REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-

0459

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		FINAL 15 April 1998- 14 November 2000

4. TITLE AND SUBTITLE
Community Builder: Structuring Agent ARachitectures to Facilitate Domain Task

5. FUNDING NUMBERS
F49620-98-1-0371

6. AUTHOR(S)
Caroline Hayes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Minnesota
Office of Research & Technology Transfer Admin.
1100 Washington South Suite 201
Minneapolis, MN 55415-1226

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
801 N. Randolph Street Room 732
Arlington, VA 22203-1977

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-98-1-0371

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF TRANSMITTAL DTIC. THIS TECHNICAL REPORT
HAS BEEN REVIEWED AND IS APPROVED FOR PUBLIC RELEASE
LAW AFR 190-12. DISTRIBUTION IS UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The results of this work include 1) Community Builder, a design methodology to assist software designers in designing mixed initiative, multi-agent intelligent decision support systems (DSSs), 2) Development of architectures for multi-agent decision support systems in several task domains and 3) Dyna-plan, a reusable framework describing the generic reasoning cycle used in most dynamic and uncertain environments. As the complexity of computing needs continually increases, multi-agent systems are becoming indispensable as approaches for making complex systems modular and manageable. However, designing effective organizations for multi-agent systems is far from simple. Each time an agent-based system is designed to support a new task, designing an effective architectural structure for the system of agents that works effectively for the task is difficult and time-consuming. The aim of the Community Builder and methodology is to provide software designers with a descriptive design methodology that will help them to identify the domain specific constraints on the agent architecture, and to organize the agents accordingly. The aim of the DynaPlan framework is to capture some of the structure common to many planning under uncertainty domains so that other developers can re-use some of the knowledge that we have gained in building related domains.

20010905 126

14. SUBJECT TERMS

15. NUMBER OF PAGES
15

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Community Builder: Structuring Agent Architectures to Facilitate Domain Tasks

Final Report

Project duration: April 1998 - November 2000

Caroline Hayes

University of Minnesota,

Minneapolis, MN.

Grant number: AF/F49620-98-1-0371

Sponsored by AFOSR Program:

"Intelligent Agents for Air Force Battlefield and Enterprise Integration"

Abstract

The results of this work include 1) Community Builder, a design methodology to assist software designers in designing mixed initiative, multi-agent intelligent decision support systems (DSSs), 2) Development of architectures for multi-agent decision support systems in several task domains and 3) Dyna-plan, a re-usable framework describing the generic reasoning cycle used in most dynamic and uncertain environments. As the complexity of computing needs continually increases, multi-agent systems are becoming indispensable as approaches for making complex systems modular and manageable. However, designing effective organizations for multi-agent systems is far from simple. Each time an agent-based system is designed to support a new task, designing an effective architectural structure for the system of agents that works effectively for the task is difficult and time-consuming. The aim of the CommunityBuilder methodology is to provide software designers with a descriptive design methodology that will help them to identify the domain specific constraints on the agent architecture, and to organize the agents accordingly. The aim of the DynaPlan framework is to capture some of the structure common to many planning under uncertainty domains so that other developers can re-use some of the knowledge that we have gained in building related domains.

1 Introduction

In this work we have developed 1) the CommunityBuilder Methodology, to help designers of multi-agent intelligent decision support systems identify domain-related architectural constraints, 2) portions of various decision support systems and architectures including MAPP, DIOS, and Joint-Advisor 3) DynaPlan, a re-usable framework which provides a generalized description of the cycle of activities involved in planning and reasoning under uncertainty.

Decision support systems (DSS) are software systems using a wide variety of techniques, including visualizations, task-oriented editors, and computational analysis, which can help people make decisions faster or better. Decision support systems are becoming increasingly necessary in complex military, engineering, and other decision making tasks to make prevent information overload, and to make rapid response possible.

Intelligent decision support systems are those that perform some part of the problem solving for the decision maker. They may completely automate all of the decision making (as was the case with old-style expert systems), or they may operate as intelligent assistants that provide solution options, evaluations, or critiques to the decision maker. This second style of decision support systems are typically much more acceptable to decision makers because they leave control of critical problem solving tasks and final decisions in the hands of the human.

Intelligent decision support systems are often designed in the form of single or multiple intelligent agents that provide advice. Additionally, since most complex problem solving is not done by individuals but by teams, decision support agents that can provide collaborative support to teams are becoming increasingly important.

Assumptions. All complex decision making tasks have an *information-dependency* structure. For example, decision A may produce information that allows decision B to be made, which in turn produces information that allows decision C to be made. An assumption behind this research is that when designers of multi-agent systems understand the information-dependency structure of a task, they can make better decisions about how to assign decision-making tasks to the various agents, so as to maximize design goals such as minimization of inter-agent communications, or maximization of individual agents' abilities to function independently, etc.) CommunityBuilder is a design methodology intended to help software system designers identify constraints placed on the agent architecture by the information-dependency structure of the task. By helping designers to identify these constraints, CommunityBuilder facilitates designers' efforts to create effective agent-architectures that best meet their design goals.

2 Background and Motivations

This work focuses not just on constructing multi-agent systems, but more specifically on multi-agent systems in complex, single and multi-user, intelligent decision support systems (DSSs). *Intelligent decision support* is an emerging technology that combines automated problem solving techniques artificial intelligence (AI) and operations research (OR) with human computer interaction (HCI). The difference between decision support systems and older style expert systems is that unlike expert systems that solve problems for users, a decision support system works jointly with users to arrive at a

solution. A DSS may have an AI or OR algorithm embedded within it to generate or evaluate solution alternatives, but the human has ultimate control over the development of the final solution. This is essential both in making systems flexible, and acceptable to people.

Engineering and other decision making tasks have become increasingly complex to the point where decision support is almost required to generate solutions quickly and accurately. Complex tasks typically require the collaboration of many specialists. Similarly decision support systems typically require the collaboration of many computer agent specialists. Thus, agent-based system design is an important part of decision support design.

However, constructing a specific agent architecture to best support a specific decision support task depends heavily on the nature of the task domain. Each domain has a specific task and information structure. This structure places strong constraints on how tasks can effectively be assigned to agents and what communication needs to occur between them. We will refer to portion of the agent architecture that is structured by these constraints to be the *task layer*. The CommunityBuilder methodology provides a design process that can guide designers of multi-agent decision support systems in eliciting these constraints. The DynaPlan framework generalizes the common properties of a family of domains, with the goal of facilitating re-use of these common parts for similar task-domains.

3 Objectives

- To assist the developers of multi-agent decision support systems in analyzing task and agent needs,
- To assist in design of a multi-agent system's task layer to support those needs,
- To facilitate rapid updating or adaptation of an existing systems task layer to suit the needs of a new application a closely related domain.

4 Approach

Our approaches for accomplishing these objectives are tightly inter-related. All results grew out of the in depth study and development of architectures for a number of existing decision multi-agent decision support systems: MAPP, Fox, and CoRaven. MAPP is a decision support system to assist engineers in creating manufacturing plans, FOX assists battlestaff to rapidly develop course of action plans, and CoRaven helps intelligence analysts to plan and analyze battlefield information. We chose to look at these particular DSS tool and domains because they were domains with which we already had experience that could be leveraged. Learning a new complex experience-centered domain typically requires several years, so leveraging existing experience was essential. We also hoped that by focusing on a military domain, to facilitate construction of tools that would be of interest to the Air Force and the armed forces in general.

By generalizing the structure of these domains, we produced DynaPlan, a re-usable framework intended for use by DSS designers as general template describing the the reasoning cycles in uncertain environments for many domains. By recording the design

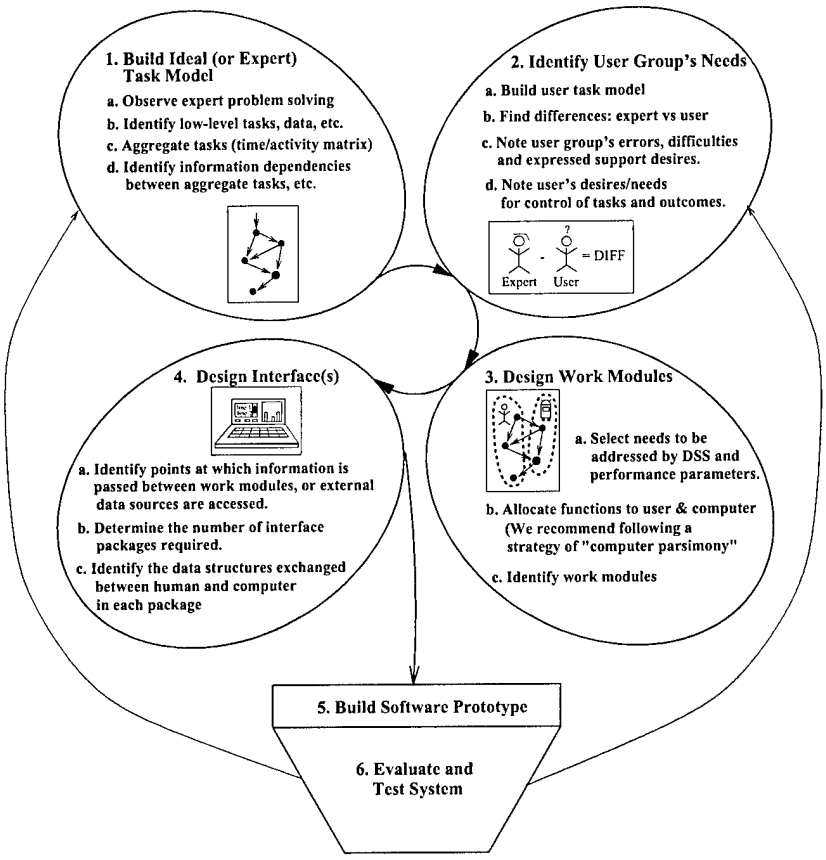

Figure 1: Overview of the Iterative DAISY Methodology

process used to create these DSSs we produced the CommunityBuilder Methodology. Like military doctrine, the CommunityBuilder methodology is intended to be viewed as a set of *guidelines* for DSS designers. It is not prescriptive but *descriptive* of what successful multi-agent DSS designers do when developing system architectures. As practices change, so should the methodology description.

5 Research Accomplishments

5.1 CommunityBuilder Design Methodology

The CommunityBuilder methodology is closely related to many cognitive-engineering methodologies [1, 14] which are commonly used for building decision support and human-computer interfaces. This type of design is also often called human-centered design. In particular, CommunityBuilder extends the the DAISY methodology. [5].

DAISY (Figure 1) is aimed specifically at assisting in design of a *single decision support agent* that interacts with an *individual user*. It assists software designers in understanding where users might desire or benefit from decision support, identifying what functions the decision support agent should perform, and what type information

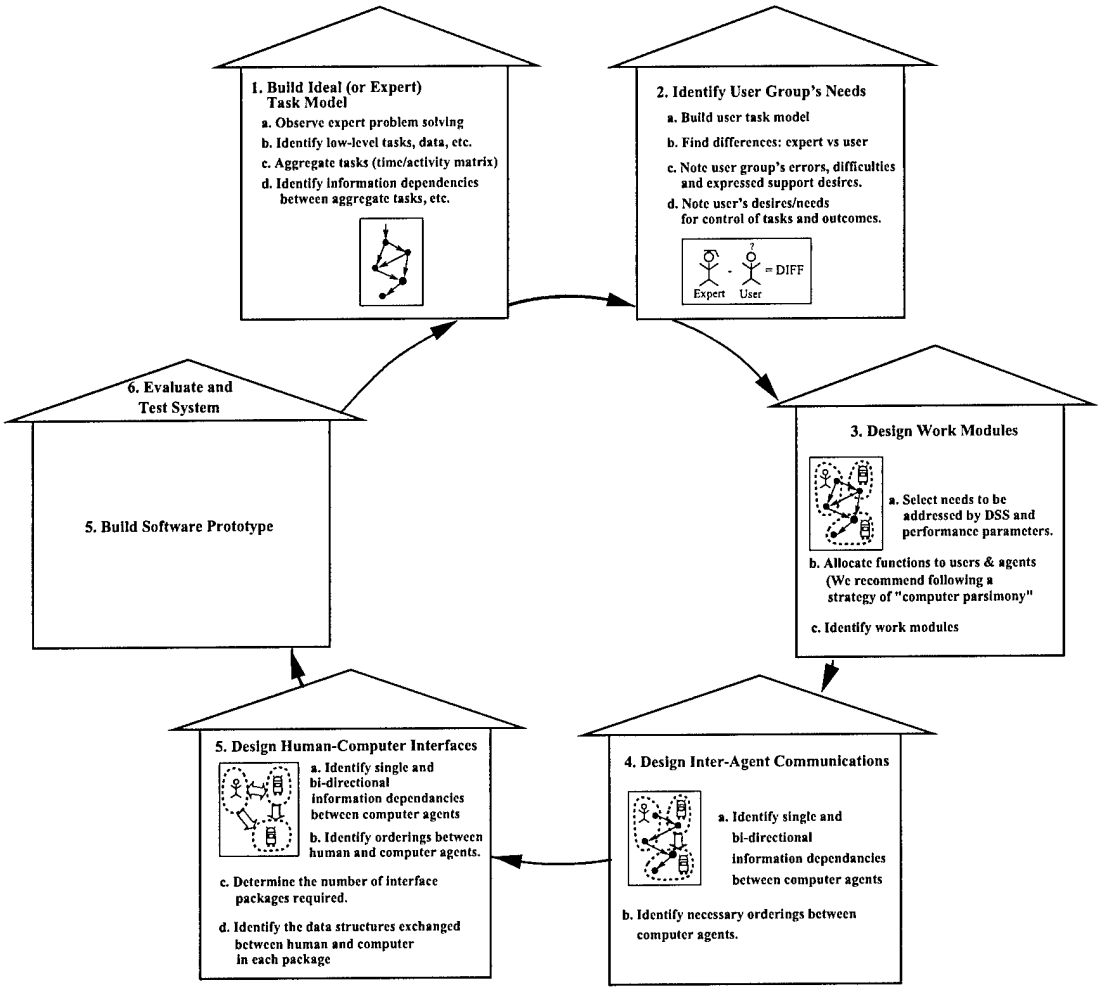

Figure 2: Overview of the CommunityBuilder Methodology

the agent and the user need to exchange. DAISY was developed under Army funding[1] to describe the development process followed to develop FOX [15]. Fox is a decision support tool to help battlestaff rapidly develop a wide variety of friendly courses of action.

CommunityBuilder (Figure 2) extends DAISY to address *multiple decision support agents,* interacting with *multiple users.* Additional challenges raised in such systems include selection of agents (and humans') roles such that work is made more efficient, inter-agent communications are reduced, and inter-agent "thrashing" (passing of control back and fourth between agents many times due to poor division of labor) is minimized.

DAISY's contributions over typical cognitive engineering design methodologies are that it focuses on techniques geared towards modeling very large scale, complex domains in which problems solvers typically need to spend many years developing a high

[1]US Army Research Laboratory, Federated Laboratory Program, Cooperative Agreement Number DAAL01-96-1-0003

level of skill in that type of problem solving. We call these *experience-centered* domains. Design in decision support systems and human-computer interactions in these domains presents very different challenges than do designing interactions in simpler domains, such as interaction with an automated back teller machine. Additionally, DAISY provides methods for identifying the specific type of support functions that users at a given level of domain-experience will need. Understanding the needs of users at different levels of advancement is critical in organizations in which high turn over and training are large issues. The DAISY Methodology and the development of the FOX COA generation tool are described in detail in [2].

CommunityBuilder's addition to DAISY is that it provides a systematic method to handle the assignment of tasks to multiple agents using the constraints inherent in the structure of the task-domain. A basic premise of this work is that those constraints are generated by the inherent information dependencies between tasks which also impose orderings between those tasks, and those constraints can be exploited to make effective assignments of tasks to agents. This can help designers to better understand the trade-offs in making specific agent/task assignments (such greater agent modularity at the expense of additional inter-agent communications) so that they can make more effective design choices.

5.2 MAPP Architectural Design (Veretennikov and Hayes)

The most effective way to study architectural design processes, is to do architectural design (or re-design). In some cases, we created architectural descriptions of existing software that been created "organically" over time, as various parts of the domain were explored and understood. Thus a working DSS tool has evolved, but lacked a "big picture." In others cases, we re-designed architectures to be used for later software development. One of the first such architectures we explored was the MAPP architecture.

MAPP (Manufacturing Architecture for Process Planning) is a multi-agent architecture to guide software developers in constructing single and multi-agent decision support systems for assisting manufacturing planners in tasks such as feature extraction, operations, setup, and fixture planning.

Discoveries made through the MAPP architecture have been very important to the development of CommunityBuilder because it is a domain in which it was very clear that in-appropriate (but typical) assignments of tasks to agents can be demonstrated to result in unnecessary backtracking and lower quality solutions. Typical assignment of tasks to agents in automated planners has been by function. For example, all feature extraction tasks may be handled by one agent, all setup decisions by another, and all fixturing decisions by a third.

However, this "functional" division of tasks may not lend itself well to the information structure of the tasks. For example, human problem solvers do not tend to make all fixturing decisions contiguously. In practice, decisions centered around one topic may need to be distributed through out many stages of problem solving. Some decisions of a certain class may need to be made early so that they can supply information for downstream decisions, while others in the same class may need to be delayed till late in the process when sufficient information becomes available. For example, some fixturing decisions provide information needed for setup sequencing, while setup se-

quencing provides information needed for detailed fixture planning. Grouping all tasks relating to a single function in a single agent can cause reduced solution quality because commitments must sometimes be made when insufficient information is available.

One approach that that some researchers have chosen to address this issue is to allow the (single function) agents to freely pass control back and fourth [3]. However, that practice can lead to unnecessarily complex control structures and much thrashing between the agents.

We feel that a more suitable approach is to first understand the information structure of the tasks (which fixturing decision generates information needed to sequence setups, etc.) and to assign tasks in a way that makes information flow between agents as orderly as possible, which we have done in MAPP. This results in what may, on the surface, appear to be a less intuitive assignment of roles to agents, for example a single agent may be assigned to perform both early fixturing decisions and setup sequencing, while another agent may be assigned to do the remaining detailed fixture design. Instead of being grouped by function, tasks are grouped and assigned to agents according to the decision(s) they support. This results in a much smoother flow of control and better solution quality.

This study and re-design of the MAPP architecture has enabled us to better understand the information structure of the decisions in this domain, and has resulted in re-organization and reassignment of tasks among MAPP's agents (described in [17]).

Through an National Science Foundation Grant, we have created a software implementation of the MAPP architecture. We are currently in the process of further generalizing the MAPP framework by working with other manufacturing planning researchers (Henderson, and Gupta) to combine MAPP with frameworks which these researchers have developed. The goal is to make a very general framework to guides software developers in creating automated planning and decision support software for a wide range of manufacturing problems [11].

5.3 CoRaven Architectural Components

The insight gained from MAPP that information dependencies between tasks could be used to guide assignment of tasks to agents, was an important part of Community-Builder. However, we needed more than the design of one architecture in one domain to create a methodology. Additionally, we explored decision support architectures in a variety of decision support tools designed to assist battlestaff in various part of the Deliberate Decision Making Cycle (DDMC).

Figure 3 shows a version of the DDMC used by all branches of the armed forces. The dark boxes show the components addressed in this project. Specifically, the architectural designs of these components, the DIOS architecture, the Joint-Advisor architecture, and portions of the CoRaven Architecture were created or re-designed as a part of this project focusing of multi-agent architectural design and methodology. Collectively we refer to all these modules together as the CoRaven project. The architectural designs of these DSSs were developed under this grant, focusing on architecture and methodology development. However the majority of the software implementations of these architectures were funded under a series of Army subcontracts focusing on tool development, except when the function of the software was to explore high-level architectural issues such as agent integration, of experiments with star vs. federated control

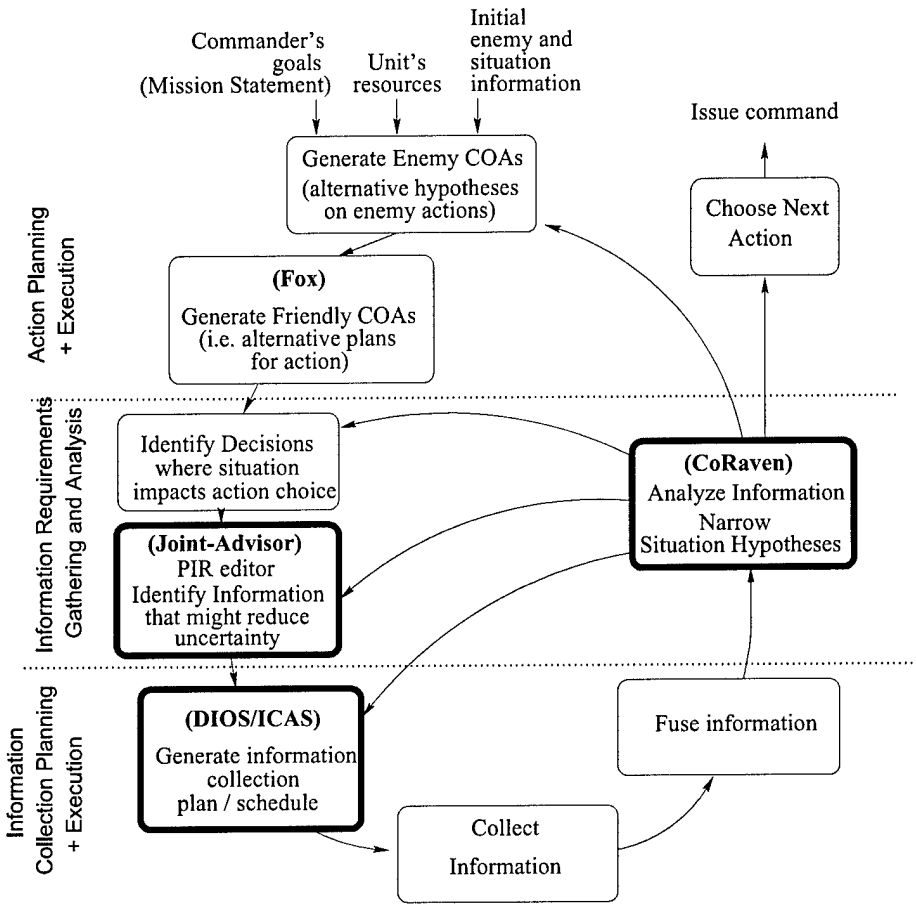

Figure 3: Tools supporting the Deliberate Decision Making Cycle

of the agents. Fox, a tool developed under Army funding and discussed earlier is also depicted in this diagram to show the inter-relations and synergies between the projects.

DIO architecture for Scheduling. (Ergan, Hayes).

The Dynamic Intelligence Operations (DIO) architecture DIO [4] is a general architecture for scheduling information collection tasks. It differs from other scheduling approaches in that its focus is on maximizing value of *information* gathered rather than on maximizing factors such as throughput of operations. Typically, information is gathered in order to make specific decisions. Furthermore, if that information is to be useful, it must be gathered in a timely (i.e. obtain the information *before* the decision has to be made) and reliable manner (send out multiple observers to obtain really critical information), using the available resources in the most cost effective way possible.

DIO was initially designed to apply specifically to scheduling of military

intelligence collection tasks. However, it is general enough to be applied to a a wide range of information gathering tasks such as medical test scheduling (for time-critical diagnostic purposes) or legal information gathering. A software implementation of the DIO architecture, called Intelligence Collection Asset Scheduler (ICAS) has been

funded by the ARMY and has been demonstrated to produce effective collection schedules. Ergan completed his thesis on DIO and ICAS in May 2000. A paper this work has been submitted to IEEE SMC.

JointAdvisor Architecture (Penner, Hayes)

The CoRaven DSS was an intelligence analysis tool developed with the goal of reducing information overload on intelligence analysts. It did so by using a Bayesian Belief Network (BB) [10] to perform much of the work of analyzing battlefield data, and determining which hypotheses about the enemy were most likely. However, analysts found it hard to use this tool in this form because they found it difficult to follow an analysis for which they had not created the logic themselves.

To address this issue, Penner and Hayes studied the existing CoRaven DSS, and created a broader more encompassing architecture, and design for a new tool called Joint Advisor. Joint Advisor added a "logic" editor to CoRaven's capabilities that allowed analysts to enter and edit a logic tree containing their questions and hypotheses about the enemy, (priority information requests), and the observations that they felt would either support or deny each of these hypotheses. Penner produced a domain model on which to base the design of JointAdvisor, designed the architecture including packages and interactions, the interaction concept and the use model. This architecture was later used in an Army funded project to guide the implementation of a JointAdvisor software tool.

5.4 Dyna-Plan (Hayes)

Dyna-Plan is a re-usable architectural framework that we developed by generalizing the problem solving phases in the deliberate decision making cycle. After study of several domains including military intelligence planning, and robotic sensor planning, we realized that this cycle could form the basis of a description for a wide variety of planning under uncertainty tasks.

A re-usable framework is not the same as re-usable software (although this is sometimes the case). Instead, it refers to a "road-map" or architectural structure of a system: the agents, the tasks they perform, and the information exchange between them. A re-usable framework saves the system designer time, by providing a system design "template" capturing the common structure of a family of domains. Only minimal domain modeling and agent structure design will need to be done to create an agent system architecture appropriate for the new domain with in the family.

Much work in planning under uncertainty focuses on specific techniques for representing uncertain concepts [8], reasoning about uncertainty [9], or planning under incomplete [12], uncertain [18] or dynamic [7] circumstances. However, little is written about high-level strategies or the structure of problem solving under uncertain circumstances.

After studying the structure of reasoning in military maneuver and intelligence planning, which occurs in an highly uncertain and dynamic environment, it because apparent to us that:

1. The structure of the over-all problem solving cycle (plan, gather information, replan, etc) was as important to success (if not more important) than the individual

Dyna-Plan

A Framework Planning and Reasoning Cycle
for Dynamic and Uncertain Environments

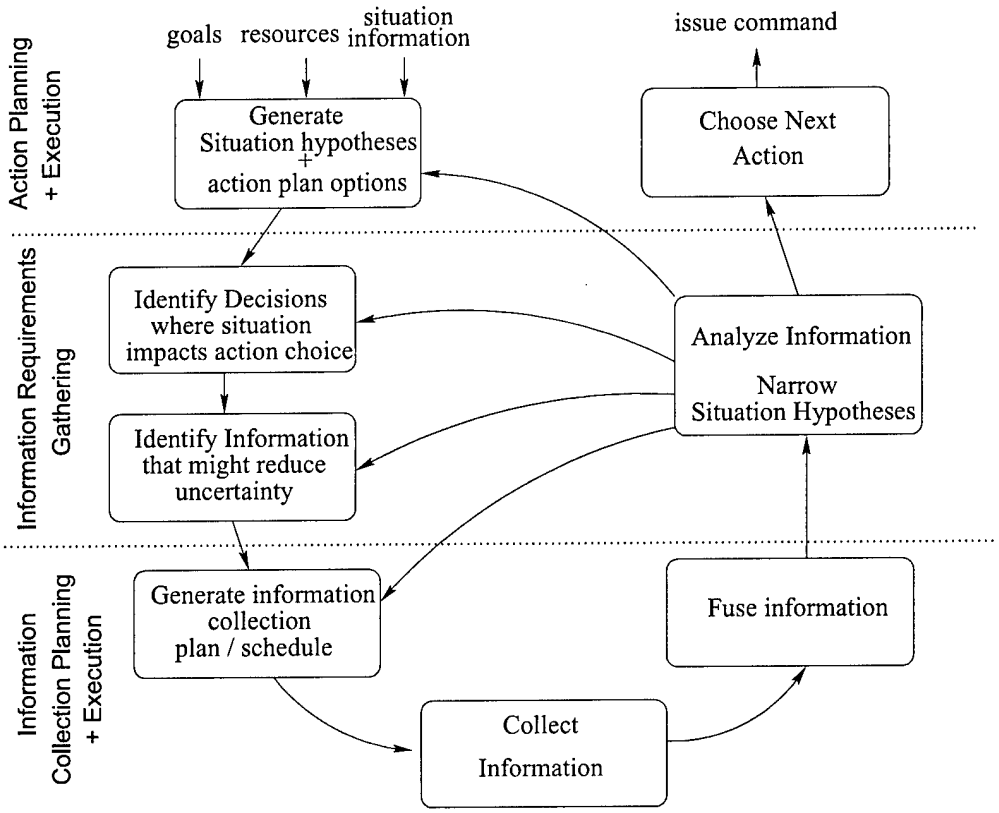

Figure 4: DynaPlan: A general framework for reasoning under dynamic and uncertain circumstances.

techniques employed, such as Bayesian Belief Networks, fuzzy logic, or certainty factors.

2. Many reasoning under uncertainty domains have a very similar problem solving cycle
3. It would be useful to many researchers and software developers to have a "template" of that cycle available to them to assist them during development efforts in understanding its various aspects.

Dyna-Plan is the framework we have created aimed at capturing the common architectural process structure of for planning and reasoning under dynamic and uncertainty circumstances in a wide range of domains. DynaPlan is outlined in Figure 4.

The major tasks in the Dyna-Plan architecture were derived and generalized from the Deliberate Decision Making Cycle. The general cycle followed in Dynaplan is: gen-

erate several hypotheses about what the current (or past, or future) situation is, based on currently available data; generate several plans to address each of those situations; identify key pieces of unknown information that will help the planner identify which plan(s) to follow; plan how to use available resources to get as much of that information as possible; use the new information to narrow the set of situation hypotheses; if an action must be taken now decide which one appears best given current information and execute it; go through the planning cycle again.

An important aspect of the DynaPlan cycle is that the planning cycle for actions under dynamic uncertain and uncertain circumstances has another planning cycle embedded inside it: a planning cycle for focused gathering of information to reduce uncertainty about what action to take next. DynaPlan both recognizes and highlights the fact that focused information gathering is a powerful and general technique reducing uncertainty.

Many problem solving cycles under uncertainty can be described by the DynaPlan framework. Figure 5 shows an example of the DynaPlan framework instantiated for the Medical domain. Not all domains will go through all stages listed in the DynaPlan framework, some steps may be absent or less prevalent in some domains. However, the over-all cycle will remain the same.

For example, robot motion planning follows the dynaplan cycle although, in some cases the information planning phase is is less prevalent. This happens when the robot's sensors have fixed capabilities, such as a simple touch sensor. When there are no choices as to what to do with the sensors, planning of information gathering is unnecessary. The only information gathering plan available may simply to be to read all the sensors. However, for robots with more complex sensors, such as a robot with a camera that can be aimed in various directions, the information (also known as sensor planning) phase becomes more complex. Information collection planning is also required when processing the information receive may exceed the robot's computational capabilities, for example a driving robot may have to choose to process only one of several images, or choose areas of a single image to process in order to avoid high speed collisions on the highway.

DynaPlan Impact. By explicitly creating a generic framework that describes the reasoning under uncertainty cycle, and by casting various domains into this framework, we hope to make clear the connections and similarities between many diverse planning domains: the military deliberate decision making cycle, robot motion and sensor planning, medical diagnostic planning, geologic exploration planning, etc.

In the future, we also hope to associate various reasoning techniques with various parts of the DynaPlan cycle. For example, many contingency planning techniques [13, 16] can be viewed as different ways of accomplishing portions of the first DynaPlan step: generate action plan options. Bayesian Belief nets or fuzzy logic [9] are examples of techniques that might be used to perform the analyze information step. We hope that by providing the framework, augmented with catalogs of techniques that have been applied to various parts of the cycle, we can help to organize planning and uncertainty work from many fields (robotics, operations research, artificial intelligence, military science) into a common framework, and assist researchers in identifying techniques from many disciplines that they might apply to their problems.

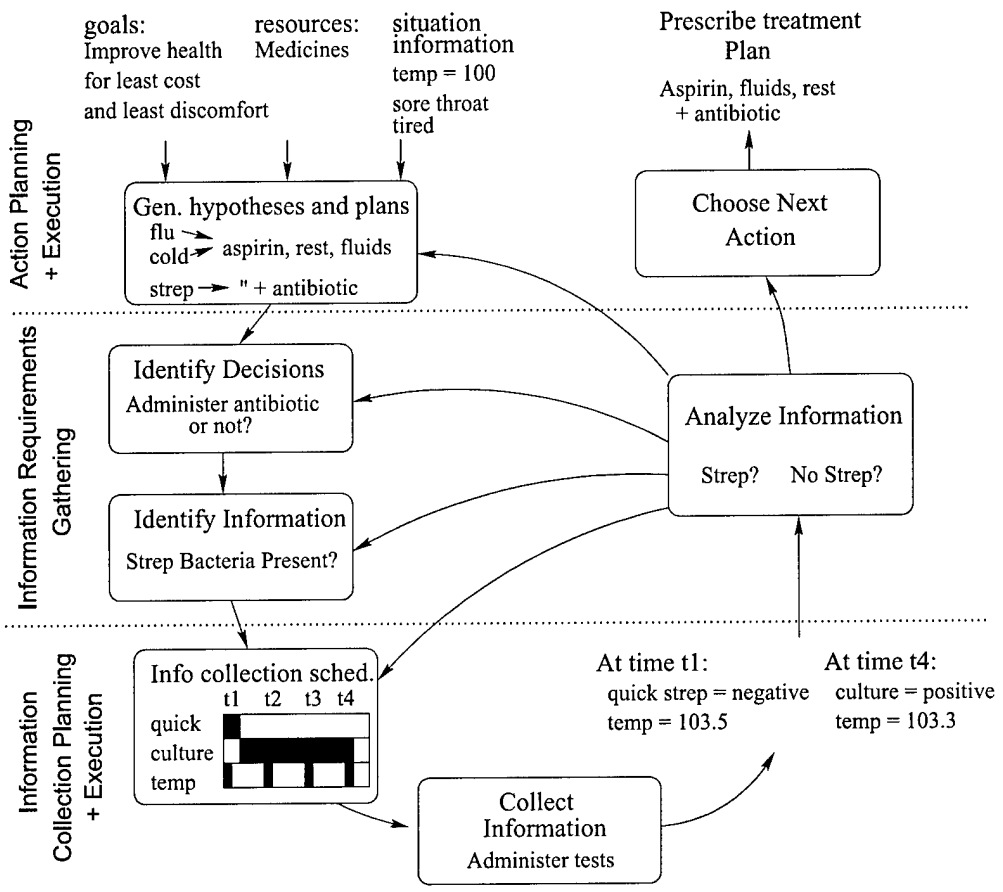

Figure 5: DynaPlan instantiated for the medical reasoning domain

6 Lessons learned

Centralized control, processing, interface, communication/client record, data store, are separate issues.

For decision support tasks, the organization of computer agents needs to closely mirror organization of the human users.

7 Summary

The contributions of this work are to provide 1) CommunityBuilder, a systematic methodology to help DSS designers identify, understand, and design for the information structure of complex experience-centered domains, 2) a demonstration of CommunityBuilder's effectiveness through development of as set of architecture for complex, practical domains, and 3) DynaPlan, a re-usable framework capturing the generic structure of these domains to facilitate re-use of these knowledge gained in these efforts. The

expected impact is faster multi-agent DDS system development, better agent organizations that work with rather than against the inherent task structure, and better performing more usable DSS tools.

8 Primary People and Activities

- Caroline Hayes, Principle Investigator, research director, developed the Dyna-Plan framework, which generalized CoRaven's structure and provides a general template for iterative reasoning cycles in dynamic and uncertain situations.
- Nan Tu, graduate graduate research assistant, designed initial CoRaven architecture, and developed "TaskICDA methodology," the first version of the Community Builder methodology.
- Aleksey Veretennikov, graduate research assistant, extended MAPP architecture.
- Miner Liang, graduate research assistant, performed experiments with star and federated control architecture for coordinating CoRaven agents.
- Hakan Ergan, graduate research assistant, designed the DIOS scheduling architecture, a portion of the CoRaven architecture.
- Li Lu, graduate research assistant, integrated all parts of the CoRaven framework into a unified architecture.
- Robin Penner, Research Scientist, Designed the Joint-Advisor portions of the CoRaven Architecture.

9 Publications and Theses supported in part or whole by this grant.

Journal and Conference Papers

Hayes, C. C. and C. B. F. Brodie, "CommunityBuilder: A Methodology for Designing Mixed-Initiative Multi-Agent Systems," *Intelligent Autonomous Systems (IAS) Conference*, Venice, Italy, July 25-27, 2000, pp. 736 - 743.

Hayes, C. C.; N. Tu, H. Ergan, L. Lu, P. Asaro, P. M. Jones, "Model-Based Design of Decision Support for Real-Time Information Assessment," *IEEE Transactions on Systems, Man and Cybernetics,* Special Issue on Model-Based Design, editors C. Mitchel and P. M. Jones, accepted for publication in 2002.

Ergan, H, and C. C. Hayes, "Design of a Decision Support System for Army Intelligence Collection Resource Scheduling", Submitted to IEEE SMC.

A journal paper on Dyna-plan is in preparation.

Master's and PhD Theses

Nan Tu, PhD Thesis,
"TASK-ICDA: A Methodology for Developing Multi-Agent Multi-User Decision Support Systems", Expected defense, Sept 2001, Department of Mechanical Engineering, University of Minnesota.

Hakan Ergan, Masters thesis
"Design of A Decision Support System for Army Intelligence Collection Resource Scheduling" May 2000, Department of Mechanical Engineering, University of Minnesota.

Miner Liang, Masters thesis (plan B)
"MIMOSA and Concept-Linker: Map Tools for Facilitating Intelligence Analysis." May, 1999, Department of Computer Science, University of Minnesota.

References

[1] R. W. Bailey. *Human Performance Engineering.* Prentice Hall, New Jersey, third edition, 1996.

[2] C. B. Brodie and C. C. Hayes. Daisy: A decision support design methodology for large-scale, complex, experience-centered domains. *IEEE Transactions on Systems, Man and Cybernetics*, 2002. Special Issue on Model-Based Design.

[3] M. Cutkosky and J. Tenenbaum. Towards a framework for concurrent design. *International Journal of Systems Automation: Research and Applications*, 1(3):239–261, 1992.

[4] Hakan Ergan. Design of a decision support system for army intelligence collection resource scheduling. Master's thesis, University of Minnesota, May 2000.

[5] Carolyn Brittan Fiebig. *DAISY: Designing Experience-Centered Decision Support Systems.* PhD thesis, University of Illinois at Urbana-Champaign, 1999.

[6] C. C. Hayes, N. Tu, H. Ergan, L. Lu, P. Asaro, and P. M. Jones. Model-based design of decision support for real-time information assessment. *IEEE Transactions on Systems, Man and Cybernetics*, 2002. Special Issue on Model-Based Design.

[7] L. P. Kaelbling. *An Architecture for Intelligent Reactive Systems*, pages 713–728. Morgan Kaufman, San Mataeo, CA, 1990.

[8] L. V. A. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, July/August 2001.

[9] G. F. Lugar and W. A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving, second edition.* Benjamin/Cummings Publishing Company, Redwood City, CA, 1993.

[10] O. J. Menshoel and D. C. Wilkins. Genetic algorithms for belief network inference: The role of scaling and niching. *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, 1998.

[11] A. Montelaro, M. R. Henderson, C. A. Roberts, N. F. Hubele, C. C. Hayes, and S. K. Gupta. A comparison method for automated manufacturability analysis systems *amas). *Submitted to Journal of Intelligent Manufacturing*, expected publication 2002.

[12] D. Olawsky and M. Gini. *Deferred Planning and Sensor Use*, pages 166–174. Morgan Kaufman, San Diego, CA, November 1990.

[13] D. Payton. Exploiting plans as resources for action. In K. P. Sycara, editor, *Proceedings of the Workshop on Innovative Approaches to Planning and Control*, pages 175–180, San Diego, CA, November 1990. Morgan Kaufman.

[14] J. Rasmussen, A. M. Pejtersen, and L. P. Goodstein. *Cognitive Systems Engineering*. John Wiley and Sons, Inc., 1994.

[15] J. L. Schlabach, C. C. Hayes, and D. E. Goldberg. Fox-ga: A genetic algorithm for generating and analyzing battlefield courses of action. *The Evolutionary Computation Journal*, 7.1, 1999.

[16] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1039–1046, Milano, Italy, 1987.

[17] Aleksey Veretennikov. Providing clampability analysis for multiple fixtures to support setup sequencing in process planning. Master's thesis, University of Minnesota, May 2000.

[18] J. S. Zelek. A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments. In *Workshop on Integrated Planning, Scheduling and Execution in Dynamic and Uncertain Environments, held at the 4th International Conference on Artificial Intelligence in Planning Systems*, Pittsburgh, PA, June 1998. AAAI Technical Report WS-98-02.