer-based decision-support systems in which the capabilities of the human decision maker are complemented with knowledgebases, expert agents, and self-activating conflict identification and monitoring capabilities.

The Critical Importance of Information Representation in the Computer.

Although technological advances in computer hardware and communication systems have been truly astounding over the past 20 years, the direct utilization of these advances in the area of decision-support has been less than remarkable. The fact is that we are still using computers largely as ***data*** processing devices that perform only the most menial and least intelligent data transmission and manipulation tasks. While computers are performing these tasks with great speed and accuracy, and while they are able to provide connectivity among a virtually unlimited number of access points, the higher level and much more rewarding tasks of analyzing, interpreting and abstracting data as ***information*** and ***knowledge*** is almost entirely left to the human users (Fig.5).

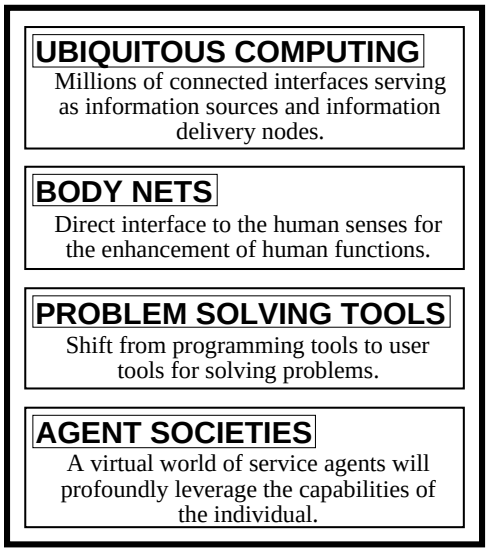


Fig.5: Evolving computer-human partnership

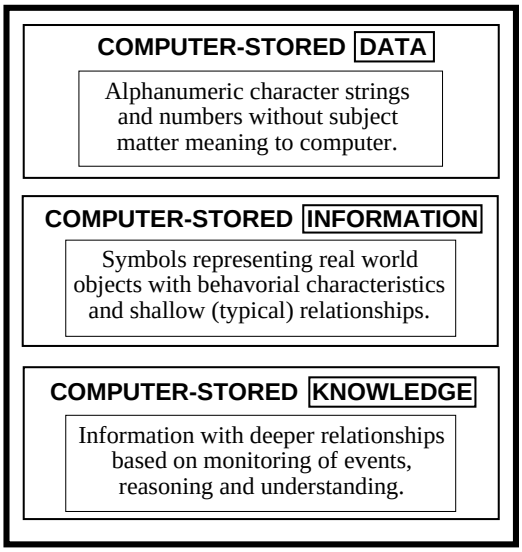


Fig.6: Data-information-knowledge

This serious deficiency has become increasingly apparent as technological advances have increased computing power, data storage capacities, and data transmission speeds by orders of magnitude in such a short period of time. Convenient global access to users and data has increased the need for information filtering, so that individuals might take advantage of the opportunities for material and personal profit that this connectivity and processing power present to the user. Needless to say, the capabilities of a computer to assist in the intelligent assessment of information are basically non-existent if the computer processes this information as bitmaps and alphanumeric text strings (Fig.6). ***Any significantly useful human-computer collaborative partnership carries with it the expectation that information is held within the system environment in a representational form that is, if not equivalent to, at least compatible with human cognition.***

The current approach for achieving this objective is to represent information in the computer as objects with behavioral characteristics and relationships to other objects (Myers et al. 1993). While this approach is hardly sophisticated it does allow real world objects (e.g., airfield, tunnel, building, weapon, tank) to be represented symbolically so that computer software modules can reason about them.

It is important to note that the relationships among these objects are often far more important than the characteristics that describe the individual behavior of each object. For example, the word *house* holds little meaning if we strip away the many associations that this word represents in our mind. However, such associations to our knowledge of construction materials, our experiences in having lived in houses, and our understanding of how our own home is impacted by external factors (such as rain, sunshine, neighbors, mortgage interest rates, and so on) constitute the rich meaning of the object *house* (Minsky 1982). Accordingly, any useful representation of information in the computer must be capable of capturing the relationships among the entities (i.e., objects) in the problem system.

While some of these associations are fairly static (e.g., a weapon is a kind of asset and a lethal weapon is a kind of weapon) many of the associations are governed by current conditions and are therefore highly dynamic. For example, as a platoon of soldiers moves through the battlefield it continuously establishes new associations (e.g., to windows in buildings from which snipers could fire on individual members of the platoon), changes existing associations (e.g., higher levels of risk as the platoon nears an active combat zone), and severs previous associations (e.g., as the platoon is forced to abandon its compromised command post).

Abstract concepts such as privacy, security and power, are less amenable to this approach since their meaning and role in our day-to-day activities is less easily defined. For example, the characteristics of *privacy* are neither static nor can they be accurately described in relational terms. They depend on a wide range of factors that relate to both environmental and personal circumstances and dispositions. These factors can be only partially accounted for through embedded knowledge and rules, and therefore become largely the purview of the human members of the collaborative human-computer partnership.

Nevertheless, even with these shortcomings this form of representation of real world objects can provide the basis of usable problem solving support and decision making assistance. Improvements are possible with the addition of knowledge bases and user interaction. In the latter case the user becomes as much a helper to the system as the system serves as an assistant to the user. However, this occurs in quite different ways. The system uses its computing and logical reasoning capabilities to monitor, analyze and evaluate the actions, requests and interests of the user in an opportunistic manner. The user, on the other hand, helps the system to understand the nature of the objects and relationships that it is processing in a more deliberate manner (Pohl 1995).

The reliance on object representations in reasoning endeavors is deeply rooted in the innately associative nature of the human cognitive system. Information is stored in long term memory through an indexing system that relies heavily on the forging of association paths. These paths relate not only information that collectively describes the meaning of symbols such as *helicopter*, *rifle* and *truck*, but also connect one symbol to another. The symbols themselves are not restricted to the representation of physical objects, but also serve as concept builders. They provide a means for grouping and associating large bodies of information under a single conceptual metaphor. In fact, Lakoff and Johnson (1980) argue that *"...our ordinary conceptual system, in terms of which we both think and act, is fundamentally metaphorical in nature..."*. They refer to the influence of various types of metaphorical concepts, such as 'desirable is up' (spatial metaphors) and 'fight inflation' (ontological or human experience metaphors), as the way human beings select and communicate strategies for dealing with every day events. Problem solvers typically intertwine the factually based aspects of objects with the less precise, but implicitly richer language of metaphorical concepts. This leads to the spontaneous linkage of essentially different objects through the process of analogy. In other words, the decision maker recognizes similarities between two or more sub-components of apparently unrelated objects and embarks upon an exploration of the discovered object seeking analogies where they may or may not exist. At times these seemingly frivolous pursuits lead to surprising and useful solutions of the problem at hand.

The need for a high level representation is fundamental to all computer-based decision-support systems. It is an essential prerequisite for embedding artificial intelligence in such systems, and forms the basis of any meaningful communication between user and computer. Without a high level representation facility the abilities of the computer to assist the human decision maker are confined to the performance of menial tasks, such as the automatic retrieval and storage of data or the computation of mathematically defined quantities. While even those tasks may be highly productive they cannot support a partnership in which human users and computer-based systems collaborate in a meaningful and intelligent manner in the solution of complex problems.

The Limited Role of *Visualization*.

Decision makers use various visualization media, such as visual imagination or simulation, drawings and physical models, to communicate the current state of the evolving solution to themselves and to others. For example, drawings, sketches and computer displayed images have become intrinsically associated with problem solving. Although the decision maker can reason about complex problems solely through mental processes, drawings and related visual images are useful and convenient for extending those processes. The failings of a drawing or sketch as a vehicle for communicating the full intent of the decision maker do not apply to the creator of the drawing. To the latter the drawing serves not only as an extension of long term memory, but also as a visual bridge to its associative indexing structure. In this way, every

meaningful part of the drawing is linked to related data and deliberation sequences that together provide an effectively integrated and comprehensive representation of the artifact.

From a technical point of view a great deal of headway has been made over the past two decades in the area of computer-based visualization. ***However, without high level representation capabilities even the most sophisticated computer generated images are nothing but hollow shells***. If the computer system does not have even the simplest understanding of the nature of the objects and their associations that are contained in the image then it cannot contribute in any way to the analysis of those objects. On the other hand, visualization in combination with high level representation becomes the most powerful element of the user interface of a decision-support system. Under these circumstances, visualization promotes the required level of understanding between the user and the computer as they collaborate in the solution of the problem.

The Complementary Role of Human *Intuition*.

Schön (1983 and 1988) has written extensively about the intuitive aspects of decision making. Although he focused primarily on engineering design as an application area, his views provide valuable insight into the solution of complex problems in general. Design has all of the common characteristics of complex problem situations, and some additional ones such as the desire for solution uniqueness, that make it a prime candidate for computer-based assistance (Pohl et al.1994).

In Schön's (1988) view designers enter into *"...design worlds..."* in which they find the objects, rules and prototype knowledge that they apply to the design problem under consideration. The implication is that the designer continuously moves in and out of design worlds that are triggered by internal and external stimuli. While the reasoning process employed by the designer in any particular design world is typically sequential and explicitly logical, the transitions from state to state are governed by deeper physiological and psychological causes. Some of these causes can be explained in terms of associations that the designer perceives between an aspect or element of the current state of the design solution and the prototype knowledge that the designer has accumulated through experience. Others appear to be related to environmental stimuli or emotional states, or interactions of both.

For example, applying Schön's view to the broader area of complex problem solving, a particular aspect of a problem situation may lead to associations in the decision maker's mind that are logically unrelated to the problem under consideration. However, when the decision maker pursues and further develops these associations they sometimes lead to unexpected solutions. Typically, the validity of these solutions becomes apparent only after the fact and not while they are being developed. In popular terms we often refer to these solutions as *creative leaps* and label the author as

a brilliant strategist. What we easily forget is that many of these intuitions remain unrelated associations and do not lead to any worthwhile result. Nevertheless, the intuitive aspect of decision making is most important. Even if only a very small percentage of these intuitive associations were to lead to a useful solution, they would still constitute one of the most highly valued decision making resources.

The reasons for this are twofold. First, the time at which the decision maker is most willing to entertain intuitive associations normally coincides with a most difficult stage in the problem solving process. Typically, it occurs when an impasse has been reached and no acceptable solution strategy can be found. Under these conditions intuition may be the only remaining course of action open to the decision maker. The second reason is particularly relevant if there is a strong competitive element present in the problem situation. For example, in command and control situations during the execution of military operations. Under these circumstances, strategies and solutions triggered by intuitive associations will inevitably introduce an element of surprise that is likely to disadvantage the enemy.

The importance of ***intuition*** in decision making would be sufficient reason to insist on the inclusion of the human decision maker as an active participant in any computer-based decision-support system. In designing and developing such systems in the CAD Research Center over the past decade we have come to appreciate the importance of the human-computer partnership concept, as opposed to automation. Whereas in some of our early systems (e.g., ICADS (Pohl et al. 1988) and AEDOT (Pohl et al. 1992)) we included agents that automatically resolved conflicts, today we are increasingly moving away from automatic conflict resolution to conflict detection and explanation. We believe that even apparently mundane conflict situations should be brought to the attention of the human agent. Although the latter may do nothing more than agree with the solution proposed by the computer-based agents, he or she should be given the opportunity to bring other knowledge to bear on the situation and thereby influence the final determination.

The Human-Computer Partnership

To look upon decision-support systems as partnerships between users and computers, in preference to automation, appears to be a sound approach for at least two reasons. First, the ability of the computer-based components to interact with the user overcomes many of the difficulties, such as representation and the validation of knowledge, that continue to plague the field of machine learning (Thornton 1992, Johnson-Laird 1993).

Second, human and computer capabilities are in many respects complementary (Figs.7 and 8). Human capabilities are particularly strong in areas such as communication, symbolic reasoning, conceptualization, learning, and intuition. We are able to

store and adapt experience and quickly grasp the overall picture of even fairly chaotic situations. Our ability to match patterns is applicable not only to visual stimuli but also to abstract concepts and intuitive notions. However, although the biological basis of our cognitive abilities is massively parallel, our conscious reasoning capabilities are essentially sequential. Therefore, human decision makers are easily overwhelmed by large volumes of information and multi-faceted decision contexts. We have great difficulty dealing with more than two or three variables at any one time, if there are multiple relationships present. Under these circumstances we tend to switch from an analysis mode to an intuitive mode in which we have to rely almost entirely on our ability to develop situation awareness through abstraction and conceptualization. While this is our greatest strength it is also potentially our greatest weakness. At this intuitive meta-level we are vulnerable to emotional influences that are an intrinsic part of our human nature and therefore largely beyond our control.

Computer capabilities are strongest in the areas of parallelism, speed and accuracy (Fig.8). Whereas the human being tends to limit the amount of detailed knowledge by continuously abstracting information to a higher level of understanding, the computer excels in its almost unlimited capacity for storing data. While the human being is prone to making minor mistakes in arithmetic and reading, the computer is always accurate. A slight diversion may be sufficient to disrupt our attention to the degree that we incorrectly add or subtract two numbers. However, if the error is large we are likely to notice that something is wrong further downstream due to our ability to apply conceptual checks and balances. The computer, on the other hand, cannot of its own accord distinguish between a minor mistake and a major error. Both are a malfunction of the entirely predictable behavior of its electronic components.

The differences between the human being and the computer are fundamental. All of the capabilities of the digital computer are derived from the simple building blocks of '0' and '1'. There is no degree of vagueness here, '0' and '1' are precise digital entities and very different from the massively parallel and largely unpredictable interactions of neurons and synapses that drive human behavior. It is not intuitively obvious how to create the high level representations of real world objects (e.g., ship, aircraft, dog, house, power, security, etc.) that appear to be a prerequisite for human reasoning and learning, in a digital computer. While these objects can be fairly easily represented in the computer as superficial visual images (in the case of physical objects such as aircraft and house) and data relationships (in the case of conceptual objects such as power and security) that in itself does not ensure that the computer has any understanding of their real world meaning. These representations are simply combinations of the basic digital building blocks that model, at best, the external shell rather than the internal kernel of the object.

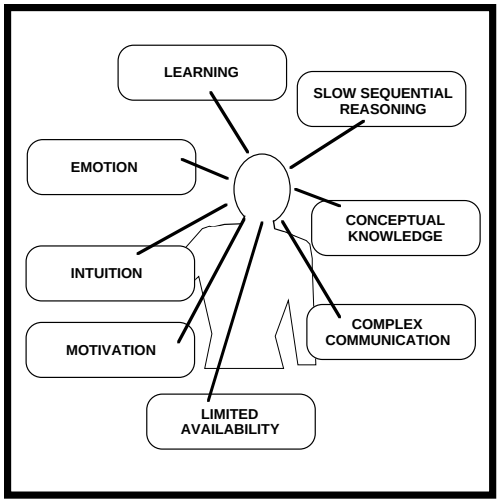


Fig.7: Human abilities and limitations

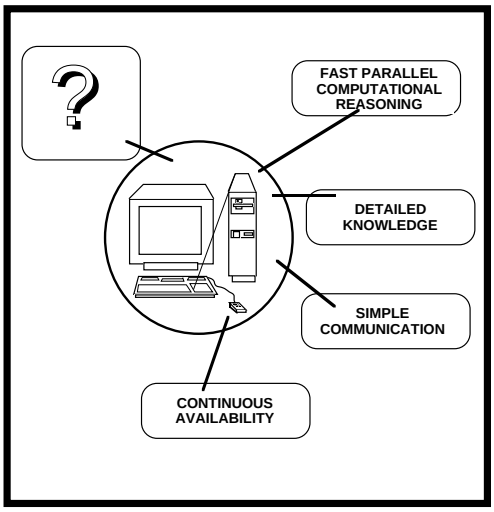


Fig.8: Computer abilities and limitations

Unfortunately, it is still not generally understood that this representational inadequacy is the single most limiting factor in virtually all existing decision-support systems. For example, current military command and control systems tend to overwhelm commanders with hundreds of detailed satellite pictures of battlefield conditions that are transmitted by computers as digital packages rather than groups of objects. As a result the interpretation, filtering and fusion of these images, areas in which computer-assistance would be highly desirable, become the burdensome task of the human decision maker.

More than 10 years ago when the CAD Research Center first embarked on the development of cooperative multi-agent systems we recognized the fundamental importance of representation, as a prerequisite for providing computer-based agents with reasoning capabilities. We discovered that while this problem was well known and had been the subject of considerable research in the artificial intelligence community, the results of this research work had generally remained the province of that close-knit community.

Early practical implementations of artificial intelligence systems were almost exclusively confined to stand-alone applications, such as expert systems (e.g., Prospector (Duda et al. 1977, Reboh 1981), MYCIN (Buchanan and Shortliffe 1984), and ASTA (Wilson et al. 1984)). Since these systems were not intended to interface with other applications the importance of representation continued to be largely ignored by the mainstream of software developers and users. Over the past decade the CAD Research Center has explored, adapted and implemented several high level representation techniques in its various decision-support applications for industry and government sponsors (Myers et al. 1993). While there is a need for a great deal more work in this area the state of technology today is, without question, capable of providing an internal representation level that can support meaningful reasoning assistance in large integrated decision-support systems.

Multi-Agent Collaborative Decision-Support Systems

Adaptation of 1st Wave software (Fig.2) to increasingly more complex real world problem situations has led to a hybrid of human and computer-based decision-support systems (Fig.9). Individual members of the human problem solving team utilize computer-based tools to assist them mostly with the computational and planning components of their tasks. However, this assistance is limited to the individual team member. While the computer can retrieve and send information from and to shared databases, it exercises these capabilities only on the request of its user. Collaboration within the problem team is largely restricted to the communications initiated by team members. The computer shares in these communications only to the extent that its user initiates queries to shared databases. The computer functions as a stand-alone tool that interacts with its user, but does not actively participate in the collaborative problem solving process.

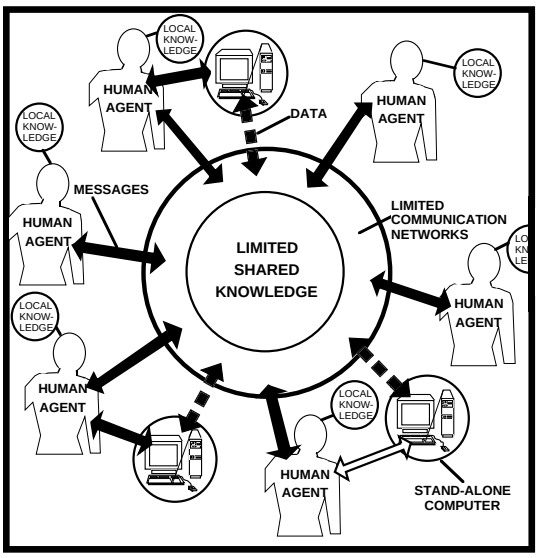


Fig.9: Limited computer assistance

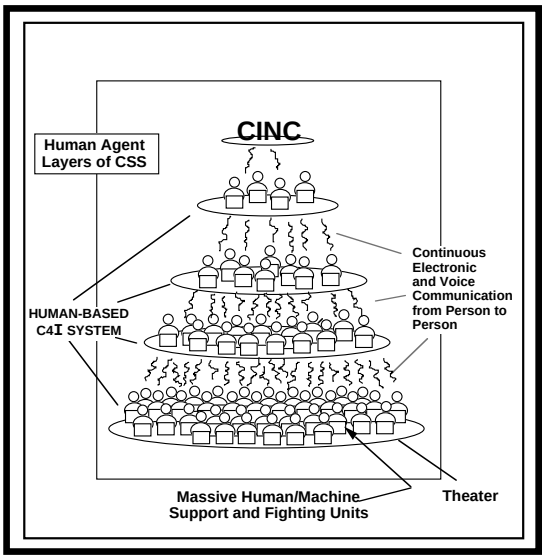


Fig.10: Hierarchical military C4I structure

In this hybrid decision-support environment, which is still representative even of the more critical transportation and military systems today, much of the collaboration is based on human to human voice communication. As a result, under severe stress conditions these systems are subject to serious communication bottlenecks that will disrupt and may even terminate the decision making process. In recent years examples of these conditions have occurred during environmental disasters, such as earthquakes in the USA, and military missions, such as Desert Storm in the Middle East. In the latter case, as shown in Fig.10, the combination of a hierarchical command and control structure with a 1st Wave software architecture produced a high potential for communication failure. A massive build-up of US and allied forces (i.e., more than 500,000 personnel) in the theater was supported by computer-based communication

facilities that reflected the chain of command through multiple levels from the commander in chief (CINC) down to the soldier in the battlefield. In this human-based C4I system environment continuous electronic and voice communication, essentially from person to person, quickly clogged the available communication channels.

During the late 1990s the limited computer-assistance capabilities (Fig.9) that are reflective of 1st Wave software will be increasingly replaced by integrated, multi-agent, cooperative systems. This signals the emergence of 2nd Wave software (Fig.11) in which the contributions of several decision-support components are coordinated through an inter-process communication facility. The components, commonly referred to as agents, may be separate processes or modules of one or more processes. They may be rule-based expert systems, procedural programs, neural networks, or even sensing devices. Increasingly, these agents will have the ability to explain their actions and proposals, as they interact spontaneously with each other either directly or through coordination facilities.

In the broadest sense an agent may be described as a computer-based program or module of a program that has communication capabilities to external entities and can perform some useful tasks in at least a semi-autonomous fashion. According to this definition agent software can range from simple, stand-alone, predetermined applications to the most intelligent, integrated, multi-agent decision-support system that advanced technology can produce today.

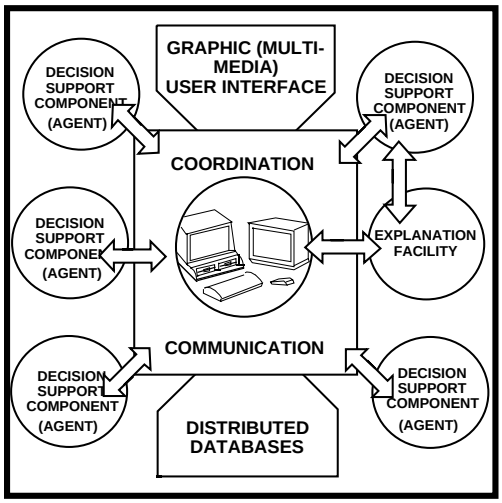


Fig.11: 2nd Wave computer applications

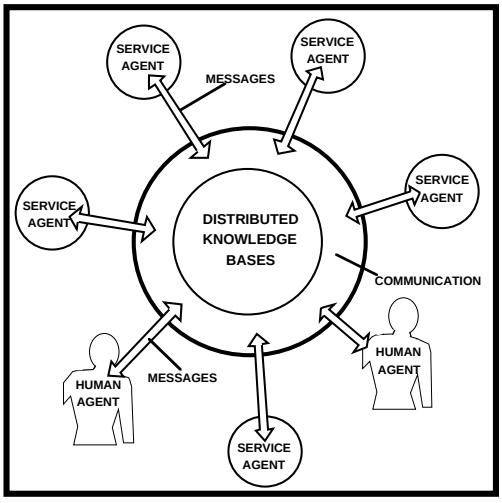


Fig.12: The service-agent architecture

As discussed previously, 2nd Wave software requires a high level internal representation of the real world objects and their relationships that are central to the problem situation. This is a prerequisite for the reasoning capabilities of the agents and also for the interaction of the user(s) with the system. The objective of 2nd Wave software is

not to automate the decision making activity, but to create an effective partnership between the human decision maker and the computer-based agents. In this partnership the human agent must be able to communicate with the computer-based agents in terms of the same real world objects that are used so effectively in all human reasoning endeavors. In their role as active collaborators the computer-based agents will have information needs that cannot be totally predetermined. Therefore, similar to the human agent, they will require the capability to dynamically generate database queries and initiate user interactions. At least some of the information sources accessed by the agents will be prototypical in nature (i.e., standard practices, case studies, and other typical knowledge pertaining to the problem situation) consistent with the notion of knowledge-based systems.

As discussed earlier, human and computer capabilities are complementary in many respects. Where we excel in the areas of abstraction, conceptualization, intuition and creativity, the performance of the computer cannot be described as being even adequate. However, when it comes to computational speed and accuracy, searching for and storing data, redundancy and parallelism, information persistence, and continuous availability, the computer outperforms us by far. It is therefore not surprising that current 2nd Wave software developments are increasingly focusing on collaborative systems in which users interact with computer-based expert agents (Fig.11). Typically, each agent is designed to be knowledgeable in a narrow domain, and represents the viewpoint of that domain in its collaborative endeavors. In this respect it provides services and can be categorized as a service-agent (Fig.12).

The service-agents are endowed with a communication facility that allows them to receive and send information. The manner in which they participate in the decision making activities depends on the nature of the application. They can be designed to respond to changes in the problem state spontaneously, through their ability to monitor information changes and respond opportunistically, or information may be passed to them in some chronological order based on time-stamped events or predefined priorities. They should be able to generate queries dynamically and access databases automatically whenever the need arises. In other words, service-agents should have the same data search initiation capabilities as the user and should not be dependent solely on the user for access to external information sources. In fact, the human users in such multi-agent systems may be categorized as very intelligent, multi-domain service agents. Examples of such service-agent systems can be found in the literature (Durfee 1988, Lesser 1995, Pohl et al. 1989, 1991 and 1997).

Within a networked environment the service-agents pertaining to a single multi-agent system (Fig.12) may be distributed over several computers, and even the coordination facilities (i.e., planning, negotiation, conflict detection, etc.) may be distributed over several nodes (Pohl et al. 1992). Alternatively, several single multi-agent systems can be connected. In this case each multi-agent system functions as an agent in a

higher level multi-agent system. Such systems are well suited to planning functions in which resources and viewpoints from several organizational entities must be coordinated. Typical application areas include military mission planning and facilities management. The user at each node should be able to plan in multiple worlds. For example, a private world in which shared information sources may be accessed but the deliberations of the user are not shared with other users, and a shared world which allows and encourages the continuous exchange of comments, plans and instructions. The capability normally exists for the user to maintain multiple views of each world to facilitate experimentation and the exploration of alternatives (Nadendla and Davis 1995). The service-agents resident in each system (i.e., at each node) should be able to differentiate between worlds and also between the views of any particular world. This normally requires a high degree of parallelism that must be supported by the system architecture.

So far we have discussed multi-agent systems involving two types of agents; namely, service-agents and human agents (i.e., users). Other agent types are certainly feasible. Of particular interest is the agentification of the information objects that are intrinsic to the nature of each application. These are the information objects that human decision makers reason about, and that constitute the building blocks of the real world representation of the problem situation.

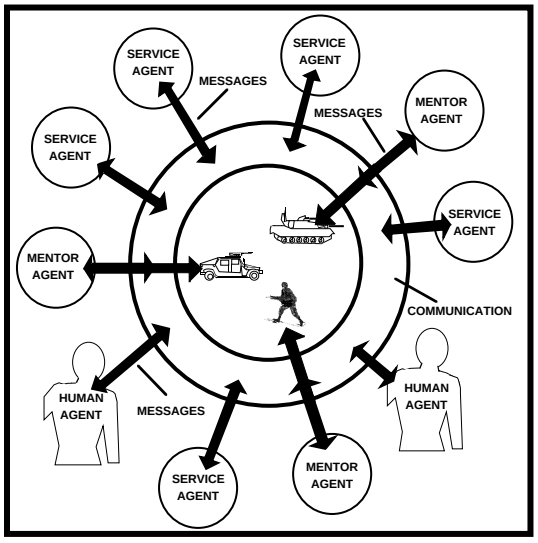


Fig.13: Object-Agent systems

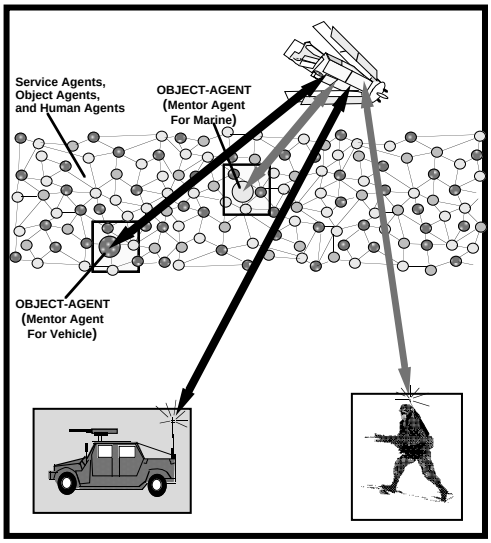


Fig.14: The object-agent as a mentor

The notion of object-agents brings several potential benefits. First, it increases the granularity of the active participants in the decision making environment. As agents with communication capabilities, objects such as armored vehicles (in military missions), aircraft (in air traffic control), or building spaces (in architectural design), can pursue their own needs and perform a great deal of local problem solving without

continuously impacting the communication and coordination facilities utilized by the higher level components of the decision-support system (Fig.13). Typically, an object-agent is a process (i.e., program) or component of a process that includes several adjuncts that provide the agent with communication capabilities, process management capabilities, information about its own nature, global objectives, and some focused problem solving tools.

Second, the ability of object-agents to request services through their communication facilities greatly increases the potential for concurrent activities. Multiple object-agents can request the same or different services simultaneously. If necessary, service-agents responding to multiple service requests can temporarily clone themselves so that the requests can be processed in parallel. Third, groups of object-agents can negotiate among themselves in the case of matters that do not directly affect other higher level components or as a means of developing alternatives for consideration by higher level components. Fourth, by virtue of their communication facilities object-agents are able to maintain their associations to other objects. In this respect they are the product of *decentralization* rather than *decomposition*. In other words, the concept of object-agents overcomes one of the most serious deficiencies of the rationalistic approach to problem solving; namely, the dilution and loss of relationships that occurs when a complex problem is decomposed into sub-problems. In fact, the relationships are greatly strengthened because they become active communication channels that can be dynamically created and terminated in response to the changing state of the problem situation.

The combination of object-agents and service-agents in the same decision-support system suggests a logical transition from 2nd Wave to 3rd Wave software in which even simple learning capabilities may eventually lead to emergent knowledge (Brooks 1990). Object-Agents may represent abstract concepts such as image and power, collective notions such as climate, virtual entities such as a building space during the design process (Pohl 1996), physical objects such as a M1A1 tank in the battlefield, or even human beings such as an individual soldier, squad or platoon. In the latter case a small communication device, embedded in a computer tag, is attached to the uniform of the soldier (Fig.14). This Radio Frequency Tag (RF-Tag) is capable of receiving and sending messages to an object-agent taking the role of a mentor within the computer-based command and control system. In this scenario the object-agent can serve many functions. It can provide several kinds of assistance to the soldier, such as medical advice, geographical position and terrain information, enemy location and strength, maneuver strategies, fire support alternatives, and so on. Conversely, the object-agent can use the soldier as part of a sensory array that continuously collects intelligence with and without the soldier's direct involvement.

Many of the service requests received by the object-agent will need to be passed onto service-agents, human agents, or other object-agents. This can be accomplished

through the appropriate use of both broadcasting and directed modes of communication. For example, a request for medical advice may initiate several actions by the mentor agent: a specific request for more detailed information to the soldier; the collection of bodily functions data from sensors embedded in the soldier's uniform, if the soldier has been wounded; a broadcast for evacuation assistance, if the wounds are serious; a request for specific self-help medical advice directed to a service-agent with medical expertise; a situation update to the Commander's mentor agent and/or the designated command and control service-agent; and so on. Even if the soldier is unable to personally communicate, the mentor agent is automatically alerted to the soldier's medical condition through sensors attached to his uniform or skin.

Conclusion

A collaborative agent-based command and control system, such as the ***Integrated Marine Multi-Agent Command and Control Systems (IMMACCS)*** that was successfully field-tested by the Marine Corps Warfighting Laboratory (MCWL) during the Urban Warrior Advanced Warfighting Experiment a few weeks ago (i.e., March 11 to 18, 1999, Monterey and Oakland, California), differs from the conventional *human-based* command and control system shown in Fig.10 in several significant respects (Porczak et al. 1999). First, the continuous and automatic monitoring of human/machine warfighting units by the various types of agents that operate spontaneously within the communication system potentially provides the warfighter with access to instantaneous advice and guidance. The agent to agent communication which facilitates this continuous access to information and intelligent analysis is not dependent on human to human interaction. In a conventional command and control system the communication channels are easily saturated by the continuous flow of human to human electronic and voice communications. Efforts to control this traffic inevitably require the imposition of communication restrictions that can easily prevent critical information from reaching the appropriate Commander or warfighter. In addition, as shown in Fig.10, the human to human interaction encourages a build-up of support personnel in and around the theater. This build-up is costly in terms of transportation and logistics, increases the danger of casualties, and places an additional burden on the already overloaded communication facilities.

Second, the multi-agent system architecture decentralizes both the collection and analysis of information. Individual human/machine warfighting units serve equally well as collectors and generators of information, as they do as recipients of information. In this way a dispersed force of warfighters can represent an important sensor array, with the ability to add value by converting data into information and knowledge close to the source. This decentralization of the data analysis process is particularly valuable in terms of distributing the communication traffic and validating the results of the analysis at the collection source.

Third, the seamless integration of planning, execution and training functions within the same command and control communication system allows the Commander and the individual warfighter to continuously and instantaneously switch from one mode of operation to another. In fact, the parallel nature of the system allows specific planning, execution and training tasks to be undertaken concurrently. For example, the Commander may wish to initiate a planning function through one set of agents while executing a specific operation in the theater, and at the same time simulate a particular *what if* scenario in anticipation of a possible future situation.

Recent studies by the US Marine Corps and the US Army have demonstrated the capabilities of relatively low cost computerized RF-Tags that are mounted on vehicular cargo. Object-Agents can be designed to communicate with tagged equipment not only for purposes of monitoring their location, but also in a service and low level decision making role.

For example, let us assume a tactical cargo loadout scenario in which a fuel truck, fitted with a RF-Tag has been loaded onto a ship. During the voyage the fuel truck starts to leak. While the volume of fuel leaked is fairly small, even this small amount constitutes a serious potential hazard on-board ship. Alerted of the situation through a simple feed-back mechanism the RF-Tag communicates to its companion object-agent, resident in the command and control system, both its location and the extent of the leakage. The object-agent analyses the situation, either through its own capabilities or by requesting supporting services from other agents, and automatically notifies appropriate command personnel, or other agents, or the ship directly. ***What is particularly noteworthy in this scenario is the fact that the command and control system was not only able to automatically detect the problem, but also analyze the situation and take action without the need for human intervention.***

In existing multi-agent system configurations which include only domain agents (i.e., service-agents), conflicts arise when agents either disagree among themselves or with a decision made by the user. For example, utilizing such a system for the load planning of a ship, the placement of a fuel truck in a particular ship compartment might provoke the latter type of conflict (CADRC 1994). If the stow-planner unknowingly places the truck in the immediate vicinity of another cargo item of a different hazardous material class, then the *hazard agent* will alert the user and explain the necessary segregation requirements. The stow-planner resolves the conflict by relocating or unloading one or both of the cargo items or, alternatively, overrules the service-agent. The fuel truck, as a passive object, is involved in the conflict resolution process only as an information source that is used by the service-agent in its deliberations. In other words, while the validation of the load planning decision is entirely dependent on the knowledge encapsulated in the object the latter is unable to actively participate in the determination of its own destiny.

There is another kind of conflict resolution scenario that becomes possible with the availability of object-agents. An object-agent may develop a solution to a sub-problem in its own domain that redirects the entire course of the overall solution plan. For example a squad, operating in dispersed mode in enemy territory and communicating with a mentor agent (Fig.14), performs its assigned enemy surveillance mission. It communicates through its object-agent certain enemy behavior that it believes could be turned to advantage if specific elements of the current overall operations plan were to be modified. However, such suggestions are rejected at operational levels below the Commander for reasons that appear to this squad to be based on erroneous intelligence. The squad judges the matter to be of a potentially serious nature and instructs its mentor agent to validate aspects of the squad's current understanding of the battlefield situation.

The object-agent commences a low level investigation by communicating with the mentor agents of several other squads and utilizing the services of domain agents (i.e., service-agents) where necessary. Soon an alarming picture emerges. It appears possible that the enemy has infiltrated one node of the command and control system and is entering erroneous information through this node. The effects of this gradually evolving deception could lead to disastrous consequences. The squad, realizing the potentially serious nature of the situation, progressively develops through the activities of its object-agent a more and more compelling case in support of its observations and suggestions.

Eventually, the overwhelming weight of evidence developed from the interactions of the squad with its object-agent and other agents in the command and control system attracts the attention of the Command Element. The Commander and his object-agent quickly undertake another analysis of the situation considering additional factors not considered in the squad's analysis. He verifies an almost certain localized penetration by the enemy of the command and control system and decides to utilize this knowledge by implementing a double-deception strategy.

This scenario demonstrates several significant capabilities of a multi-agent command and control system, like IMMACCS, incorporating object-agents. First, it is significant that the likely enemy penetration of the information system has been discovered at all. If the squad had been restricted to communicating its information as passive objects for processing by service-agents there would not have been any desire on the part of the command and control system to pursue the problem after the initial conflict resolution. Second, the squad's object-agent was able to undertake its investigation in a decentralized fashion without impacting higher level command and control activities until it was ready to present a strong case for reconsideration. However, it was able at any time to alert higher levels of the command structure as soon as the results of its investigation warranted such action.

Third, if the squad's projections had been rejected at all higher agent levels, the squad's object-agent could have appealed directly to the Commander or his object-agent. Under these circumstances the Commander would have several alternative courses of actions open: also reject the squad's suggestions; require one or more of the higher level agents (i.e., object-agents and service-agents) to explain their ruling; reset certain parameters that allow the higher level agents to reconsider their ruling; over-rule the higher level agents and accept the proposal; or, capture the current state of the battlefield situation as a recoverable view and use the squad's proposition as the basis for the exploration of alternative solution paths.

Apart from their immediate action capabilities, object-agents support the highly desirable goal of decentralization through localized decision making and communication. In this kind of distributed, cooperative environment it would be useful if messages themselves could be endowed with agent capabilities. At least certain types of messages would benefit greatly from action capabilities. For example, a message-agent sent by an object-agent or service-agent to find particular information could clone itself to seek the information concurrently in several potential sources. Once apparently relevant information has been found it could be synthesized to formulate a meaningful response to the originator of the query. Clearly, message-agents would add another level of granularity, decentralization and action capability within the distributed, collaborative decision-support system architecture.

References

Archea J. (1987); 'Puzzle-Making: What Architects Do When No One Is Looking'; in Kalay (ed.) Principles of Computer-Aided Design: Computability of Design'; Wiley, New York, New York.

Bohm D. (1983); 'Wholeness and the Implicate Order'; Ark Paperbacks, London, England.

Brooks R.A. (1990); "Elephants Don't Play Chess"; in Maes P. (ed.) Designing Autonomous Agents, MIT Press, Cambridge, Massachusetts (pp.3-15).

Buchanan B. and E. Shortliffe (1984); 'Uncertainty and Evidential Support'; in Buchanan and Shortliffe (eds.) Rule-Based Expert Systems, Addison-Wesley, Reading, Massachusetts (pp.209-232).

CADRC (1994); 'ICODES: Proof-of-Concept System: Final Report'; Contract #: N47408-93-7347, Naval Civil Engineering Laboratory (Port Hueneme, California), CAD Research Center, Cal Poly, San Luis Obispo, California.

Duda R., P. Hart, N. Nilsson, R. Reboh, J. Slocum and G. Sutherland (1977); 'Development of a Computer-Based Consultant for Mineral Exploration'; SRI Report, Stanford Research Institute, Menlo Park, California, October.

Durfee E. (1988); 'Coordination of Distributed Problem Solvers'; Kluwer Academic, Boston, Massachusetts.

Johnson-Laird P. (1993); 'Human and machine Thinking'; Erlbaum, Hillsdale, New Jersey.

Kuhn T. (1977); 'The Essential Tension: Selected Studies in Scientific Tradition and Change'; University of Chicago Press, Chicago, Illinois.

Lakoff G. and M. Johnson (1980); "Metaphors We Live By"; University of Chicago Press, Chicago, Illinois.

Lesser V. (ed.) (1995); 'Proc. First International Conference on Multi-Agent Systems'; ICMAS-95, AAAI Press/MIT Press, Cambridge, Massachusetts.

Mackinder M. and H. Marvin (1982); 'Design Decision Making in Architectural Practice'; Building Research Establishment, Department of Environment, HMSO, London, England, July.

Mallen G. and P. Goumain (1973); 'The Analysis of Architectural Design Activity in the Working Environment'; Report 108/4 DDR, Royal College of Art, London, England, November.

Minsky M. (1982); "Why People Think Computers Can't"; AI Magazine, Vol.3(4), Fall.

Myers L., J. Pohl, J. Cotton, J. Snyder, K. Pohl, S. Chien, S. Aly and T. Rodriguez (1993); 'Object Representation and the ICADS-Kernel Design'; CAD Research Center, Technical Report (CADRU-08-93), Cal Poly, San Luis Obispo, California.

Nadendla R. and A. Davis (1995); 'FEAT: Distributed Problem Solving in a Military Mission Planning Environment'; Master Thesis, Computer Science Department, Cal Poly, San Luis Obispo, California.

Pohl J., A. Chapman, L. Chirica, R. Howell and L. Myers (1988); 'Implementation Strategies for a Prototype ICADS Working Model'; Technical Report, CADRU-02-88, CAD Research Unit, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl J., L. Myers, A. Chapman and J. Cotton (1989); 'ICADS: Working Model Version 1'; Technical Report, CADRU-03-89, CAD Research Unit, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl J., L. Myers, A. Chapman, J. Snyder, H. Chauvet, J. Cotton, C. Johnson and D. Johnson (1991); 'ICADS Working Model Version 2 and Future Directions'; Technical Report, CADRU-05-91, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl J., J. La Porta, K. Pohl and J. Snyder (1992); 'AEDOT Prototype (1.1): An Implementation of the ICADS Model'; Technical Report, CADRU-07-92, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Pohl J. and L. Myers (1994); 'A Distributed Cooperative Model for Architectural Design'; in G. Carrara and Y. Kalay (eds.), Knowledge-based Computer-Aided Architectural Design, Elsevier, Amsterdam (pp.205-242).

Pohl J., L. Myers and A. Chapman (1994); 'Thoughts on the Evolution of Computer-Assisted Design'; Technical Report, CADRU-09-94, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California, September.

Pohl J. (1995); 'The Representation Problem in CAD Systems: Solution Approaches'; in Pohl J. (ed.) Advances in Cooperative Computer-Assisted Environmental Design Systems, focus symposium: 8th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 16-20.

Pohl J. (1996); 'Agents and their Role in Computer-Based Decision Support Systems'; in Pohl J. (ed.) Advances in Cooperative Environmental Design Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 14-18 (pp.41-54).

Pohl J., A. Chapman, K. Pohl, J. Primrose and A. Wozniak (1997); 'Decision-Support Systems: Notions, Prototype, and In-Use Applications'; Technical Report, CADRU-11-97, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, California.

Porczak M., K. Pohl, R. Leighton, A. Davis, H. Assal and L. Vempati (1999); 'IMMACCS: Urban Warrior Advanced Warfighting Experiment After Action Report'; CAD Research Center, Cal Poly, San Luis Obispo, California, April.

Reboh R. (1981); 'Knowledge Engineering Technologies and Tools in the Prospector Environment'; SRI Technical Note 243, Stanford Research Institute, Menlo Park, California, June.

Schön D. (1983); 'The Reflective Practitioner: How Professionals Think in Action'; Basic Books.

Schön D. (1988); 'Designing: Rules, Types and Worlds'; Design Studies, 9(3), July (pp.181-190).

Senge P. (1993); 'Transforming the Practice of Management'; Human Resource Development Quarterly, Jossey-Bass, San Francisco, California.

Thornton C. (1992); 'Techniques in Computational Learning'; Chapman and Hall, Computing Series, London, England.

Wilson G., A. Cremarty, T. Adams, M. Grinberg, C. Tollander and J. Cunningham (1984); 'AI Assists Analysts in Identifying Soviet Radar Systems'; Defense Systems Review, January (pp.23-26).

Distributed Intelligent Agents

Katia Sycara, Keith Decker, Anandeep Pannu, Mike Williamson, Dajun Zeng
The Robotics Institute — Carnegie Mellon University

Abstract

We are investigating techniques for developing distributed and adaptive collections of agents that coordinate to retrieve, filter and fuse information relevant to the user, task and situation, as well as anticipate a user's information needs. In our system of agents, in- formation gathering is seamlessly integrated with decision support. The task for which particular information is requested of the agents does not remain in the user's head but it is explicitly represented and supported through agent collaboration. In this paper we present the distributed system architecture, agent collaboration interactions, and a reusable set of software components for constructing agents. We call this reusable multi-agent computational infrastructure RETSINA (Reusable Task Structure-based Intelligent Network Agents). It has three types of agents. *Interface agents* interact with the user receiving user specifications and delivering results. They acquire, model, and utilize user preferences to guide system coordination in support of the user's tasks. *Task agents* help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents. *Information agents* provide intelligent access to a heterogeneous collection of information sources. We have implemented this system framework and are developing collaborating agents in diverse complex real world tasks, such as organizational decision making (the PLEIADES system), and financial portfolio management (the WARREN system).

1 Introduction

Effective use of the Internet by humans or decision support machine systems has been hampered by some dominant characteristics of the Infosphere. First, information available from the net is unorganized, multi-modal, and distributed on server sites all over the world. Second, the number and variety of data sources and services is dramatically increasing every day. Furthermore, the availability, type and reliability of information services are constantly changing. Third, information is ambiguous and possibly erroneous due to the dynamic nature of the information sources and potential information updating and maintenance problems. Therefore, information is becoming increasingly difficult for a person or machine system to collect, filter, evaluate, and use in problem solving. As a result, the problem of locating information sources, accessing, filtering, and integrating information in support of decision making, as well as coordinating information retrieval and problem solving efforts of information sources and decision-making systems has become a very critical task.

The notion of Intelligent Software Agents (e.g., [1, 19, 20, 25, 13, 22]) has been proposed to address this challenge. Although a precise definition of an intelligent agent is still forthcoming, the current working notion is that Intelligent Software Agents are programs that act on behalf of their human users in order to perform laborious information gathering tasks, such as locating and accessing information from various on-line information sources, resolving inconsistencies in the retrieved information, filtering away irrelevant or unwanted information, integrating information from heterogeneous information sources and adapting over time to their human users' information needs and the shape of the Infosphere. Most current agent-oriented approaches have focussed on what we call *interface agents*—a single agent with simple knowledge and problem solving capabilities whose main task is information filtering to alleviate the user's cognitive overload (e.g., [15, 16]). Another type of agent is the *Softbot* ([6]), a single agent with general knowledge that performs a wide range of user-delegated information-finding tasks. We believe that such centralized approaches have several limitations. A single general agent would need an enormous amount of knowledge to be able to deal effectively with user information requests that cover a variety of tasks. In addition, a centralized system constitutes a processing bottleneck and a "single point of failure". Finally, because of the complexity of the information finding and filtering task, and the large amount of information, the required processing would overwhelm a single agent.

Another proposed solution is to address the problem by using multi-agent systems to access, filter, evaluate, and integrate this information [23, 17]. Such multi-agent systems can compartmentalize specialized task knowledge, organize themselves to avoid processing bottlenecks, and can be built expressly to deal with dynamic changes in the agent and information-source landscape. In addition, multiple intelligent coordinating agents are ideally suited to the predominant characteristics of the Infosphere, such as the heterogeneity of the information sources, the diversity of information gathering and problem solving tasks that the gathered information supports, and the presence of multiple users with related information needs. We therefore believe that a distributed approach is superior, and possibly the only one that would work for information gathering and coherent information fusion.

The context of multi-agent systems widens the notion of intelligent agent in at least two general ways. First, an agent's "user" that imparts goals to it and delegates tasks might be not only a human but also another agent. Second, an agent must have been designed with explicit mechanisms for communicating and interacting with other agents. Our notion is that such multi agent systems may comprise *interface agents* tied closely to an individual human's goals, *task agents* involved in the processes associated with arbitrary problem-solving tasks, and *information agents* that are closely tied to a source or sources of data. An information agent is different from an interface agent in that an information agent is tied more closely to the data that it is providing, while an

interface agent closely interacts with the user. Typically, a single information agent will serve the information needs of many other agents (humans or intelligent software agents). An information agent is also quite different from a typical World Wide Web (WWW) service that provides data to multiple users. Besides the obvious interface differences, an information agent can reason about the way it will handle external requests and the order in which it will carry them out (WWW services are typically blindly concurrent). Moreover, information agents not only perform information gathering in response to queries but also can carry out long-term interactions that involve monitoring the Infosphere for particular conditions, as well as information updating.

In this paper, we report on our work on developing distributed collections of intelligent software agents that cooperate asynchronously to perform goal-directed information retrieval and information integration in support of performing a variety of decision making tasks [23, 2]. We have been developing RETSINA, an open society of reusable agents that self organize and cooperate in response to task requirements. In particular, we will focus on three crucial characteristics of the overall framework that differentiate our work from others:

- ours is a *multi-agent* system where the agents operate asynchronously and collaborate with each other and their users,
- the agents *actively* seek out information,
- the information gathering is *seamlessly integrated* with problem solving and decision support

We will present the overall architectural framework, our agent design commitments, and agent architecture to enable the above characteristics. We will draw examples from our work on Intelligent Agents in the domains of organizational decision making (the PLEIADES system), and financial portfolio management (the WARREN system).

The rest of the paper is organized as follows. Section 2 briefly lists some agent characteristics we consider desirable. Section 3 motivates the distributed architecture for intelligent information retrieval and problem solving, and presents an overview of the system architecture, the different types of agents in the proposed multi agent organization, and agent coordination mechanisms. Section 4 presents in detail the reusable agent architecture and discusses planning, control, and execution monitoring in agent operations. Description and examples from the application of RETSINA to everyday organizational decision making and financial portfolio management are given in Section 5. Section 6 presents concluding remarks.

2 Desirable Agent Characteristics

Many different definitions of intelligent agents have been proposed. In this section, we give a brief list of what we see as essential characteristics of intelligent agents.

- *taskable*. By "taskable" we mean agents that can take direction from humans or other agents.
- *network-centric*: by this we mean that agents should be distributed and self organizing. When situations warrant it, agent mobility may also be desirable.
- *semi-autonomous* rather than under direct human control all the time. For example, in an information gathering task, because of the large amount of potential requests for information, humans would be swamped, if they had to initiate every single information request. The amount of agent autonomy should be user controllable.
- *persistent*, i.e. capable of long periods of unattended operation.
- *trustworthy*: An agent should serve users' needs in a reliable way so that users will develop trust in its performance.
- *anticipatory*: An agent should anticipate user information needs through task, role and situational models as well as learning to serve as an intelligent cache, acquiring and holding information likely to be needed.
- *active*: An agent should initiate problem solving activities (e.g. monitor the infosphere for the occurrence of given patterns), anticipate user information needs and bring to the attention of users situation-appropriate information, deciding when to fuse information or present "raw" information.
- *collaborative* with humans and with other machine agents. Collaborative agent interactions allow them to increase their local knowledge, resolve conflicts and inconsistencies in information, current task and world models, thus improving their decision support capabilities.
- *able to deal with heterogeneity* of other agents and information resources.
- *adaptive* to changing user needs, and task environment.

3 Distributed Intelligent Agents in Information Processing and Problem Solving

In this section, we motivate and describe the distributed agent framework for intelligent information retrieval and problem solving, and then present the agent coordina-

tion mechanisms. The issues of how to engineer these agents are the topics of Section 4. RETSINA has been motivated by the following considerations:

- *Distributed information sources*: Information sources available on-line are inherently distributed. Furthermore, these sources typically are of different modalities. Therefore it is natural to adopt a distributed architecture consisting of many software agents specialized for different heterogeneous information sources.

- *Sharability*: Typically, user applications need to access several services or resources in an asynchronous manner in support of a variety of tasks. It would be wasteful to replicate agent information gathering or problem solving capabilities for each user and each application. It is desirable that the architecture support sharability of agent capabilities and retrieved information.

- *Complexity hiding*: Often information retrieval in support of a task involves quite complex coordination of many different agents. To avoid overloading the user with a confusing array of different agents and agent interfaces, it is necessary to develop an architecture that hides the underlying distributed information gathering and problem solving complexity from the user.

- *Modularity and Reuseability*: Although software agents will be operating on behalf of their individual patrons—human users, or other agents, pieces of agent code for a particular task can be copied from one agent to another and can be customized for new users to take into consideration particular users' preferences or idiosyncrasies. One of the basic ideas behind the distributed agent-based approach is that software agents will be kept simple for ease of maintenance, initialization and customization. Another facet of reuseability is that pre-existing information services, whose implementation, query language and communication channels are beyond the control of user applications, could be easily incorporated in problem-solving.

- *Flexibility*: Software agents can interact in new configurations "on-demand", depending on the information requirements of a particular decision making task.

- *Robustness*: When information and control is distributed, the system is able to degrade gracefully even when some of the agents are out of service temporarily. This feature of the system has significant practical implications because of the dynamic and unstable nature of on-line information services.

- *Quality of Information*: The existence of (usually partial) overlapping of available information items from multiple information sources offers the opportunity to ensure (and probably enhance) the correctness of data through cross-validation. Software agents providing the same piece of information can interact and negotiate to find the most accurate data.

- *Legacy Data*: Many information sources exist prior to the emergence of the Internet-based agent technology. New functionalities and access methods are necessary for them to become full- edged members of the new information era. Directly updating these systems, however, is a nontrivial task. A preferable way of updating is to construct agent wrappers around existing systems. These agent wrappers interface to the information sources and information consumers and provide a uniform way of accessing the data as well as offer additional functionalities such as monitoring for changes. This agent wrapper approach offers much flexibility and extensibility. Practically speaking, it is also easier to implement since the internal data structure and updating mechanism of the legacy information systems don't need to be modified.

The above considerations clearly motivate the development of systems of distributed software agents for information gathering and decision support in the Internet-based information environment. The critical question then is how to structure and organize these multiple software agents. Our major research goal is to construct reusable software components in such a way that building software agents for new tasks and applications and organizing them can be relatively easy. It seems difficult to engineer a general agent paradigm which can cover *in an efficient manner* a broad range of different tasks including interaction with the user, acquisition of user preferences, information retrieval, and task-specific decision making. For example, in building an agent that is primarily concerned with interacting with a human user, we need to emphasize acquisition, modeling and utilization of user information needs and preferences. On the other hand, in developing an agent that interacts with information sources, issues of acquiring user preferences are de-emphasized and, instead, issues of information source availability, efficiency of data access, data quality and information source reliability become critical. Therefore, reusable software components must efficiently address the critical issues associated with each of these three agent categories.

3.1 Agent Types

In the RETSINA framework, each user is associated with a set of agents which collaborate to support him/her in various tasks and act on the user's behalf. The agents are distributed and run across different machines. The agents have access to models of the user and of other agents as well as the task and information gathering needs associated with different steps of the task. Based on this knowledge, the agents decide how to decompose and delegate tasks, what information is needed at each decision point, and when to initiate collaborative searches with other agents to get, fuse and evaluate the information. In this way, the information gathering activities of the agents are automatically activated by models of the task and processing needs of the agents rather than wholly initiated by the user. The user can leave some of the information gathering decisions to the discretion of the agents. This saves user time and cognitive load and increases user productivity. The degree of agent autonomy is

user-controlled. As a user gains more confidence in the agents' capabilities, more latitude over decisions is given over to them. During search, the agents communicate with each other to request or provide information, find information sources, filter or integrate information, and negotiate to resolve conflicts in information and task models. The returned information is communicated to display agents for appropriate display to the user.

RETSINA has three types of agents (see Figure 1): *interface* agents, *task* agents and *information* agents. Interface agents interact with the user receiving user specifications and delivering results. They acquire, model and utilize user preferences to guide system coordination in support of the user's tasks. For example, an agent that filters electronic mail according to its user's preferences is an interface agent. The main functions of an interface agent include: (1) collecting relevant information from the user to initiate a task, (2) presenting relevant information including results and explanations, (3) asking the user for additional information during problem solving, and (4) asking for user confirmation, when necessary. From the user's viewpoint, having the user interact only through a relevant interface agent for a task hides the underlying distributed information gathering and problem solving complexity and frees the user from having to know of, access and interact with a potentially large number of task agents and information seeking agents in support of a task. For example, the task of hosting a visitor in a university (see Section 5.1), one of the tasks supported by our intelligent agents, involves more than 10 agents. However, the user interacts directly only with the visitor hoster interface agent.

Task agents support decision making by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents. Task agents have knowledge of the task domain, and which other task assistants or information assistants are relevant to performing various parts of the task. In addition, task assistants have strategies for resolving conflicts and fusing information retrieved by information agents. A task agent performs most of the autonomous problem solving. It exhibits a higher level of sophistication and complexity than either an interface or an information agent. A task agent (1) receives user delegated task specifications from an interface agent, (2) interprets the specifications and extracts problem solving goals, (3) forms plans to satisfy these goals, (4) identifies information seeking subgoals that are present in its plans, (5) decomposes the plans and coordinates with appropriate task agents or information agents for plan execution, monitoring and results composition. An example of a task agent from the financial portfolio management domain is one that makes recommendations to buy or sell stocks.

Information agents provide intelligent access to a heterogeneous collection of information sources depicted at the bottom of Figure 1. Information agents have models of the associated information resources, and strategies for source selection, information access, conflict resolution and information fusion. For example, an agent that

monitors stock prices of the New York Stock Exchange is an information agent. An information agent's activities are initiated either top down, by a user or a task agent through *queries*, or bottom up through *monitoring* information sources for the occurrence of particular information patterns (e.g., a particular stock price has exceeded a predefined threshold). Once the monitored-for condition has been observed, the information agent sends notification messages to agents that have registered interest in the occurrence of particular information patterns (See Section 5.2). For example, in the financial domain, a human or machine agent may be interested in being notified every time a given stock price has risen by 10%. Thus, information agents are active, in the sense that they actively monitor information sources and *proactively* deliver the information, rather than just waiting for and servicing one-shot information queries.

An information agent may receive in messages from other agents three important types of goals: (1) Answering a one-shot query about associated information sources, (2) Answering periodic queries that will be run repeatedly, and the results sent to the requester each time (e.g., "tell me the price of IBM every 30 minutes"), and (3) Monitoring an information source for a change in a piece of information (e.g., "tell me if the price of IBM drops below $80 within 15 minutes of the occurrence of that event").

A useful capability that can be added to all types of agents is *learning*. The agents can retain useful information from their interactions as training examples and utilize various machine learning techniques to adapt to new situations and improve their performance [18, 26, 16].

3.2 Agent Organization and Coordination

In RETSINA, agents are distributed across different machines and are directly activated based on the top-down elaboration of the current situation (as opposed to indirect activation via manager or matchmaker agents [12], or self-directed activation)[1]. These agent activations dynamically form an organizational structure "on-demand" that fits in with the task, the user's information needs and resulting decomposed information requests from related software agents. This task-based organization may change over time, but will also remain relatively static for extended periods. Notice that the agent organization will not change as a result of appearance or disappearance of information sources but the agent interactions could be affected by appearance (or disappearance) of agents that are capable of fulfilling task subgoals in new ways. Information that is important for decision-making (and thus might cause an eventual change in organizational structuring) is monitored at the lowest levels of the organization and passed upward when necessary. In this type of organization, task-specific agents continually interleave planning, scheduling, coordination, and the execution of domain-level problem-solving actions.

[1] Matchmaking is, however used for locating agents.

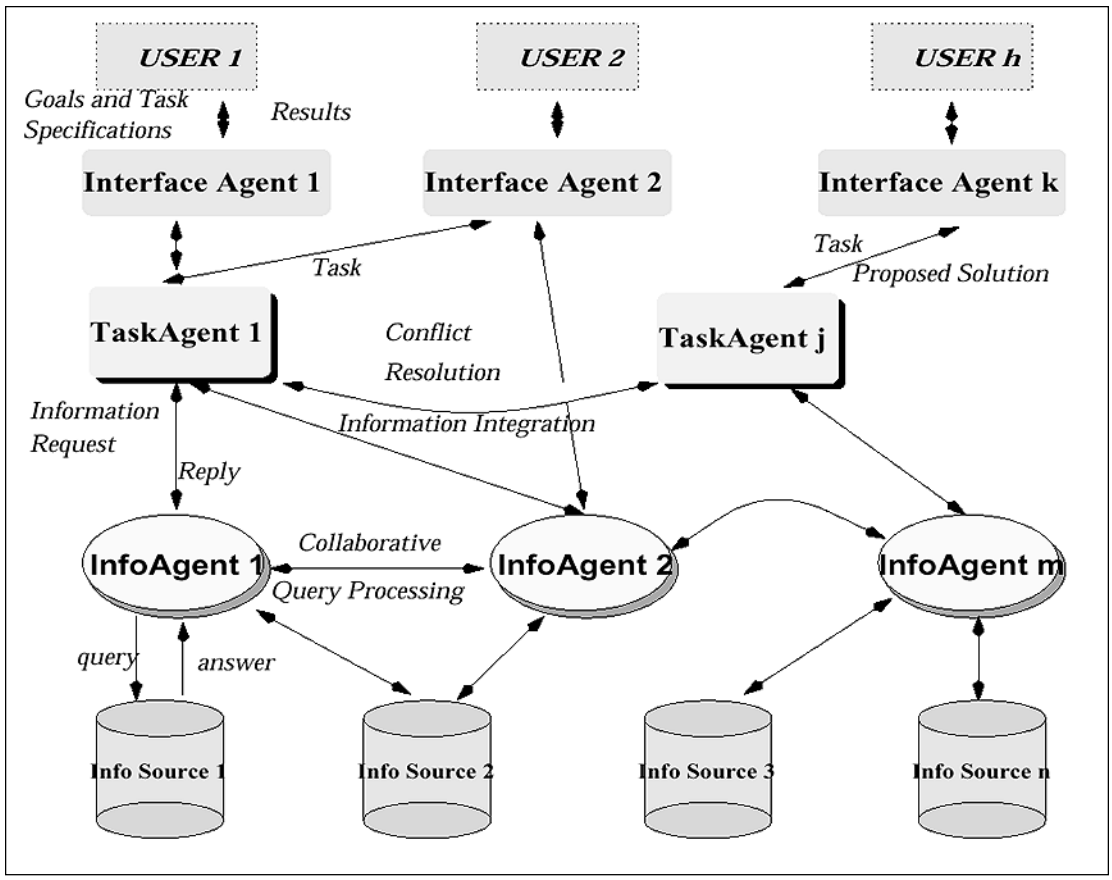


Figure 1: The RETSINA distributed agent organization

This system organization has the following characteristics:

- There is a finite number of task agents that each agent communicates with.
- The task agents are eventually responsible for resolving information conflicts and integrating information from heterogeneous information sources for their respective tasks.
- The task agents are responsible for activating relevant information agents and coordinating the information finding and filtering activity for their task.

In our organization, the majority of interactions of interface agents are with the human user, the most frequent interactions of information agents are with information sources, whereas task agents spend most of their processing interacting with other task agents and information agents. We briefly describe the distributed coordination processes. When a task-specific agent receives a task from an interface agent or from another task-specific agent, it decomposes the task based on the domain knowledge it has and then delegates the subtasks to other task-specific agents or directly to information-specific agents. The task-specific agent will take responsibility for collecting data, resolving conflicts, coordinating among the related agents and reporting

to whoever initiated the task. The agents who are responsible for assigned sub-tasks will either decompose these sub-tasks further, or perform data retrieval (or possibly other domain-specific local problem solving activities).

When information sources are partially replicated with varying degrees of reliability, cost and processing time, information agents must optimize information source selection. If the chosen information sources fail to provide a useful answer, the information agent should seek and try other sources to re-do the data query. Because of these complexities, we view information retrieval as a planning task itself[11]. The plans that task-specific agents have (see 4) include information gathering goals, which, in turn are satisfied through relevant plans for information retrieval. This type of intelligent agent differs from traditional AI systems since information-seeking during problem solving is an inherent part of the system. In effect, the planning and execution stages are interleaved since the retrieved information may change the planner's view of the outside world or alter the planner's inner belief system.

Information is filtered and fused incrementally by information or task agents as the goals and plans of the various tasks and subtasks dictate, before it is passed on to other agents. This incremental information fusion and conflict resolution increases efficiency and potential scalability (e.g., inconsistencies detected at the information-assistant level may be resolved at that level and not propagated to the task agent level) and robustness (e.g., whatever inconsistencies were not detected during information assistant interaction can be detected at the task-assistant level). A task agent can be said to be proactive in the sense that it actively generates information seeking goals and in turn activates other relevant agents.

Obviously, one of the major issues involved in multi-agent systems is the problem of interoperability and communication between the agents. In our framework, we use the KQML language [7] for inter-agent communication. In order to incorporate and utilize pre-existing software agents or information services that have been developed by others, we adopt the following strategy: If the agent is under our control, it will be built using KQML as a communication language. If not, we build a gateway agent that connects the legacy system to our agent organization and handles different communication channels, different data and query formats, etc.

In open world environments, agents in the system are not statically predefined but can dynamically enter and exit an organization. This necessitates mechanisms for agent locating. This is a challenging task, especially in environments that include large numbers of agents, and where information sources, communication links and/or agents may be appearing and disappearing. We have made initial progress in implementing *matchmaker agents* [12, 3] that act as yellow pages[9]. When an agent is instantiated, it advertises its capabilities to a matchmaker. An agent that is looking to find another that possesses a particular capability (e.g. can supply particular information, or achieve a problem solving goal) can query a matchmaker. The matchmaker returns appropriate

lists of agents that match the query description, or "null" if it does not currently know of any agent that has this capability. Architecturally, matchmakers are information agents. A matchmaker is an information agent who can find other agents rather than finding pieces of information. One nice property that falls out of this matchmaker design is that, if currently a matchmaker does not know of any agent that can provide a particular requested service, the requesting agent can place a monitoring request that directs the matchmaker to keep looking for an agent whose advertised capability matches the service specification of the requesting agent (the customer). When the matchmaker finds such an appropriate agent, it *notifies* the customer.

Matchmaking is advantageous since it allows a system to operate robustly in the face of agent appearance and disappearance, and intermittent communications (the customer can go back to the matchmaker, looking for a new supplier agent). Matchmaking is significant in another respect: it lays the foundation for evolutionary system design where agents with enhanced capabilities can be gracefully integrated into the system.

4 Agent Engineering: How to Structure an Agent?

In order to operate in rich, dynamic, multi-agent environments, software agents must be able to effectively utilize and coordinate their limited computational resources. As our point of departure in structuring an agent, we use the *Task Control Architecture* [21] and *TAEMS*[4], which we extend and specialize for real-time user interaction, information gathering, and decision support.

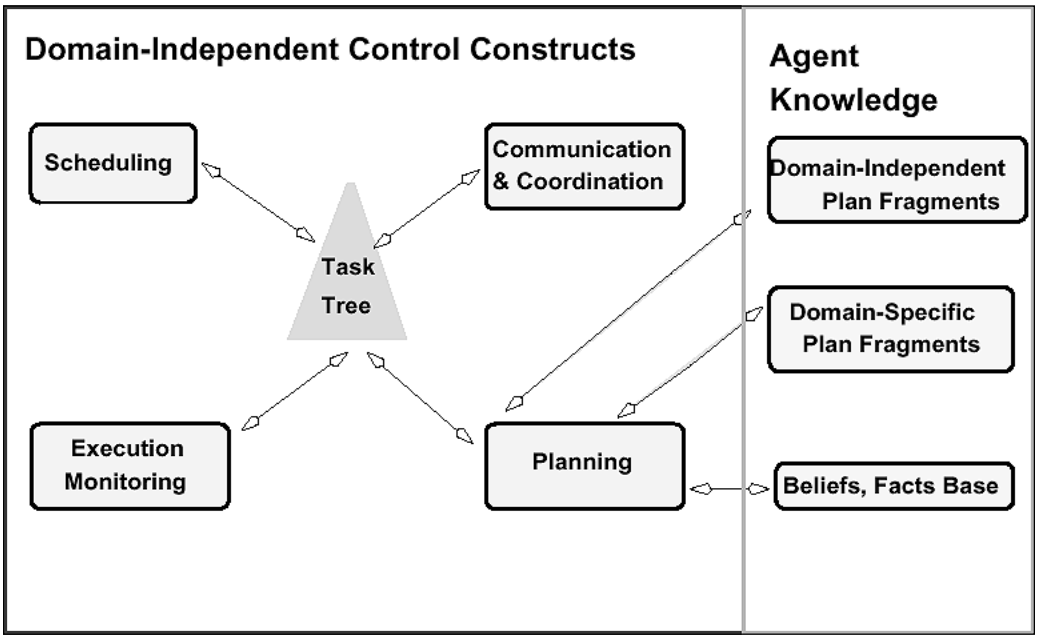


Figure 2: The agent architecture: a functional view

The *planning* module takes as input a set of goals and produces a plan that satisfies the goals. The agent planning process is based on a hierarchical task network (HTN) planning formalism. It takes as input the agent's current set of goals, the current set of task structures, and a library of task reduction schemas. A task reduction schema presents a way of carrying out a task by specifying a set of sub-tasks/actions and describing the information-flow relationships between them. That is, the reduction may specify that the result of one sub-task (e.g. deciding the name of an agent) be provided as an input to another sub-task (e.g. sending a message). Actions may require that certain information be provided before they can be executed, and may also produce information upon execution. For example, the act of sending a KQML messages requires the name of the recipient and the content of the message, while the act of deciding to whom to send some message would produce the name of an agent. An action is *enabled* when all the required inputs have been provided. (See [24] for a complete description of our task network representation.)

The *communication and coordination* module accepts and interprets messages from other agents in KQML. In addition, interface agents also accept and interpret e-mail messages. We have found that e-mail is a convenient medium of communicating with the user and/or other interface agents, for example agents that provide event notification services. Messages can contain request for services. These requests become goals of the recipient agent.

The *scheduling* module schedules each of the plan steps. The agent scheduling process in general takes as input the agent's current set of plan instances, in particular, the set of all executable actions, and decides which action, if any, is to be executed next. This action is then identified as a fixed intention until it is actually carried out (by the execution component). Whereas for task agents, scheduling can be very sophisticated, in our current implementation of information agents, we use a simple earliest-deadline-first schedule execution heuristic.

To operate in the uncertain, dynamic Infosphere, software agents must be reactive to change for robustness and efficiency considerations. Agent reactivity considerations are handled by the *execution monitoring* process. Execution monitoring takes as input the agent's next intended action and prepares, monitors, and completes its execution. The execution monitor prepares an action for execution by setting up a context (including the results of previous actions, etc.) for the action. It monitors the action by optionally providing the associated computation limited resources—for example, the action may be allowed only a certain amount of time and if the action does not complete before that time is up, the computation is interrupted and the action is marked as having failed.

When an action is marked as failed, the exception handling process takes over to replan from the current execution point to help the agent recover from the failure. For instance, when a certain external information source is out of service temporarily, the

agent who needs data from this information source shouldn't just wait passively until the service is back. Instead, the agent might want to try another information source or switch its attention to other tasks for a certain period of time before returning to the original task.

The agent has a *domain-independent library of plan fragments* (task structures) that are indexed by goals, as well as *domain-specific library of plan fragments* from which plan fragments can be retrieved and incrementally instantiated according to the current input parameters. The retrieved and instantiated plan fragments are used to form the agent's instantiated task tree that is incrementally executed.

The *belief and facts* data structures contain facts and other knowledge related to the agent's functionality. For example, the belief structures of an interface agent contain the user profile, and the belief structures of an information agent contain a local data base that holds relevant records of external information sources the agent is monitoring. Since an information agent does not have control of information sources on the Internet, it must retrieve and store locally any information that it must monitor. For example, suppose an information agent that provides the New York Stock Exchange data is monitoring the Security APL Quote Server web page to satisfy another agent's monitoring request, for example, "notify me when the price of IBM exceeds $80". The information agent must periodically retrieve the price of IBM from the Security APL web page, bring it to its local data base and perform the appropriate comparison. For information agents, the local data base is a major part of their reusable architecture. It is this local database that allows all information agents to present a consistent interface to other agents, and re-use behaviors, even in very different information environments [2].

An agent architecture may also contain components that are not reusable. For example, the architecture of information agents contains a small amount of site-specific external query interface code. The external query interface is responsible for actually retrieving data from some external source or sources. The external query interface is usually small and simple, thus minimizing the amount of site-specific code that must be written every time a new information agent is built.

Since task structure management, planning, action scheduling, execution monitoring, and exception handling are handled by the agent in a domain- independent way, all these control constructs are reusable. Therefore the development of a new agent is simplified and involves the following steps:

- Build the domain-specific plan library
- Develop the domain-specific knowledge-base
- Instantiate the reusable agent control architecture using the domain-specific plan library and knowledge-base

5 Application Domains

We have implemented distributed cooperating intelligent agents using the concepts, architecture, and reusable components of the RETSINA multi-agent infrastructure for everyday organizational decision making and for financial portfolio management.

5.1 Everyday Organizational Decision Making

In performing everyday routine tasks, people spend much time in finding, filtering, and processing information. Delegating some of the information processing to Intelligent Agents could increase human productivity and re- duce cognitive load. To this end, recent research has produced agents for e-mail filtering, [15], calendar management [5], and filtering news [13]. These tasks involve a single user interacting with a single software agent. There are tasks, however, which have more complex information requirements and possible interaction among many users. A distributed, multi-agent collection of Intelligent Agents is then appropriate and necessary. Within the context of our PLEIADES project, we have applied the distributed RETSINA framework to multi user tasks of increased complexity, such as

- distributed, collaborative meeting scheduling among multiple human attendees [14, 8]
- finding people information on the Internet
- hosting a visitor to Carnegie Mellon University [22]
- accessing and filtering information about conference announcements and requests for proposals (RFPs) from funding organizations and notifying Computer Science faculty of RFPs that suit their research interests [18].

5.1.1 An Extended Example: The Visitor Hosting Task

We will use the task of hosting a visitor to Carnegie Mellon University (CMU) as an illustrative example of system operation. Hosting a visitor involves arranging the visitor's schedule with faculty whose research interests match the interests that the visitor has expressed in his/her visit request. A different variation of the hosting visitor task has also been explored in [10].

For expository purposes, we refer to the collection of agents that are involved in the visitor hosting task as the Visitor Hosting system. The Visitor Hosting system takes as input a visit request, the tentative requested days for the meeting and the research interests of the visitor. Its final output is a detailed schedule for the visitor consisting of time, location and name of attendees. Attendees in these meetings are faculty members whose interests match the ones expressed in the visitor's request and who have been

automatically contacted by the agents in the Visitor Hosting system and have agreed to meet with the visitor at times convenient for them. The Visitor Hosting system has an interface agent, referred to as the Visitor Hoster, which interacts with the person hosting the visit. It also has the following task agents: (1) a Personnel Finder task agent, who finds detailed information about the visitor, and also finds detailed information about CMU faculty for better matching the visitor and the faculty he/she meets, (2) the visitor's Scheduling task agent and (3) various personal calendar management task agents that manage calendars of various faculty members. In addition, the Visitor Hosting system has a number of information agents that (1) retrieve information from a CMU data base that has faculty research interests (Interests agent), and (2) retrieve personnel and location information from various university data bases.

We present a detailed visitor hosting scenario to illustrate the interactions of the various agents in the Visitor Hosting task.

- The user inputs a visitor request to the Visitor Hoster agent.

  Suppose Marvin Minsky wants to visit CMU CS department. Minsky has requested that he would prefer to meet with CMU faculty interested in machine learning. The user inputs relevant information about Minsky, such as first name, last name, affiliated organization, date and duration of his visit, and his preference as to the interests of faculty he wants to meet with, to the Visitor Hoster agent.

- The Visitor Hoster agent extracts the visitor's areas of interest and visitor's name and organization.

- The Visitor Hoster agent passes to the Interests agent the visitor's areas of interest and asks the Interest agent to find faculty members whose interest areas match the request.

- The Visitor Hoster agent passes the name and organization of the visitor to the Personnel Finder agent and asks it to find additional information about the visitor.

- The Personnel Finder agent accesses Internet resources to find requested information about the visitor, such as visitor's title, rank, office address etc. The visitor information is used by faculty calendar software agents, such as CAP (see [16]), to decide level of interest of a faculty member to meet with the visitor.

- Meanwhile, the Interests agent queries the faculty interests data base and returns names of CMU faculty whose research matches the request. Using "machine learning" as the keyword to search through faculty interests database, the Interests agent finds a list of faculty whose interest areas match machine learning.

- The Visitor Hoster agent passes the returned faculty names to the Personnel Finder agent requesting more information on these faculty.

- The Personnel Finder agent submits queries to three personnel information sources (finger, CMU Who's-Who, CMU Room Database) to find more detailed information about the faculty member (e.g., rank, telephone number, e-mail address), resolves ambiguities in the returned information, and integrates results.

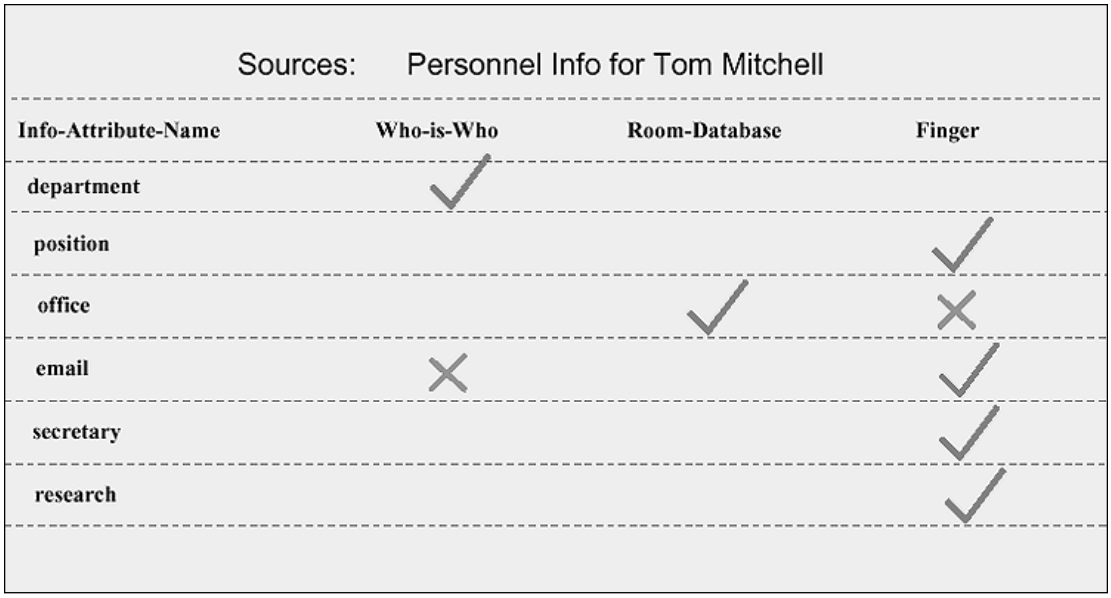


Figure 3: Information sources and returned items

Figure 3 shows in detail the information sources used for querying personnel information about Tom Mitchell, one of the Machine Learning faculty found by the Interests agent, and the information attributes returned by these sources. The columns correspond to different information sources. The rows are the attributes of personnel information that can be obtained from the sources. The checks and cross marks indicate which information sources return answers for which attributes. From this figure, we observe that for some

information attributes (e.g., office room number), more than one information source (Room Database and finger) offer answers, which may be potentially conflicting. To resolve this conflict, the Personnel Finder applies one of the rules kept in its domain-specific knowledge base saying that the office information based on Room Database is always more relevant and up-to-date than other sources. In this case, the value as to office room number returned by finger is overruled by the one returned by Room Database (indicated by the check mark). The cross mark in the "Office" row and "CS-FINGER" column means that although finger finds the office information, the retrieved value is overruled by another information source (Room Database).

- Based on the information returned by the Personnel Finder, the Visitor Hoster agent selects an initial set of faculty to be contacted. The user can participate in this selection process.

- The Visitor Hoster agent automatically composes messages to the calendar assistant agents of the selected faculty asking whether they are willing to meet with the visitor and at what time. For those faculty that do not have machine calendar agents, e-mail is automatically composed and sent.

- The Visitor Hoster agent collects responses and passes them to the visitor's Scheduling agent.

- The visitor's Scheduling agent composes the visitor's schedule through subsequent interaction and negotiation of scheduling conflicts with the attendees' calendar management agents[2]. The final calendar is shown in Figure 4.

The Visitor Hosting system has many capabilities. It automates information retrievals in terms of finding personnel information of potential appropriate meeting attendees. It accesses various on-line public databases and information resources at the disposal of the visit organizer. It integrates the results obtained from various databases, clarifies ambiguities (e.g., the same entity can be referred by different names in different partially replicated data bases) and resolves the conflicts which might arise from inconsistency between information resources. It creates and manages the visitor's schedule as well as the meeting locations for the various appointments with the faculty members (e.g., a faculty's office, a seminar room). It interacts with the user, getting user input, confirmation or dis-confirmation of suggestions, asking for user advice and advising the user of the state of the system and its progress.

[2] For details on the distributed meeting scheduling algorithm, see [14, 8].

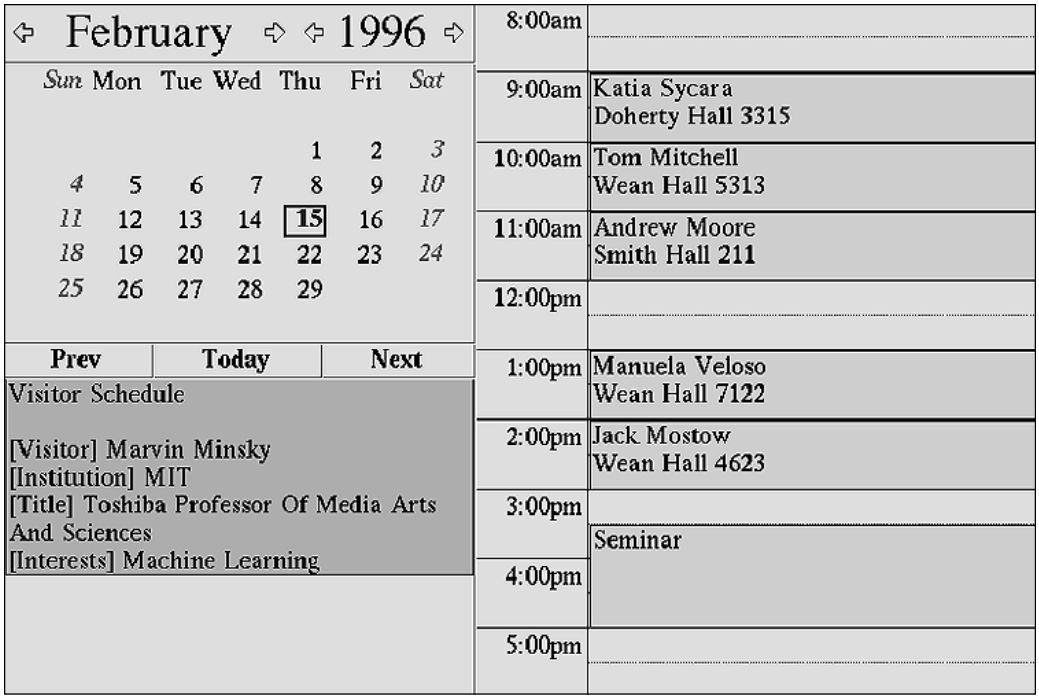


Figure 4: Final schedule of Minsky's visit

5.2 Financial Portfolio Management

The second domain of applying the RETSINA framework is financial portfolio management (the WARREN system[3]). In current practice, portfolio management is carried out by investment houses that employ teams of specialists for finding, filtering and evaluating relevant information. Based on their evaluation and on predictions of the economic future, the specialists make suggestions about buying or selling various financial instruments, such as stocks, bonds, mutual funds etc. Current practice as well as software engineering considerations motivate our multi-agent system architecture. A multi-agent system approach is natural for portfolio management because of the multiplicity of information sources and the different expertise that must be brought to bear to produce a good recommendation (e.g. a stock buy or sell decision).

The overall portfolio management task has several component tasks. These include eliciting (or learning) user profile information, collecting information on the user's initial portfolio position, and suggesting and monitoring a reallocation to meet the user's current profile and investment goals. As time passes, assets in the portfolio will no longer meet the user's needs (and these needs may also be changing as well). Our initial system focuses on the on-going portfolio monitoring process.

[3] The system is named after Warren Buffet, a famous American investor and author about investment strategies.

We briefly describe the main agents in the portfolio management task, shown in Figure 5:

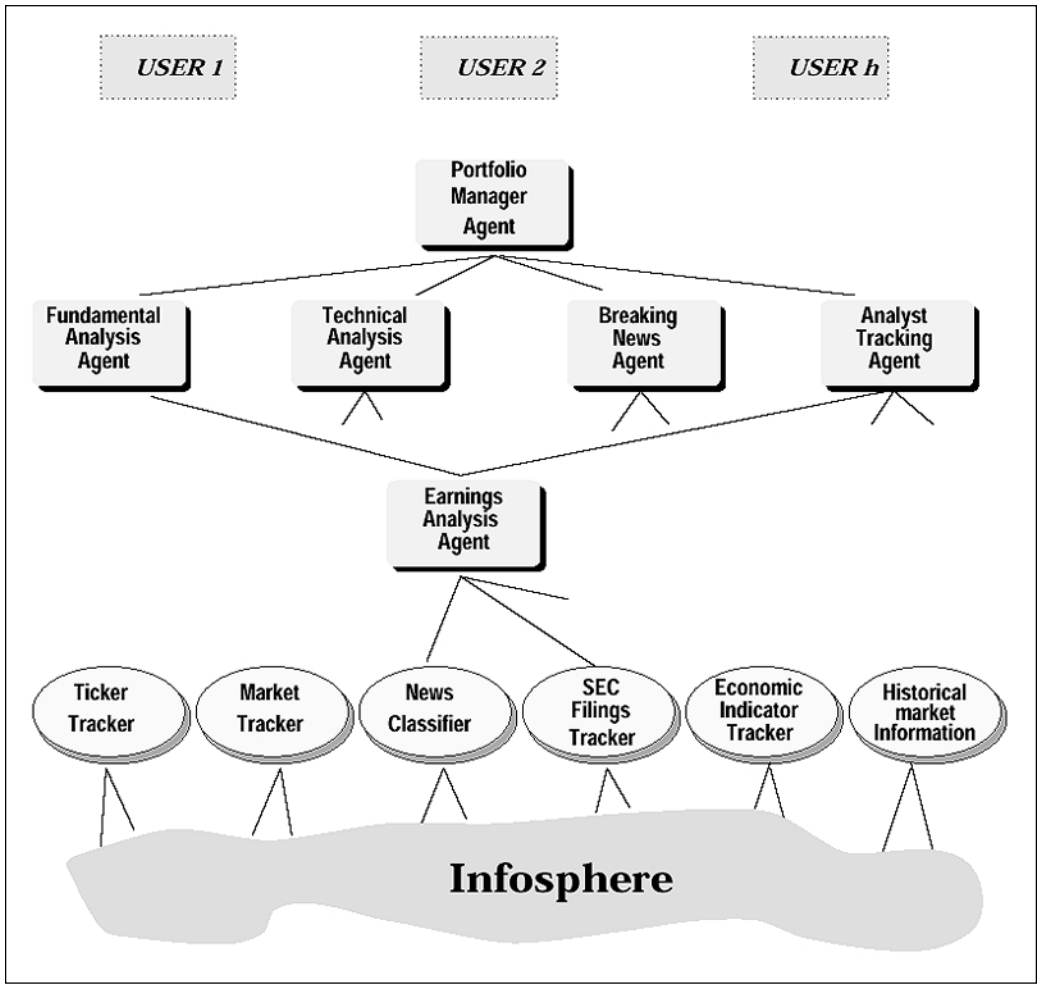


Figure 5: The portfolio management application

The portfolio manager agent is an interface agent that interacts graphically and textually with the user to acquire information about the user's profile and goals. *The fundamental analysis agent* is a task assistant that acquires and interprets information about a stock from the viewpoint of a stock's (fundamental) "value". Calculating fundamental value takes into consideration information such as a company's finances, forecasts of sales, earnings, expansion plans etc. The *Technical Analysis agent* uses numerical techniques such as moving averages, curve fitting, complex stochastic models, neural nets etc., to try to predict the near future in the stock market. The *Breaking News agent* tracks and filters news stories and decides if they are so important that the user needs to know about them immediately, in that the stock price may be immediately affected. The *Analyst Tracking agent* tries to gather intelligence about what human

analysts are thinking about a company. These agents gather information through information requests to information agents. The information agents that we have currently implemented are the *Stock Tracker* agents that monitors stock reporting Internet sources, such as the Security APL, the *News Tracking* agents that track and filter Usenet relevant financial news articles (including CMU's Clarinet and Dow Jones news feeds), and the SEC (Securities and Exchange Commission) fillings of companies financial information tracker agent that monitors the EDGAR database. The information retrieved by these information agents is passed to the display agents which display in an integrated fashion the retrieved information to the user.

Figure 6 shows an example WARREN portfolio. Currently, a user may interact with his or her own portfolio display (interface) agent via HTML forms and a web browser.[4] The portfolio display consists of a summary of the user's portfolio, including which issues are owned, and for each issue the total number of shares owned, the current price, the date of the last news article, and the current value. Below the portfolio table, the current value of the entire portfolio is displayed along with the portfolio's net gain in equity (current values compared to purchase values minus commissions). The interface also allows the user to buy and sell stocks (Trade) and to request the preparation of a Financial Data Summary (Fetch FDS), which uses historical price, earnings, and revenue information from the SEC's EDGAR database to do a simple fundamental analysis of the stock.

The other display available to the user (by clicking on a stock's current value) is a price/news graph that dynamically integrates intra-day trading prices and news stories about a stock. Figure 7 shows an example for Netscape Communications (NSCP) during the period of roughly December 5 to December 23. Prices are plotted at mostly 1 hour (sometimes 15 minute) intervals, and connected during the trading day (there's no trading at night or during the weekends). The numbers on the graphs correspond to the news articles whose subjects are listed below the graph. The articles are numbered from earliest to latest (left to right on the graph). Each article number is positioned at the time the news story appeared, and vertically at the approximate price of the stock at that time. The article subjects are hyperlinked to the actual news stories themselves.

The example graph covers a time period just after the $30 price rise in NSCP triggered by the joint Sun and Netscape announcement of JavaScript (2). However, the new record high caused some profit-taking, and then the Dec 7 news hits that Smith Barney had begun coverage of Netscape and recommended SELL (4,5), dropping the stock for the rest of the day. Although our University access is to delayed price and news sources, such information from realtime data feeds is the bread and butter of many types of institutional investment decision-making.

[4] We are currently constructing a more interactive Java interface.

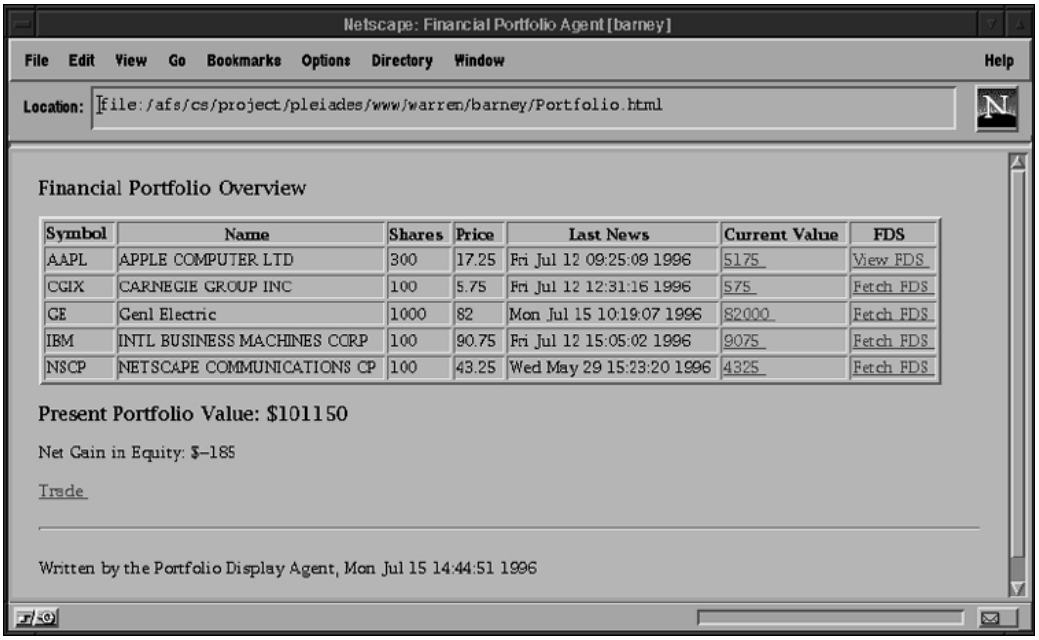


Figure 6: WARREN's Netscape interface.

6 Conclusions

In this paper, we have described our implemented, distributed agent framework, RETSINA, for structuring and organizing distributed collections of intelligent software agents in a reusable way. We presented the various agent types that we believe are necessary for supporting and seamlessly integrating information gathering from distributed internet-based information sources and decision support, including (1) *Interface agents* which interact with the user by receiving user specifications and delivering results, (2) *Task agents* which help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents, and (3) *Information agents* which provide intelligent access to a heterogeneous collection of information sources. We have also described and illustrated our implemented, distributed system of such collaborating agents. We believe that such flexible distributed architectures, consisting of reusable agent components, will be able to answer many of the challenges that face users as a result of the availability of the vast, new, net-based information environment. These challenges include locating, accessing, filtering and integrating information from disparate information sources, monitoring the Infosphere and notifying the user or an appropriate agent about events of particular interest in performing the user-designated tasks, and incorporating retrieved information into decision support tasks.

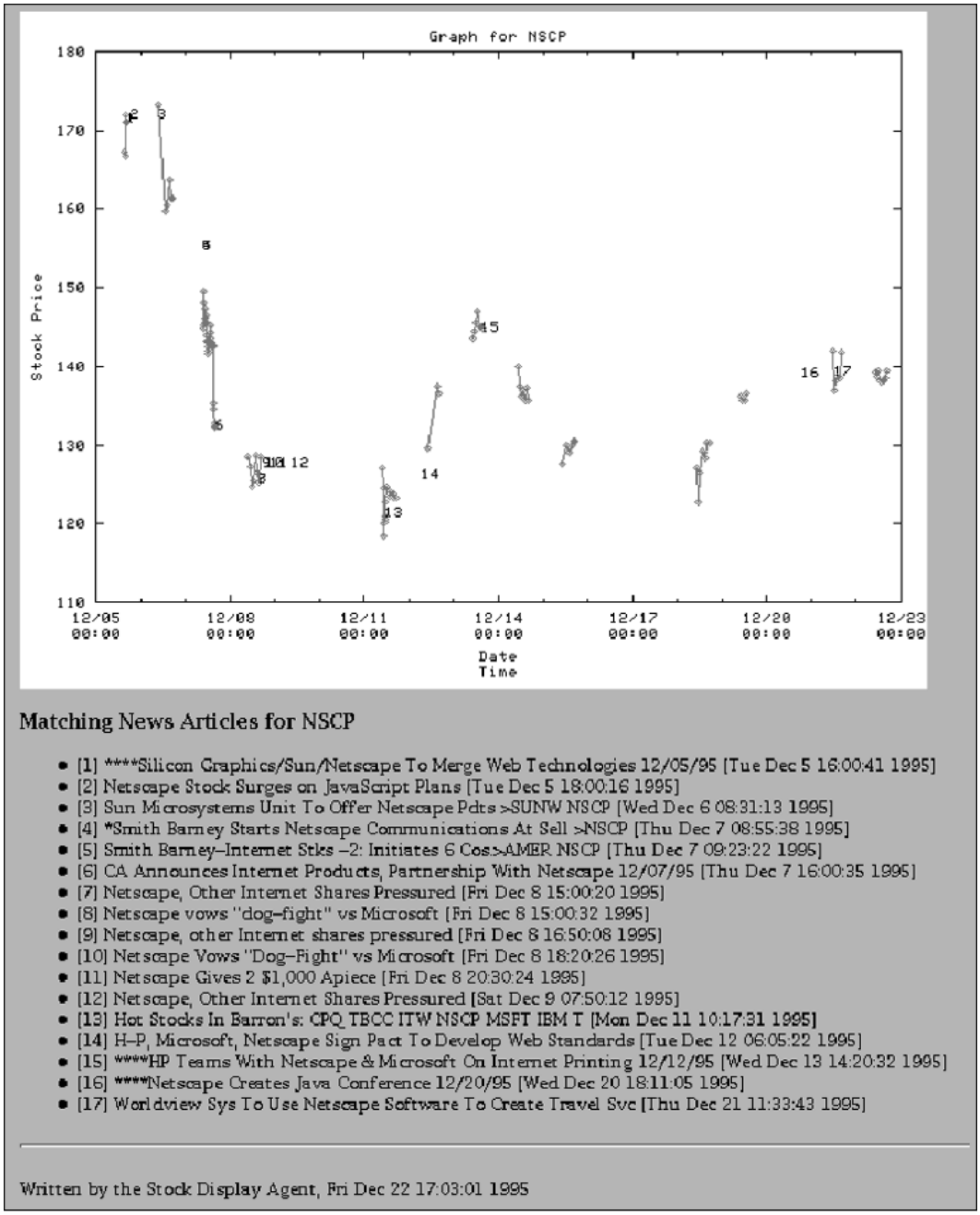


Figure 7: A Price/News graph constructed by the WARREN system for Netscape Communications (ticker symbol NSCP).

7 Acknowledgements

The current research has been sponsored in part by ARPA Grant #F33615-93-1-1330, by ONR Grant #N00014-95-1-1092, and by NSF Grant #IRI-9508191. We want to thank Tom Mitchell, Dana Freitag, Sean Slittery, and David Zabowski for insightful discussions.

References

[1] P. R. Cohen and H. J. Levesque. Intention=choice + commitment. In *Proceedings of AAAI-87*, pages 410–415, Seattle, WA., 1987. AAAI.

[2] K. Decker, K. Sycara, and M. Williamson. Modeling information agents: Advertisements, organizational roles, and dynamic behavior. In *Proceedings of the AAAI-96 Workshop on Agent Modeling*, Portland, Oregon, August 1996. AAAI.

[3] K. Decker, M. Williamson, and K. Sycara. Matchmaking and brokering. In *Proceedings of the Second International Conference in Multi-Agent Systems* (ICMAS'96), Kyoto, Japan, December 1996.

[4] Keith Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, University of Massachusetts, 1995.

[5] Lisa Dent, Jesus Boticario, John McDermott, Tom Mitchell, and David Zabowski. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI, 1992.

[6] Oren Etzioni and Daniel Weld. A softbot-based interface to the internet. *Communications of the ACM*, 37(7), July 1994.

[7] Tim Finin, Rich Fritzson, and Don McKay. A language and protocol to support intelligent agent interoperability. In *Proceedings of the CE and CALS Washington 92 Conference*, June 1992.

[8] Leonardo Garrido and Katia Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In *Proceedings of the Second International Conference in Multi-Agent Systems (ICMAS'96)*, Kyoto, Japan, December 1996.

[9] M. R. Genesereth and S. P. Katchpel. Software agents. *Communications of the ACM*, 37(7):48–53,147, 1994.

[10] Henry A. Kautz, Bart Selman, and Michael Coen. Bottom-up design of software agents. *Communications of the ACM*, 37(7), July 1994.

[11] Craig K. Knoblock. Integrating planning and execution for information gathering. In Craig Knoblock and Alon Levy, editors, *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, Stanford, CA, March 1995. AAAI.

[12] D. Kuokka and L. Harada. On using KQML for matchmaking. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 239–245, San Francisco, June 1995. AAAI Press.

[13] Kan Lang. Newsweeder: Learning to filter netnews. In *Proceedings of Machine Learning Conference*, 1995.

[14] JyiShane Liu and Katia Sycara. Distributed meeting scheduling. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, Georgia, August 13-16 1994.

[15] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7), July 1994.

[16] Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7), July 1994.

[17] Tim Oates, M. V. Nagendra Prasad, and Victor R. Lesser. Cooperative information gathering: A distributed problem solving approach. Technical Report 94-66, Department of Computer Science, University of Massachusetts, September 1994.

[18] Anandeep Pannu and Katia Sycara. Learning text filtering preferences. In *1996 AAAI Symposium on Machine Learning and Information Access*, 1996.

[19] Anand S. Rao and Michael P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of IJCAI-93*, pages 318–324, Chambery, France, 28 August - 3 September 1993. IJCAI.

[20] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[21] Reid Simmons. Structured control for autonomous robots. *IEEE Journal of Robotics and Automation*, 1994.

[22] Katia Sycara and Dajun Zeng. Towards an intelligent electronic secretary. In *Proceedings of the CIKM-94 (International Conference on Information and Knowledge Management) Workshop on Intelligent Information Agents*, National Institute of Standards and Technology, Gaithersburg, Maryland, December 1994.

[23] Katia Sycara and Dajun Zeng. Coordination of multiple intelligent software agents. *International Journal of Cooperative Information Systems*, To Appear, 1996.

[24] M. Williamson, K. Decker, and K. Sycara. Unified information and control flow. In *Proceedings of the AAAI-96 Workshop on Theories of Action, Planning and Control: Bridging the Gap*, Portland, Oregon, August 1996. AAAI.

[25] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[26] Dajun Zeng and Katia Sycara. Bayesian learning in negotiation. In *1996 AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, 1996.

*Anticipation, Delegation, and Demonstration: Why Talking to Agents is Hard**

Katia Sycara and Michael Lewis
The Robotics Institute — Carnegie Mellon University
School of Information Sciences — University of Pittsburgh

Abstract

Interacting with a computer requires adopting a metaphor to guide our actions and expectations. When we choose to build and use agents we are committing to interact with a domain indirectly. Whether this is a good choice or not will depend on the ease and accuracy with which we can instruct our agents. Current approaches range from specialized agents programmed to perform specific tasks, to learning programs which get on-the-job training looking over a user's shoulder. In the expert case very little communication is needed because the agent already knows what it is going to do. In the novice case the raison d'etre of agent learning is to relieve the user of the tedium of instructing it. The vast middle ground of tasks of moderate complexity too infrequent for targeted implementations or empirical learning goes largely untouched. Three current projects at our laboratory address aspects of this problem. The first, a series of experiments in which subjects are aided by fallible agents examines the role of trust in providing a context for communication. The second project compares the effectiveness of communication to adapt an agent plan with human planning which is critiqued by an equally informed agent. The third project uses a variety of learning and interaction techniques to help a user communicate the information needed to access and extract information on a subsequent autonomous visit.

1. Introduction

Interacting with a computer requires adopting some metaphor to guide our actions and expectations. Most human-computer interfaces can be classified according to two dominant metaphors: *agent* or *environment*. Interactions based on an agent metaphor treat the computer as an intermediary which responds to user requests. In the environment metaphor a model of the task domain is presented for the user to interact with directly.

The power of the environment approach which provides advertisement and unique identification and selectability of available objects and actions is reflected in the ascendance of graphical user interfaces (GUI's). The value of the agent metaphor to interaction only becomes apparent when objects are not present or fully visualizable and actions are repetitive, delayed, or poorly specified. The distinctions between agent

* A version of this paper will be presented at CIA-99 (Cooperative Information Agents) workshop

and environment based HCI are very similar to those between manual and automated action in the physical world. It is much simpler for us to drive a car or set a table than to instruct a robot to do so, yet we would rather adjust a thermostat or program a milling machine than repeatedly performing these actions by hand. While the computer offers the ultimate in flexible automation, instructing it do what we wish may be arbitrarily hard for humans as demonstrated by the difficulty experienced in using traditional programming and scripting languages. The growing popularity of "agent-based" interaction reflects the emergence of an increasingly powerful and complex computing environment bringing with it desires to perform flexible tasks involving multiple or unknown objects by users who do not wish or may not have the ability to program.

The greatest impediment to assisting human users lies in communicating their intent and making results intelligible to them.

By this we mean that today in almost all cases the limiting factor in HCI is not computing cycles or connectivity to information sources or characteristics of peripherals (the machine side) but in the user's ability and/or willingness to communicate these desires and sift, organize, and represent the machine's response to satisfy them (the human side). So, for example, although I have a Perl interpreter for which I could in principle write a script which would visit a list of web sites, extract particular information from each, perform comparisons, and return the result to my Inbox, I will almost certainly not do so. The effort of prescribing how to navigate, how to parse HTML at each of the sites, and how to analyze and return the results is infinitely more difficult than pushing buttons and following links on my browser myself. It would probably remain more difficult even if I had to repeat the search ten or fifteen times. Even when the time to search repeatedly equaled the time to program, I would still prefer the manual search because of the lower cognitive effort. As this example suggests, scripting languages may fit our definition as maximally powerful instructable "agents", yet they fail miserably in satisfying Negreponte's [6] desire for an implacable butler or mine for a no hassle autonomous web searcher. The problem is a human variant of Turing equivalency. Scripting languages or for that matter assembly code may meet the letter of a definition of agents but the spirit clearly lies in the ease with which our desires can be communicated.

1.1 A Cybernetic Model

Don Norman (1986) characterizes human-computer interaction as the problem of bridging twin gulfs of execution and evaluation. The execution side of the cycle involves translating a goal into a sequence of actions for achieving that goal. The evaluation side involves using feedback from the domain to compare the result of the action to the goal. The model is cybernetic rather than logical in that attention to parts of the environment and processing of these inputs are determined by prior actions and subsequent actions are in turn functions of previous feedback. A crucial

feature of this model is that tampering with either side of the loop can lead to detrimental or unanticipated results. If the execution side is automated the human may fail to observe effects of actions and be unable to correct errors or modulate ongoing behavior. If the evaluation side is automated the human may be unable to track the effect of actions and adjust to their results. Norman proposes seven stages of action in this model to link the user's goals to the world The stages of execution are: forming an intention to act, translating this intention into a planned sequence of actions and executing that sequence. The stages of evaluation are perceiving the state of the world, interpreting this perception in light of prior action and evaluating that change with respect to initial goal. The gulfs refer to the interface/metaphor which separates the user's goals from the application domain in which they must be realized.

While some agent-based systems may require no more human interaction than a conventional GUI, they should support the same cycle of action, feedback, and interpretation. This will in general be more difficult because the greater flexibility and autonomy which made a task suitable for an agent also make monitoring and evaluating more difficult for the human. Except in cases where task performance is completely correct and deterministic (as in the case of a command to open a file or delete a document) uncertainties may need to be addressed even in the simplest interactions.

Just as our networked computing infrastructure has given rise to multi-agent systems and cooperative computing paradigms, it has become a medium for human coordination and cooperation. The role of agents in this environment for facilitating human-human interactions is a second crucial research issue.

1.2 Desiderata for Human Agent Interaction

Graphical user interfaces have succeeded by providing advertisement, unique identification and selectability of available objects and actions to their users. To be a viable alternative, intelligent agents must also convey these types of information. More precisely they must:

1) Advertise their availability

   users must be made aware of the agent's existence and how to access it

2) Advertise their service domains

   users must be made aware of the types of services an agent can perform, or arrange to have performed (multi-agent systems)

3) Advertise their capabilities

   users must be made aware of the precise nature of services actually available

4) Advertise their "instruction language"

   users must be made aware of how to specify parameters and objects to "customize" the services they request

5) Advertise opportunities for monitoring

   users must be made aware of how they may monitor task performance

Over the past three years we have conducted a series of experiments aimed at identifying strategies for improving human-agent performance in the presence of errors, aiding in human-human cooperation, choosing between human/machine initiative, and hybrid forms of instruction combining programming by demonstration, learning, and command. In conducting these experiments we have attempted to: identify issues likely to be important for proposed uses of agents such as aggregation, interpretation, and presentation of information and test them at simplified tasks which allow us full experimental control..

2. The Tandem Simulation

Two of our experiments used a low fidelity simulation (TANDEM) of a target identification task, jointly developed at the Naval Air Warfare Center-Training Systems Division and the University of Central Florida and modified for these experiments. The TANDEM simulation was developed under the TADMUS (tactical decision making under stress) program of the US Office of Naval Research and simulates cognitive characteristics of tasks performed in the command information center (CIC) of an Aegis missile cruiser.

The cognitive aspects of the Aegis command and control tasks which are captured include time stress, memory loading, data aggregation for decision making and the need to rely on and cooperate with other team members (team mode) to successfully perform the task. The more highly skilled tasks of the individual team members involving extracting and interpreting information from radar, sonar, and intelligence displays is not modeled in the simulation. Instead of interpreting displayed signals to acquire diagnostic information about targets, TANDEM participants access this information manually from menus. In accessing new information, old information is cleared from the display creating the memory load of simultaneously maintaining up to 5 parameter values and their interpretation.

In the TANDEM task subjects must identify and take action on a large number of targets (high workload) and are awarded points for correctly identifying the targets (type, intent, and threat). and taking the correct action (clear or shoot). A maximum of 100 points is awarded per target for correct identification and correct action. Users

"hook" a target on their screen by left-clicking on the target or selecting "hook" from a menu and specifying a target's unique contact number. Only after a target is hooked can they access information relative to that target. In team configuration TANDEM consists of three networked pc's each providing access through menus to five parameters relative to a "hooked" target. Their tasks involve identifying the type of contact (submarine, surface, or aircraft), its classification (military or civilian), and its intent (peaceful or hostile). Each of these decisions is made at a different control station and depends on five distinct parameter values, only three of which are available at that station. Subjects therefore must communicate among themselves to assure that they have all hooked the same target and subsequently exchange parameter values to classify the target. It the team finds a target to be hostile it is shot, otherwise it is cleared and the team moves on to another target.

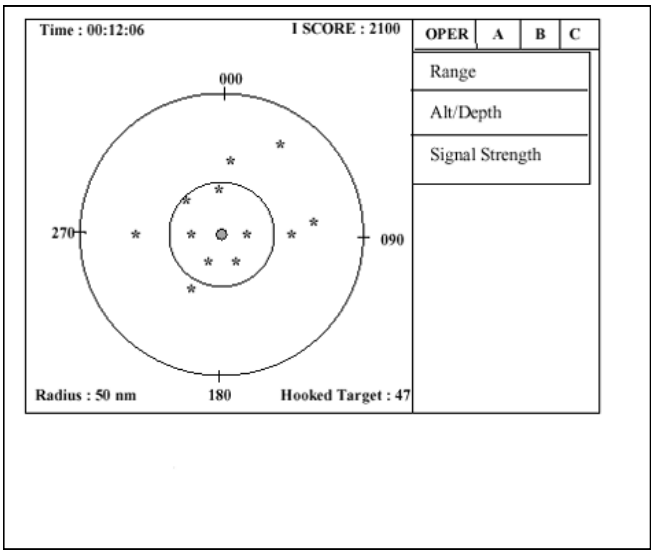


Figure 1. The TANDEM display

In standalone mode all of the information is made available on a single pc with the station specific parameters accessed using three distinct menus. Menus in standalone mode present 5 parameters each. In team mode the three menus present 3 (overlapping among team members) parameters per menu. Just as TANDEM simulates cognitive aspects of the Aegis missile command and control task, it provides a context to simulate the gathering, aggregation, and presentation of C2I information by intelligent agents. The information found on menus remains ground truth while the validity of agent processed information can be manipulated by the experimenter. To investigate impacts on human-human coordination presentations of aggregated information can be tailored to support different aspects of the participants' cognitive tasks.

2.1 Trust, Error, and Uncertainty

Many of the complex of issues involving mutual human-machine modeling, awareness, and coordination are captured by the anthropomorphic term trust. If we examine the considerations that enter into our decision to delegate a task to a subordinate, instruct the subordinate in how to perform the task, monitor that performance, or authorize some class of tasks without follow-up, our trust in the subordinate will almost certainly play an explanatory role. Closer consideration will show our use of the term to be multidimensional. The trust we have that our secretary will remember to pick up the mail, is distinct from our trust that she will compose a postable business letter, which in turn is distinct from our trust in the lawyer who assures us that the letter is not actionable. A merger of several taxonomies proposed by Lee and Moray (1992) distinguishes:

1) trust which is based on observed consistency of behavior (*persistence* or *predictability*)I trust my watch to keep relatively accurate time

2) trust which is based on a belief in competence or well formedness (*competence* or *dependability*).I trust the recipe for hollandaise

3) trust which is based on faith in purpose or obligation (*fiduciary responsibility* or *faith*)I trust my physician to monitor my health

As bases for human modeling of machine agents this taxonomy suggests that agents can be made predictable by: 1) consistently pairing simple observable actions with inputs, or 2) making the causes and rules governing an agent's behavior transparent or 3) making the purpose, capability, and reliability of the agent available to the user. Muir [5] refers to the process of acquiring predictive models of these sorts as *trust calibration*. The idea being that performance will be better for human-machine systems in which trust is accurately calibrated because the human's model will allow more accurate predictions. The greater predictability of consistent or competent agents should also make boundary conditions and "brittleness" more apparent and remediable. Agents trusted on faith, by contrast, would require a very high degree of reliability across their range and more communication to maintain accurate coordination.

Where information sources are unreliable or information processing algorithms are uncertain, brittle, or error prone, usefulness of their services will depend on how readily they can be incorporated into their user's model of the situation. The first TANDEM experiment manipulates error, error source, and transparency of presentation to address the questions of task allocation (under what conditions should automated information processing be curtailed or eliminated) and information presentation (can choice of presentation context affect the usability of processed information).

We hypothesize that effective human/agent performance requires a precise calibration of trust so that the decision maker can accurately interpret an agent's communications and anticipate its limitations. This calibration will depend both on experience and supporting evidence.

2.2 Standalone TANDEM Experiment

We classify information presentations into three types which roughly parallel the level of trust they rely upon for interpretation:(1) aggregated, (2) integrated, and (3) synthesized. The reported experiment pairs error-making and error-free data presentations with differing degrees of inspectability to observe the effects on decision quality, reliance on agent provided information and reported confidence. .

Displays

Each agent provided one of three possible levels of information.

To manipulate the subjects' trust of the agents' presentations, errors were introduced. In the control condition, both menus and agent presented errorless values. Errors were of three types: data errors (display levels 1, 2, & 3), classification errors (display levels 2 & 3), or decision errors (display level 3). Data errors occurred when the agent displayed different data than the ground truth shown on the corresponding menus. This type of error was explained to subjects as "problems with the agent's sensors". Classification errors occurred when the agent placed data in the wrong column of the display table (level 2, placing an altitude reading of 1000 feet in submarine column for example) or used a wrong classification to determine assignment type and certainty factor (level 3). Decision errors occurred when the oracle assigned an incorrect "type" to a target independent of target data. Classification and Decision errors were explained to the subjects as "software problems". Errors of the different types were equated by matching corresponding rates to the multinomial reference distribution followed by data errors (5 independent parameters with 1/3 probability of error {.132, .33, .33, .164, .04, .004} for 0-5 errors). Only one source of error was presented during a TANDEM session. Buttons presses to access agent information, menu selections, target hooks, classifications, and final actions and times were collected along with simulation states for each subject. Agent displays tested showed:

1) aggregated information (list) — a list of parameters and values

2) integrated information (table) — a table showing categorized values

3) synthesized information (oracle) — target type assignment with certainty factor.

Method

Sixty targets were distributed in several concentric rings on the screen. The circle closest to the center is referred to as the "circle of fear" and the amount of time a target spent in this circle before being identified was measured as penalty time. The number of targets identified while in this penalty circle, targets identified outside of the penalty circle, and targets hooked but not resolved were all measured. Ratings of "trust" of simulated information agents using scales developed and validated by Muir [5] were also gathered from each subject. These ratings, on a scale of 1 (low) to 5 (high) focused on issues of dependability, predictability, accuracy, reliability and an overall "assessment of trust in the agent. Seventy-eight paid subjects recruited from the University of Pittsburgh community participated in the experiment. Data from eight subjects were excluded from this analysis due to equipment failure, incomplete data, or failure to follow instructions. Subjects received standard instructions and a sheet of tables showing the correspondence between parameter values and identification decisions. Subjects were assisted through a five minute training session operating the simulation and then completed two 15 minute experimental trials, concluding the session by completing a "trust in automation" [5] survey.

Results

Performance was analyzed using a repeated measures analysis of variance with session as the within subject factor and types of error and level of agent as between group factors. Effects of session were significant ($p < .05$) for each of the dependent measures reported. Where differences were found between groups, data was pooled across the two sessions and Post hoc analyses conducted using Tukey's HSD to identify reliable differences among the conditions.

Performance measures reported in preliminary form in Lenox et al. [3] fell into three groups which can be categorized as targets engaged, targets engaged within penalty circle, and agent use/correct identification. The number of targets shot or cleared, number of targets engaged, and score were affected by the type of presentation, the type of error and the interaction between presentation and error type. Errorless presentations led to processing more targets and the table presentation led to processing more targets ($p < .04$).

The number of non-penalty targets engaged, number of penalty targets engaged, and time targets remained in the penalty circle were affected by the presence of errors and particularly decision errors ($p < .04$). Subjects' willingness to activate an agent depended on the type of agent and the presence/absence of errors. Subjects activated the agents more often in the no error condition and activated the oracle more often than the other agents ($p <.05$). Subjects' ratings on 10 of the 11 scales on Muir's trust in automation questionnaire were lower for agents committing errors ($p < .05$)

Ratings of trust in automation using scales developed by Muir [2] were collected at the conclusion of the experiment. Subjects' ratings on 10 of the 11 scales were lower ($p < .05$) for agents committing errors and were not affected by the level of agent.

The level 2 (table) agent provided the best support for the target identification task. Although subjects consulted the level 3 (probability assignment) agent more than either of the other two agents, their scores were lower than subjects using level 2 agents and errors in the probability assignment led to longer penalty times. Regardless of their source, errors affected subjects' performance, reliance on agents and ratings of trust in a similar manner. Contrary to our expectations, presentation did not appear to affect the subjects' ratings of trust or penalty times.

These results demonstrate the dangers of using agents to collect, aggregate, and process information without providing their user the ability to monitor and evaluate their product. The participants rated their trust of the level-3 oracle as highly as the others and accessed it more often yet performed more poorly than the others. Equally clear was the failure of processed information to provide added value when errors occurred. Subjects using the level-2 (Table) agent with classification errors, for example, had available on their agent display exactly the same information as those using the level-1 (List) agent without errors yet performed less well.

2.3 Supporting Individuals vs. Teams

We have developed a framework for examining the different ways that machine agents can be deployed in support of team performance. One option is to support the individual team members in completion of their own tasks. Another option is to allocate to the machine agent its own subtask as if we were introducing another member into the team. In this case all the issues associated with communication and coordination among team members become relevant [1], [8], [9].

The third option is to support the team as a whole by facilitating communication, allocation of tasks, coordination among the human agents, and attention focus. A basic tenet of this approach is that teamwork skills exist independent of individual competencies. The performance of teams, especially in tightly coupled tasks, is believed to be highly dependent on these interpersonal skills.

The second TANDEM study examines different ways of deploying machine agents to support multi-person teams: 1) supporting the individual (within a team context) by keeping track of the information he has collected and in sense, helping the individual with his task and with passing information to team mates (Individual Clipboard); 2) supporting communication among team members by automatically passing information to the relevant person which should reduce communication errors and facilitate individual classification (Team Clipboard); and 3) supporting task prioritization and

coordination by providing a shared checklist (Team Checklist). We hypothesized that the Individual Agent should aid the individual task and aid communication among team members (Figure 2). This agent shows all data items available to an individual team member (in this case, ALPHA) and fills in the values for the data items as the subject selects them from them from the menu. The values under the TYPE

heading assist the individual with their task while the other team members may need the remaining values. The Team Clipboard Agent should also aid the individual task and aid team communication to a greater degree than the Individual Agent (Figure 3) should. This agent aggregates values from all members of the team to help the individual with his/her task. It automatically passes values as they are selected from the menu to the appropriate team member. Thus, when altitude/depth is selected from a menu, it is passed to an individual team member (ALPHA) who can use it to make the type identification. We hypothesized that this agent should reduce verbal communication among team members and reduce communication errors. The third agent, Team Checklist, should aid team coordination (Figure 4). This agent shows who has access to what data. For example, all three team members (ALPHA, BRAVO, CHARLIE) have access to speed, but only BRAVO has access to "Intelligence". The final condition is a control where we observed team performance without the aid of any machine agent. This is the standard TANDEM paradigm used by Jentsch, et al [9]. The goal of the study is to examine the impact of the aiding alternatives on: 1) communication patterns, 2) data gathering strategies, 3) reliance (i.e., use of) on the intelligent agents, and 4) performance.

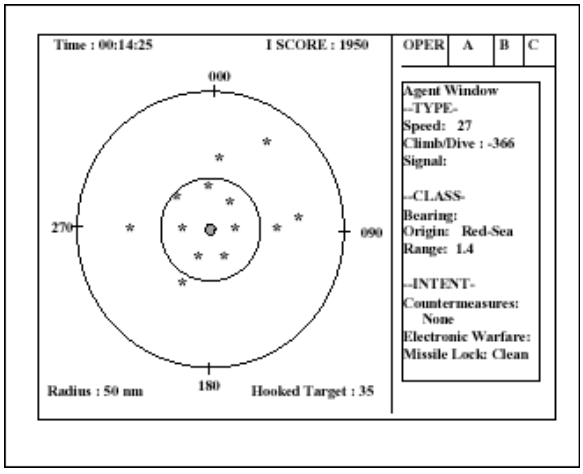


Figure 2: Individual agent

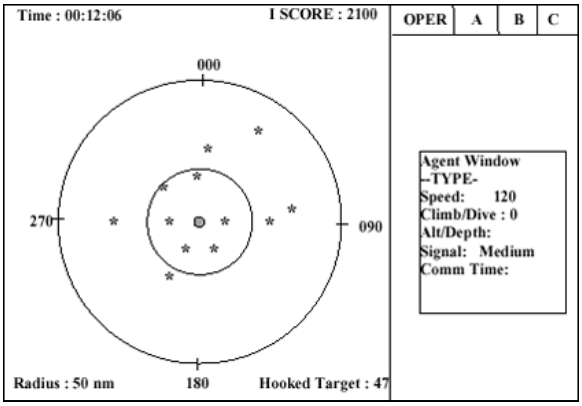


Figure 3 Team clipboard

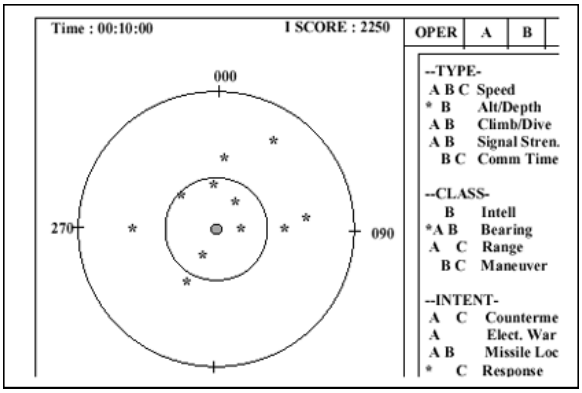


Figure 4 Team checklist

Method

Teams of three subjects were recruited for this study. Each team was assigned to one of four conditions: 1) control, 2) individual agent, 3) team clipboard agent, or 4) team checklist agent. TANDEM was used with three-person teams, each member with a different identification task to perform (air/surface/submarine, military/civilian, and peaceful/hostile). One person was assigned to ALPHA, one to BRAVO and one to CHARLIE. ALPHA, BRAVO and CHARLIE had different items on their menus and different tasks during the trials. ALPHA identified the type of target (air, surface or submarine); BRAVO determined whether the target was civilian or military; CHARLIE determined whether the target was peaceful or hostile. In addition, CHARLIE acted as the leader by indicating the type, classification and intent of each target to the system and taking the final action (shoot or clear).

There were five pieces of information for each identification task, three of which must agree in order to make a positive identification. These pieces of information were distributed among the three team members. Each team member saw different data items on the menus and had three data items required for his/her identification task and several other items that the other team members might need to complete their tasks. Thus, the subjects needed to communicate with one another to perform their tasks for roughly two-thirds of the targets. All five pieces of information might agree for a particular target, however, in many cases, the ambiguity of the data was manipulated such that only three pieces agreed.

The targets were divided into three groups: 1) easy—all three pertinent items on the individual's menu agree; 2) medium—only two items on the menu agree, a team member must ask one or both teammates for data; and 3) hard—two items on the menu agree, but do not provide the correct solution. For example, ALPHA's task was to identify the type of target. If the target was easy, all three items on ALPHA's menu indicated the same type (e.g., air). If the target was of medium difficulty, one or two values would indicate air and the other indicate submarine. If the target was hard, both of ALPHA's menu items indicate air, but the remaining three items from ALPHA's menu and the other team members indicate surface. Thus, the target is a surface vessel. Subjects had no way of knowing the difficulty level of the targets.

Each team participated in a 90-minute sessions which began with a 15- minute training session in which the TANDEM software and team goals were explained. The team was told to identify as many targets as possible, as accurately as possible during the 15-minute trial. After the training session, the team participated in three 15-minute trials. At the conclusion, subjects were asked to complete a brief questionnaire.

Several forms of data were collected during the trials: 1) performance data from Tandem logs including the type and number of targets hooked and classified, the percentage of targets correctly identified, and the number of times the agents were activated; 2) communication data encoded from observers or audio tapes including the number of requests for data (e.g., does anyone have initial range?), the number of responses (e.g., range is 5.6 nm), the number of target identifications (e.g., it's civilian), and the number of confirmations (e.g., target is sub, civilian); 3) observer data including ratings on team communication, situation assessment, leadership ad supporting behaviors; and 4) questionnaires completed by the subjects before they leave.

Results

The performance data reported in this paper are based on five teams per condition. Time per target varies for both the target difficulty and across conditions. For example, teams took approximately 450 seconds per target to process hard targets in the control condition, 350 seconds in the individual agent condition, 250 seconds in the team clipboard condition, and 150 seconds in the team checklist condition.

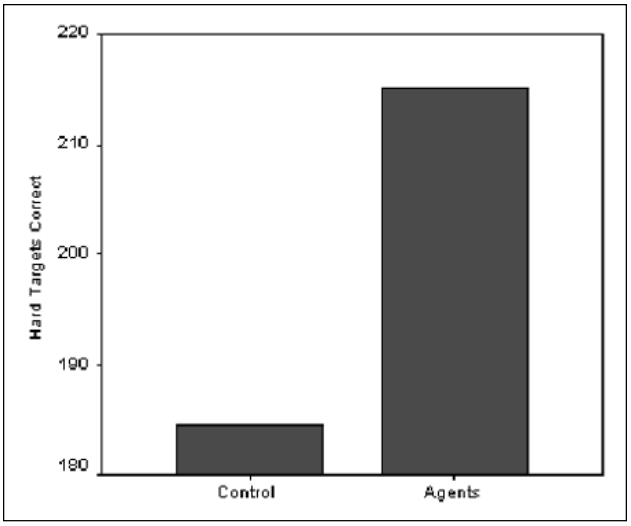


Figure 5

Using a repeated measures design with four conditions there were significant order effects across the three trials for the proportion of correct targets (p<.009), the time per target (p<.0001), and the total targets hooked by a team (p<.0001). Effects were also found across the target difficulties (easy, medium, and hard) for the proportion of correct targets (p<.0001) and the time per target (p<.0001). In pairwise comparisons for time per target, the control condition differed from the team clipboard agent (p<.03) and the control condition differed from the team checklist agent (p<.02).

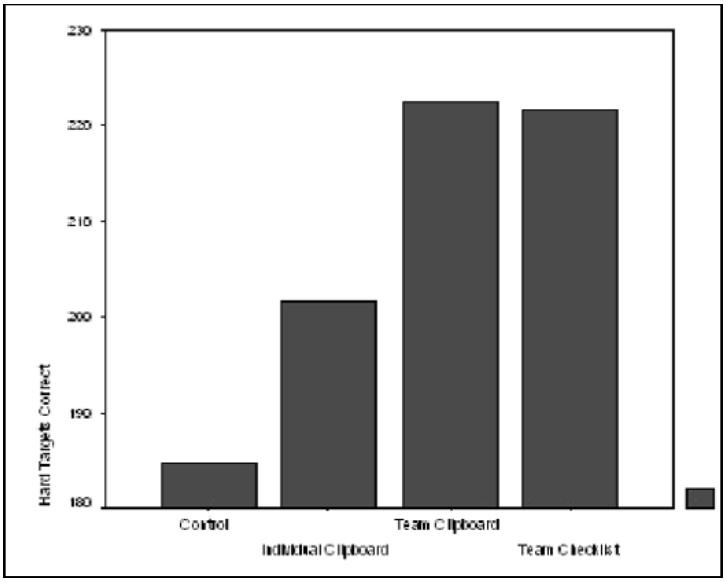


Figure 6. Percentage correct for hard targets

Grouping all agent conditions (individual agent, team clipboard agent and team checklist agent) into one condition, showed that agent aiding was superior to the no

aiding condition (control) over the three trials on the proportion of correct targets ($p<.0001$), time per target ($p<.0001$) and total targets hooked ($p<.0001$).

Subjects learned across the trials, hooking more targets, spending less time on any particular target and getting more targets correct. As Figure 6 shows, aiding teamwork directly (team clipboard/checklist) proved more effective than supporting team members at their individual tasks despite the reductions in memory load and ready accessibility to parameters for sharing provided by the individual clipboard. The potential for coordinating human-human interactions through agent systems seems a particularly promising approach because of the high payoff and the reusable and largely domain independent character of the agents' tasks.

3. MokSAF Experiment

Human decision-makers, particularly military commanders, face time pressures and an environment where changes may occur in the task, division of labor, and allocation of resources. Information such as terrain characteristics, location and capabilities of enemy forces, direct objectives and doctrinal constraints are part of the commander's "infosphere." This information is routinely gathered and organized through geographical and other information systems. Information within the infosphere has the opportunity for data fusion, situation visualization, and "what-if" simulations. Software agents have access to all information in the infosphere and can plan, criticize, and predict the consequences of actions using the information at a greater accuracy and finer granularity than the human commanders can

These agents cannot consider information outside the infosphere unless it is explicitly translated in a compatible form.. This extra-infosphere data consists of intangible or multiple objectives involving morale, the political impact of actions (or inaction), intangible constraints, and the symbolic importance of different actions or objectives. Military commanders, like other decision-makers, have vast experiential information that is not easily quantifiable. Commanders must deal with idiosyncratic and situation-specific factors such as non-quantified information, complex or vaguely specified mission objectives and dynamically changing situations (e.g., incomplete/changing/new information, obstacles, and enemy actions). When participating in a planning task, commanders must translate these intangible constraints into physical ones to interact with planning agents. The issue then becomes how should software agents interact with their human team members to incorporate these intangible constraints into the physical environment effectively.

3.2 The Planning Environment: MokSAF

A computer-based simulation called MokSAF was developed for these experiments and two agent interfaces are currently undergoing evaluation. MokSAF is a simplified

version of a virtual battlefield simulation called ModSAF (modular semi-automated forces). MokSAF allows two or more commanders to interact with one another to plan routes in a particular terrain. Each commander is tasked with planning a route from a starting point to a rendezvous point by a certain time. The individual commanders must then evaluate their plans from a team perspective and iteratively modify these plans until an acceptable team solution is developed.

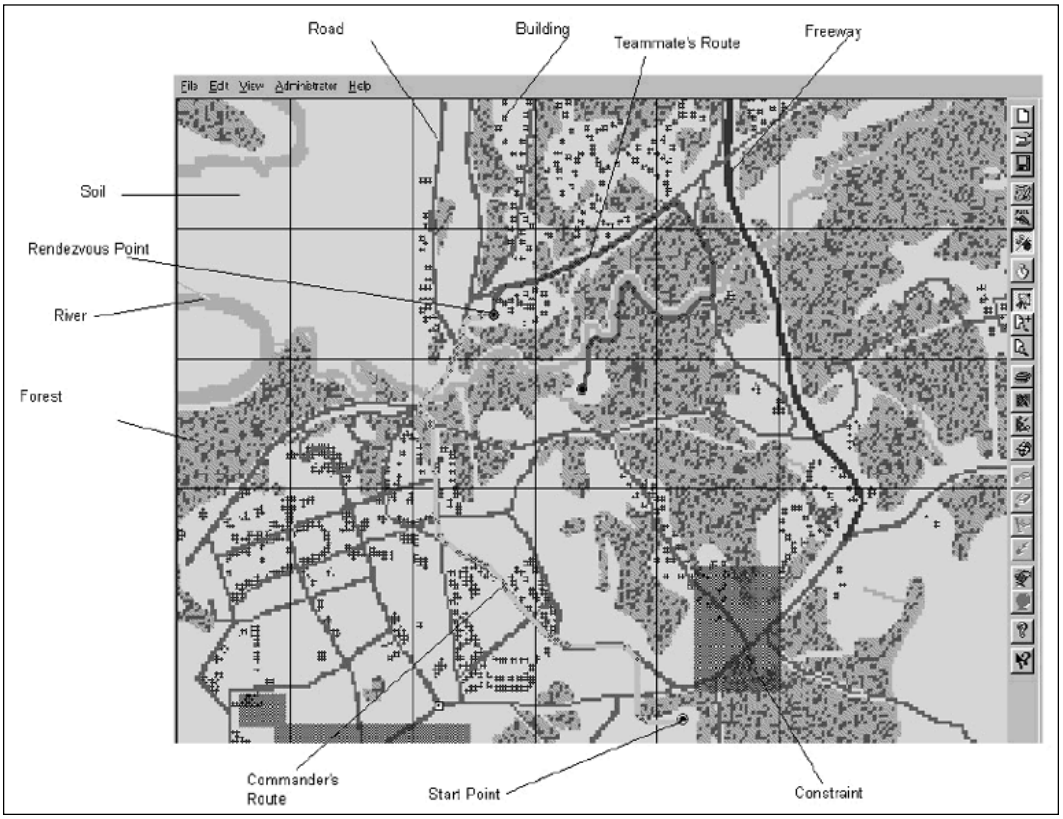


Figure 7 MokSAF

Figure 7 shows the MokSAF Environment, including the terrain map and the toolbar. The terrain consists of soil (plain areas), roads (solid lines), freeways (thicker lines), buildings (black dots), rivers and forests. The rendezvous point is a red circle and the start point is a yellow circle on the terrain map. As participants create routes with the help of the Path Planner Agent or the Critique Agent, the routes are shown in bright green. The second route shown is from another MokSAF commander who has agreed to share a route.

The partially transparent rectangles are social constraints that the user has drawn on the terrain map to indicate to the agents which areas should be avoided. Once these constraints have been drawn on the map, the Path Planner Agent will not draw a route through these coordinates and the Critique Agent will inform the user that a constrained area has been violated.

3.3 MokSAF Agents

Two different software agents which interact with the human team members in the planning task have been developed for MokSAF. The first agent, the Path Planner Agent, guides the human team members through the route-planning task and performs much of the task itself. This agent acts much like a "black box." The agent creates the route using its knowledge of the physical terrain and an artificial intelligence planning algorithm that seeks to find the shortest path. The agent is aware of physical constraints only. Commanders must translate the intangible constraints into physical ones by drawing constrained areas on the maps.

The second agent, the Critique Agent, analyzes the routes drawn by the human team members and helps them to refine their plans. In this mode, the human and agent work jointly to solve the problem (e.g., plan a route to a rendezvous point). The workload should be distributed such that each component matched to its strengths. Thus, the commander, who has a privileged understanding of the intangible constraints and utilities associated with the mission, can direct the route around these constraints as desired. However, the commander may not be as knowledgeable about the terrain and so the agent can indicate where the path is sub-optimal due to violations of physical constraints.

The commander draws the desired route and requests that the Critic Agent review the route for physical violations or to indicate ways in which the path could be improved. The commander can iteratively improve the plans until a satisfactory solution is reached.

Method

In the current MokSAF pilot experiments, a deliberative, iterative and flexible planning task is examined. There are three commanders (Alpha, Bravo and Charlie), each with a different starting point but the same rendezvous point. Each commander selects units for his/her platoon from a list of available units. This list currently contains M60A3 tanks, M109A2 artillery units, M1 Abrams tanks, AAV-7 amphibious assault vehicles, HMMWVs (i.e., hummers), ambulances, combat engineer units, fuel trucks and dismounted infantry. This list can be easily modified to add or delete unit types. With the help of one of the software agents, each commander plans a route from a starting point to the rendezvous point for the specified platoon.

Once a commander is satisfied with the individual plan, he/she can share it with the other commanders and resolve any conflicts. Conflicts can arise due to several issues including shared routes and/or resources and the inability of a commander to reach the rendezvous point at the specified time. The commanders must coordinate about

the number and types of vehicles they are planning to take to the rendezvous point. The mission supplied to the commanders provides them with a final total of vehicles required at the rendezvous point. In addition, the commanders are told that they should not plan a route that takes them on the same path as any other commander and that they should coordinate their routes to avoid shared paths.

MokSAF 1.0 was used for this pilot study. It consists of the standard terrain map and markings, a toolbar as seen in Figure 5, a communication window where commanders can send and receive messages and share plans, and a constraint tree. The two different agent interfaces described above were evaluated. Fifteen teams consisting of three-persons were recruited (10 teams in the Planner Agent condition and five in the Critic Agent condition) from the University of Pittsburgh and Carnegie Mellon University communities. Participants were recruited as intact teams, consisting of friends or acquaintances. Each team member had a different starting point, but all had the same rendezvous point. Teammates needed to communicate with one another to complete their tasks successfully.

Each team participated in a 90-minute session that began with a 30-minute training session in which the MokSAF environment and team mission were explained. The team was told to find the optimal path between the start and rendezvous points, to avoid certain areas or go by other areas, to meet the mission objectives for numbers and types of units in their platoon, and to avoid crossing paths with the other commanders. After the training session, the team participated in two 15-minute trials. Each trial used the same terrain, but different start and rendezvous points and different platoon requirements. At the conclusion, participants were asked to complete a brief questionnaire.

Results

We examined time to share a route for the three commanders and found that the Planner Agent interface had an advantage over the Critic Agent interface ($p < .005$ for Alpha, $p < .063$ for Bravo and $p < .006$ for Charlie). Groups using the Planner Agent spent less time creating their individual plans before sharing them with their teammates.

We also examined the individual path lengths for each commander at two points in each trial when routes were first shared with the team and at the end of the 15-minute trial. The ending path lengths for Alpha ($p < .001$), Charlie ($p < .001$) and combined were better using the Planner Agent Interface than with the Critic.

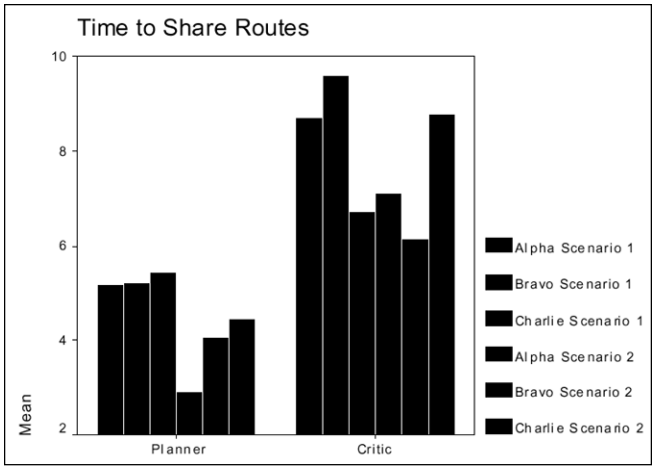


Figure 8.

It is expected that path lengths between the first time a route was shared and at the end of a trial would vary due to issues related to conflict resolutions among the teammates. There was a significant difference in the change in path lengths from these two points in time ($p < .018$). Figure 6 shows that participants using the Critic Agent interface made more changes in their paths. This change could be due to the state in which the route was in when first shared; that is, the routes drawn by the participants may have required additional refinement during the trial. Another possible reason for the change in the paths could be due to interactions with teammates.

Participants were asked to create optimal routes given certain confounding factors (e.g., avoiding constraints, going to designated areas, and avoiding traveling on the same paths as other commanders). They were also asked to plan as a group numbers and types of units at the rendezvous point. We found that there was no difference in this selection of units in either agent interface condition.

In its current form, the Path Planner has been shown to provide a better interface for both individual route planning and team-based re-planning. While the individual plans for Critic users in the Alpha and Bravo roles were not significantly different from Planner users in quality, it took them substantially more time to construct their routes. The eventual coordinated routes were uniformly better for each of the individual positions in the planner group and for the team as a whole. Despite this clear superiority, participants in the Planner group frequently expressed frustration with the indirection required to arrange constraints in the ways needed to steer the planner's behavior and often remarked that they wished they could "just draw the route by hand".

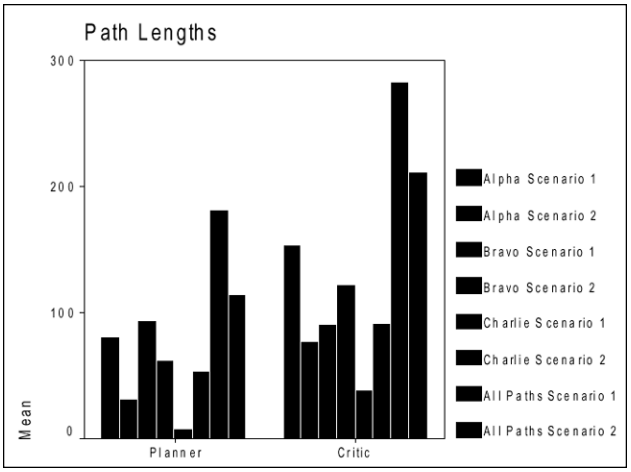


Figure 9

In the Critic condition complaint s focused more closely on the minutiae of interaction. In its current form, the user "draws" a route in MokSAF by specifying a sequence of points at the resolution of the terrain database. To do this, she clicks to specify an initial or intermediate point in the path and then clicks again at a second point. A sequence of points is then drawn in a straight line between these locations. A route is built up incrementally by piecing together a long sequence of such segments. Although MokSAF provides tools for deleting unwanted points and line segment "rubberbanding" for moving control points, the process of manually constructing a long route is both tedious and error prone. While the Planner automatically avoids local obstacles such as trees and closely follows curves in roads due to their less costly terrain weights, a user constructing a manual route is constantly fighting unseen obstacles which void her path or line segments which stray a point or two off a road into high penalty terrain. The anticipated advantages of heuristic planning and cooperation among human users were largely lost due to the necessity of focusing on local rather than global features of routes.

The experience we have gained in developing and evaluating the critic interface will be used to redesign and test a new hybrid version of the task in which automated path planning will be performed within an approximate path drawn by the human user. Of the lessons learned in this initial test of our agent-based alternatives, the difficulty of creating good interfaces for communicating human intent stands out. The Planner interface which minimizes this communication was very successful in its initial implementation. The Critic interface, by contrast, will require substantial revision before it approaches the planner in articulatory directness and fluency. We hope that subsequent refinements to the Critic may allow a more thorough comparison of the effects of agent and human initiative on team planning and re-planning tasks.

4. InfoWrapper

One of the most commonly cited tasks to be solved by agent technologies is seeking out and extracting information from the World Wide Web. The problem is actually much more difficult than it seems and despite many efforts agents do not yet intelligently seek out new sites and extract information from them although many systems will return URLs likely to contain desired information. While parsing HTML is fairly straight forward, specifying the information to be extracted from a content/user oriented view is much more complex. A promising approach to this problem is to treat it as one of human-agent cooperation. Web pages have been designed and formatted to promote human discriminations and judgements so while the tagging conventions of HTML may be commonly violated their appearance is tightly controlled and "intelligible" to humans. If we can devise the means for the human to communicate her discriminations and semantic judgements to the agent it should be possible for the joint system to develop site specific information extractors fairly efficiently. What is needed is some form of "programming by example" with provisions for indicating and discriminating among procedural instructions (accessing successive pages for instance), informational templates (the boundaries and constituents of classified ads or example) and string constants/variables for matching. From the user's perspective, one would like to access a page through a browser, select (highlight) an instance of the items to be searched, and associate the selection's constituents with a schema for testing and extracting matching instances. The design challenge is to build an interface which make this kind of direct manipulation specification as transparent as circling a classified ad and underlining an object's name and price.

Unlike the controlled experiments with TANDEM and MokSAF, our efforts in developing the InfoWrapper follow an iterative prototyping plan where through development and testing we explore a range of potential interaction schemes for bridging the gulf between what the user sees and what the agent parses. The InfoWrapper has by far the most complex task of conveying human intent of the systems we have studied. Like TANDEM, it must operate in an open environment where errors are likely and mechanisms for monitoring and evaluation must be designed in. Like MokSAF, there are many possible ways to allocate tasks and control with good choices likely to emerge only after repeated testing. We believe that the future of human agent systems will lie in such multi- method interactions which can combine demonstration, direct manipulation, machine learning, and command language in effective enough ways to bridge Norman's gulfs and allow communication of complex intents and perceptions

References

1. P. M. Jones and C. M. Mitchell. Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent associate system. *IEEE Transactions on Systems, Man, and Cybernetics*. SMC-25(7), pages 1039-1053, 1995.

2. J. Lee and N. Moray. Trust, Control Strategies, and Allocation of Function in Human-Machine Systems. *Ergonomics* 35(10), pages 1243-1270, 1992.

3. T. Lenox, L. Roberts, and M. Lewis. Human-Agent Interaction in a Target Identification Task. In *1997 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2702-2706, Orlando, FL: IEEE, 1997.

4. J.T. Malin, D.L. Schreckenghost, D.D. Woods, S.S. Potter, L. Johannesen, M. Holloway, and K. D. Forbus. *Making Intelligent Systems Team Players: Case Studies & Design Issues. Human-Computer Interaction Design* (NASA Technical Memorandum 104738). Houston, TX: NASA Johnson Space Center, 1991.

5. B. Muir. Trust in Automation: Part I. Theoretical Issues in the Study of Trust and Human Intervention in a Process Control Simulation. *Ergonomics*, 39(3), pages 429-460, 1994.

6. N. Negroponte. *Being Digital*. New York, NY: Alfred Knopf, 1996.

7. D. Norman. *Cognitive Engineering. In User Centered System Design: New Perspectives on Human-Computer Interaction*, eds. D. Norman and S. Draper, pages 31-61, Hillsdale, NJ: Lawrence Erlbaum, 1986.

8. E. M. Roth, J. T. Malin, J. T and D. L. Schreckenghost. Paradigms for Intelligent Interface Design. In M. Helander, T. K. Landauer, and P. Prabhu (Eds) *Handbook of Human-Computer Interaction*, Second Edition, pages 1177-1201, 1997.

9. K. Smith-Jentsch, J. H. Johnston, S. C. Payne (in press). Measuring team-related expertise in complex environments. To appear in J. A. Cannon-Bowers and E. Salas (eds.), *Decision Making Under Stress: Implications for Individual and Team Training*. Wash., DC: American Psychological Association.

The Combat Decision Range: Multimedia Training in Decisionmaking under Stress

Francis J. West
President, GAMA Corporation

In March 1997, the Marine Corps Warfighting Laboratory under the command of Col. Anthony A. Wood, conducted the *Hunter Warrior* Advanced Warfighting Experiment in multiple sites in Southern California. Among other experimental objectives, this week-long experiment deployed individual rifle squads 150 miles from their parrent platoons and companies located in simulated ships at sea. Each squad had to rely upon its own skills to survive and ability to employ long-range fire support to attack a much larger, more mobile enemy force.

The experiment demonstrated the potential capability infantry small units leaders have to assume an expanded tactical role if given the proper training and equipment It also demonstrated that the one most important discriminator in effectiveness of the various squads was the ability of the squad leader as a combat decision maker.

The squad leaders in this experiment were not hand picked superstars. Instead, they were an average infantry battalion's non-commissioned officers. They had typical training in leadership and small unit tactics. They also received some instruction in the theory of small unit decision making. During pre-experiment training, some failed to demonstrate the ability to make effective combat-related decisions under pressure. Where possible these small unit leaders were replaced. Others improved rapidly once they were put in the role and provided the opportunity to learn on the job.

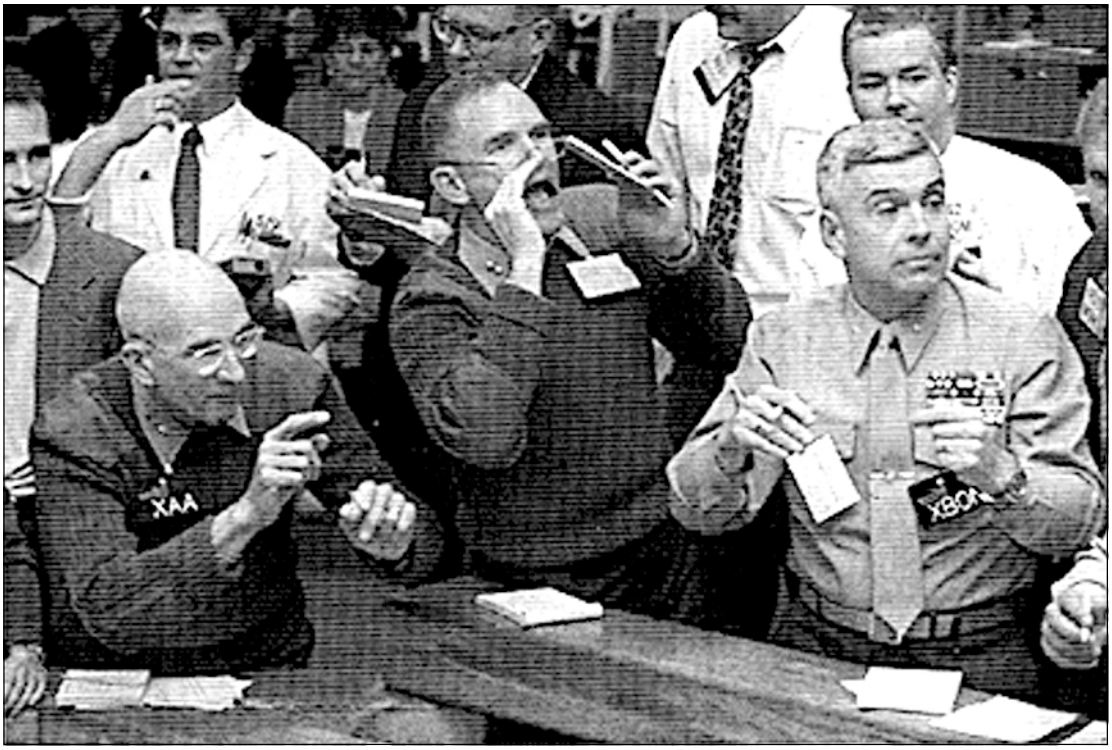
The experiment identified two specific problems. How do you determine who has the ability to develop into an effective combat decision makeer? And second, how can the Marine Corps train small unit leaders to be effective decision makers in the chaos and confusion of combat?

How to Identify Effective Decision Makers

The Marine Corps Wargaming Division, supported by GAMA Corporation of Falls Church Virginia, VA, had been seeking answers to these questions through a series of wargames with the New York Mercantile Exchange. The Wall Street Traders make hundreds of decision every day in an atmosphere of chaos and confusion that resembles combat in its intensity and ambiguity. The Traders are decisive, intuitive decision makers. They absorb information with all of their senses and make second by second assessments as to the direction of the market.

The traders indicated that there is no formula for predicting who will make a good trader and who will not. Some successful traders have advanced degrees; more do not. Some have a background in business; most do not. The one characteristic all have in common is an aura of self-confidence. However, the traders indicated that this aura emerges over time and is not necessarily identifiable in advance.

The only way to identify a successful trader is through watching him trade. The traders do this through a program of mock trades. To add realism they add as much of the confusion and chaos as they can to mimic the atmosphere of the actual exchange floor. Over time, the candidate traders begin to show whether they can intuitively identify the patterns of trades and prices. The traders often cannot identify why they made a specific trade. They are making decisions based on ambiguous — and often conflicting — information. Like combat, they cannot wait for more information. The profits go to those who first identify the direction of the market.



Those that demonstrate in these mock trades that they have potential are sponsored onto the actual Mercantile Exchange. They then trade using their sponsors money. Those that make good decisions — that show a profit — gain the time to make additional trades. Those that do not, find another line of work. The traders, indicate that success in the first few trades are essential because only success then earns the opportunity to gain sufficient experience to truly become an effective trader later.

This experience is not unlike that of combat. Historically, the only true indicator as to the effectiveness of combat leaders is success in combat. Many who are very successful in training cannot handle the chaos and pressure of combat and fail to survive. But using combat to weed out the effective combat leaders is not only inefficient but leads to ineffective units in the first battle. There must be a way to simulate the pressures of combat sufficiently to permit an organization to assess its small unit leaders and to permit those that are chosen to practice making sound decision making.

Practicing Decision Making

The Marine Corps Warfighting Laboratory turned to analogous organizations that face similar problems in training. Firefighters offered a possible comparative decision making problem. On behalf of the Wargaming Division of the Marine Corps Warfighting Lab, GAMA Corporation conducted a series of exploratory seminars with the New York City Fire Department — the largest and arguably the best-trained fire department in the world.

A fire department was chosen because urban fire fighting closely approximates many of the same situations resident in urban combat. The New York Fire Department aggressively attacks fires deep inside high-rise concrete buildings on a daily basis that require communicating in dense smoke and limited visibility.

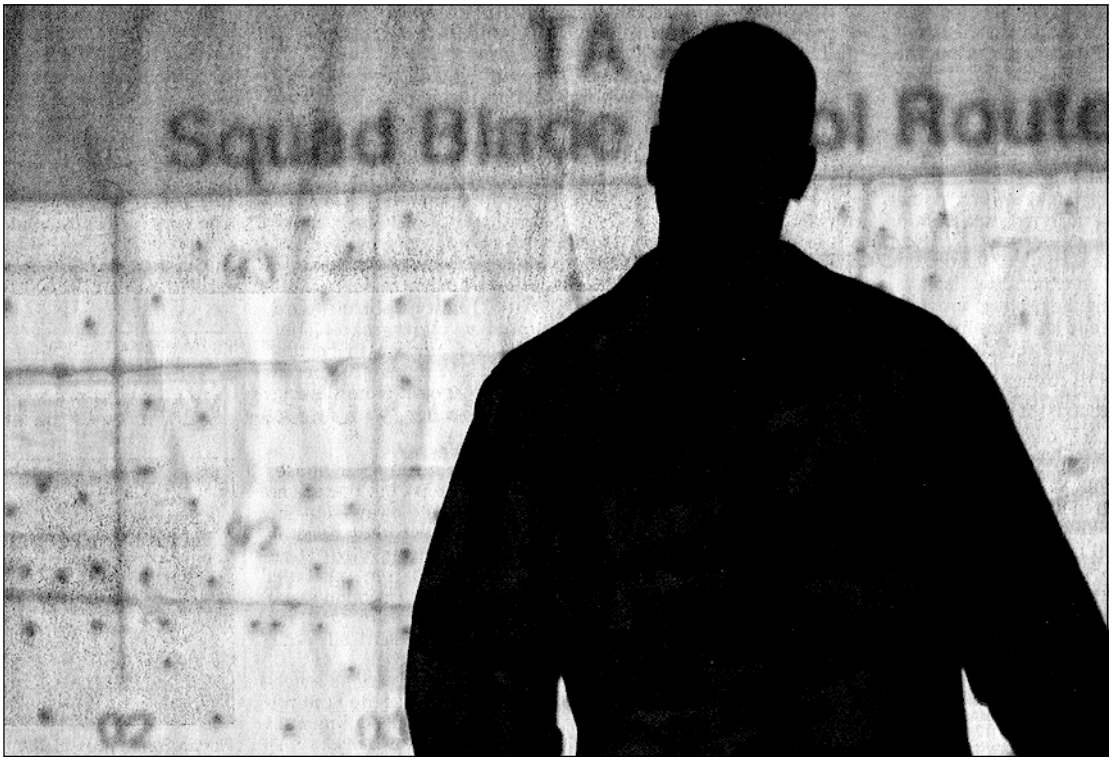
How do they do it? Their capability appears to have three key components. First, small handheld UHF radios are distributed throughout the ranks; "breaking news" is instantly known by all. Second, a fire is geographically "gridded" by use of voice codes; each firefighter has situational awareness of adjacent unit's locations and the location of all "hot zones." Third, every new fire chief goes through a computer-based classroom training course. Under the tutelage of experienced chiefs, he must fight twelve different types of city fires. On a large screen, he watches pictures of a spreading fire while the older, more experienced chiefs act as subordinates and shout confused calls. As "on-scene commander," the new chief must respond promptly, determine the extent of the blaze, deploy his men, and call for reinforcements.

The old fire chiefs stressed that a new chief does not have broad fire fighting experience. He has not yet encountered many types of fires. The scenarios and the tutors are the means of giving him vicarious experience by placing him under stress and insisting he make decisions on the spot. Part of the stress comes from being evaluated by his peers. When he completes this test, the new chief is prepared to encounter any conceivable fire in the real world. In short, he is expected to look at a fire and be able to say, "I have seen this situation before, I know what needs to be done, and I know how to do it."

The Combat Decision Range

The fire fighters technique led to the development of a Marine Corps Combat Decision Range to be used to assess the capabilities of small unit combat leaders and to permit them to practice combat decision making. The first prototype was developed by GAMA Corporation, based in part on the fire fighters example as modified by corporate experience in over 200 wargames and simulations conducted over the past 15 years. The prototype was developed specifically to stress and train the squad leaders to be used in the Urban Warrior experiment. It was called a "range" because Marines prepare for combat by training on the rifle range. Similarly, with combat decision making, combat leaders need to train to make sound decisions under stress – *before they go into combat.*

GAMA adapted the basic technique, added multimedia computer programming, and wrote a series of tactical scenarios based on real firefights. These scenarios were then used in the training of the infantry battalion squad leaders used in Urban Warrior. Subsequently, these scenarios have been modified and are now being delivered to all of the infantry regiments in the Marine Corps as the Collins' Combat Decision Range. It is named after a retired Marine Colonel Pat "Paddy" Collins, who died while working for the Marine Corps Warfighting Lab developing prototype squad level training programs. By the end of this year, the Combat Decision Range will be in use through out all eight active and three reserve infantry regiments in the Marine Corps.



The CDR technique is simple. In a darkened classroom, the squad leader stands before a six-foot screen. The video he is watching is as tall as he is, and the virtual sounds of combat echo from several surround-sound speakers. A company officer or non commissioned officer gives a succinct order, accompanied by a map, which is also provided in hard copy. As the squad leader proceeds on the mission, with the video showing combat scenes, one or two officers or non-commissioned officers request urgent instructions while acting as his fire team leaders. At the same time, his platoon commander, or company commander, is on a handheld radio requesting situation reports. Supporting arms are on-call.

The scenarios vary: Civilian refugees plead for water and offer vague intelligence; angry mobs block the fire teams; stray shots ring out; a deadly sniper opens up; a fortified building holds an enemy platoon; a bridge must be seized; enemy attack out of the darkness and rain, etc. In a half-hour of one scenario, the squad leader must make 15 to 30 rapid-fire decisions while issuing both orders and situation reports up and down the chain of command.

There is a plan to this multimedia madness. The squad leader is constantly reading his map and adjusting to events (Situation Awareness), directing his fire teams (Command), and providing information and seeking support (Coordination). Meanwhile, his own company officers are able to assess both the squad leader's level of training and decision making capabilities against a battlefield scenario.

The Combat Decision Range is running to good reviews from both the squad leaders and the company staffs. Each scenario takes 30 to 45 minutes to complete. Very quickly, the company can assess the level of squad leader training and capability. Under stress, many show difficulty reading grid coordinates even though they exhibited little prior problem. Others lose track of the battle while focusing on casualties. Few are adept at the use of supporting arms, although familiarity and confidence improves dramatically with practice. Reporting to higher headquarters is rarely adequate and information reported often lacks significant intelligence-related information. Radio traffic is often garbled. These are but a few examples.

The CDR employs a commercially available computer of 150 MHz or faster using CD-ROMs. The regiments have been successful in employing a lance corporal with one day of training as an operator. The company officers and NCOs were trained for two days; then they became the facilitators for their company.

The CDRs stress situation awareness, internal unit command, and external coordination. This situational awareness training requires the squad leaders to identify location — both his and his fire teams — through map reading and the use of a commercially available GPS. The command and coordination require the means to communicate, which is demonstrated through the use of commercially available 14 channel squad

radios. The intent is for the squad leader to know at all times where he is and to be able to communicate effectively with his team leaders and platoon commander. Each commercial GPS has its own 'white board' map, and the radios have multiple frequencies.

Most squad leaders, after the training, are noticeably more confident in their abilities. They are truly excited about their progress so far, although not without many hard lessons learned. The point is that the tempo of operations appears to be increasing dramatically. The result could be similar to the "West Coast Offense" in football. The San Francisco 49ers do nothing other teams do not do, but they execute faster, and once they have momentum, they build on it, generating an overwhelming tempo which other teams cannot match. Similarly, more confident, experienced, and better-equipped squad leaders add a dimension to maneuver warfare. The CDR provides a valuable inexpensive tool for training small unit leaders to execute quickly and with confidence. Once that happens, the operational level of war quickens — not because those in the rear think they know more — but rather because the senior commanders can seize and exploit the momentum gained by the initiative of confident small unit leaders.

The goal is to develop a full set of scenarios equivalent to those of the New York City Fire Department, so that when Marines deploy to the equivalent of our next "fire," every Marine squad leader will be able to say: "I have seen this sort of situation before. I can react fast, because I have confidence in myself and my ability to direct my Marines."

The underlying premise behind the CDR is that decisionmaking under stress depends on matching the crisis at hand with a mental image of a similar case – learning based on experience. When the direct experience is lacking, the CDR substitutes combat cases that impart experience. The analogy is to the flight trainer, where a pilot learns by crashing and walking away unhurt, but vividly remembering what he did wrong.

Theoretically, the technique can be applied across a variety of subject areas. Technically, the technique resembles making a movie. It is an art more than a skill. If the scenario chosen – the plot – is not credible, or if the actions are stilted, or if the technical military systems are incorrect, or if the facilitator stumbles, then the CDR will fail. GAMA proceeds by identifying the lessons to be imparted, designing a combat situation which will impart the lessons, selecting a venue for filming, writing a story line and a script, assembling actors (often marines), filming the scenarios, digitizing the results, adding animation, sound explosions, night shadings, etc., writing both a computer program (in Multimedia Director) and a text for the facilitators, selecting key strokes to match the action scenes, burning a CD master, debugging, practicing, changing and finally 'going public'. It's hard work, but it is fun when the squad leader gets so involved he forgets where he is and starts screaming for his supporting arms or for his fire teams to respond more quickly.

*Using Guidelines to Constrain Interactive Case-Based HTN Planning**

Héctor Muñoz-Avila †‡, Daniel C. McFarlane ‡, David W. Aha ‡, Len Breslow ‡,
James A. Ballas ‡, & Dana S. Nau †

† Department of Computer Science — University of Maryland
‡ Navy Center for Applied Research in AI — Naval Research Laboratory

Abstract.

This paper describes *HICAP*, a general-purpose, interactive case-based plan authoring architecture that can be applied to decision support tasks to yield a hierarchical course of action. It integrates a hierarchical task editor with a conversational case-based planner. HICAP maintains both a task hierarchy representing guidelines that constrain the final plan and the hierarchical social organization responsible for these tasks. It also supports bookkeeping, which is crucial for real-world large-scale planning tasks. By selecting tasks corresponding to the hierarchy's leaf nodes, users can activate the conversational case-based planner to interactively refine guideline tasks into a concrete plan. Thus, HICAP can be used to generate context sensitive plans and should be useful for assisting with planning complex tasks such as noncombatant evacuation operations. We describe an experiment with a highly detailed military simulator to investigate this claim. The results show that plans generated by HICAP were superior to those generated by alternative approaches.

1 Introduction

Planning a course of action is difficult, especially for large hierarchical organizations (e.g., the U.S. Navy) that assign tasks to elements (i.e., groups or individuals) and constrain plans with guidelines (e.g., doctrine). In this context, a concrete plan must adhere to guidelines but should also exploit organizational knowledge where appropriate (e.g., standard procedures for solving tasks, previous experiences when reacting to unanticipated situations). Case-based reasoning (CBR) can be used to capture and share this knowledge.

In large planning environments, automatic plan generation is neither feasible nor desirable because users must observe and control plan generation. We argue that, rather than relying on an automatic plan generator, users prefer and can greatly benefit from the assistance of an intelligent plan formulation tool with the following characteristics:

- *Guidelines-driven*: Uses guidelines to constrain plan generation.

* Presented at the 1999 International Conference on Case-Based Reasoning

- *Interactive*: Allows users to edit any detail of the plan.
- *Provide Case Access*: Indexes plan segments from previous problem-solving experiences, and retrieves them for users if warranted by the current planning scenario.
- *Perform Bookkeeping*: Maintains information on the status of and relations between task responsibilities and individuals in the organizational hierarchy.

This paper describes HICAP, a general-purpose plan formulation tool that we designed to embody these characteristics.[1] HICAP (Hierarchical Interactive Case-Based Architecture for Planning) integrates a task decomposition editor, HTE (Hierarchical Task Editor) (Muñoz-Avila et al., 1998), with a conversational case-based planner, NaCoDAE/HTN. The former allows users to edit and select guidelines for refinement, while the latter allows users to interactively refine plans encoded as hierarchical task networks (HTNs) (Erol et al., 1994). Refinements use knowledge of previous operations, represented as cases, to augment or replace standard procedures.

The following sections describe the application task, HICAP's knowledge representation, its architecture, its empirical evaluation, and a discussion of related work.

2 Planning Noncombatant Evacuation Operations

Noncombatant evacuation operations (NEOs) are conducted to assist the U.S.A. Department of State (DoS) with evacuating noncombatants, nonessential military personnel, selected host-nation citizens, and third country nationals whose lives are in danger from locations in a host foreign nation to an appropriate safe haven. They usually involve the swift insertion of a force, temporary occupation of an objective (e.g., an embassy), and a planned withdrawal after mission completion. NEOs are often planned and executed by a Joint Task Force (JTF), a hierarchical multi-service military organization, and conducted under an American Ambassador's authority. Force sizes can range into the hundreds and involve all branches of the armed services, while the evacuees can number into the thousands. More than ten NEOs were conducted within the past decade. Publications describe NEO doctrine (DoD, 1994), case studies (Siegel, 1991; 1995), and more general analyses (e.g., Lambert, 1992).[2]

The decision making process for a NEO is conducted at three increasingly-specific levels: strategic, operational and tactical. The strategic level involves global and political considerations such as whether to perform the NEO. The operational level involves considerations such as determining the size and composition of its execution force. The tactical level is the concrete level, which assigns specific resources to specific tasks.

[1] Implemented in Java 2, the HICAP applet can be run from www.aic.nrl.navy.mil/hicap. HICAP was introduced in (Muñoz-Avila et al., 1999), which did not include the evaluation described here.

[2] See www.aic.nrl.navy.mil/~aha/neos for more information on NEOs.

JTF commanders plan NEOs in the context of doctrine (DoD, 1994), which defines general guidelines (e.g., chain of command, task agenda) for designing strategic and operational plans; tactical considerations are only partly addressed. Doctrine is abstract; it cannot account for the detailed characteristics of specific NEOs. Thus, JTF commanders must always adapt doctrine to a NEO's specific needs, and do so in two ways. First, they dynamically modify doctrinal guidance by eliminating irrelevant planning tasks and adding others, depending on the operation's needs, resource availabilities, and relevant past experiences. For example, although NEO doctrine states that a forward command element must be inserted into the evacuation area with enough time to plan the insertion of the JTF's main body, this is not always feasible (e.g., in *Operation Eastern Exit*, combined elements of the JTF were inserted simultaneously due to the clear and imminent danger posed to the targeted evacuees (Siegel, 1991)). Second, they employ experiences from previous NEOs, which complement doctrine by suggesting tactical refinements suitable for the current NEO. For example, they could draw upon their previous experiences to identify whether it is appropriate to concentrate the evacuees in the embassy or to plan for multiple evacuation sites.

3 Knowledge Representation

Because HTNs are expressive representations for plans, we used a variant of them in HICAP. A HTN is a set of tasks and their ordering relations, denoted as $N = \langle \{T_1, \ldots, T_m\}, \prec \rangle$ ($m \geq 0$). The relation $\prec$ has the form $T_i \prec T_j$ ($i \neq j$), and expresses temporal restrictions between tasks.

Problem solving with HTNs occurs by applying *methods* to decompose or reduce tasks into subtasks. Each method has the form $M = \langle l, T, N, P \rangle$, where l is a label, T is a task, N is a HTN, and $P = \langle p_1, \ldots, p_k \rangle$ a set of preconditions for applying M. When P is satisfied, M can be applied to a task T to yield N.

HICAP's HTN consists of three task types. First, *non-decomposable* tasks are concrete actions and can occur only at a network's leaves. Next, *uniquely decomposable* tasks correspond to guideline tasks (e.g., doctrine), and are solved by unconditional methods ($k = 0$). Finally, *multi-decomposable* tasks must be solved in a specific problem-solving context.

There are two sources of knowledge for decomposing multi-decomposable tasks: standard operating procedures (SOPs) and recorded episodes. SOPs describe how to reduce a task in a typical situation. Recorded episodes describe how tasks were reduced in situations that are not covered by SOPs. In our representation, SOPs and recorded episodes are both represented as methods and we loosely refer to both as *cases*. However, there is an important difference in the way SOPs and recorded episodes are applied. To apply a SOP to reduce a task, all its preconditions must be matched because they are typically rigid in their use. In contrast, recorded episodes can be applied to reduce a task even if some of its preconditions are not satisfied.

When reducing a task *T*, HICAP retrieves all cases (i.e., standard procedures and recorded episodes) that can decompose *T*. If all the preconditions of a SOP are met, then it should be used to decompose *T*. Otherwise, a case corresponding to the most similar episode should be used. For example, standard NEO procedures state that the evacuees must be concentrated in the embassy prior to troop deployment, but this is not always possible: in Operation Eastern Exit, only some of the evacuees were concentrated in the embassy after the Joint Task Force was deployed. This occurred because escorted transports were not available to gather these evacuees, who were unable to reach the embassy due to the dangerous conditions in the surrounding areas (Siegel, 1991). Likewise, the evacuees of *Operation Sharp Edge* (Sachtleben, 1991) were concentrated in several places, forcing multiple separate evacuations.

4 HICAP: An Interactive Case-Based Planner

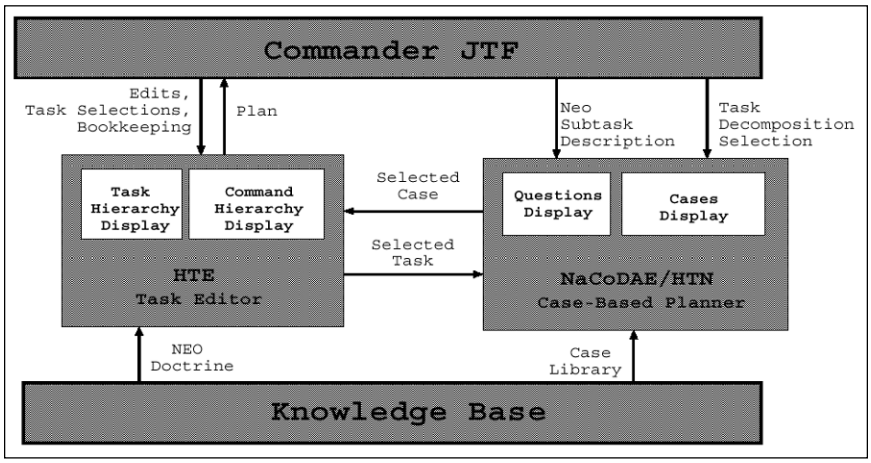


Fig. 1. The HICAP architecture.

HICAP (Figure 1), which integrates HTE with NaCoDAE/HTN, inputs a HTN describing the guidelines for an application along with a set of cases for each multi-decomposable subtask. It displays all uniquely decomposable tasks as expanded. Under user control, HICAP outputs an elaborated HTN whose leaves are concrete actions as specified by case applications and manual edits. In this way, HICAP satisfies the requirements stated in Section 1. First, all plans formulated using HICAP are in accordance with the guidelines or user modifications of them. Second, HICAP supports interactive task editing and triggers conversations for tasks that can be decomposed by case application. Third, it incorporates knowledge from previous problem solving episodes as cases, which serve as task decomposition alternatives. Finally, it allows users to visually check that all tasks are assigned to JTF elements, and to record/update their completion status.

4.1 Hierarchical Task Editor

In complex environments where dozens of tasks must be performed by many people, tracking the completion status for each task can be challenging. For example, during the NEO Operation Eastern Exit, the task to inspect evacuees prior to embarkation was not assigned (Siegel, 1991). One of the evacuees produced a weapon during a helicopter evacuation flight. Although it was immediately confiscated, this oversight could have resulted in tragedy and illustrates the difficulties with planning NEOs manually.

The Hierarchical Task Editor (HTE) (Muñoz-Avila et al., 1998) serves HICAP as a bookkeeping tool to track the status of each task. HTE inputs a knowledge base consisting of a HTN task agenda, its ordering relations, the organization's command hierarchy, and an assignment of tasks to command elements. It allows users to edit the knowledge base and select tasks to refine by invoking NaCoDAE/HTN, thus tailoring the plan to the particular circumstances of the current NEO.

For our NEO application, we encoded a HTN to capture critical planning doctrine (DoD, 1994), yielding 200+ tasks and their ordering relations. Next, we used this doctrine to elicit the JTF command hierarchy commonly used in NEO operations. Finally, we elicited relations between tasks and the JTF elements responsible for them. The mapping of tasks to command elements is many-to-one. Figure 2 displays (left) the top level tasks that, according to doctrine, must be performed during a NEO and (right) the elements in the JTF responsible for them.

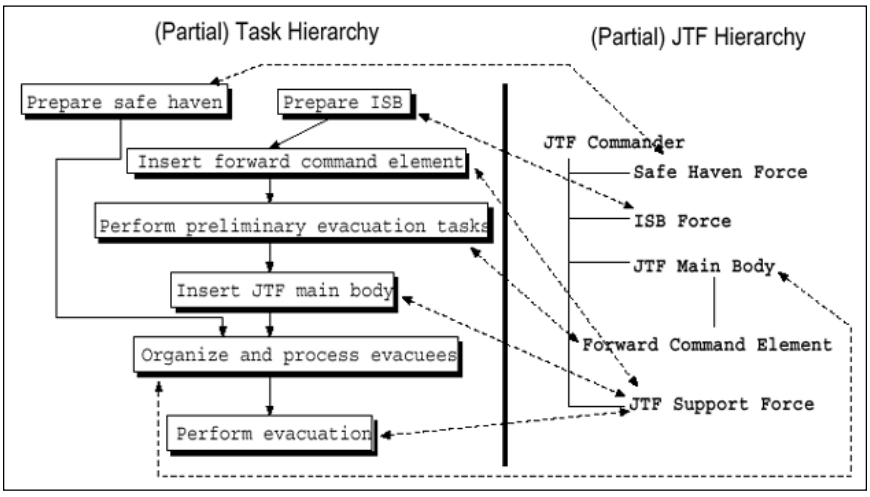


Fig. 2. Top level NEO tasks and their assignment to JTF command elements (double arrows denote assignments; arrows denote task orderings; ISB = intermediate stage base).

4.2 Conversational Task Decomposer

NaCoDAE/HTN, an extension of the NaCoDAE conversational case retrieval tool (Aha & Breslow, 1997; Breslow & Aha, 1997), supports HTN planning by allowing users to refine selected tasks into concrete actions. When given a task T to refine by HTE, NaCoDAE/HTN uses T as an index for initial case retrieval and conducts an interactive *conversation*, which ends when the user selects a case $C = \langle I, T, N, P \rangle$. Network N is then used to decompose T (i.e., into a set of subtasks represented as T's child nodes). Subtasks of N might themselves be decomposable, but non-decomposable tasks corresponding to concrete actions will eventually be reached. Task expansions are displayed by HTE.

During conversations, NaCoDAE/HTN displays the labels of the top-ranked cases that can decompose the selected node and the top-ranked questions fromthese cases whose answers are not yet known for the current situation. The user can select and answer any displayed question; question-answer pairs are used to compute the similarity of the current task to its potential decomposition methods (cases). Cases are ranked according to their similarity to the current situation (Aha & Breslow, 1997), while questions are ranked according to their frequency among the top-ranked cases. Answering a question modifies the case and question rankings. A conversation ends when the user selects a case for decomposing the current task.

Some of the displayed cases are standard procedures; they can only be selected to decompose a task after all of their questions have been answered and match the current planning scenario. That is, preconditions of the standard procedures must match before they can be applied. In contrast, cases based on previous experiences can be selected even if some of their questions have not been answered, or if the user's answers differ. Thus, they support partial matching between their preconditions and the current planning scenario.

5 Example: NEO Planning

During NEO planning, users are first shown the tasks corresponding to doctrine, and revise them as needed. They can expand any task and view its decomposition. In Figure 3, the user has selected the *Select assembly areas for evacuation & Evacuation Control Center sites* task, which is highlighted together with the command element responsible for it.

Standard procedure dictates that the embassy is the ideal assembly area. However, it is not always possible to concentrate the evacuees in the embassy.Alternative methods can be considered for decomposing this task. When the military planner selects this task, HICAP displays the alternatives and initiates a NaCoDAE/HTN conversation (see Figure 4 (top)).

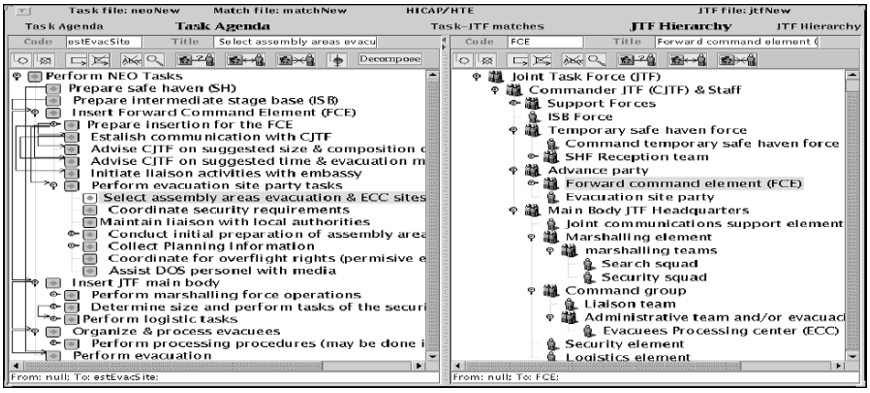


Fig. 3. HTE: Task agenda (left) and command hierarchy (right) displays (arrows denote ordering constraints).

If the user answers *Are there any hostiles between the embassy and the evacuees?* with *uncertain*, a perfect match occurs with the case labeled "Handle situation in which it is unknown whether hostiles are present," which now becomes the top-ranked case (Figure 4 (bottom)). Figure 5 (left) shows the decomposition when selecting this case to decompose this task in HTE; two new subtasks are displayed, corresponding to this case's decomposition network. *Send unmanned air vehicle to …* is a non-decomposable concrete action. If the user tells HICAP to decompose *Determine if hostiles are present*, HICAP will initiate a new NaCoDAE/HTN dialogue (Figure 5, right).

The user can again prompt a dialogue by selecting the *The UAV detects hostiles* alternative and decomposing its subtasks. This cycle, in which HICAP displays alternatives and the user answers questions and selects an alternative, continues until non-decomposable tasks (i.e., concrete actions) are reached, which form part of the final plan.

6 The Case-Based Planning Cycle in HICAP

The case-based planning component of HICAP, NaCoDAE/HTN, typically performs three steps: retrieval, revise, and retain. As illustrated in Section 5, the adaptation process can be viewed as embedded in the conversational retrieval process.

6.1 Case Retrieval

We previously explained that, during a conversation, cases are ranked according to the proportion of their question-answer pairs that match the current scenario. More specifically, a case c's similarity score is computed with a query q using

$$\text{case\_score}(q,c) = \frac{\text{num\_matches}(q,c) - \text{num\_mismatches}(q,c)}{\text{size}(c)} \qquad (1)$$

where num_matches(q,c) (num_mismatches(q,c)) is the number of matches (mis-

matches) between the states (i.e., $\langle q,a \rangle$ pairs) of q and c, and size(c) yields the number of $\langle q,a \rangle$ pairs in c's state.[3]

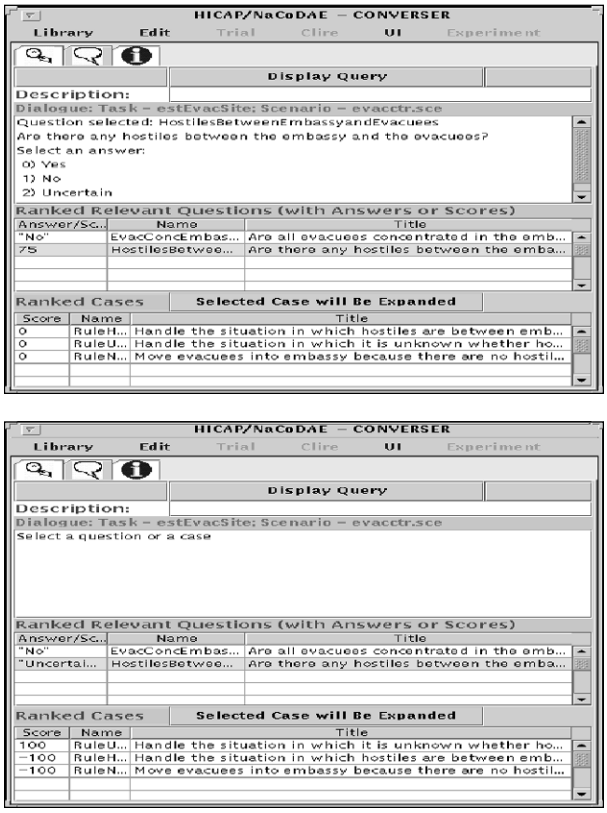


Fig. 4. NaCoDAE/HTN: Before (top) and after (bottom) answering a question. The top window lists possible answers to selected questions, while the lower windows display the ranked questions and cases.

6.2 Case Revision

The user can revise the current solution by editing the task hierarchy (in HTE) and by selecting alternative cases during a NaCoDAE/HTN conversation. In addition, the user can revise their answers to previously selected questions, which can modify case rankings. Although, revising an answer does not alter the plan automatically, the new ranks may prompt the user to change their case selection, which in turn may prompt additional edits to the task hierarchy.

This ability to explore alternatives (i.e., "what-if" analyses) is particularly important in NEO planning for two reasons. First, military planners typically plan for a main course of actions and for contingency alternatives should certain key events occur. These events may trigger changes to answers and case rankings, thus helping the user

[3] Matching for numeric-valued questions is implemented using a suitable partial matching routine, but we focus on symbolic and boolean questions here.

formulate these alternatives. Second, NEO planning is dynamic in nature and the user must be able to replan due to unforeseen contingencies.

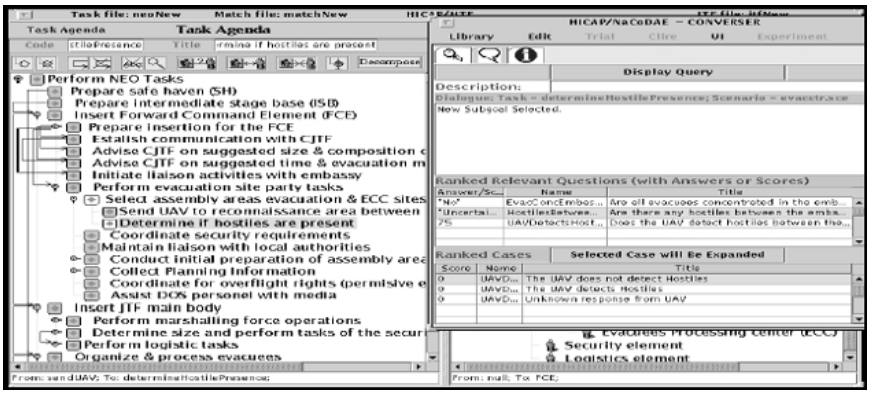


Fig. 5. HICAP's interface after selecting the *determine hostile presence* task.

6.3 Case Retention

NaCoDAE incorporates an approach introduced by Racine and Yang (1997) for maintaining case libraries. It evaluates whether any case "subsumes" another case (i.e., whether its question-answer pairs are a proper subset of the question-answer pairs of another case). If so, the subsuming case will block the subsumed case from being retrieved. A case library evaluation function alerts the user to all such pairs of cases in the case library. The user can then decide which of the two cases to revise and/or delete.

7 Empirical Validation

An experiment was run to test HICAP's effectiveness in choosing successful plans for an example NEO subtask. In particular, we showed the importance of considering episodic records over standard procedures. A larger experiment, demonstrating the capability of HICAP to generate a complete NEO plan, is currently under development.

Two researchers performed the experiment: one operated a military simulator while the other operated HICAP. A strict blind was imposed to ensure that the HICAP user had no advance knowledge concerning the simulated hostile forces, and had to take appropriate, realistic actions to acquire this knowledge. This tests HICAP's utility for planning under realistic situations where decision makers have uncertain information about the state of the world. We hypothesized that HICAP would allow users to choose a relatively successful plan from among known tactical options. HICAP's strategy was evaluated versus three other planning strategies: *random choice*, *heuristic choice*, the *most frequently used plan* used in previous NEOs. Because their definitions require explaining the scenario, we define them in Section 7.3.

7.1 The ModSAF Simulation System

We used Marine Corps SAF (MCSF), a variant of ModSAF (Modular Semi-Automated Forces), to evaluate the quality of NEO plans elicited using HICAP. ModSAF, developed by the U.S.A. Army to inject simulated auxiliary forces into training exercises, has been deployed to simulate real-world military scenarios (Ceranowicz, 1994). It is a finite state simulation with modular components that represent individual entities and parts of entities. For example, a simulated tank would have physical components such as a turret. It would also have behavioral components that represent its nominal tasks such as move, attack, target, and react to fire. Certain 3D aspects are also represented (e.g., terrain elevation, trees and vegetation, rivers, oceans, atmospheric conditions) that can affect sensory and movement behavior. The realism of ModSAF/MCSF simulations is sufficient for training exercises.

Figure 6's MCSF snapshot displays a simulated American embassy, a host country government compound, and some simulated objects. For example, a simulated transport helicopter is positioned at the heliport within the embassy site.

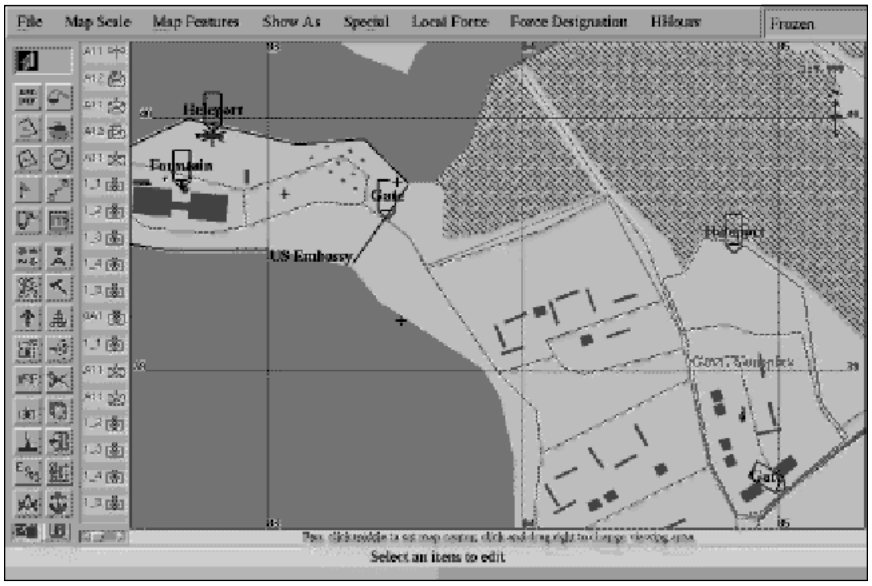


Fig. 6. A MCSF snapshot.

MCSF is a non-deterministic simulator that models several sources of stochastic variation. Some events are determined by a random number generator; others are highly sensitive to the initial startup conditions. MCSF simulates the behavior of military units in context as they follow given tactical orders. Therefore, MCSF can simulate simplified NEO subtasks in which a single planning decision determines tactical orders.

7.2 Experimental Setup

We created a NEO subtask scenario for this evaluation concerning how to move 64 evacuees from a meeting site to an embassy. The meeting site was at a crossroads in an uninhabited area outside but nearby the embassy's city. Evacuees had to be transported (8 per vehicle) through this undeveloped area, which had heavy tree cover, and out through the city to the embassy. Evacuees had to pass near a local government complex to enter the embassy grounds. This NEO context requires only a single tactical plan decision with four distinct choices:

1. Land evacuation using 8 armored trucks
2. Land evacuation using 8 armored trucks with an escort of 8 tanks
3. Air evacuation using 8 transport helicopters
4. Air evacuation using 8 transport helicopters with an escort of 8 attack helicopters

The kind of military units used in the simulation are typical of those available to the Marine Expeditionary Units that frequently perform NEO's. A detailed terrain database of the Camp Lejeune (North Carolina, U.S.A.) area was chosen to simulate the environment. We chose this location because Marine Expeditionary Units train there for NEOs.

Two scenarios were defined that were identical except for the type of hostile forces. All hostiles were two-person dismounted infantry teams. Hostile teams in both scenarios were armed with two automatic rifles and a portable missile launcher. Each scenario included only one type of missile for hostile teams (i.e., either anti-tank missiles or anti-air missiles, but not both). These types of infantry teams, positioned in an urban environment, are typical of the kinds of hostile forces encountered in real NEO's. The positions of the hostile teams were the same for both scenarios and selected to ensure that the opposing forces will meet.

All four plan options were simulated ten times for each of the two scenarios. This resulted in 80 (2 scenarios × 4 plan choices × 10 simulations) total MCSF runs. Each of the eight plan-and-scenario combinations was repeated ten times because MCSF is non-deterministic. For example, slight differences produced by MCSF's stochastic movement models yield strikingly different formations of friendly units when they first encountered the hostile teams. These differences can often yield drastically different simulated battle outcomes.

	Scenario 1 (anti-tank)						Scenario 2 (anti-air)					
Tactical Plans	**Evacuees**		**Friends**		**Hostiles**		**Evacuees**		**Friends**		**Hostiles**	
Land	6.4	5.1	0.8	0.6	5.5	1.3	0		0		4.2	0.8
Land/Escort	3.2	10.1	7.4	1.5	6.5	1.8	0		0		7.6	0.6
Air	56.0	9.2	7.0	1.2	0		64.0	0.0	8.0	0.0	0	
Air/Escort	0		0.8	1.5	8.0	0.0	20.0	18.6	6.3	4.4	5.7	2.9

Table 1. Summaries of casualties, to individual evacuees and military teams (mean & standard deviation), averaged over 80 MCSF simulations.

The HICAP user had no knowledge of the scenarios being tested; scenario information was gradually extracted through the questions prompted by NaCoDAE/HTN. That is, case-based planning was done with incomplete information about the world. Furthermore, the effects of actions were uncertain; the only way to learn the effects of an action was to actually execute it. This contrasts with traditional planning approaches that assume an action's effects are known a priori (Fikes and Nilsson, 1971).

7.3 Alternative Planning Strategies

HICAP's decision-making performance was compared with three baseline strategies. First, *random choice* simply averaged the results of all four planning choices. Second, *heuristic choice* always sent an escort, and its results were the average of the choices that include escorts. Finally, the *most frequently used* plan strategy for this subtask in recent NEOs (i.e., conducted during the past decade) was to move evacuees using escorted land vehicles.

7.4 Results

Table 7.4 summarizes the casualty results for the 80 total simulations, which each required approximately 15 minutes to run. The success measures were taken from the U.S.A. Navy's Measures of Effectiveness (MOE's) published in the Universal Naval Task List. Recommended MOEs are specified for evaluating each kind of military operation. There are several MOE's for the tactical aspects of NEO's, but only three were chosen as most important for evaluating the results of this experiment: (1) the number of evacuees safely moved, (2) the number of casualties to friendly forces, and (3) the number of casualties to hostile forces.

HICAP did not choose the same tactical plan for both scenarios. For the first (anti-tank) scenario, it chose to move the evacuees by helicopter with an attack helicopter escort. For the second (anti-air) scenario, it chose to move evacuees by armored truck with a tank escort.

HICAP's conversational case-based planning method was evaluated by comparing the success of its chosen plans to plans chosen by the other three plan selection strategies. Figure 7 compares the effectiveness of these four strategies. Overall, HICAP selected plans of higher quality than the other strategies because its plan selection decisions are tailored to the characteristics of each scenario.

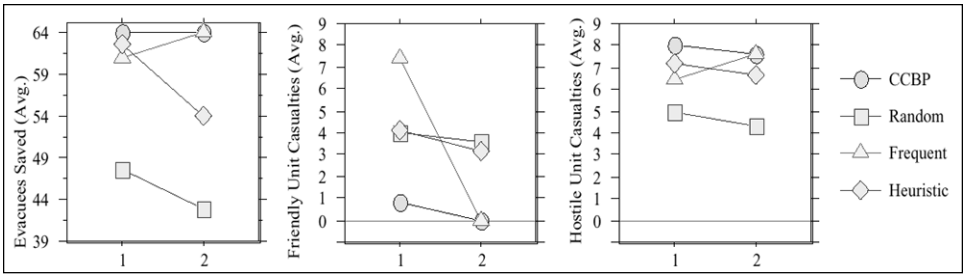


Fig. 7. Comparison of plan selection strategies using Navy MOEs for NEOs.

8 Related Research

Case-based planning (CBP) has been extensively researched (Bergmann et al., 1998). Our research is closely related to studies on hierarchical CBP (e.g., Kambhampati, 1993; Bergmann & Wilke, 1995; Branting & Aha, 1995). HICAP differs from these other approaches in that it includes the user in its problem solving loop. This is particularly important for applications like NEO planning, where completely automated tools are unacceptable. MI-CBP (Veloso et al., 1997) uses rationale-directed CBP to suggest plan modifications in a mixed-initiative setting, but does not perform doctrine-driven task decomposition.

Some researchers have used CBP with HTNs for military tasks. For example, Mitchell (1997) used integrated CBP to select tasks for a tactical response planner. NEO planning requires that each task be addressed — no choice is involved — and we use CBP to instead choose *how* to perform a task. HICAP's interactions instead focus on retrieval rather than plan adaptation and learning.

9 Conclusion and Future Work

The HICAP case-based planner helps users to formulate a course of action for hierarchical tasks. It is the first tool to combine a task guideline decomposition process with CBR to support interactive plan formulation. It yields plans that benefit from previous experiences and conform to predefined guidelines. HICAP also supports experience sharing, thus allowing planners to exploit knowledge from other planning experts. These design characteristics enhance HICAP's acceptance by military planning personnel.

We are currently integrating HICAP with a generative HTN planner that can evaluate numeric expressions (Nau et. al., 1999), which is particularly important for NEOs because decisions often depend on resource capability and availability (i.e., determining whether a helicopter requires in-flight refueling for a given mission). HICAP will serve as the plan formulation component for the Space and Naval Warfare Systems Command's Interactive Decision Support (IDS) system. When completed, IDS will perform distributed NEO plan formulation, execution, monitoring, and replanning.

Our collaborative research with IDS partners will focus on associating temporal durations with tasks, developing a resource tracking module (i.e., to solve resource conflicts), implementing a strategy for justifying case rankings, integrating HICAP with a powerful dynamic planner (i.e., SIPE-2 (Wilkins, 1998)), and integrating existing GUIs for plan authoring. We will also investigate methods for performing information gathering in HICAP using a planning approach (e.g., Carrick et al., 1999).

Acknowledgements

Thanks to ONR Program Managers Michael Shneier and Paul Quinn, and Program Officer Lt. Cdr. Dave Jakubek, for their encouragement throughout this project. This research was supported by grants from the Office of Naval Research, the Naval Research Laboratory, and the Army Research Laboratory. Many thanks to members of the Center for Naval Analyses and ONR's Naval Science Assistance Program for their guidance and support. And thanks to our ICCBR-99 reviewers for their thoughtful suggestions, which improved this paper.

References

Aha, D. W., & Breslow, L. A. (1997). Refining conversational case libraries. *Proceedings of the Second International Conference on CBR* (pp. 267–278). Providence, RI: Springer.

Bergmann, R., Muñoz-Avila, H., Veloso, M., Melis, E. (1998). Case-based reasoning applied to planning tasks. In M. Lenz, B. Bartsch-Spoerl, H.-D. Burkhard, & S. Wess (Eds.) *CBR Technology: From Foundations to Applications*. Berlin: Springer.

Bergmann, R. & Wilke, W. (1995). Building and refining abstract planning cases by change of representation language. *Journal of AI Research*, *3*, 53–118.

Branting, L. K., & Aha, D. W. (1995). Stratified case-based reasoning: Reusing hierarchical problem solving episodes. *Proceedings of the Fourteenth International Joint Conference on AI* (pp. 384–390). Montreal, Canada: Morgan Kaufmann.

Breslow, L., & Aha, D. W. (1997). *NaCoDAE: Navy Conversational Decision Aids Environment* (TR AIC-97-018). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Carrick, C., Yang, Q., Abi-Zeid, I., & Lamontagne, L. (1999). Activating CBR systems through autonomous information gathering. To appear in *Proceedings of the Third International Conference on Case-Based Reasoning*. Munich, Germany: Springer.

Ceranowicz, A. (1994). *Modular Semi-Automated Forces. Proceedings of the Winter Simulation Conference of the ACM* (pp. 755–761). New York, NY: IEEE.

DoD (1994). *Joint tactics, techniques and procedures for noncombat evacuation operations* (Joint Report 3-07.51, Second Draft). Washington, DC: Department of Defense.

Erol, K., Nau, D., & Hendler, J. (1994). HTN planning: Complexity and expressivity. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 1123–1128). Seattle, WA: AAAI Press.

Fikes, R.E., & Nilsson, N.J. (1971). Strips: A new approach to the application of theorem proving in problem solving. *Artificial Intelligence*, *2*, 189–208.

Kambhampati, S. (1994). Exploiting causal structure to control retrieval and refitting during plan reuse. *Computational Intelligence*, *10*, 213–244.

Lambert, Kirk S. (1992). *Noncombatant evacuation operations: Plan now or pay later* (Technical Report). Newport, RI: Naval War College.

Mitchell, S.W. (1997). A hybrid architecture for real-time mixed-initiative planning and control. *Proceedings of the Ninth Conference on Innovative Applications of AI* (pp. 1032–1037). Providence, RI: AAAI Press.

Muñoz-Avila, H., Breslow, L.A., Aha, D.W., & Nau, D. (1998). *Description and functionality of HTE* (TR AIC-98-022). Washington, DC: NRL, NCARAI.

Muñoz-Avila, H., Aha, D.W., Breslow, L. & Nau, D. (1999). HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. To appear in *Proceedings of the Ninth National Conference on Innovative Applications of Artificial Intelligence*. Orlando, FL: AAAI Press.

Nau, D. S., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple Hierarchical Ordered Planner. To appear in *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Stockholm, Sweden: Morgan Kaufmann.

Racine, K., & Yang, Q. (1997). Maintaining unstructured case bases. *Proceedings of the Second International Conference on CBR* (pp. 553–564). Providence, RI: Springer.

Sachtleben, G.R. (1991). Operation Sharp Edge: The Corps MEU (SOC) program in action. *Marine Corps Gazette*, *11*, 76–86.

Siegel, A.B. (1991). *Eastern Exit: The noncombatant evacuation operation (NEO) from Mogadishu, Somalia, in January 1991* (TR CRM 91-221). Arlington, VA: Center for Naval Analyses.

Siegel, A.B. (1995). *Requirements for humanitarian assistance and peace operations: Insights from seven case studies* (TR CRM 94-74). Arlington, VA: CNA.

Veloso, M., Mulvehill, A.M., & Cox, M.T. (1997). Rationale-supported mixed-initiative case-based planning. *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence* (pp. 1072{1077). Providence, RI: AAAI Press.

Wilkins, D.E. (1998). *Using the SIPE-2 planning system: A manual for Version 5.0* (Working Document). Menlo Park, CA: Stanford Research International, Artificial Intelligence Center.

Towards an Effective Information Sharing System: Shared Net

Dr. Thomas McVittie
Jet Propulsion Laboratory
California Institute of Technology

1 Introduction

Today's decision-maker (whether they are a corporate executive, military commander, or spacecraft mission planner) is faced by almost insurmountable challenges. Ever increasing amounts of data are available from a bewildering number of sources such as: overhead imagery, sensor nets, telemetry systems, intelligence assets, and in-situ personnel. The pace at which a decision-maker must make critical choices has decreased from days to minutes. The decision-maker is expected to manage multiple simultaneous (and often conflicting) dynamic missions rather than a single monolithic and statically planned mission. Additionally, the classical hierarchical decision making approach where decision-makers made all the decisions based on input from a few individuals is rapidly giving way to a distributed decision-making process where decisions are made simultaneously at all levels within an organization – often by people who have never met.

Decision-makers are increasingly being overwhelmed by data, but remain starved for information. They are surrounded by mountains of data, but don't have the resources to turn the data into insightful information they need to make decisions. Also, once information is discovered, we have only primitive tools to share that information among a heterogeneous collection of systems and humans that may themselves be widely distributed.

This paper discusses one approach being jointly investigated by NASA's Jet Propulsion Laboratory and the US Marine Corps (USMC) to efficiently gather and share information among people and systems. The approach advocates the use of collaborative agents as a decision support tool, and an object sharing system providing a powerful mechanism for both representing and sharing information and data.

The paper is organized as follows. Section 2 discussed methods of sharing information. It contrasts a message passing approach (widely used by the military, NASA, and industry) to the use of an object sharing system. Section 3 briefly introduces IMMACCS, an experimental implementation of this approach developed for the USMC. Section 4 drills into the IMMACCS architecture and briefly discusses the component which implements the object sharing system – the Shared Net. Finally section 5 presents concluding remarks and acknowledgements.

2 Models of Sharing Information

Most organizations need to share information in order to make effective decisions. However, the specific mechanisms that are used to share information between users and systems vary greatly. For example, information may be exchanged between people using free text email messages, or between applications using a rigorously defined language and protocol.

In this section, we'll examine two methods for sharing and representing information with an eye towards how they support the decision-making process. The first approach is based on a commonly used message passing approach. This approach is widely used in both industry and government organizations. The second approach is based on an object sharing system that can greatly enhance our ability to represent and share information.

In order to contrast these two approaches, we'll use the situation depicted in Figure 2-1 below. Lets assume that we have a number of intelligence assets (a.k.a. spotters) monitoring an evolving situation involving the ABC insurgents. At some time n, one of our spotters reports that a woman, matching the description of the leader of the ABC insurgents and wearing a pink dress, was seen in a taxi heading eastbound on Main Street. At a later time, another spotter, located on the other end of town, reports that a woman wearing a pink dress was scene exiting a taxi and entering a building located at 123 Maple Street.

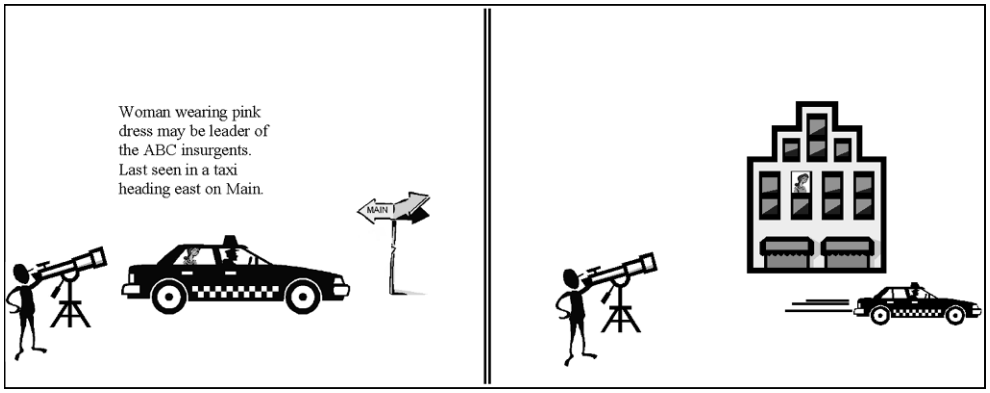


Figure 2-1 Data from the field

Based on this information we need to determine whether both reports reference the same woman. The information must be effectively communicated to others who will use this information to make decisions (e.g., to investigate the building further, etc.)

2.1 Message Passing Systems

In most DoD and industry, message passing is the primary mechanism used to exchange information between users and systems. For example, email is often used to convey complex concepts between users. Likewise, most software systems use internal data structures to represent some facet of the real world, and use structured messages to communicate some of that data to other systems. In most message based environments, information is fragmented across multiple systems and users (e.g., email boxes, databases, file formats, etc.) with each user and/or system maintaining only a slice of the corporate knowledge.

The format and contents of the message may be rigidly defined (as in a military POSREP message), or may be ad-hoc (say, an email message). Where automatic processing support is desired, the messages tend to be rigidly formatted and terms/values are well defined. However, where humans are the intended audience, messages tend to be free form.

Figure 2-1 depicts how our two messages might be processed in a typical command center.

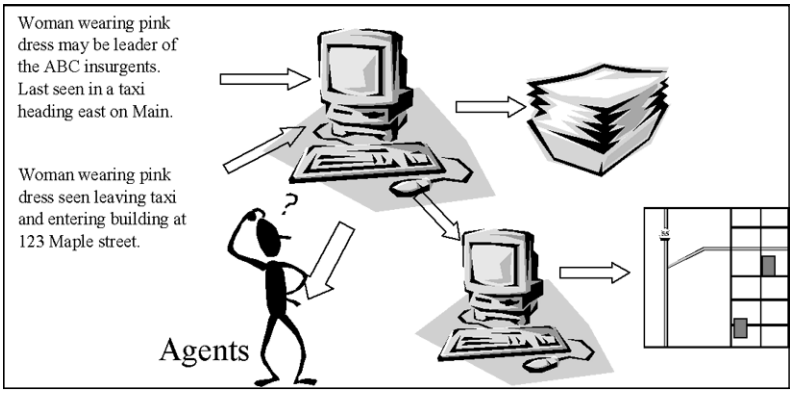


Figure 2-2 A message passing environment

First the messages are received by an automated processing center. Once received, the processing center may:

1) Parse the incoming message(s) for key words and route the message in its entirety to one or more individuals or desks. For example, the Automated Message Handling System (AMHS), used in most major DoD command centers, could be used to route messages containing the key word "insurgents" to the intelligence watch officer.

2) Extract data from the message (for example the location where the report was made) and display the message as an icon on a map. This is traditionally how

messages such as SALUTE and SPOT reports appear. In some instances, such as a POSREP (position report), the contents of the message is extracted and used to update the position of the reporting unit on the display map.

3) Simply store the incoming message and present it to a human for routing and disposition. This is typical of some command centers where all AUTODIN traffic is routed to a desk that reads the message and based on the reader's previous experience, routes it to the appropriate users.

The ability for the automated system to perform each of these activities is based on the message's format. Automated systems are best able to handle messages that have a rigid and well-defined format. For example, in order to display the message on a map, the automated system must be able to locate the part of the message that contains coordinate information. Further, the coordinate information must be in a well-understood format (e.g., latitude-longitude, or MGRS). However, humans are much better equipped to process natural language messages and infer structure and format.

Returning to Figure 2-1, we see that both messages have been routed to a user's inbox and have also been displayed as icons on the common map. At some point we trust that some combination of users and computer system will examine the two messages and determine whether or not the "lady in pink" in both messages is indeed the same lady. For example, a correlater might be used to determine whether a taxi could move from the first reported position to the second reported position within the time allowed. Similarly, a human could ask the spotters for more information about the reported lady, such as her hair color and height, that could be used to aid in determining whether the reports detail the same lady.

Once the relationship has been identified or disproved, it must be shared with other interested users. Receiving this new piece of information may also impact the processes and decisions of other users. For example, the fact that the leader of the ABC insurgents is located at 123 Maple St. may prompt the intelligence officer to investigate the building to determine whether it is an insurgent safe house or whether there is only a casual relationship between the building and the lady. If the new information is not shared, then each user receiving the messages must make the inference independently, and it is likely that some set of users will recognize the relationship while others will not.

Unfortunately, most message formats provide only limited tools for representing and sharing complex relationships with other users. For example, the SALUTE report does not contain the ability to relate the message to another message, nor does it have the capability to specify a complex relationship in anything other than free text. Therefore, its likely that the new information and relationship would be transmitted in the form of yet another message which would go through the same routing system

as the original messages. Users receiving the message must recognize that this new message is related to the first messages. Only by reading all three messages is a user able to construct a mental model of the situation that they can use to make decisions. This process is repeated for each individual receiving the messages. If the volume of messages is large, or if messages arrive frequently, users may have difficulty in maintaining a correct model of the situation.

Likewise, agent based decision support systems must rely on complex natural language processing to extract the information and relationships from free text messages. Additionally, they must possess a detailed understanding of the format and meaning (ontology) of the messages produced by each system. For example, they must understand that the messages produced by system A express coordinates in latitude/longitude, but that system B uses MGRS. Thus in order to reason about the data contained in the message, the agent must develop a translator for each message type it must handle. Further, the agent is responsible for "knowing" how to compare/convert the similar (but not identical) data provided in different messages (e.g., how to convert between latitude/longitude and MGRS). Thus, an agent needing to extract data from a large number of different message types must be very sophisticated. Unfortunately, these approaches often yield poor results and agents are rarely productive in this type of environment.

We need a better approach if we want to move from data sharing to information sharing.

2.2 Object Sharing Systems

Object Sharing Systems assume that all users and systems use a common object model to represent and exchange information about the real world. Objects are modeled after their real-world counterparts and contain a rich set of attributes. More importantly, the object model allows us to create relationships between objects which are immediately available to all other users and system.

To better understand the concept, we'll examine how the same two messages would be processed in an Object Sharing System (OSS). First we'll assume that the object model has been defined, and populated with a variety of different objects such as:

> Infrastructure Objects (buildings, roads, rivers, etc.)
> Organization Objects (e.g., ABC Insurgents, our peace keeping forces, etc.)
> Transportation Objects (e.g., Taxis, trains, planes, etc.)

To a reasonable extent the attributes for these objects have been populated. For example, in creating the building infrastructure objects, we may utilize data from publicly available maps (or GIS systems), but may not have the information necessary

to populate attributes detailing the type of construction. In our simple example, we'll assume that the OSS contains objects for:

ABC Insurgents (an organization)
123 Maple Street (a building)
Main Street (a street)

It also contains object definitions (templates) for defining a Person, and a Vehicle (in this case a taxi).

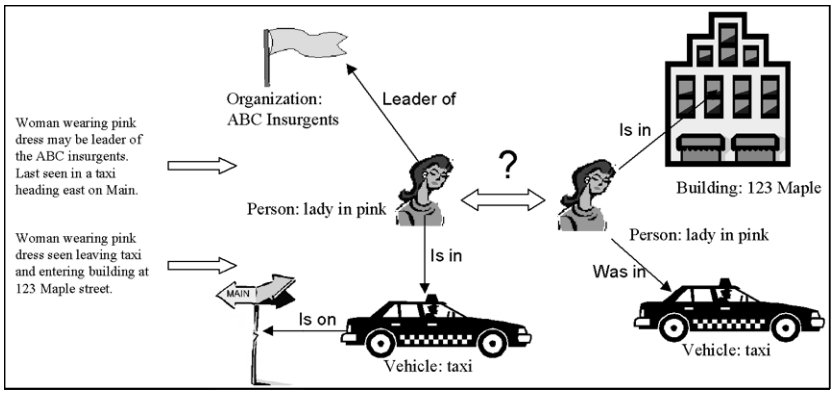


Figure 2-3 Objects and relationships

The first report causes an object (of type Person) to be created and populated with any available attributes about "the lady in pink" (e.g., her sex, color of her clothing, etc.) The report also causes an object (of type Vehicle) to be created to represent the reported taxi. Again, the taxi object is populated with any available attribute information (such as the taxi's color, current location, and direction/speed of travel.) More importantly, the report causes relationships to be established between the various objects. For example, the report's statement that the "Woman wearing pink dress may be leader of the ABC insurgents" causes a "leader of" relationship to be constructed between the "lady in pink" object and the "ABC Insurgents" object. Similarly, the fact that the taxi is reported to be driving on Main street would be represented as a "is on" relationship between the taxi and the "Main Street" object. Finally, the fact that the woman is in the taxi is likewise represented by an "is in" relationship between the "lady in pink" and the "taxi".

The ability to connect objects using relationships is very powerful. It represents information in a manner that is very close to the way in which humans model information. For example, the taxi is associated with Main Street, and the "lady in pink" is associated with the ABC insurgents. However, the model tells us that there is only an

indirect relationship between the ABC insurgents and Main Street. The approach also allows us to easily express changing information while preserving other information. For example, if we later determine that the lady is NOT the leader of the ABC insurgents, we can easily break the relationship or replace it with a more appropriate one (e.g., "sympathizes with"). The lady's association with the taxi is still valid. This type of flexibility is very difficult to achieve using message passing.

Continuing with our example, we receive the second report that indicates that a lady wearing a pink dress exited a cab and entered the building at 123 Maple Street. The report likewise creates objects for a "lady in pink" and a "taxi". Additionally it indicates that the "lady in pink" is "in" the building at 123 Maple, and that she "was in" the taxi.

Figure 2-1 correctly depicts our understanding of the situation at the moment – i.e., we have reports on two women. We still need to apply resources to determining whether or not the lady reporting in the first and second reports is the same. However, unlike the message-based system, automated decision support systems can reason about the object model, and therefore can aid in determining whether they are indeed the same people.

Lets assume that an automated system notices similarities between the two objects and their associated relationships and suggests to a human decision-maker that they may indeed be the same people. If the human agrees, he merges the object models. As shown in Figure 2-2, the object model now correctly depicts our model of the real world.

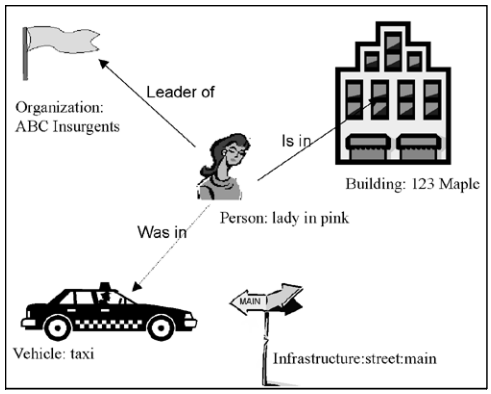


Figure 2-4 The merged object model

Any other users of the system (be they humans, agents, or software systems) are automatically aware of the new relationships (Figure 2-3). For example, a user may

wish to be informed of any buildings that are either directly or indirectly associated with the ABC insurgents. Likewise, we may want to display each of the objects on a map – i.e., instead of the map displaying the location of the reports, the map displays an icon representing the lady in pink, the building, and maybe the association between the lady in pink and the ABC insurgents. More importantly, since all of the systems share the same object model, a user seeing the icon representing the "lady in pink" could choose to explore the relationships stored in the OSS.

For example, a user recognizing that the suspected leader of the ABC insurgents is in the building may choose to examine the other objects (say people) which are also associated with the building. The needed information (the objects and relationships) is already available in the shared object model.

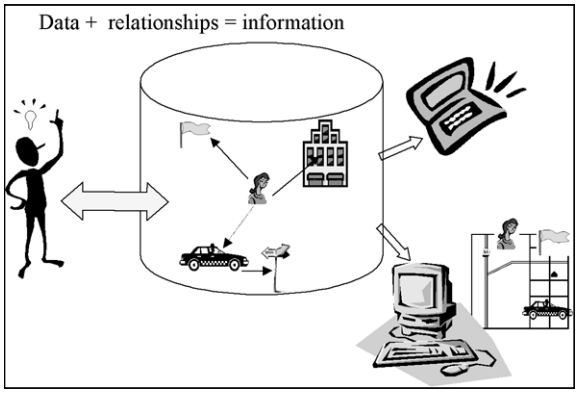


Figure 2-5 Information sharing in an pobject system

3 IMMACCS

During the past two years, the Marine Corp Warfighting Laboratory (MCWL) has been experimenting with various technologies with a focus on improving situational awareness and supporting rapid decision making. One of the outgrowths of this experimental process is the Integrated Maritime Multi-Agent Command and Control System (IMMACCS) which has been cooperatively developed by researchers from government, universities, and industry. The IMMACCS architecture is built around the object sharing approach, in which all components share and represent information exclusively through a shared object system.

IMMACCS is an integrated suite of applications with automated decision support tools. Currently, IMMACCS is composed of five primary components shown in Figure 3-1: GIDB, MCSIT, Agent Engines, 2-DV/IOB, and the Shared Net.

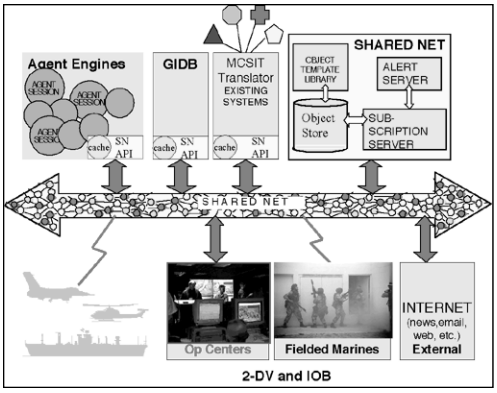


Figure 3-1 High-Level IMMACCS architecture

Geographic Information Database (GIDB)[1] provides the interface between IMMACCS and information provided by National Imagery and Mapping Agency (NIMA). It is responsible for establishing and maintaining all infrastructure objects (roads, buildings, rivers, topology) stored in the Shared Net.

MCSIT [2] provides the bi-directional interface between IMMACCS and existing message based Command and Control (C4I) systems that are deployed throughout DoD. Messages received from these systems are translated into updates to the appropriate IMMACCS objects and relations. For example, MCSIT takes naval track updates from JMCIS and uses it to update the coordinates for the appropriate ship object in IMMACCS. Similarly, updates to the IMMACCS object model may result in MCSIT constructing and transmitting a correctly formatted message to one of the Existing C4I systems (for example, a POSREP back to JMCIS). MCSIT allows IMMACCS integrate with existing C4I systems as peers.

The **Agent Engine** [3] provides IMMACCS and the users with automated decision support tools that can be applied to solving a number of problems. For example, a NBC (nuclear, biological, and chemical) agent may be instructed to watch for signs of this type of event. Upon detecting the event, the agents could then use other IMMACCS data (such as wind direction, force deployment, etc.) to suggest courses of action to fielded troops and their commanders (e.g., safe evacuation routes, appropriate protective measures, etc.) By providing a tool set rather than a pre-canned solution to a specific problem, the Agent Engine can be rapidly applied to new problem domains. Additionally, the agents constantly monitor the object model and interact with users to suggest and establish new relationships between existing objects.

IMMACCS provides two user- interfaces, the InCon **2-D viewer** (2-DV) [4] and **IMMACCS Object Browser** (IOB) [5]. Both interfaces allow users to view and update the common map (including the location of all friendly and reported hostile forces, and infrastructure objects), query objects to gain more information (e.g., What is the construction of this bridge?), issue/receive basic reports (in object form), and add information into the Shared Net by creating object and relationships. Additionally, they allow the user to subscribe to information of interest in the Shared Net (e.g., automatically update the position of hostile forces within 1 mile of my present position). The 2-DV can also be integrated with a Global Positioning System (GPS) receiver, and laser range finding binoculars, both of which allow it to automatically report the user's position, and more accurately report the position of any target. Conversely, the IOB provides more sophisticated tools for interacting with the Agent Engine's decision support tools. These interfaces were used to support users in the Enhanced Combat Operations Center (ECOC) aboard ship as well as Marines ashore (via an RF network).

The **Shared Net** provides the Shared Object store, and serves object updates to all other IMMACCS components. In effect, it is the common object bus for all IMMACCS component, and ensures that they all have the common information necessary to provide the *same* common picture of the battlespace. The Shared Net supports one-time queries, as well as standing requests for information. In the latter case, if the information in the Shared Net changes, it is automatically pushed out to the subscribing client(s). The Shared Net also maintains a local replicative cache on each subscribing client. This cache contains the latest state of the subscribed objects and can be used by the client if the network connection to the Shared Net is temporarily unavailable. The capabilities and architecture of the Shared Net will be described in more detail in the next section.

The functional capability of IMMACCS was demonstrated during the recent Urban Warrior Advanced Warfighting Exercise in San Francisco. All IMMACCS components and users were able to share and update the common set of more than 17,000 objects and their associated relations. Changes made by one component were immediately available to all other components. Users were able to successfully tailor their information feeds and interact with agents to support their decision-making processes.

During the AWE, IMMACCS was employed within the Experimental Combat Operations Center (ECOC) aboard the USS Coronado, with the city of Monterrey, CA in their Emergency Operations Center over a wireless LAN, and to Marines ashore down to the squad level. While operating in the Caribbean, the HMS Marlborough was also able to employ IMMACCS via an INMARSAT connection to the servers aboard the USS Coronado in San Francisco bay. During the final phase of the Urban Warrior AWE, the Common Tactical Picture was posted to an interactive World Wide Web site. Development of the IMMACCS system is ongoing.

4 Overview of the Shared Net

The Shared Net is the primary information storage, management and distribution system in IMMACCS. It is intended to provide the tools necessary to get the right information to the right decision-maker (from General to fielded Marine). Clients may update information stored in Shared Net, can issue on-time requests for information (a.k.a. queries), or can set up standing requests for information (a.k.a. subscriptions). Subscriptions can be at the class, object or attribute level - e.g., a user could subscribe to the creation of a friendly aircraft, movement of ANY hostile unit within an area, or the change in life-status of a particular squad-mate. Whenever a change in the Shared Net satisfies a subscription request, the requesting client is notified.

In addition, the design of the Shared Net is heavily influenced by the following operational considerations:

1) The Shared Net must support the information needs of several hundred simultaneous clients.

2) The Shared Net must be able to support a sustained rate of 100 - 200 object updates per second from its aggregate clients.

3) Shared Net users (a.k.a. subscribers) will likely have widely different information interests. For example, data concerning the fuel level in a supply truck is of primary interest to the logistics officer, and generally of little interest to the intelligence officer.

4) In general, a user will need only a small fraction of the information available in the Shared Net to support their information needs.

5) Similar types of users will likely have common subscriptions. For example, most members of a squad would subscribe to changes in the reported positions of their squad-mates as well as any nearby hostile forces.

6) Even if users subscribe to the same data, they will assign a different level of importance (priority) to a change in the data.

7) Users must be able to handle higher priority changes before lower priority changes.

8) Users will view the battle space at various levels of detail. For example, a commander in an ECOC may want to maintain the overview of the battlespace, while the squad leader may only want information concerning their local area.

9) A user's subscriptions may change dynamically. The change may be caused by the situation or by geography - e.g., in an urban canyon, tell me if enemy aircraft are within 10 miles. However, on an open field outside of the city, notify me if they are within 50 miles.

10) The Communication channels used by the Shared Net may be relatively small and unreliable especially tactical communications to fielded Marines. (RF communication in Urban Canyons, jamming, equipment failure, etc.)

11) Some (but not all) information handled by the Shared Net is deemed "life/mission critical".

12) The Shared Net shall not be a single point of failure.

13) Commercial Off-The-Shelf products should be used where feasible.

These considerations have driven an architecture that uses a hybrid of various distributed computing techniques. A traditional client-server architecture, built on CORBA, is used when clients need to reliably update the contents of the Shared Net. A distributed cache model has been used to guarantee that individual clients can continue to function (to a limited extent) even if the Shared Net is unavailable. Finally, a modified "publish and subscribe"[6] approach has been used to efficiently distribute changes in the Shared Net to subscribing clients. By transmitting only the changes to the object model, we can overcome many of the problems associated with distributing a large object infrastructure across a narrow communication link.

While many of these objectives were completely met in the initial system, others are being addressed as part of the on-going Phase II design and implementation.

4.1 Architecture

The Shared Net is comprised of the five major components shown in figure 4-1. The components are connected via common internet protocols such as CORBA/IIOP or IP. Servers are hosted on a Solaris Ultra-2 processor and written in C++. Client applications are hosted on NT, Solaris, and HP platforms and written in Java for portability.

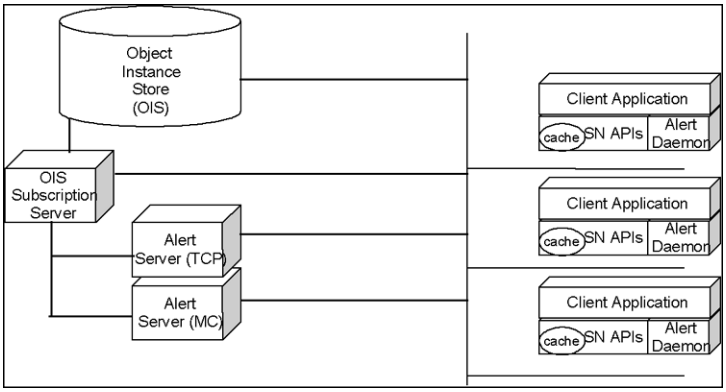


Figure 4-1 Shared Net components

The major components include:

The **Object Instance Store** (OIS) is the primary object factory and repository for the Shared Net. It is responsible for managing object creation, deletion, and modification of object attributes. The OIS provides a CORBA interface which is invoked by the clients via the Shared Net API (SNAPI), through a straight CORBA/IIOP interface, or though a local management interface. The OIS provides object persistence by periodically saving object changes to an object oriented database. The OIS notifies the OIS Subscription Server whenever a change is made to the OIS (e.g., an object is created/destroyed, or the attribute's value changes.)

The **Shared Net Application Programmer Interface** (SNAPI) provides an abstract set of client-side APIs that are used by all clients to access Shared Net services. SNAPI isolates the client from the particular distributed computing model (CORBA, TCP, etc.). SNAPI allows the Shared Net to define and manage network diagnosis/recovery policies (e.g., when to retry a failed connection). It also distributes the processing load associated with first order business rules (e.g., data integrity checks) to the clients rather than the OIS. Finally, SNAPI maintains an up-to-date local cache of subscribed objects on each client. Changes made to the OIS, by the client, are automatically written to the local cache (write-through policy). Likewise, the cache is automatically updated (via the alert and subscription system) if another client changes the object. The cache also allows the client to read from the local store (rather than retrieving the value via a CORBA connection to the OIS) which reduces the load on the OIS for non time-critical retrievals. More importantly, the cache provides the ability for a Shared Net client to continue to work (albeit on potentially old data) even if the network connection to the OIS is severed. For example, a fielded Marine who has lost communications would have, at least, the latest position of friendly and hostile forces. SNAPI uses the services of both the OIS and the Alert Daemon.

The **OIS Subscription Server** (SS) is responsible for maintaining the list of client subscriptions and ensuring that they are notified when a change in the OIS satisfies one or more of their subscription requests. Clients communicate with the SS via an SNAPI, and indicate the combination of objects, classes or attributes that make up their subscription (e.g., hostile tank movement within 1 mile of my current position). The subscription request also indicates HOW the client needs to be notified (e.g., reliable TCP or broadcast), and the priority at which the client wants to be notified when a subscription is met. Note that several different subscribers may assign different priorities to the same subscription. The SS stores the subscription information locally, and passes information concerning whom is to be notified when the subscription is satisfied to the appropriate Alert Server (AS). When a change is made to the OIS, the OIS sends a summary of the change to the SS. The summary includes sufficient information to update the distributed cache maintained by SNAPI on each client. At a minimum, it includes the object reference, its class, and the name and value of any attributes that have changed. The SS server compares this information with its list of subscriptions. If a subscription is met, the SS passes the summary information to the appropriate AS and requests that it raise the appropriate alert to any subscribed clients.

The **Alert Server** (AS) is responsible for notifying subscribing clients when their subscriptions have been met. Currently there are two forms of AS, one for reliable TCP notification, and one for broadcast (currently using UDP, but being modified to support multicast). The AS receives a summary message from the SS and forwards the message to the appropriate subscribers using the appropriate model (e.g., via a TCP connection to each subscriber, or a message sent to a multicast group, etc.) The AS is also responsible for maintaining a reasonable cache of previous alerts and ensuring that they are delivered to the subscriber upon request. For example, the TCP implementation must be able to maintain a finite ordered set of alerts that meet the subscription request of a client which is currently out of range. Likewise, the UDP implementation supports a request to rebroadcast a subset of recent alerts. The alerts generated by the AS are received and processed by the client's Alert Daemon (AD).

The **Alert Daemon** (AD) is responsible for receiving alerts from various Alert Servers. Once it validates the alert as being of interest to the local client (necessary for some broadcasts), it uses the client's original subscription request to place the alert in the appropriate priority queue. It then notifies the client that an alert is waiting to be processed.

4.2 A Subscription Example

The heart of the Shared Net's ability to efficiently distribute information to a large number of clients across possibly unreliable networks is largely provided by the Subscription and Alert system. As an example of how these systems functions, we'll

assume that we have three clients. The first is a medevac agent that is responsible for monitoring life status readings and proposing medical evacuations if the life signs of an individual reach a critical threshold. The second client is a squad leader who is naturally concerned about the heath of his squad-members. The third client is an operations agent responsible for monitoring the assets (human and machinery) assigned to a particular operation to ensure that the operation can be completed according to plan.

In order to complete their missions, each of these clients subscribes to information within the Shared Net.

- The medevac agent subscribes to "life status of any blue force personnel which fall outside a specified norm." The medevac agent indicates that this information should be processed at the "priority" level. The subscription system determines that this is a new (unique) subscription and returns a unique alert ID to the medevac agent. In addition, the medevac agent subscribes to all changes in blue force position at the "priority" level. The subscription system again determines that this is a new (unique) subscription and returns a new unique alert ID.

- The squad leader needs the most up-to-date information concerning his team, and so subscribes to all changes in the life status of members of his squad. He indicates that this information should be processed at the "critic" level. The subscription system determines that this is a new (unique) subscription and returns a unique alert ID. To keep his map current, the squad leader also subscribes to all blue force position changes, but at the 'normal' priority. The subscription system determines that an identical subscription has already been entered, and returns the original alert ID to the client. (Note that the priority assigned to the subscription is ignored by the subscription system.) Finally, the squad leader indicates that at the moment, he does not want to handle anything below a "priority" alert.

- The operations agent subscribes to the life status of the personnel assigned to a particular mission. The agent assigns a "flash" priority to this information. The subscription system determines that this is a new (unique) subscription and returns a unique alert ID. Like the other system, the operations agent subscribes to all blue position changes, again at the normal priority. The subscription system recognizes that an existing subscription meets this request and returns the original alert ID.

The subscription system has now been configured to watch for the following subscriptions:

1. Changes in blue force life status that falls outside the norm.
2. Changes in the position of any blue force.
3. Changes to the life status of any member of the squad.
4. Changes to the life status of any member of a mission.

Again, the priorities assigned by the client to the subscription impact only the Alert Daemon, not the subscription or alert servers.

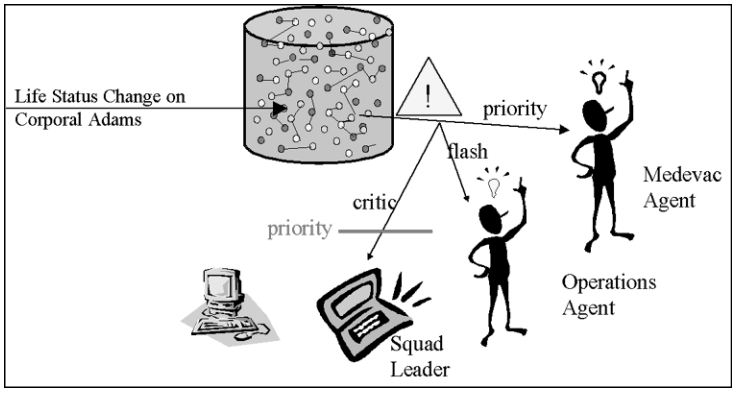


Figure 4-2 Priority-Based alerts

We'll assume that one of the members of the mission and squad is Corporal Adams. As part of its normal operation, Corporal Adams' 2-DV terminal periodically reports on his position and any significant change in life status. We'll look at two different reports issued by Adams' system.

In the first report, Corporal Adams' 2-DV updates the Shared Net (via SNAPI) and indicates that only his position has changed. The OIS updates the appropriate object's attribute and notifies the subscription server that a change has been made. The subscription server examines its subscription list, and determines that subscription # 2 has been met. It sends a message (including the summary data it received from the OIS) to the Alert Server and indicates that it should notify subscription #2 clients that their subscription has been met. The AS (in this case we'll assume a broadcast server), broadcasts an alert message to the appropriate group. The message contains the subscription #, and the original summary received from the OIS. Each client's AD is responsible for receiving alert messages transmitted by the Alert Server(s), and determining whether the alert is relevant to the particular machine. In this case, all three ADs recognize that it is an alert of interest. However, here the processing for each AD differs. The squad leader has assigned a priority of "normal" to alerts associated with subscription #2, and has also instructed his system to ignore

alerts with a priority of less than "priority". In this case, the AD pends the alert to the "normal" queue but does NOT notify the client that an alert is waiting to be processed (note that if at some later time, the squad leader lowers their notification threshold, the information will still be locally available). The operations agent's AD receives the alert, adds it to the "normal" priority queue, and notifies the client that an alert is waiting to be processed. When the client chooses to process the alert, it uses the summary information to automatically update the client's cached copy of the object to reflect the new coordinates. The medevac agent's AD performs similarly, but adds the alert to the "priority" rather than "normal" queue.

In the second report, Corporal Adams' unit updates the Shared Net (via SNAPI) and indicates that only his life status has changed and that it is outside normal parameters. The OIS updates the appropriate object's attribute and notifies the subscription server that a change has been made. The subscription server examines its subscription list, and determines that subscriptions #s 1, 3 and 4 have all been met. It is important to note that a single object update can satisfy multiple subscription requests. The SS sends three independent messages to the Alert Server each of which contain a unique alert ID, but the same summary information. As before, the Alert Server generates the appropriate alert messages that are received by the subscribing Alert Daemons. The ADs again determine whether the alert is of interest, append it to the appropriate queue, and notify the client that a subscription has been met.

While confusing at first, the priority based publish and subscribe system allows a great deal of flexibility in dealing with a large number of clients, and subscriptions which are common to a large number of clients as well as those associated with only a single client.

4.3 Continuing Work

The first version of the Shared Net was fielded with IMMACCS as part of the Urban Warrior Exercises in spring of 1999. This "phase I" version of the Shared Net was able to handle a small number of clients (10-20) and support a sustained transaction rate of 60 – 70 object updates per second. While it performed well during the exercises, a number of modification need to be made before the Shared Net (or IMMACCS) can be fielded. In order of importance the issues are:

- The Shared Net must provide and enforce strong authorization and authentication. The Phase I system allowed any client to modify objects. The phase II system must recognize that only certain users are authorized to modify certain objects or attributes. We will likely leverage off of the Public Key Infrastructure (PKI) which will be fielded Department of Defense wide by DISA and NSA in the spring of 2000.

- The Shared Net must be distributed across multiple servers in a variety of configurations. At a minimum we should support both fully and partially replicated, and cooperating autonomous servers. In the former case, some or all of the data on one Shared Net server is replicated on another Shared Net server. In case of a primary failure, or for load balancing, the replicate server can serve the information. The existing subscription mechanisms can easily support these requirements. In the latter case, various parts of the object model are maintained on independent Shared Net nodes. For example, logistics information could be maintained on the logistics ship, while operational information would be maintained on another ship or even on a Shared Net node with the Marines ashore. We are currently evaluating techniques that can be used to support this capability.

- The Shared Net must be scalable to support a much larger number of clients (100 – 200) and a larger transaction rate (hundreds of updates per second). Currently CORBA and the object-oriented database, that provides persistence to the OIS, are significant processing bottlenecks. While replicated Shared Net nodes can be used to distribute some of the processing load across multiple systems, it is likely that we will investigate the use of Real Time CORBA, as well as more efficient methods of providing persistence to the OIS.

5 Conclusion

Today's decision-makers are faced by almost insurmountable challenges. Ever increasing amounts of data are available from a bewildering number of sources, and the speed at which decisions must be made is rapidly increasing. Efficient methods of sharing information (not just data) must be employed in order to provide the decision-maker with the right information at the right time.

Government and industry currently rely heavily on message passing systems to support the exchange of information between users and systems. Unfortunately, these approaches are often inflexible, ill-suited to wide spread information exchange, and do little to support automated decision support systems.

Object Sharing Systems represent information as objects and relationships. This approach allows us to express complex relationships in a manner that is not only flexible, but also able to support the information needs of automated systems, decision support tools, and users. Changes to the information stored in the Object Sharing System are immediately available to all participants.

The IMMACCS system is a successful on-going application of the Object Sharing approach. IMMACCS provided the ability for distributed users and wide variety of existing and experimental systems to efficiently share information. Decision support

tools (agents) were able to effectively collaborate with users to recognize dangers and suggest courses of action.

The Shared Net provides the Object Sharing infrastructure for IMMACCS. The Shared Net provides a distributed object architecture, and provides a powerful mechanism which allows users/systems to subscribe to the information that they need to make their decisions (i.e., to get the right information to the right user at the right time.) Development of the Shared Net and IMMACCS is on going.

Acknowledgements

The success of the Shared Net is largely due to the incredible efforts of the programming team: Joe Hutcherson, Mike Guadarama, Ken Childress, Nguyen Hiep, Chris Alexander, and Steve Scandor. Joe also spent countless hours fleshing out the original designs as well as supervising the development effort. Thanks also to Lt.Col Bull Durham and Lt.Col Carl Bott for their constant support and *motivation*.

This work was supported by the Marine Corps Warfighting Laboratory under a contract with the National Aeronautics and Space Administration.

References

[1] Maria A. Cobb, Harold Foley III, Ruth Wilson, Miyi Chung, and Kevin B. Shaw, " An OO Database Migrates to the Web", IEEE Software, Vol. 15, No. 3, May/June 1998.

[2] Multi-C4I System/IMMACCS Translator (MCSIT) Software Requirements Specification, IMMACCS-MCSIT-SRS-3.6, Space and Warfare Systems Command (SPAWAR), 15 January 1999.

[3] Pohl J., M. Porczak, K. Pohl, R. Leighton, H. Assal, A. Davis, L. Vempati, A. Wood, and T. McVittie; "IMMACCS: A Multi-Agent Decision-Support System;" Technical Report CADRU-12-99, CAD Research Center, Cal Poly State University, San Luis Obispo, California, August 1999 (Sections 2 and 3.3).

[4] InCon, is a product of SRI International. Further information is available at http://sri.systech.com/InCON.

[5] Pohl J., M. Porczak, K. Pohl, R. Leighton, H. Assal, A. Davis, L. Vempati, A. Wood, and T. McVittie; "IMMACCS: A Multi-Agent Decision-Support System;" Technical Report CADRU-12-99, CAD Research Center, Cal Poly State University, San Luis Obispo, California, August 1999 (Section 3.5).

[6] Gamma, Helm, Johnson and Vlissides, "Design Patterns. Elements of Reusable Object-Oriented Software", Addison-Wesley, 1995.

*SPOOK: A System for Probabilistic Object-Oriented Knowledge Representation**

Avi Pfeffer, Daphne Koller, Brian Milch, Ken T. Takusagawa
Computer Science Department
Stanford University

Abstract

In previous work, we pointed out the limitations of standard Bayesian networks as a modeling framework for large, complex domains. We proposed a new, richly structured modeling language, Object-oriented Bayesian Networks, that we argued would be able to deal with such domains. However, it turns out that OOBNs are not expressive enough to model many interesting aspects of complex domains: the existence of specific named objects, arbitrary relations between objects, and uncertainty over domain structure. These aspects are crucial in real-world domains such as battlefield awareness. In this paper, we present SPOOK, an implemented system that addresses these limitations. SPOOK implements a more expressive language that allows it to represent the battlespace domain naturally and compactly. We present a new inference algorithm that utilizes the model structure in a fundamental way, and show empirically that it achieves orders of magnitude speedup over existing approaches.

1 Introduction

Bayesian networks are a graphical representation language in which complex probabilistic models can be represented compactly and naturally. The power of the representation comes from its ability to capture certain structure in the domain — the locality of influence among different attributes. This structure, which is formalized as probabilistic conditional independence, is the key to the compact representation. It also supports effective inference algorithms.

In previous work [KP97], we argued that, despite their power, Bayesian networks (BNs) are not adequate for dealing with large complex domains. Such domains require an explicit representation of additional types of structure: the notion of an object a complex structured domain entity with its own properties; and the notion of a *class* of objects, that captures properties common to an entire set of similar objects. Our *Object-Oriented Bayesian Networks* extended the language of BNs with these additional concepts.

By introducing objects and classes, OOBNs provide us with a representation language that makes it much easier to specify large models in a compact and modular way.

* Submitted to UAI-99

However, these new concepts also reveal the shortcomings of the OOBN framework. As soon as we have objects, we want to encode various relationships between them that go beyond part-whole. For example, we may have an object representing some physical location (with its own properties). We may well wish to assert that another object, such as a military unit, is at the location. This relation is not a part-whole relation, and thus does not fit naturally into the OOBN framework.

In [KP98], we described a language that allows a much richer web of relations between objects. It also extends the expressive power of the language in several significant ways. For example, by making relations first-class citizens in our ontology, we can express knowledge we might have about them; just as importantly, we can express *lack of know*ledge about relations. For example, we can express the fact that we do not know which of several locations a unit is at; we can even quantify this uncertainty using probabilities. We can also express uncertainty about the number of subunits that a unit has.

Although the additional expressive power provided by OOBNs and its extensions is natural and even desirable, one still needs to make the case that it actually helps model real-life domains. We need also to show that we have the capability to answer interesting queries using a reasonable amount of computation. In this paper, we address both of these points. We present an implemented system called SPOOK — System for Probabilistic Object-Oriented Knowledge. We show that it can be used to represent and reason about a real-world complex domain.

The domain we have chosen for this test is military situation assessment [ML96]. This domain is notoriously challenging for traditional Bayesian networks. It involves a large number of objects, related to each other in a variety of ways. There is also a lot of variability in the models appropriate to different situations. We started with a set of Bayesian networks constructed for this domain by IET, Inc. We then used our SPOOK language to construct a single unified model for this domain, one with a rich class hierarchy. The resulting model was compact, modular, natural, and easy to build.

We also investigate our ability to answer queries effectively using such a complex model. One approach (the one we proposed in [KP98]) is based on *knowledge based model construction* (KBMC) [WBG92] — converting the complex model into a traditional BN, and using standard BN inference. The BNs constructed from a complex SPOOK model are large enough to stretch the limitations of existing inference algorithms. Even the network for a single SCUD battalion involves over 1000 nodes and requires 20 minutes to answer a query. A network for many interacting units in a battlespace would be orders of magnitude larger.

The challenges posed by real-life complex models require a more sophisticated approach to inference. In our original OOBN paper [KP97], we described an inference

algorithm that makes use of the structure made explicit in the more expressive language: the encapsulation of one object within another, and the model reuse implied by the class hierarchy. The OOBN algorithm is too simple to apply to our much richer SPOOK language. However, it turns out that many of the same ideas can be adapted to this task. We present a new inference algorithm, implemented in the system, that utilizes encapsulation and reuse in a fundamental way. We present experimental results for our algorithm on realistic queries over the battlespace model, and show that by utilizing encapsulation and model reuse, we obtain orders of magnitude speedup over the KBMC approach.

2 The SPOOK Language

In this section we review the SPOOK representation language. The language presented here is based on the probabilistic frame systems of [KP98]; it extends the language of object-oriented Bayesian networks (OOBNs) [KP97] in several important ways.

The basic unit in the SPOOK language is an *object*. An object has *attributes*, which may be either *simple* or *complex*. A simple attribute is a function from objects to values in some specified domain; it is similar to a variable in a standard BN. A complex attribute represents a relationship between objects. If the value of complex attribute A of object X is Y (notated $X.A = Y$), the relation $A(X,Y)$ holds. Complex attributes may be *single-valued*, corresponding to functional relationships, or multi-valued, corresponding to general binary relations. A complex attribute may have an *inverse*: if the inverse of attribute A is B, and Y is a value of $X.A$, then X must be a value of $Y.B$.

For example, a scud-battalion object has a simple attribute under-fire, whose value ranges over {*none, light, heavy*}. It has a single-valued complex attribute at-location, whose value is an object corresponding to the location of the battalion. It has a multi-valued complex attribute has-battery, each of whose values is a battery in the battalion. The has-battery attribute has an inverse in-battalion, which is a single-valued complex attribute of a battery object. If battery-1 is a value of scud-battalion-charlie.has-battery, then battery-1.in-battalion = scud-battalion-charlie. The dot notation can be extended to *attribute chains* $A_1.A_2. \ldots .A_k$, denoting the composition of the relations $A_1, \ldots, A_k$. If $A_1, \ldots, A_{k-1}$ are single-valued complex attributes, and A_k is a simple attribute, we call the attribute chain *simple*.

The probability model for an object is specified by defining a local probability model for each of its simple attributes. As in BNs, The local probability model consists of a set of parents, and a conditional probability distribution (CPD). A parent can be either another simple attribute of the same object, or a simple attribute chain. Allowing attribute chains as parents provides a way for the attributes of an object to be influenced probabilistically by attributes of related objects. If two objects are inverses of each other, each can be influenced by the other.

Continuing our example, the under-fire attribute of scud-battalion has a parent at-location.defense-support, and the CPD for under-fire indicates that the battalion is more likely to be under heavy fire if it is in a location with poor defense support. The battery object has a hit attribute whose parent is in-battalion.under-fire, thus creating an in-direct chain of influence from the location, through the battalion at the location, to the battery in the battalion. Since in-battalion is an inverse of has-battery, the battalion can in turn be influenced by the battery it contains. For example the attribute scud-battalion.next-activity depends on has-battery.launch-capability. (Section 2.2 explains how to specify dependence on multi-valued attributes.)

2.1 Classes and Instances

In SPOOK, a probability model is associated with a class, which corresponds to a type of entity in the domain. An *instance* of a class corresponds to a domain entity of the appropriate type, and derives its probability model from its class. We use *object* to denote either a class or an instance. For example, scud-battalion is a class and scud-battalion-charlie is an instance of scud-battalion.

Classes provide reusable probability models, that can be applied to many different objects. Classes are organized in a *class hierarchy*. A *subclass* inherits the probability model of its *superclass*, and it can also override or extend it. The inheritance mechanism facilitates model reuse by allowing the commonalities between different classes to be captured in a common superclass. For example, the battalion super-class captures those features common to all battalions.

Classes also provide a type system for the SPOOK language. Every complex attribute A has a type $T(A)$, and for any object X, the value of $X.A$ must be an instance of $T(A)$. If A no particular value is specified for $X.A$, we use the *unique names assumption*, which states that the values of $X.A$ are generic, unnamed instances of $T(A)$, that are not related in any other way to the instances in the model.

The unique names assumption implies that in the class models, no two battalions can be at the same location. In-stances provide a way to specify such webs of inter-related objects. In this example, there are two battalion instances, battalion-1 and battalion-2, and a location instance location-a. By stating that battalion-1.at-location = location-a and that battalion-2.at-location = location-a, the objects are hooked together appropriately.

2.2 Multi-Valued Attributes and Structural Uncertainty

As discussed above, a complex attribute can be multi-valued, but a parent of a simple attribute must be a simple attribute chain, in which the attributes are single-valued. In order to allow the attributes of an object to be influenced by attributes of related objects when the relationship is multi-valued, we introduce a *quantifier attribute*. A quantifier

attribute has the form #($A.\rho = v$), where A is a multi-valued complex attribute, ρ is a simple attribute chain, and v is a possible value of ρ. If X is an object with attribute A, $X.\#(A.\rho = v)$ denotes the number of objects Y such that $A(X,Y) \wedge Y.\rho = v$.

Quantifier attributes allow attributes of an object to depend on aggregate properties of a set of related objects. Continuing our running example, we may specify that a parent of scud-battalion.next-mission is the quantifier attribute #(has-battery.launch-capability=*high*). The value of the quantifier is determined by the value of launch-capability for each of the batteries in the battalion. If the set of batteries in the battalion is fixed, the quantifier simply expresses an aggregate property of the set. However, we may also have uncertainty over the number of batteries in the battalion. This is an example of *structural uncertainty*, which is uncertainty not only over the properties of objects in the model but over the relational structure of the model itself.

The type of structural uncertainty encountered in this ex-ample is *number uncertainty*: uncertainty over the number of values of a multi-valued complex attribute. Number uncertainty is integrated directly into the probability model of an object using a *number attribute*. If A is a multi-valued complex slot, the number attribute $\#A$ denotes the number of values of A. A number attribute is a standard random variable whose range is the set of integers from 0 to some upper bound n. It can participate directly in the probability model like any other variable. In our example, scud-battalion.#has-battery depends probabilistically on scud-battalion.country. Under number uncertainty, the value of a quantifier depends on the value of the number attribute, as well as on the values of the related objects.

Another kind of structural uncertainty is *reference uncertainty*, which is uncertainty over the value of a single-valued complex attribute. For example, we may have uncertainty over whether a battalion is located in a mountain or a desert location. As with number uncertainty, reference uncertainty can be introduced directly into the probability model of an object using a *reference attribute*. If A is a single-valued complex attribute whose value is uncertain, $R(A)$ is a reference attribute whose range determines the possible values of A. An element of the range of $R(A)$ may either be a subclass C of $T(A)$, or an instance I of $T(A)$. If the value of $X.R(A)$ is the type C, then the value of $X.A$ is a generic instance of C; if the value of $X.R(A)$ is the instance I, then the value of $X.A$ is I. As with number attributes, reference attributes participate in the probability model, and can depend on and be influenced by other attributes. We call this type of uncertainty "reference uncertainty" because we do not know which object is being referred to when we refer to the value of A.

A SPOOK knowledge base consists of a set of classes and instance models. In [KP98], we defined a data structure called a *dependency graph* that can be used to make sure that all the probabilistic influences, including the influences between different objects, are acyclic. We defined a semantics for SPOOK models, based on a

generative process that randomly generates values for attributes of instances in the domain, including number and reference attributes. We showed that if the dependency graph is acyclic, then the knowledge base defines a unique probability distribution over the values of all simple attributes of all named instances in the KB.

3 Modeling the Battlespace Domain

To demonstrate the representational power of the SPOOK language, we implemented a model for reasoning about military units in a battlespace. In [LM97], Mahoney and Laskey describe how they model this domain using *network fragments*. In this section, we introduce the domain, discuss why it is difficult to model using BNs, and describe how we modeled it using SPOOK.

The purpose of the battlespace model is to reason about the locations and status of enemy military units based on intelligence reports. Our model deals specifically with missile battalions, the batteries within those battalions, and the individual units — vehicles, radar emplacements, missile launchers, etc. — within the batteries. A scenario consists of multiple battalions, some of which may be at the same location. A battalion typically has four batteries, each with about 50 individual units. Thus, the model for a battalion includes about 200 units, and a scenario may include 1000 units.

Let us consider trying to model our domain directly with a BN. With four or five variables for each unit, a flat BN for a battalion model will typically contain over a thousand nodes. The sheer size of this network is a major obstacle to its construction. In addition, the resulting BN will be too rigid for practical purposes. The configuration of a battalion is highly flexible, with the exact number of units of each type varying considerably between different battalions. These difficulties have led to an alternative approach, in which several different BNs are used, one for each aspect of the model. Figure 1(a) shows a Bayesian network for an SA3 battalion. There are similar networks for other types of units, such as Scud battalions and batteries. Although a Scud battalion contains Scud batteries, the battalion model does not replicate all the details of the battery model; rather, it summarizes the status of all the batteries with nodes, indicating the initial number of batteries, the number of damaged batteries, and the current number. These summaries serve two purposes: to keep the network reasonably simple; and to account for changing model configuration by making the initial number of subunits a variable.

A major disadvantage of this approach is that it is very difficult to reason between the different networks. The only way to reason from one network to another is to reach conclusions about the state of variables in one network and assert them as evidence in the other network. For example, the only way to transfer conclusions from a battery to a battalion is to condition one of the summary nodes in the battalion model; going from one battery to another requires conditioning the battalion model, reasoning

about the battalion, and then conditioning the other battery model. This type of reasoning has no sound probabilistic semantics. There is no way to combine evidence about multiple different units in a probabilistically coherent manner. Furthermore, this type of reasoning between fragments must be performed by a human or a program. It requires some model of the relationship between the fragments, e.g., that the status node of the battery model is related to the number-damaged-batteries node of the battalion model. Nowhere is this relationship made explicit.

Another disadvantage is that multiple BNs do not allow us to take advantage of redundancy within a model and similarities between models. For example, the battalion model in Figure 1(a) contains many similar substructures, summarizing groups of units of different kinds. In addition, different battalions may all have substructures describing their locations, as shown in the bottom right corner of the figure. In the multiple BNs approach, the only mechanism for exploiting these redundancies is cut-and-paste. This makes it very hard to maintain these models, because each time one of the reused components is changed, it must be updated in all the different networks that use it.

OOBNs solve the problems inherent in the multiple BN approach. By allowing a battalion to contain a battery as a sub-object, we can easily have the battalion model encompass the complete models of the different batteries in it, which in turn contain complete models of their subunits, without making the battalion model impossibly complex. We can then reason between different objects in the part-of hierarchy in a probabilistically coherent manner. In addition, by allowing us to define a class hierarchy, OOBNs allow us to exploit the redundancy in the model.

However, the language of OOBNs is quite restricted, in a way that is problematic in our domain. If we want to model the effect of a unit's location on the unit, we need to represent the relationship between the unit and its location. In our model, this was the only relationship that did not fall into the part-of hierarchy, but richer models of the battlespace domain require more sophisticated relationships, such as that between a unit supporting another unit. In addition, our domain requires multi-valued attributes and quantifiers. A battalion contains several batteries, and each battery contains several units of different types. The higher level objects do not depend directly on the individual lower level objects, but only on aggregate properties of the set of objects, expressed through quantifier attributes. The ability to create named instances and hook them together via relations is also important in our domain, as illustrated by the example from the previous section of two battalions in the same location. Finally, the battlespace domain contains a great deal of structural uncertainty, in particular number uncertainty over the number of subunits. One may also have reference uncertainty as to the actual location of a unit.

SPOOK includes all the capabilities of OOBNs to represent part-of and class hierarchies, and also handles relations between objects, multi-valued attributes, named instances, and structural uncertainty, all of which cannot be expressed in OOBNs. Our SPOOK model of the battlespace domain includes a natural class hierarchy, with Military-Unit, Environment, Location and Weather as root classes. The Battalion, Battery, Group, and Unit families are all part of the Military-Unit hierarchy. Similarly, part-of relationships are easy to model in SPOOK using inverse relations. The has-battery attribute of a battalion, and the in-battalion at-tribute of a battery, are inverses, allowing the battalion and its contained battery to influence each other. Batteries do not contain individual units directly, but instead contain a Group object for each type of unit. For instance, a battery has (among others) groups of missile launchers, command vehicles, and anti-aircraft artillery units. Each Group has a multi-valued attribute relating it to the individual units, as well as a number attribute and a set of quantifier attributes that summarize the status of the units. Using Group objects is convenient because we summarize the same attributes for all types of units.

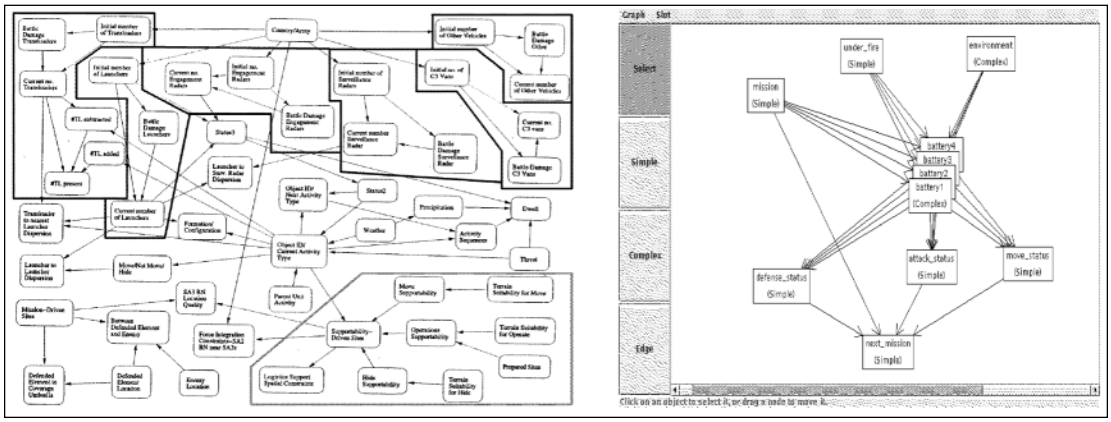


Figure 1: (a) SA3 Battalion Bayesian network, (b) SPOOK model of Scud Battalion

An object of class Unit has simple attributes reported, operational, damaged and reported-damaged. These attributes are influenced by the location of the battalion — specifically, the location's support for concealment and defense — and by the battalion being under fire. We represent these influences in SPOOK by specifying, for example, at-location.defense-support as a parent of damaged. The number of damaged units in turn influences the battery's operational attribute, and a quantifier slot that counts the number of operational batteries in a battalion influences the battalion's current-activity. Subclassing gives us the ability to provide models for certain types of units that are similar to the general unit model but not exactly the same. For instance, Missile-Launcher has an additional activity attribute that indicates whether it is launching, reloading, or idle. While we only modeled the domain up to the battalion level, we could easily extend our model to higher-level groups in the military hierarchy.

In our current model, all units in a battalion share a common environment, which is referred to by the in-environment of the battalion. The environment is composed of Location and Weather objects, which between them determine the current support of the environment for various activities such as moving, hiding and launching missiles. We could have associated a different environment with each battery or unit, making locations of lower-level objects related probabilistically to higher level objects.

To give an example of the power of reasoning at multiple levels of the hierarchy and between different objects, we present a series of queries we asked the model. First we queried the prior probability that a particular Scud battery was hit, and found it to be 0.06. We then observed that the containing battalion was under heavy fire, and the probability that the battery was hit went up to 0.44. We then observed, however, that none of the launchers in the battery had been reported to be damaged, and the probability that the battery was hit went down to 0.28. We then explained away this last observation, by observing that the environment has good support for hiding; the probability that the battery was hit went back up to 0.33. This example combines causal, evidential and intercausal reasoning, and involves battery and battalion objects, individual launcher objects, the launcher group, and the environment object.

4 Inference

In the previous sections we described the SPOOK language, and how we used it to model the battlespace awareness domain. Of course, in order for the language to be useful, we need an efficient algorithm for performing inference in it. Ideally, we would like the language features to lend themselves to efficient inference. Indeed, as we argued in [KP97], one of the measures of a successful representation language is that it makes the structure of the domain explicit, so that it can be exploited by an appropriately designed inference algorithm.

One way to perform inference in SPOOK is to use the technique of *knowledge-based model construction (KBMC)* [WBG92]. In this approach, we construct a BN that represents the same probability distribution as that defined by our model, and answer queries in this BN using standard BN inference algorithms. We described the KBMC process for our language in detail in [KP98], and showed that if the dependency graph is acyclic, it always terminates.

While the KBMC approach provides a sound and complete method for answering queries in SPOOK it is somewhat unsatisfactory. It fails to exploit the language's ability to make explicit the structure of the domain. All notions of objects and relationships are lost when the model is turned into a flat BN. In [KP97], we argued that the object structure of a model can and should be exploited for efficient inference.

We argued that two aspects of the structure in particular should be exploited: the fact that the internal state of each object is encapsulated from the remainder of the model via its interface, and that locality of interaction typically induces small interfaces; and the fact that the same type of object may appear many times in a model. Since the flat BN produced by KBMC has no notion of an object, the KBMC algorithm cannot exploit these structural features.

We now present an object-based inference algorithm that does exploit the structural features of a model. The algorithm is based on the ideas presented in [KP97], but it is significantly more complex due to the increased expressivity of our language. The added complexity arises principally from four new language features.

First, the "multi-centeredness" of the language implies that each object can be accessed by a multitude of other objects, in a variety of different ways. In OOBNs, we assumed that each type of object had a unique set of inputs and outputs, and that we could precompute a conditional distribution over the outputs given the inputs. This is no longer the case. Because an object can be accessed in many different ways, its outputs can be arbitrarily complex. In addition, its inputs are not fixed, but are determined by the way the object is accessed, and the particular set of outputs required, as will be explained below. Thus, for each object referred to by another object, our algorithm must determine its inputs and outputs on the fly, during the processing of a query.

The second relevant language feature is the ability to create instances and hook instances together via relations. As we shall explain later, this property implies that encapsulation, although still present, no longer holds in exactly the same way as in OOBNs. The third feature is multi-valued attributes and quantifier attributes that depend on them, which do not appear in OOBNs, and require a new treatment.

The final complicating feature is structural uncertainty. The naive approach to dealing with structural uncertainty, that could be applied to OOBN models, is as follows. To compute $P(Q)$, enumerate all possible structural hypotheses h, and compute $P(Q \mid h)$ for each such hypothesis. $P(Q)$ is then equal to $\sum_h P(h)P(Q \mid h)$. Unfortunately, the number of structural hypotheses is exponential in the number of structural variables, rendering this approach completely infeasible with more than a very small number of structural variables. In the battlespace awareness domain, the number of structural variables is large, since we have uncertainty over the number of units in many different groups. Therefore we need a much better way of doing inference with structural uncertainty.

Our inference algorithm is related to the KBMC algorithms, but its recursive nature makes it quite different, so we describe it in detail. It is fairly complex, so we present it in stages. We begin with the basic algorithm, for class objects without multi-valued complex attributes, we then extend it to deal with instances, and finally we show how

we deal with multi-valued attributes, quantifier attributes, number uncertainty and reference uncertainty.

4.1 Basic Algorithm

Our inference algorithm is recursive. The main function of the algorithm answers a query on an object. The key to understanding the algorithm is to understand how the function is recursively called, and the relationship between the object calling the function and the object on which it is called. Suppose that, during the process of solving a query on an object X, we encounter a complex attribute D of X. For now, let us assume that D is single-valued. There is some object Y that is the value of $X.D$. Let us assume for now that no value is asserted in the knowledge base for $X.D$, so that Y is a generic (unnamed) instance of $T(D)$.

Recall that other attributes of X may depend on the value of D, i.e., on Y. Specifically, let $\sigma_1, \ldots, \sigma_n$ be the complete set of attribute chains, such that attributes of X depend on each of the $Y.\sigma_1$ but on no other aspects of the value of Y. In order to solve a query on X, we will need to solve a subquery on Y, to obtain a distribution over $Y.\sigma_1, \ldots, Y.\sigma_n$. Recall, however, that Y may itself depend on X. This will happen if D has an inverse E, so that the value of $Y.E$ is X. Let $\tau_1, \ldots, \tau_m$ be the complete set of attribute chains through which Y depends on X. The distribution over $Y.\sigma_1, \ldots, Y.\sigma_n$ will depend on the values of $X.\tau_1, \ldots, X.\tau_m$. The subquery on Y needs to return a conditional distribution over $\sigma_1, \ldots, \sigma_n$ given $\tau_1, \ldots, \tau_m$. The issue is further complicated by the fact that, while solving a query for object X, we do not yet know the set $\tau_1, \ldots, \tau_m$, through which Y depends on X. This information can only be computed within Y itself. Therefore, when answering the subquery on Y, we return not only the conditional distribution over $\sigma_1, \ldots, \sigma_n$, but also the conditioning variables $\tau_1, \ldots, \tau_m$.

The main function of our algorithm, **SolveQuery**, takes three arguments, one of which is optional. The two required arguments are an object Y, called the *target* of the query, and a set of attribute chains $\sigma = \sigma_1, \ldots, \sigma_n$, called the *outputs* of the query. The optional argument is an attribute E, called the *entry point* of the query; E is the entry point into Y if $Y.E$ is X. The entry point is used for discovering the dependencies of Y on X:Y depends on $X.\tau$ only when some attribute in Y depends on $B.\tau$. In this case, τ is said to be an input to the query. **SolveQuery** returns two values: the set of inputs $\tau = \tau_1, \ldots, \tau_m$ to the query, and a conditional probability distribution over σ given τ. A query may have no entry point if it is the top-level call to **SolveQuery** or if it was called for an attribute D of X that has no inverse. In that case, Y cannot get inputs from X, so that τ will be empty, and the distribution returned over σ will be unconditional.

The basic procedure of **SolveQuery** is as follows. **SolveQuery** constructs a local BN, which it will eventually use to solve the query. The BN consists of nodes for each of the attributes of the query target Y, nodes for the inputs and outputs, and other nodes

that help communicate between different attributes. In order to add a node to the network, we must complete four steps: create it, specify its range, specify its parents, and specify its conditional probability distribution (CPD). These steps are not always per-formed together; in some cases, we cannot specify the CPD of a node at the time that it is created.

SolveQuery begins with an initialization phase. First it creates a node $v(A)$ in the network for every attribute A of Y. For each simple attribute A, we specify the range of $v(A)$ to be the range of A. If A is complex, we want the range to be the product of all the attribute chains through which Y depends on A, but we do not yet know this set. For this reason, we maintain a set *needed*(A) for each complex attribute A.

Next, **SolveQuery** creates an *output node* v(*output*),to represent the query output. The range of v(*output*) is $\times^{n}_{i=1}$ Dom(σ_i), where Dom(σ_i) is the range of the simple attribute at the end of the attribute chain σ_i. For each σ_i, we call the function **GetChainNode** (see below) to obtain a node $v(\sigma_i)$, whose range is Dom(σ_i), and make it a parent of v(*output*). The CPD for v(*output*) simply implements the cross-product operation: if the values of $v(\sigma_1)$, … ,$v(\sigma_n)$ are v_1, … ,v_n, the value of v(*output*) is $\langle v_1, \ldots, v_n \rangle$ with probability 1.

GetChainNode is called whenever we need to produce a node to represent the value of an attribute chain σ. If $v(\sigma)$ is already in the BN, we simply return it. This will always be the case if σ is just a simple attribute A. Otherwise, σ must have the form $A.\rho$, where A is a complex attribute. The algorithm thus needs to ensure that the processing of A will give the required information about $A.\rho$. We therefore add ρ to the set *needed*(A). We can extract the value from the output of A by creating a new *projection node* $v(\sigma)$, whose range is Dom(σ), and set its lone parent to be $v(A)$. As we will see below, the projection node performs the inverse operation to that of the cross-product node.

The main phase of **SolveQuery** performs a backward-chaining process to determine the interfaces of complex attributes. First, we order the attributes of Y in an order consistent with the dependency graph. Such an order must exist if the model is well-defined. We then process the attributes one by one, in a bottom-up order. Children must precede their parents, since processing a child tells us what "information" we need from its parents. Processing a simple attribute A is easy. We simply obtain the set of attribute chain parents of A, as specified in the model of Y. For each such parent σ, we convert it into a BN node $v(\sigma)$ by calling **GetChainNode**, and add it as a parent of $v(A)$. We then set the CPD of $v(A)$ as specified in the model of Y.

Processing a complex attribute A requires a recursive call. If A is the entry point of the query, we ignore it — it gets special treatment later. If *needed*(A) is empty, we can simply prune A. Otherwise, we will need to ask a subquery to obtain a distribution

over *needed*(*A*). For now, we assume that *Y.A* has no asserted value in the knowledge base, so that the value of *Y.A* is some unnamed instance *Z* of class *T*(*A*). Since the model of *Z* is the same as that of every other unnamed instance of *T*(*A*), we can ask the subquery on the class object *T*(*A*). We therefore make a call to **SolveQuery**, in which the target is *T*(*A*) and the set of outputs is *needed*(*A*). In addition, if *A* has an inverse *B*, the entry point is *B*, otherwise there is no entry point. The call to **SolveQuery** will return a set of inputs τ_A, and a conditional probability distribution over *needed*(*A*) given τ_A. We treat the inputs τ_A to *A* in the same way as the parents of a simple slot, using **GetChainNode**. We set the range of $v(A)$ to be $\times_{\sigma \in needed(A)} \mathrm{Dom}(\sigma)$, and set the CPD of $v(A)$ to be that returned by the recursive call.

When we have finished processing all of the attributes, we can fill in the CPDs for the projection nodes. Each such node $v(\sigma)$ represents a component of the value of a complex attribute *A*. We could not specify the CPDs for these nodes at the time they were created, since we did not yet know the range of $v(A)$. Once all the nodes have been created, we simply set the CPD of $v(\sigma)$ to implement the projection function from $\times_{\sigma \in needed(A)} \mathrm{Dom}(\sigma)$ onto σ.

At this point, we have almost built the complete network for solving the query. Recall that we have not yet processed the entry point *E*. The node $v(E)$ is the *input node*, representing the input of the query. We set the range of $v(E)$ to be $\prod_{\tau \in needed(E)} \mathrm{Dom}(\tau)$. The node $v(E)$ has no parents and no CPD. We are now done building the local BN for the object *Y*. If the knowledge base asserts a value *v* for a simple attribute *A* of *Y*, we assert the value of $v(A)$ to be *v* as evidence in the network. We then use a standard BN algorithm to compute the conditional probability of the output node given the input node, and return this conditional probability, along with the optional set of inputs *needed*(*E*).

To summarize, let us consider how our algorithm exploits the two types of structure described in [KP97]. Each recursive call computes a distribution over the interface between two related objects. The algorithm exploits the fact that all the internal details of the callee are encapsulated from the caller by the interface. Much of the work of the algorithm, in particular maintaining the *needed*() sets and returning the set of inputs τ, is concerned with computing these interfaces on the fly.

As for exploiting the recurrence of the same type of object many times in the model, observe that different calls to **SolveQuery** with the same set of arguments will always return the same value. In order to reuse computation between different objects of the same type, we maintain a cache, indexed by the three arguments to **SolveQuery**. Note that we cannot reuse computation between different queries on the same object, because they may generate very different computations down the line. However, if the two queries are similar, many of the recursive computations they generate will be the same, and we will be able to reuse those.

4.2 Dealing with Instances

If an instance has a value asserted for one of its attributes, we can no longer use the generic class model for that instance. In addition, if one instance is related to another, the internals of the two instances are not necessarily encapsulated from each other. Consider, for example, three instances I, J and K, such that $I.A = J$, $I.B = K$, and $J.C = K$. In this case K is not encapsulated from I by J. Hence, the interface between I and J does not separate the internals of I from the internals of J. When answering a query on I, we cannot simply perform a recursive call to obtain a distribution over the interface between I and J, as we would lose the fact that the internals of I and J may be correlated by their mutual dependence on K. A possible solution to this problem is to include K in the interface between I and J. However, including entire objects in an interface creates huge interfaces, and defeats the purpose of using an object-based inference algorithm.

In order to deal with this issue, we create a new top-level object T in the knowledge base. This object contains an attribute $I::A$ for every named instance I and each of its attributes A. If A is simple, $\mathrm{Dom}(I::A) = \mathrm{Dom}(A)$; if it is complex, $T(I::A) = T(A)$. If the KB asserts a value for $I.A$, $I::A$ has the same value. Every user query will be directed through the top-level object. More precisely, a user query will have the form $I.\sigma = I_1.\sigma_1, \ldots, I_n.\sigma_n$, where each I_j is an instance (not necessarily distinct), and σ_j is an attribute chain on I_j. The query is answered using a call **SolveTopLevel**$(T, I.\sigma)$. Since this is the top-level query, there is no entry point, and we will simply return a distribution over $I.\sigma$.

SolveTopLevel is very similar to **SolveQuery**, so we omit the details. The main difference is in the way attribute chains are treated. On the top level, all attribute chains are attached to an instance. This is true both for the attribute chains required in the query output, and for the parents of any top-level attribute $I::A$. We replace **GetChainNode** with a function **GetTopLevelChainNode** that takes two arguments: an instance and an attribute chain. This function behaves similarly to **GetChainNode**, except for one situation. If the chain σ is of the form $A.\rho$, and $I.A = J$, then $I.A.\sigma = J.\rho$. The algorithm eliminates this step of indirection, and continues to follow the rest of the chain.

4.3 Multi-Valued Attributes and Structural Uncertainty

Multi-valued attributes can be dealt with quite easily within the context of **SolveQuery**. If A is a multi-valued complex attribute of Y, with n values, then instead of creating a single node $v(A)$ in the BN, we create an array of nodes $v(A_1), \ldots, v(A_n)$. The function **ProcessMultiValued** is very similar to **ProcessComplex**. The subquery on $T(A)$ is solved as usual, and the results are then applied to all the nodes in the array.

Dealing with quantifiers requires some care. Consider the quantifier $Q = \#(A.\rho = v)$. Q will depend on the values of $A_1.\rho, \ldots, A_n.\rho$. We therefore create n projection nodes for these different values, and make each projection node $v(A_i.\rho)$ depend on $v(A_i)$. The chain ρ is added to $needed(A)$. The main issue with quantifiers is specifying their CPD. The naive approach is to have the quantifier depend directly on all of the $A_i.\rho$, and have the CPD implement the counting function. Unfortunately, a naive implementation of this idea results in a CPD whose size is exponential in n. We can resolve this issue using an approach similar to the noisy-or decomposition of [HB94]. We decompose the quantifier node using a cascade of $n-1$ intermediate nodes $count_1, \ldots, count_{n-1}$. The node $count_i$ represents the number of parents among $A_1.\rho, \ldots, A_i.\rho$ whose value is v. The node $count_1$ depends only on $A_1.\rho$; its value is 1 if $A_1.\rho = v$ and 0 otherwise. For $i > 1$, $count_i$ depends on $count_{i-1}$ and $A_i.\rho$. Its value is either $count_{i-1} + 1$or $count_{i-1}$, depending on whether $A_i.\rho = v$ or not. The quantifier node $v(Q)$ depends on $count_{n-1}$ and $A_n.\rho$ in the same way. Using this cascaded series of incrementers, the total size of all the CPDs used to determine the value of the quantifier is only linear in n.

As mentioned earlier, we cannot deal with structural uncertainty by reasoning about all possible structures. Instead, we need to exploit the fact that many of the structural variables do not interact. Each of the complete structural hypotheses can be decomposed into many independent or conditionally independent sub-hypotheses. For example, in the battlespace domain, the number of units of each type in a battery may all be independent of each other given the country to which the battery belongs. This type of reasoning about the conditional independence between different structural hypotheses is ideally performed in a BN. We need to express all the possible structures within a single network, so that the BN inference algorithm can exploit these independencies. We address this issue separately for number uncertainty and for reference uncertainty, although the solution is quite similar.

Let A be a multi-valued complex attribute with number uncertainty, and let n be the upper bound on the number of values of A. Even though A may have less than n values, we create nodes $v(A_1), \ldots, v(A_n)$ for all of the *possible* values of A. The number attribute $\#A$ is treated just like any other simple attribute. Now consider a quantifier $Q = \#(A.\rho = v)$. As above, we create the projection nodes $v(A_1.\rho), \ldots, v(A_n.\rho)$. If the value of $\#A$ is k, then Q depends on $v(A_1.\rho), \ldots, v(A_k.\rho)$, and ignores $v(A_{k+1}.\rho), \ldots, v(A_n.\rho)$. We can simulate this effect by having Q depend on all of $v(A_1.\rho), \ldots, v(A_k.\rho)$, as well as on $\#A$. As above, we can decompose the CPD for Q into a cascaded series of incrementers, where each incrementer now also depends on $\#A$.

We deal with reference uncertainty in a similar manner. Let A be a single-valued complex attribute with reference uncertainty. For each value v in the range of $R(A)$,we create a dummy attribute A_v, and a corresponding BN node $v(A_v)$. If v is equal to the type C, we set the type of A_v to be C. This operation ensures that A_v will later be processed correctly, as a generic attribute of type C. If v is equal to the instance I, we set the value of A_v to be I. In this case, A_v will be processed as a named instance.

While we do not know the actual value of A, introducing all these nodes into the network accounts for all possible hypotheses over its value. We can now deal with dependencies on A. Consider a projection node $v(A.\rho)$. We introduce a set of projection nodes $v(A.\rho)$ for each value v of $R(A)$.We make $v(A.\rho)$ depend on all the $v(A.\rho)$ as well as on $R(A)$, and set its CPD to select the value of the parent specified by $R(A)$. We can think of the CPD of $v(A.\rho)$ as implementing a multiplexer, where the selector is $R(A)$. (See [BFGK96] for a similar construction.)

5 Experimental Results

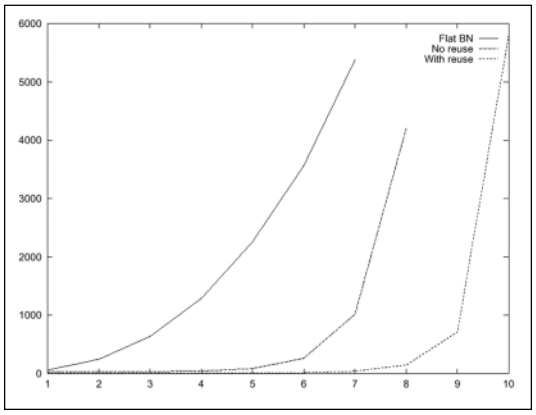


Figure 2: Experimental results

We have implemented the SPOOK system for representing models in the SPOOK language, and performing inference on these models. At the core of the system is a module containing the data structures necessary to represent the SPOOK data model. On top of this module is a user interface (see Figure 1)(b) in which the user can create class and instance objects, build probability models, observe values and query probabilities. SPOOK models can be stored in an external knowledge server such as Ontolingua. SPOOK communicates with the knowledge server using the *OKBC protocol* [CFF + 98], a generic protocol for communication between knowledge-based systems. Inference can be performed in SPOOK either by using the KBMC algorithm, or the object-based inference algorithm described in Section 4. Both inference methods use the same home-grown underlying BN inference engine. While SPOOK is already a fairly large system (about 80000 lines of code, not counting the BN inference engine), we have not yet implemented all of the language features and nuances of the inference algorithm. In particular, the support for structural uncertainty is still very basic.

In our experiments, we compared the performance of the object-based algorithm with the KBMC algorithm on models of different sizes. Each model consists of a single battalion with four batteries, each containing 11 groups of different kinds with the number of units in each group varying from 1 to 9. The model also contains objects

for the environment, location and weather, as described in Section 3. The size of the constructed BN grows linearly in the number of units per group, and varies from 750 to 5500 nodes.

In order to measure separately the benefits from exploiting interfaces and from reusing computation, we tried two different versions of the object-based algorithm, with and without reuse. We compared the three algorithms on a query on Battalion.NextMission, which depends (indirectly) on the status of most of the individual units in the model. The results are shown in Figure 2. The first graph compares the performance of the three algorithms in a model without number uncertainty. For the KBMC algorithm, the graph shows only the running time of the actual BN inference, and not the time taken to construct the BN. In practice the cost of inference dominated the cost of constructing the BN, which took up to one minute. From the graph, we see that both versions of the object-based algorithm outperform the KBMC algorithm by a large margin, and that the algorithm with reuse outperforms the algorithm without reuse. For example, with four units per group, the object-based algorithm with reuse takes 9 seconds, without reuse takes 46 seconds, while the KBMC algorithm takes 1292 seconds.

The reason for this great disparity is that the BN reasoning algorithm is failing to find optimal junction trees in the flat BN. The largest clique constructed for the flat BN contains 18 nodes, whereas the largest clique over all of the local BN computations for the structured algorithm contains only 8 nodes. The BN inference engine uses the standard minimum discrepancy triangulation heuristic to construct the junction tree. We see that at least for a standard BN implementation, exploiting the object structure and the small interfaces between objects is vital to scaling up BN inference. While algorithms do exist for computing optimal triangulations [SG97], most implementations of Bayes nets do not use them; in addition, these algorithms do not address the issue of reuse.

As the number of units per group grows, we start to see an exponential blowup for the object-based algorithm. The reason is that the cascaded decomposition of quantifier CPDs has not yet been implemented. Thus the size of these CPDs grows exponentially in the number of units per group. We believe that when this feature is implemented, we will see a much flatter performance curve.

6 Discussion

An alternative approach to ours to modeling large, complex domains probabilistically is the network fragments approach of Laskey and Mahoney [LM97]. They provide *network fragments* for different aspects of a model, and operations for combining the fragments to produce more complex models. Network fragments exploit the same types of domain structure as do OOBNs. Because they allow complex fragments to be

constructed out of simple fragments, they allow models to be composed hierarchically. Similarly, because they allow the same fragment to be reused in multiple different fragments, they exploit the redundancy present in the domain.

The main difference between the two approaches is that ours focuses on building structured models, while theirs focuses on exploiting the domain structure for the knowledge engineering process, but the constructed models themselves are unstructured. An analogy from programming languages is that network fragments are like macros, which are preprocessed and substituted into the body of a program before compilation. SPOOK class models, on the other hand, are like defined functions, which become part of the structure of the compiled program. The advantages of the two approaches are comparable to those of their programming language analogues. Network fragments, like macros, have the advantage of flexibility, since no assumptions need be made about the relationship between combined fragments. For example, the restriction of OOBNs to part-of relationships was never an issue in the network fragment approach. The SPOOK language, on the other hand, provides a stricter, more semantic approach to combining models. Like structured programming languages, it allows strong type-checking in the way objects are related to each other. The most important advantage of the SPOOK approach is that the models are themselves structured. The domain structure can then be exploited for efficient inference, as explained in Section 4. As our experimental results in Section 5 show, exploiting the domain structure can lead to great computational savings. In addition, because the domain structure is an explicit part of the model, we can now integrate uncertainty over the structure directly into the probability model.

SPOOK provides a bridge between probabilistic reasoning and traditional logic-based knowledge representation. Because it utilizes explicit notions of objects and the relationships between them, SPOOK is able to incorporate and augment the relational models used in many knowledge representation systems. This capability is enhanced by the ability of SPOOK to communicate with such systems through the OKBC protocol.

Our experiences with SPOOK are encouraging. Our hypothesis that exploiting the object structure of a domain can help both in knowledge representation and inference seems to be correct. Of course, we have only worked with one domain, and it remains to be seen if the advantages carry over to other domains. If they do, perhaps the door will be we have opened to a wide range of new applications of Bayesian network technology.

Acknowledgements

Grateful thanks to Suzanne Mahoney, KC Ng, Geoff Woodward and Tod Levitt of IET Inc. for their battlespace models; to Uri Lerner, Lise Getoor and all the other

Phrog hackers; to Barbara Engelhardt and Simon Tong for help with the knowledge engineering; and to Jim Rice for help with integrating with Ontolingua. This work was supported by ONR contract N66001-97-C-8554 under DARPA's HPKB program, by DARPA contract DACA76-93-C-0025 under subcontract to Information Extraction and Transport, Inc., and through the generosity of the Powell foundation.

References

[BFGK96] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. UAI*, 1996.

[CFF+98] V.K. Chaudhri, A. Farquhar, R. Fikes, P. Karp, and J.P. Rice. A programmatic foundation for knowledge base interoperability. In *Proc. AAAI*, 1998.

[HB94] D. Heckerman and J.S. Breese. A new look at causal independence. Technical report, Microsoft Research MSR-TR-94-08, 1994.

[KP97] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proc. UAI*, 1997.

[KP98] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. AAAI*, 1998.

[LM97] K. Laskey and S.M. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *Proc. UAI*, 1997.

[ML96] S.M. Mahoney and K. Laskey. Network engineering for complex belief networks. In *Proc. UAI*, 1996.

[SG97] K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulations. In *Proc. AAAI*, 1997.

[WBG92] M.P. Wellman, J.S. Breese, and R.P. Goldman. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35–53, 1992.