DISSERTATION

ARTIFICIAL INTELLIGENCE BASED DECISION SUPPORT FOR TRUMPETER SWAN MANAGEMENT

Submitted by

Richard S. Sojda

Department of Forest Sciences

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2002

COLORADO STATE UNIVERSITY

December 13, 2001

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY RICHARD S. SOJDA ENTITLED ARTIFICIAL INTELLIGENCE BASED DECISION SUPPORT FOR TRUMPETER SWAN MANAGEMENT BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

[redacted]
Leigh H. Fredrickson, Forest Sciences

[redacted]
Adele E. Howe, Computer Science

[redacted]
John B. Loomis, Agricultural and Resource Economics

[redacted]
Denis J. Dean
Adviser

[redacted]
Susan G. Stafford
Department Chair

ABSTRACT OF DISSERTATION

ARTIFICIAL INTELLIGENCE BASED DECISION SUPPORT FOR TRUMPETER SWAN MANAGEMENT

The number of trumpeter swans (*Cygnus buccinator*) breeding in the Tri-State area where Montana, Idaho, and Wyoming come together has declined to just a few hundred pairs. However, these birds are part of the Rocky Mountain Population which additionally has over 3,500 birds breeding in Alberta, British Columbia, Northwest Territories, and Yukon Territory. To a large degree, these birds seem to have abandoned traditional migratory pathways in the flyway. Waterfowl managers have been interested in decision support tools that would help them explore simulated management scenarios in their quest towards reaching population recovery and the reestablishment of traditional migratory pathways. I have developed a decision support system to assist biologists with such management, especially related to wetland ecology. Decision support systems use a combination of models, analytical techniques, and information retrieval to help develop and evaluate appropriate alternatives. Swan management is a domain that is ecologically complex, and this complexity is compounded by spatial and temporal issues. As such, swan management is an inherently distributed problem. Therefore, the ecological context for modeling swan movements in response to management actions was built as a multiagent system of interacting intelligent agents that implements a queuing model representing swan migration. These agents accessed ecological knowledge about swans, their habitats, and flyway management principles from three independent expert systems. The agents were autonomous, had some sensory capability, and could respond to changing conditions. A key problem when developing ecological decision support systems is empirically determining that the recommendations provided are valid. Because Rocky Mountain trumpeter swans have been

surveyed for a long period of time, I was able to compare simulated distributions provided by the system with actual field observations across 20 areas for the period 1988-2000. Applying the Matched Pairs Multivariate Permutation Test as a statistical tool was a new approach for comparing flyway distributions of waterfowl over time that seemed to work well. Based on this approach, the empirical evidence that I gathered led me to conclude that the base queuing model does accurately simulate swan distributions in the flyway. The system was insensitive to almost all model parameters tested. That remains perplexing, but might result from the base queuing model, itself, being particularly effective at representing the actual ecological diversity in the world of Rocky Mountain trumpeter swans, both spatial and temporally.

The Distributed Environment Centered Agent Framework (DECAF) was successful at integrating communications among agents, integrating ecological knowledge, and simulating swan distributions through implementation of a queuing system. The work I have conducted indicates a need for determining what other factors might allow a deeper understanding of the effects of management actions on the flyway distribution of waterfowl. Knowing those would allow the more refined development of algorithms for effective decision support systems via collaboration by intelligent agents. Additional, specific conclusions and ideas for future research related both to waterfowl ecology and to the use of multiagent systems have been triggered by the validation work.

Richard S. Sodja
Forest Sciences Department
Colorado State University
Fort Collins, CO 80523
Spring 2002

ACKNOWLEDGEMENTS

Research that spans nearly ten years is bound to need the cooperation of many good-hearted souls; it is hard to count and name them all. My dear family, Mary Ann, Kate, and Neal, have had to contribute the most. To say I am grateful is the truth, but seems trivial.

My Graduate Committee...

I have been blessed with an outstanding committee. Denis Dean had the fortitude to take on a student who had been deemed "too old" by others, who was working full-time, and who wanted to do interdisciplinary research. I will always appreciate his insights, and thank him for recommending "Flatland." The counsel of Adele Howe saved me from burnout. Her willingness to share her expertise and insights with a novice has been phenomenal. The direction she provided in helping me comprehend the artificial intelligence literature and apply algorithms to real problems is truly the foundation of this dissertation. The initial ecological ideas for my work were launched during discussions with Leigh Fredrickson in the early 1990s. I am so very thankful for all his time, and all he has tried to teach me about wetlands and waterbirds. There is nothing like seeing a marsh through his eyes. Along with Leigh, John Cornely helped form the initial concept of all that is involved in "thinking like a flyway." He was willing to take a risk on what looked like weird research to many refuge managers; still, he found funding for the project throughout its duration. John Loomis' perspectives on public lands management helped me realize that there is a socioeconomic and political basis for my interest in

connecting science and management, as well as an ecological one. Although Susan Stafford is not an official member, she gets the credit for ending the stagnation and insisting that there was light at the end of the dissertation tunnel.

Additional Friends and Colleagues...

D. Hamilton first introduced me to expert systems on some now ancient computer equipment at Blackwater National Wildlife Refuge. He was the ideal partner in conducting the knowledge engineering sessions. This brings me to thanking all the experts who participated in those sessions: J.B. Bortner, S. Bouffard, T. Grant, J. Kadlec, M. Laubhan, D. Sharp, R. Trost, and G. Will. The experimental testing of the system would not have been possible without direct involvement of the following individuals in running all three expert systems for 13 years for their respective areas: V. Hirschboeck (Bear River Migratory Bird Refuge); K. Hobbs (Harrimann State Park); C. Mitchell (Grays Lake National Wildlife Refuge); R. Munoz (Southeast Idaho National Wildlife Refuge Complex); D. Olson (Red Rock Lakes National Wildlife Refuge); and S. Patla (Wyoming Department of Fish and Game).

There are programmers who humored my algorithms and system administrators who managed to remain good friends: L. Bogar, K. Bowers, S. Lee-Chadde, and C. Wright. But, most of all, multiple "thank you's" to D. "T.Y." Zarzhitsky who inherited all the loose ends. He not only made sense where others left off, but was responsible for documenting almost all the code. Many thanks to K. Decker and J. Barbour for their assistance in learning DECAF, sharing many key papers, and understanding some of its theoretical underpinnings. L. Lucke provided superb library services. L. Landenburger prepared Figure 3-4. J. Cherry was responsible for final word processing of the dissertation.

I have had the pleasure of being taught by many outstanding professors over the years, nearly all retired now, whose ideas found their way into this dissertation; D.F. Cox (Iowa State University) helped de-mystifying statistics by always having me look at data plots and think about experimental units; E. Klaas (Iowa State University and U.S. Fish and Wildlife Service) taught me about conceptualizing research projects, start to finish; P. Mielke (Colorado State University) helped my feeble brain try to understand permutation methods and then apply them to multivariate analyses; R. Oglesby (Cornell University) is remembered for insisting to include "time" as a parameter in ecological processes; and B. Wilkins (Cornell University) has been a mentor and friend in so many ways. Other colleagues who provided expertise and moral support over the years include: D.H. Cross, F. D'Erchia, A. Gallant, D. Goodman, L. Hanson, D. Jennings, W. Ladd, D. Ouren, J. Ringleman, R.C. Solomon, T. Stohlgren, and W. King.

R. Stendell was always a strong supporter and backed his own professional support with that of our mother agency and her funds. He turned an ugly duckling of a bureaucratic situation into a beautiful swan! And, thanks to R. Jachowski for picking up where Rey left off.

Funding Credits...

This project was funded jointly by units of the Department of Interior: the Geological Survey, Biological Resources Division - Midcontinent Ecological Science Center and Northern Rocky Mountain Science Center; and the Fish and Wildlife Service, Region 6 - Division of Migratory Birds, and (earlier) Region 8 - Research. It was administered as Geological Survey, Biological Resources Division Project Number 915. I acknowledge the technical support of the Pacific Flyway Council, Subcommittee on the Rocky Mountain Population of Trumpeter Swans of the Pacific Flyway Study Committee.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 The Ecological Context: Flyway Management of Trumpeter Swans

Trumpeter swans were once thought to be extinct, but the discovery of breeding birds in the Centennial Valley of Montana and in Yellowstone National Park led to the establishment of Red Rock Lakes National Wildlife Refuge where swan management became an important function. Additional national wildlife refuges and other national, state, and provincial lands often have capabilities for managing wetlands as trumpeter swan habitats. The system of refuges and other wildlife management areas has purposefully been dispersed along migratory pathways at areas important to pre-breeding, breeding, post-breeding, and wintering migratory birds, especially those dependent on wetland habitats. Most migratory birds travel such corridors linking northern breeding areas with more southern wintering grounds.

During the mid-twentieth century, a system of four Flyway Councils for collaboratively managing such waterfowl populations and their habitats in North America was established between the U.S. Fish and Wildlife Service, Canadian Wildlife Service, and the provincial and state wildlife agencies. This system has been quite successful and has made significant contributions towards ensuring that waterfowl management in North America is based on empirical research. I was approached by swan managers from the U.S. Fish and Wildlife Service (on behalf of the Subcommittee on the Rocky Mountain Population of Trumpeter Swans of the Pacific Flyway Study Committee) who expressed an interest in having a decision support tool that would simulate and test

management options for trumpeter swans throughout migration corridors. The ultimate objective of using such a tool is to progress towards both population recovery and migration path development.

The number of trumpeter swans breeding in the Tri-state Region, where Montana, Idaho, and Wyoming adjoin, has always been limited, but has declined approximately 30 percent since the peak numbers of the 1960s, and now there are fewer than 400 total birds. Apparently they also have abandoned, to a large degree, what were thought to be traditional migratory pathways. National wildlife refuges such as Grays Lake, Red Rock Lakes, National Elk Refuge, and Bear River Migratory Bird Refuge; national parks such as Yellowstone and Grand Teton; and other areas such as Harrimann State Park (ID) share swans at different times of the year. However, these birds are part of the Rocky Mountain Population which additionally has over 3,500 birds breeding in Alberta, British Columbia, Northwest Territories, and Yukon (Subcommittee on Rocky Mountain Trumpeter Swans 1998; Reed 2000.) The northerly breeding birds have tended to share wintering habitats, and to a lesser degree postbreeding and prebreeding habitats, with their Tri-state Region counterparts, increasing management complexity. At its most fundamental level, this complexity stems from a lack of published ecological knowledge and the inability to integrate knowledge across temporal and geographical scales. Such understanding and integration should involve management of wetlands at a site specific level while incorporating flyway management principles. This is compounded by the need for mechanisms and tools by which the many agencies involved can collectively develop, simulate, and empirically evaluate management options and activities.

Managers recognize that decision making is cyclic, and they wish to iteratively plan, implement, evaluate, and improve their management strategies. Unfortunately, optimizing management of migratory birds throughout a flyway with cyclic planning is so

complex that it is often all but impossible to implement without computerized decision support (Sojda, Dean, and Howe 1994). Also, past conditions and future needs are ecological constraints to current decisions. Swan management is complex, requiring reasoning through time and space among geographically dispersed areas. Spatial interactions are inevitably intertwined with temporal components as swans migrate. This is further compounded by ecological issues that exist at specific wetlands in each location. Another component of complexity arises from local decisions having ramifications not only at other sites but also at a population level within the migration corridor.

1.2 The Decision Support Context: Distributed Problems and Multiagent Systems

Decision support systems use a combination of models, analytical techniques, and information retrieval to help develop and evaluate appropriate alternatives (Adelman 1992; Sprague and Carlson 1982); and such systems focus on strategic decisions and not operational ones. More specifically, decision support systems should contribute to reducing the uncertainty faced by managers when they need to make decisions regarding future options (Graham and Jones 1988). Distributed decision making suits problems where the complexity prevents an individual decision maker from conceptualizing, or otherwise dealing with the entire problem (Boland et al. 1992; Brehmer 1991). It is in this light that I chose to develop a decision support system to assist biologists with swan management, especially related to wetland ecology. At this time, there are no such systems available for swan managers, nor any common databases for them to access. Furthermore, many managers are either located in relatively remote locations or simply distant from each other, making it difficult to meet frequently. On national wildlife refuges and some other areas, annual water management plans are prepared for individual wetlands. These typically are prepared

manually and often do not take into account conditions in other areas of the flyway except in a general sense. Plans are not usually updated during the course of the year. In light of all this, population goals have not yet been achieved from both the past and current situations for managing trumpeter swans, of which planning is only a part.

Management of migratory birds is inherently distributed in time and space. Many artificial intelligence-based methodologies, particularly those related to cooperative distributed problem solving (Carver, Cvetanovic, and Lesser 1991; Durfee, Lesser, and Corkill 1989) and multiagent systems (Weiss 1999) also are designed to address distributed problems and therefore were deemed appropriate for adapting to the swan management domain. I chose to use intelligent agents linked in a multiagent system as the overall building blocks. Agents should have three characteristics: (1.) some degree of autonomy, (2.) the ability to collect information about their environment, and (3.) the capability to independently take action, or at least respond to their perceptions. I attempted to ensure that these criteria were incorporated in the agents I developed.

The belief-desires-intentions (BDI) agent architecture summarized by Wooldridge (1999) and Rao and Georgeff (1995) is the foundation upon which intelligent agents often are conceptualized. It is such an architecture upon which the Distributed Environment Centered Agent Framework (DECAF) (Graham and Decker 2000; Graham 2001) was developed. DECAF was used to build the overall decision support system for trumpeter swan management, handling agent operation and management as well as requests to the user for information. The ecological context for modelling swan movements in response to management actions was conceptualized as a queuing model (Dshalalow 1995; Hillier and Lieberman 1995). Interacting agents, provided through DECAF, provide the knowledge and problem solving capabilities of the multiple entities needed to implement the ecological model.

The queuing model, itself, represents the spatial and temporal distributions of swans. It uses as one primary input the number of swans in year, t, observed in 27 areas and the probability of movement from season to season among those areas as another input. The model then projects the number of swans at time, $t+1$, across those same areas. Key ecological knowledge is used to adjust the inputs to the queuing model, resulting in adjustments to the predicted numbers of swan across the 27 areas. To provide this knowledge, fundamental elements of the 1998 Management Plan for the Rocky Mountain Population of Trumpeter Swans (Subcommittee on Rocky Mountain Trumpeter Swans 1998), plus vital ecological knowledge about swans, their habitats, and flyway management principles were developed as expert systems. These latter components assist the user in providing information to the overall system.

A key problem when developing ecological decision support systems is empirically determining that the recommendations provided are valid. However, because trumpeter swans have been routinely surveyed in the Tri-state Region since the early 1970s, the opportunity existed to compare predicted numbers and distributions of swans over a long period of time. The use of historic data allowed me to look back in time and address questions such as: "Had swans been managed in the manner suggested in the plan, would the distribution of swans been different than actually observed?" This of course, is based on the observation that time and both administrative and management changes did not allow unimpaired implementation of the 1998 Plan. The system is now available to swan managers so that they can examine the effect of potential management actions on the distribution of swans.

1.3 Organization of the Dissertation

The broad question I have tried to answer is: Are multiagent systems an effective platform for integrating flyway management into site-specific decision support for trumpeter swan management? In this context, I also was interested in determining the effect that encoded ecological knowledge could have on simulated distribution of swans as represented by a queuing system. This required choosing an appropriate, artificial intelligence methodology for the ecological issues; implementing the system in a manner amenable to verification and validation; and, then empirically evaluating that system. This dissertation is therefore organized accordingly. Section III, "Applying Cooperative Distributed Problem Solving Methods to Trumpeter Swan Management", discusses the inherent distributed nature of trumpeter swan management and what artificial intelligence methodologies are appropriate for distributed problems. Section IV, "Implementation of a Multiagent Model to Decision Support for Trumpeter Swan Management", describes intelligent agents, queuing systems, and the framework developed for constructing the decision support system. Section V, "Empirical Evaluation of a Multiagent System for Trumpeter Swan Management", explains the verification and validation deployed. It also presents the conclusions reached regarding decision support system development and the use of multiagent systems for trumpeter swan management.

CHAPTER 2

APPLYING COOPERATIVE DISTRIBUTED PROBLEM SOLVING METHODS TO TRUMPETER SWAN MANAGEMENT

2.1 NOTE: Prior Publication

This chapter has been published as the following paper:

> Sojda, R. S., and A. E. Howe. 1999. Applying cooperative distributed problem solving methods to trumpeter swan management. Pages 63-67 *in* U. Cortes and M. Sanchez-Marre, eds. Environmental Decision Support Systems and Artificial Intelligence. American Association for Artificial Intelligence Technical Report WS-99-07. AAAI Press. Menlo Park, CA.

Chapter III was written approximately two years prior to actually building the system, itself. During that time, certain commercial expert system software options ceased to exist, forcing my re-examination of using them as an implementation of blackboards and cooperative distributed problem solving methods. During that same time, multiagent system software options in the public domain increased and were then adopted as an alternative methodology. Therefore, some of the implementation details mentioned in Chapter III were supplanted by those described in Chapter IV. However, basic concepts related to distributed problem solving remained the same.

2.2 Abstract

We are developing a decision support system in an effort to assist biologists who are managing habitats for the Rocky Mountain population of trumpeter swans. Swan management is a domain that is ecologically complex, and this complexity is compounded by spatial and temporal issues. We are focused on providing decision support that allows managers to develop habitat management plans for local sites while

recognizing that such decisions have ramifications not only at other sites but in the flyway as a whole. Because swan management is an inherently distributed problem, our system utilizes artificial intelligence methods including cooperative distributed problem solving, blackboards, and expert systems. The system will be made available to swan managers through the World Wide Web, using commercially available software that provides a common gateway interface between the web server software and an inference engine.

2.3 The Inherently Distributed Nature of Trumpeter Swan Management

We are developing a decision support system to assist biologists with the management of the Rocky Mountain population of trumpeter swans (*Cygnus buccinator*). The number of swans breeding in the Tri-State area where Montana, Idaho, and Wyoming come together has declined to just a few hundred pairs. They have abandoned, to large degree, what were thought to be traditional migratory pathways. Swans, like most migratory birds in North America, travel along migration corridors that link northern breeding areas with more southern wintering grounds. National wildlife refuges such as Grays Lake, Red Rock Lakes, National Elk Refuge, and Bear River Migratory Bird Refuge, share swans at different times of the year with national parks such as Yellowstone and Grand Teton, and other areas such as Harrimann State Park. Swan management is a complex domain requiring reasoning across time and space among geographically dispersed managers (Figure 2-1.) Spatial interactions are inevitably intertwined with a temporal component as swans migrate. This complexity is further compounded by ecological issues that exist at specific wetlands in each location.

Another component of complexity arises from local decisions having ramifications not only at other sites but in the flyway as a whole.

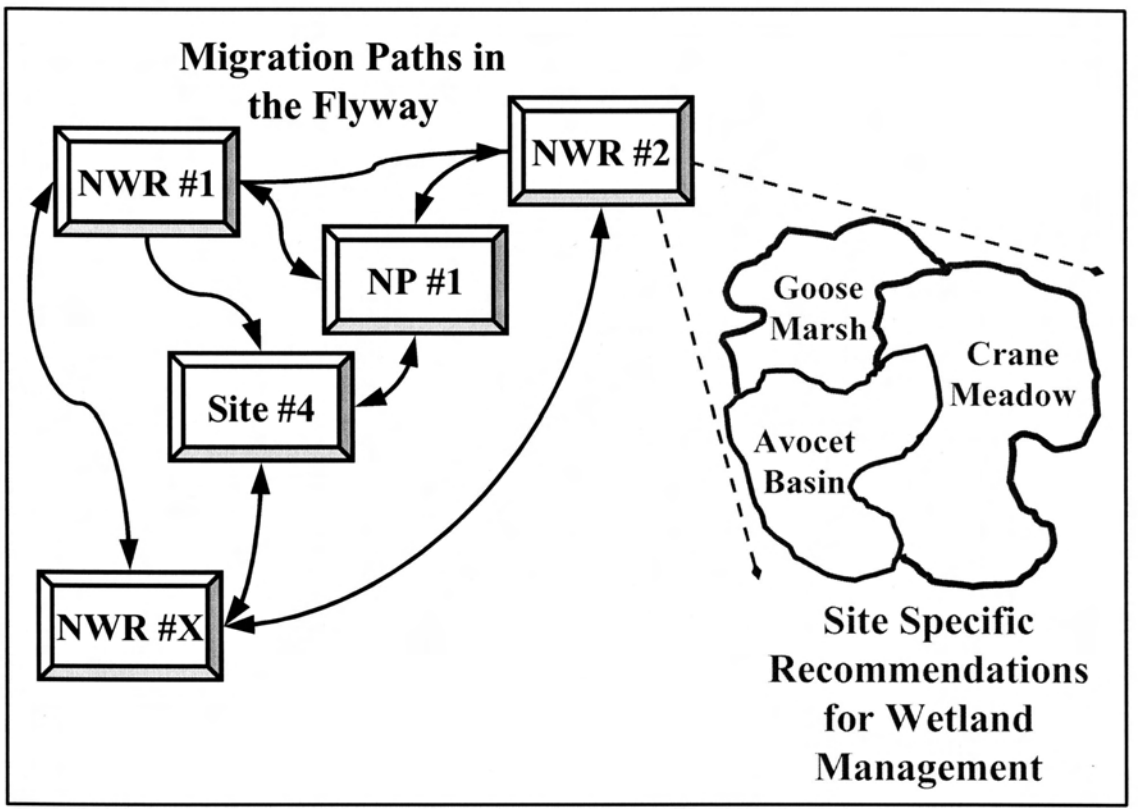

Figure 2-1. Spatial complexity shown as hypothetical migration paths among national wildlife refuges (NWR), national parks (NP), and other sites combined with recommendations needed for management of local wetlands.

Swan managers have requested a decision support tool that will simulate and test management options for trumpeter swans throughout such corridors. The ultimate objective is to contribute to both population recovery and migration path development. They recognize that their decision making is cyclic, and they wish to iteratively plan, implement, evaluate, and improve their management strategies. Biologists also are concerned by their lack of ability to objectively assess critical information gaps, identifying those that contribute the most uncertainty to the selection of management options.

Unfortunately, optimizing any management of migratory birds throughout a flyway with cyclic planning is so complex that it is often all but impossible to implement without computerized decision support (Sojda, Dean, and Howe 1994). And, past conditions and future needs are ecological constraints to current decisions. Distributed decision making approaches suit problems where the complexity prevents an individual decision maker from conceptualizing, or otherwise dealing with the entire problem (Boland et al. 1992; Brehmer 1991). Our system is focused, then, on providing support for realistic and ecologically-based management of migratory birds at multiple geographic and temporal scales.

At this time, there are no such decision support systems available for swan managers, nor any common databases for them to access. Furthermore, many managers are physically either located in relatively remote locations or simply distant from each other, making it difficult to meet frequently. They currently get together once or twice a year to discuss and select broad management options for the flyway and specific recommendations for specific sites as deemed necessary. Additionally, on national wildlife refuges and some other areas, annual water management plans are prepared for individual wetlands. These are prepared manually, and often can not take into account conditions in other areas of the flyway except in a general sense. Plans are not usually updated during the course of the year. The past and current holistic situation for management of trumpeter swans, of which planning is only a part, has not yet resulted in population recovery for Tri-State swans. New planning approaches are welcome, and an approximately 80 percent increase in breeding pairs is still desired.

2.4 Requirements Analysis and Cooperative Distributed Problem Solving

Our research is pursuing three objectives. (1.) We intend to provide a decision support system that allows swan managers to examine management actions addressing population and migration objectives at a flyway scale, and allows them to evaluate management actions at a site specific scale. (2.) We will test the hypothesis that decision support technology which allows planning in multiple geographic and temporal scales results in an increased ability for managers to identify and capitalize on trumpeter swan management potentials. For our technology to give managers this capability, we must verify that the decision support system simulates future swan distributions that meet flyway goals; that habitat recommendations are satisfactory for supporting increasing populations; and, all recommendations remain reliable over a specified time period. Management potentials are those ecological conditions that can be exploited in pursuing trumpeter swan objectives. Included are habitat quality, quantity, distribution, and availability, as well as freedom from disturbance. (3.) We will test the hypothesis that our implementation of cooperative distributed problem solving among refuges, parks, other management areas, and the internal knowledge bases effectively integrates local management actions with small-scale landscapes. This integration will occur if information is shared among human and electronic nodes, if individual knowledge bases contribute to recommendations, and if principles of adaptive management (Holling 1978; Walters 1986) are incorporated.

Based on input from swan managers, we have identified four management questions to be addressed through decision support system simulations. Each of these is a relatively course-grained approach to extrapolate possible future scenarios, while retaining the need to address the practicality of the fine-grained needs of individual managers. This is being tackled by paying close attention to knowledge engineering efforts and the use of

expert systems to connect the relatively qualitative knowledge of the domain experts with the heuristic guidance needed by managers.

> *Simulation #1.* If a particular management action is implemented at a particular site and particular time, what are the consequences for that site and for other sites in the flyway?
>
> *Simulation #2.* Given an objective for spatial and temporal distribution of swans, what is the best set of management actions across all sites to achieve this? The decision support system also will have the capability for the manager to provide an alternative objective.
>
> *Simulation #3.* Given some subset of management action(s) across all sites, and given an objective for spatial and temporal distributions of swans, what is the best complementary subset of management actions at other sites to achieve this?
>
> *Simulation #4.* Given a satisfactory set of management actions across all sites to achieve an objective for swan distribution, if an alternative management action were to be implemented at a particular site, what are the consequences for that site and for other sites in the flyway in terms of reaching their respective objectives?

To address Simulation #1, a blackboard approach will be taken. When a particular management action is proposed for a particular site, that information will be posted to the blackboard. Daemons residing there will fire as necessary to activate the use of appropriate rules and expert systems to simulate the effects of the proposed action for the current time at that site, as well as at other sites in the flyway. New and impending

constraints that the proposed action will impose on future management also will be generated and presented.

To address Simulations #2-4, a more complex search of the solution space will be required, and cooperative distributed problem solving will be used. Each geographic node in the system will need to function both independently and collaboratively with themselves and with the knowledge bases, exchanging tentative and partial results in order to converge on a solution (Carver, Cvetanovic, and Lesser 1991). These more complex simulations will require the concurrent development and posting of partially completed plans and potential management options from all geographic sites. The goal is to find a satisfactory set of solutions for management at all sites. This will be done by sharing information among geographic nodes and with the knowledge bases and databases. Then, a recursive search will be made for a set of management options that satisfices the population level and distribution objectives, and that addresses the constraints in the system.

The swan decision support system will use a combination of artificial intelligence methods including expert systems, blackboards (Corkill 1991; Nii 1986a, 1986b), and cooperative distributed problem solving (Carver, Cvetanovic, and Lesser 1991; Durfee, Lesser, and Corkill 1989). Four basic modules form the system's framework (Figure 2-2): cooperative distributed problem solving, knowledge bases (expert systems), databases, and web interface. The decision space consists of knowledge and constraints, including population objectives, on-the-ground management capabilities, ecological principles, and implementations of adaptive management. In addition, an area's past management history, as well as its future needs, represent further temporal constraints to forming recommendations in the present, particularly related to wetland manipulations.

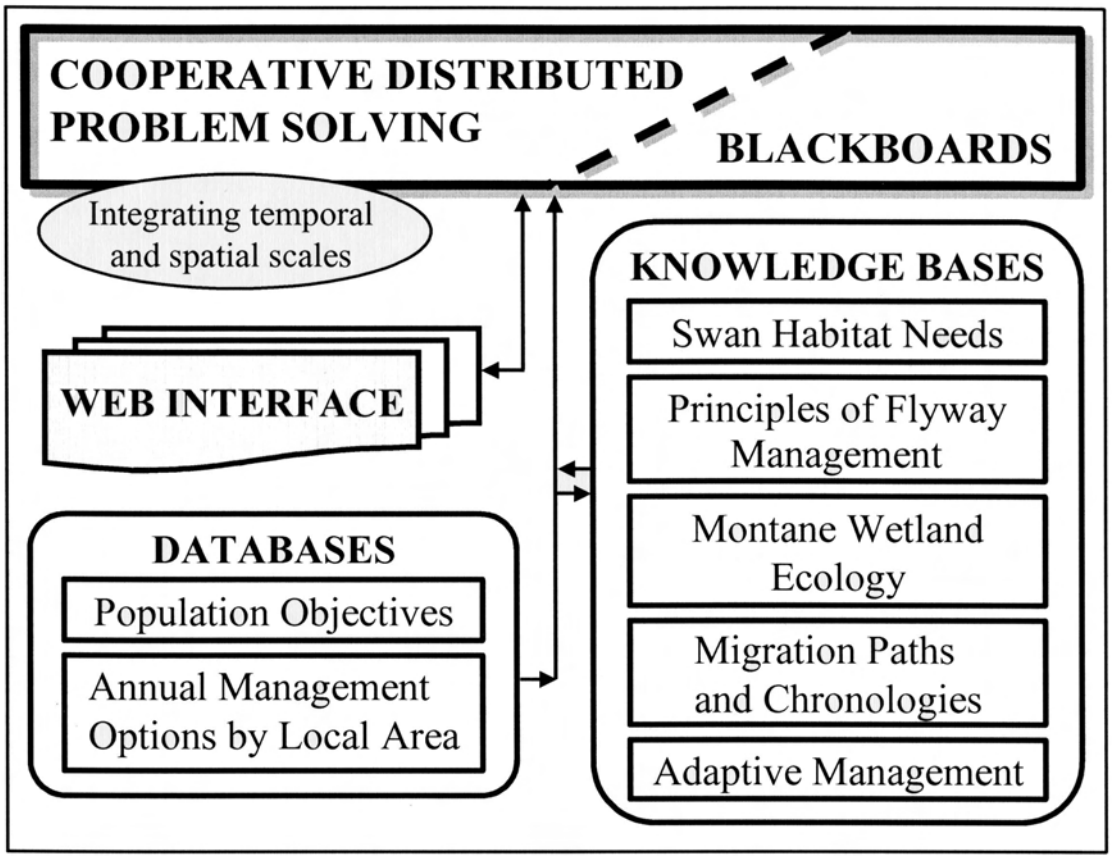

Figure 1-2. The framework for the swan management decision support system.

The essence of the distributed nature of swan ecology stems from birds moving among areas as seasons and other ecological conditions change, especially habitat availability. Migration stimuli also are related to annual life cycle events and physiological condition in individual swans. Meta-rules for handling the integration of all such spatial and temporal issues in the domain will be developed and integrated at a high level in the system.

It is clear to us that wetland ecology is a domain where the complexity of relationships, the interactions among ecological parameters, and the lack of empirical data makes the programming of rule-bases and decision trees complicated. By the same token, we are becoming increasingly convinced that the complexity of ecological systems is, in fact,

what makes the application of expert systems, cooperative distributed problem solving, and other artificial intelligence methods so potentially useful. This domain has a nearly infinite number of ecological conditions, but the number of potential recommendations is more limited. Backward chaining approaches are allowing us to appropriately search the decision space in a goal-directed manner. Artificial intelligence based multi-agent methods are another approach that might be used for such a planning problem. However, their contribution often lies in searching exceptionally large and dispersed information sources, in providing real-time solutions, or utilizing the reasoning power of individual agents. None of these attributes exists in our domain. On the other hand, there are some similarities in our approach to the asynchronous backtracking algorithm presented by Armstrong and Durfee (1997), except that we are not using a complex agent implementation.

2.5 Status of System Implementation

We are developing the system on a personal computer using a commercially available expert system development shell that has blackboard capabilities. The system is deployed on a Unix workstation acting as a web server connected directly to the Internet through Montana State University. This is accomplished using software affiliated with the development shell that provides a common gateway interface between the web server and the inference engine, developing HTML web pages on the fly.

Our primary goal is to explore whether cooperative distributed problem solving can solve actual ecological problems characterized by geographically distributed issues that are compounded by temporal scales. There were several, general institutional concerns governing our selection of technologies. These included palatability to end users, availability of off-the-shelf software, probability of long-term software support, and cost. Following the scheme describing expert system use and research provided by Hollnagel

(1991), our project is using known methods and addressing unknown problems. However, this categorization is not clear-cut because, to our knowledge, the application of cooperative distributed problem solving has not been implemented using our current software.

We have developed knowledge bases for swan habitat needs and management of montane wetlands. Each of our knowledge engineering sessions was approximately three days in length, and utilized one to two experts each. In some cases, one of the experts has been involved in previous expert system development, making knowledge acquisition efforts relatively easy. In particular, this individual was more apt to provide us detailed chains of logic in his reasoning without us needing to continually prompt him to do so. One technique that we used extensively was to provide the experts with detailed slide shows of actual field situations depicting wetland condition and management options. This seemed quite effective. It continues to be difficult to have our experts delineate their level of confidence in pieces of knowledge so that we might assign uncertainties within a knowledge base. Although our only evaluation to date has been qualitative, we have been pleased with the acceptance of the knowledge bases that we have demonstrated to swan managers.

Looking towards the future, there are some issues that we envision will be particularly challenging. First, developing the rules to implement cooperative distributed problem solving as a specific expert system, in essence a meta-system guiding the rest, has never been tried in this type of ecological venue. We are examining a number of ways to utilize blackboard algorithms (Carver and Lesser 1992) in domains such as ours. The multi-agent system of Pinson, Louca, and Moraitis (1997) which includes artificial agents, blackboards, and a constraint base may hold promise. Similar to their system, ours will be able both to make satisfying recommendations and to present incompatible

management options through the use of subgoals. Our subgoals are represented by output from the knowledge bases as well as partially completed habitat plans for individual management areas (Figure 2-2.) Local control structures on the blackboard will critique and assemble partial plans from individual users, using rules to determine when knowledge base or database interaction is necessary. The scheduling of when such knowledge base or database output is necessary will be handled at a meta-level similar to that described by Maitre and Laasri (1990). Our constraint satisfaction approach will be implemented as knowledge bases invoked as part of the meta-level control structure.

Another challenge will be determining the best way to propagate uncertainties in the system. In the development of the current prototype, we intend to accept uncertainties provided as output from individual modules at face value. Then, the global propagation issues will be tackled within the cooperative distributed problem solving algorithm, allowing each knowledge base module to pass its own internal confidence assessment, essentially unchallenged, to the broader system. Future system development may address uncertainty issues within individual modules.

Finally, although empirical evaluation of the system is planned, we anticipate that it will not be straightforward. Gold standards for validation of various ecological components and models do not exist, plus the system will be predicting and guiding future scenarios as they, in fact, unfold. And, from a holistic perspective, we are developing decision support for issues that appear to beyond the capability of single persons to conceptualize and solve.

CHAPTER 3

IMPLEMENTATION OF A MULTIAGENT SYSTEM TO DECISION SUPPORT FOR TRUMPETER SWAN MANAGEMENT

3.1 Introduction: Issue Definition, Decisions Supported, and Conceptual Background

Waterfowl biologists and managers charged with the conservation of the Rocky Mountain Population of Trumpeter Swans have expressed an interest in having decision support tools and systems that would assist in evaluating management alternatives. Such alternatives are oriented towards expanding the breeding distribution in Idaho, Montana, and Wyoming, as well as expanding the wintering distribution, and re-establishing traditional migration paths to the greatest extent feasible. To this end, I chose to develop a multiagent system (Weiss 1999) that has, as its core, a queuing model (Dshalalow 1995; Hillier and Lieberman 1995) to simulate the distribution of swans in a large portion of the Pacific Flyway. Moreover, the decision support system incorporates aspects of ecological knowledge related to swan breeding habitat quality, to the ecology of palustrine wetlands, and to principles of flyway management of migratory birds. Because this domain represents a problem inherently distributed in time and space, the very nature of multiagent systems methodology seemed appropriate to apply to such a domain (Sojda and Howe 1999.) For the same reason, I also attempted to draw on the concept of cooperative distributed problem solving (Carver, Cvetanovic, and Lesser 1991; Durfee, Lesser, and Corkill 1989).

The completed product is a decision support system for the management of trumpeter swans, and here I describe the development of the multiagent system, its architecture, and the methodologies employed in its construction and deployment. The purpose of the system is to simulate the migration of swans via the use of agents as a way of implementing a queuing system.

3.1.1 *Overall Purpose: Temporal and Spatial Distributed Problem Solving*

Queuing systems (Dshalalow 1995; Hillier and Lieberman 1995), generally, represent the inherent spatial and temporal attributes of interrelated entities moving and waiting in lines (queues). In the swan decision support system, one class of such attributes are habitats scattered throughout a migration corridor. The entities are the swans, themselves. Throughout the scattered habitats, climatic, ecological, and management changes are occurring at many temporal scales. And, swans are continually moving among habitats and areas, particularly as they proceed through annual life cycle events. The agents have been designed to perform tasks and actions to cooperatively reach a solution to the problem of predicting swan distribution via the central queuing model. They do this by listening for, requesting information about, and reacting to changing ecological knowledge and conditions, as well as interacting directly with the user.

Cooperative distributed problem solving (Carver, Cvetanovic, and Lesser 1991; Durfee, Lesser, and Corkill 1989) and a related methodology, partial global planning (Durfee and Lesser 1991), are based on the concept of individual entities (in this case agents) being able to only solve the portion of a problem visible to them, and, they collaboratively converge on comprehensive solution as information is shared among the entities. From a broad point of view, my intent was to incorporate aspects of temporal and spatial distributed problem solving into the decision support system (Figure 3-1), and specific

agents were designed to do just that. There are refuge agents that only assess the ecological knowledge at select geographic areas. There also is a move agent that only encapsulates general knowledge about how swans move among a broader set of sites during transitions from season to season. Yet, through sharing information, not only are swan movements simulated among the broader set of sites, but conditions at the specific areas actually affect swan distributions throughout the network of sites.

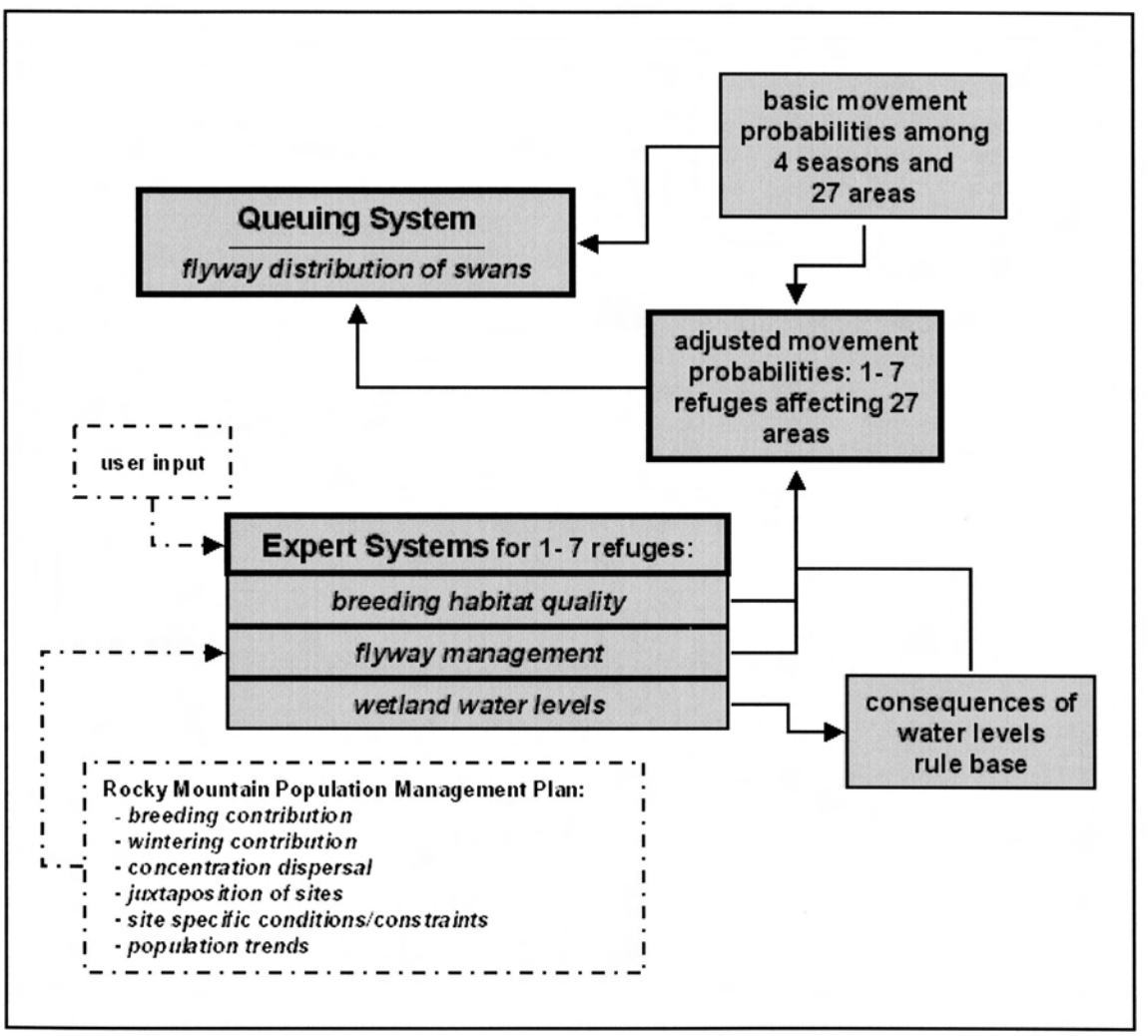

Figure 3-1. Aspects of Temporal and Spatial Distributed Problem Solving Within the Move and Refuge Agents.

The more specific purpose of the decision support system is to provide a predicted distribution of trumpeter swans of the Rocky Mountain Population in one year increments following a specified time line representing the major events in the annual life cycle of swans (Figure 3-2.) It uses as one primary input the number of swans actually observed at 27 areas in year, *t* , and the probability of movement from season to season among those areas as another input. The queuing system then projects the number of swans at time, *t+1*, across those same areas. Key ecological knowledge is used to adjust the second input to the queuing model, eventually resulting in adjustments to the predicted numbers of swan across the 27 areas. Fundamental elements of the 1998 Management Plan for the Rocky Mountain Population of Trumpeter Swans (Subcommittee on Rocky Mountain Trumpeter Swans 1998), plus vital ecological knowledge about swans, their habitats, and flyway management principles were incorporated as expert systems to provide this ecological knowledge. These expert systems assist the user provide and update ecological information to the overall system.

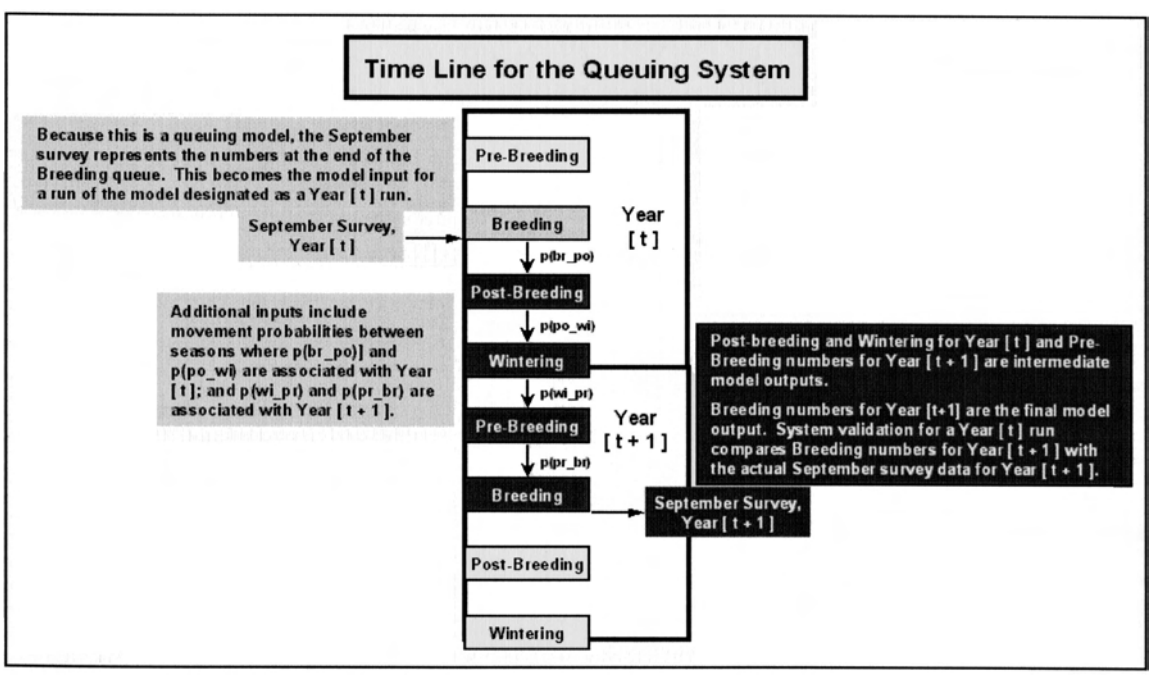

Figure 3-2. The time line of inputs and outputs for the swan decision support system, focusing on the queuing model.

3.1.2 *A System of Intelligent Agents: the Underlying Concepts*

Wooldridge (1999) states that no accepted definition of an agent exists, although his writings have helped to overcome this. I will accept the definition of an intelligent agent as a computer system based in artificial intelligence, that is autonomous, collects information about its environment (either virtual or real environment), and is capable of independently taking the initiative to react to that input as appropriate (Weiss 1999; Wooldridge 1999; Wooldridge and Jennings 1995). The decision support system for trumpeter swan management consists of a minimum of three such agents (facilitator, movement simulator, and refuge agents), but can include up to six more refuge agents depending on the geographical complexity of interest by the user. Additional agents also are embedded, somewhat transparently, within the DECAF software (Graham 2001; Graham and Decker 2000; Graham et al. 2001) that was used to implement the entire multiagent system. The framework I have implemented is conceptually similar to that of Decker et al. (1997) who describe interface agents, task agents, and information agents in their financial portfolio management system.

Different theoretical concepts have been developed as foundations for building intelligent agents, and the belief-desires-intentions (BDI) agent architecture described by Rao and Georgeff (1991; 1995) is at the core of my agents. Graham (2001) describes DECAF as conceptually based on a BDI architecture and his particular interest was relating intentions to the planning and scheduling features of DECAF. So, my system inherits those characteristics. The aspect of BDI theory that I actively brought into play when building my system is the concept of updating beliefs. It is implemented in my agents as they determine the need for, currency of, and communicate about, the ecological knowledge needed during a consultation. This is the underlying notion of agents understanding their own environment and updating that knowledge when necessary.

The term, multiagent system, implies more than one agent interacting with each other within an underlying communication infrastructure and without a procedural control mechanism; and, the individual agents often are distributed and autonomous (Huhns and Stephens 1999.) My system seems to fit such a definition. However, Durfee and Rosenschein (1994) consider the differences and similarities between distributed problem solving and multiagent systems, concluding that there are different perspectives from which to view the distinctions. It is clear that my research borrows from both fields (their "View 3"), and that my system emanated from concurrent emphasis on individual agents and system behavior. My system was designed as a multiagent system (focus on autonomy) that addresses problem resolution, at least partially, from a distributed problem solving approach (focus on shared goals).

3.1.3 *Expert Systems for Representing and Using Natural Resource Knowledge*

Ecological knowledge is embedded in many parts of the decision support system, e.g., as logic within DECAF agents and as problem solving methods within individual agents. However, the most substantial ecological knowledge accessible to the system originates in expert systems that provide information within the context of individual refuges (geographic areas). Expert systems (Russell and Norvig 1995) have been used in a variety of domains where knowledge about that domain can be symbolically represented through knowledge engineering (Scott et al. 1991). Often, such domains are characterized by a lack of quantitative information about cause-effect relationships and their associated uncertainties. Therefore, one must develop models and symbolic representations of knowledge using heuristics and the problem solving methods of limited experts. This seems to be the case in the field of natural resource management generally, and in the domain of trumpeter swan ecology and management specifically. I have, accordingly, developed three expert systems: (1.) Breeding Habitat Needs for

Trumpeter Swans, (2.) Management of Palustrine Wetlands in the Northern Rockies, and (3.) Assessing the Contribution an Area Can Make Towards the Flyway Management Plan for the Rocky Mountain Population of Trumpeter Swans. The use of this knowledge is explained later.

White et al. (1985) provided a critique of expert systems in wildlife management but referred to no actual systems. Davis and Clark (1989) list 203 citations in their bibliography of expert systems in natural resource management. Also in 1989, the journal, *Ecological Modelling*, devoted an entire issue to the use of artificial intelligence. Hushon (1990) tallied 68 environmental expert systems in their review. And, from 1987 to 1997, the journal, *AI Applications*, was devoted solely to the use of artificial intelligence in natural resources with many papers about expert systems. Clearly, the field has expanded, and expert system development in the general arena of natural resources has become more common. However, application of expert system or other artificial intelligence methodologies to provide ecological knowledge for flyway management of waterfowl or other migratory birds is not known to have been attempted before.

3.1.4 *Queuing Systems and Ecological Applications*

Queuing systems are part of a broader schema known as systems of flow (Kleinrock 1975), where some commodity (e.g., swans) move from one place to another (e.g., refuges, wildlife areas) but are constricted in that movement through channels (e.g., life cycle events). Queuing theory originally developed as a way to optimize manual switching of telephone lines at the turn of the Twentieth Century in Denmark. It has developed immensely since then and theory and applications have been developed within Mathematics, Statistics, Computer Science, Operations Research, Transportation

Engineering, Telephony, and others (Dshalalow 1995). In queuing systems, stochastic processes typically are used to model some or all of the various events, especially the arrival of customers and the servicing of customers by a server(s) as related to waiting line states. The purpose usually is to estimate waiting times and provide algorithm(s) for optimizing commodity or customer service. In the swan queuing system, events related to arrival and service are deterministic, representing the simple class of queuing system described by Gross and Harris (1974). The use of movement probabilities in the swan queuing model, although treated like percentages and not as probability distributions, does change its deterministic flavour somewhat and allows for future modification using stochastic-based service functions.

Applications of queuing theory in ecological and natural resource models are not extensive and none are known to have been made to waterfowl or other bird migration. There are applications where only the modelling of waiting times via a queue are discussed, such as territoriality in oystercatchers (*Haematopus* ostralegus) (Ens et al. 1995) and visitor crowding in Yangminshan National Park, China (Chang 1997). Kokko et al. (1998) focused on the queuing discipline, alone, in a model of lekking behavior of black grouse (*Tetrao tetrix*). Another conceptual application to the optimization of reservoir releases has been reported by Maniak and Trau (1974). Other models are more developed in terms of queuing theory, itself. These include applications to harvest of white-tailed deer (*Odocoileus virginianus*) (Jacobs and Dixon 1982; Cohen 1984), social behavior of ants and other organisms (Blanckenhorn and Caraco 1992; Burd 1996), and groundwater management (Batabyal 1996).

3.2 **Multiagent System Architecture**

3.2.1 *Basic Structure for a System of Cooperating Intelligent Agents*

The decision support system was designed as a network of cooperating intelligent agents, and DECAF is software that provides a framework for building such multiagent systems. DECAF agents are programmed at a high level using a graphical user interface that depicts tasks and actions and the relationships and programming logic among them. These DECAF programs are called "plan files", and each agent is uniquely represented by one; although multiple instantiations of the same agent type are instantiations of the same plan. Such agents and their plans represent hierarchical task network planning methodology (Erol et al. 1994.) DECAF tasks are the roots of the task networks, and DECAF actions are leaf nodes (Graham 2001.) It is DECAF that provides the embedded capability of scheduling and coordinating the various actions to be undertaken as represented in the plan files of instantiated agents; and, it is DECAF that executes the best approach for logical completion of the task networks. Such planning and scheduling activity, obviously, must be somewhat dynamic as the system progresses through a consultation.

At a lower level, each task and its actions symbolized in a DECAF plan file are programmed as a JAVA class and its methods, respectively. The agents communicate internally and among each other by passing messages that follow the KQML (Knowledge Query and Management Language) specification (Labrou and Finin 1997) and via DECAF's internal architecture. Agents in my system either request other agents to accomplish tasks (achieve performatives) or notify other agents that some aspect of the sending agent's belief system has changed (tell performative). The latter is used in responding to an achieve performative. Advertise performatives also are used by start-up tasks in each agent to allow the Matchmaker agent (part of DECAF's internal

software engine) know the capabilities of each agent. Because the version of DECAF used has not yet implemented complex planning and scheduling algorithms, the Matchmaker advertisements in my system have been implemented in anticipation of future augmentation of the swan decision support system.

All agents are started manually by the user through standard DECAF interfaces. The facilitator agent, called "fac", handles interaction with the user and notifies the distribution simulator agent, called "move", that the user wishes to proceed with a simulation. The facilitator agent also actively listens for knowledge about specific refuges that could be changing. The move agent requests refuge agents to provide information, assembles all necessary system data, runs the queuing model, and writes output files. From one to seven agents representing knowledge about specific areas also are activated by the user. These latter agents are instantiations of a generic "refuge" agent and are given two letter names corresponding to real geographic locations (br = Bear River, ca = Camas, cv = Centennial Valley, gl = Grays Lake, ip = Island Park, jh = Jackson Hole, and um = Upper Madison; see Table 3.1.) Each refuge agent is charged with determining the status of its own knowledge regarding water level management, flyway management, and breeding habitat quality. If it is determined that additional knowledge is necessary, the facilitator agent then handles interaction with the user by requesting that specific expert systems be run.

A configuration file (Appendix 1) is manually edited by the user and used automatically by the various agents to set system parameters regarding how the queuing system will be run. This includes values, discussed later, for the "breeding_threshold_value" and the "remain_in_queue_multiplier". Through the configuration file, one also can disable the interaction of either one, two, or all ecological knowledge bases with all active refuge

agents. Because they can change from run to run, configuration parameters are archived as part of automatically generated output files pertaining to each run.

Table 3-1. Names of servers in the queuing model and associated geographic locations. Refuge, here, is used to denote servers for which their associated probabilities of movement can be directly affected by the move agent.

[1] locations are denoted in Figure 3-4.

Server Name	Geographic Location	Treated as a Refuge
Canada	everything in Canada except Okanogan	
Okanogan	South Central BC	
Freezeout Lake	near Fairfield, MT	
Flathead Valley	in Northwest MT	
Upper Madison River	Quake Lake (MT) and all upstream areas [1]	X
Mid-Madison River	Quake Lake through Ennis Lake (MT) [1]	
Lower Madison River	below Ennis Lake (MT)	
Centennial Valley	in Southwest Montana, includes Lima Reservoir. Red Rock Lakes NWR, Elk, Cliff, and Wade Lakes [1]	X
Teton Basin	Victor to Teton, ID [1]	
Swan Valley	vicinity of Swan Valley, ID along South Fork of the Snake River [1]	
Island Park	in Southeast ID and Northwest WY [1]	X
Lower Henry's Fork	Ashton to Roberts, ID [1]	
Paradise Valley	Yellowstone River between Yankee Jim Canyon and Livingston (MT) [1]	
Yellowstone Lake/River	Yellowstone NP (WY)	
Jackson Hole	includes National Elk Refuge and Grand Teton NP (WY)	X
Green River	in Southwest WY, includes Seedskadee NWR [1]	
Camas NWR	near Hamer, ID [1]	X
American Falls Reservoir	West of Pocatello, ID, includes [1]	
Grays Lake NWR	near Wayan, ID [1]	X
Salt River	along ID/WY border above Palisades Reservoir [1]	
Bear Lake/Soda Springs	Southeast ID, includes Bear Lake NWR and areas along the Bear River [1]	
Bear River MBR	near Brigham City, UT [1]	X
Lower Snake River	from Minidoka NWR (ID) to the Oregon border, includes Deer Flat NWR	
Ruby Lake NWR	Northwest NV	
Malheur NWR	Southeast OR	
Summer Lake WMA	South Central OR	
Central Valley, CA	North Central California, includes several NWRs	

3.2.2 Queuing System Configuration

Trumpeter swans of the Rocky Mountain Population have been surveyed routinely in September across large regions of Western North America, and that information was organized for 27 specific areas. This forms the structure of the D/BB/28 queuing system (Dshalalow 1995; Kendall 1953) which I developed. It models the movement of birds through a large portion of the Pacific Flyway, distributing them across time and space in one year increments.

Hillier and Lieberman (1995) describe four key components to any queuing process from an Operations Research perspective: the *input source* (customer arrival times and patterns), the *queue* (number of customers awaiting service), the *queue discipline* (order of customer service), and the *service mechanism* (the process and associated time to serve customers). Dshalalow (1995), considering a more theoretical Mathematics perspective, also includes three additional components: the *number of servers*, the vacation or *idle discipline* (the process and time when a server has no customers to serve), and the *network configuration* (direction of service among multiple servers and steps). The configuration of the swan movement queuing system is depicted in Figure 3-3.

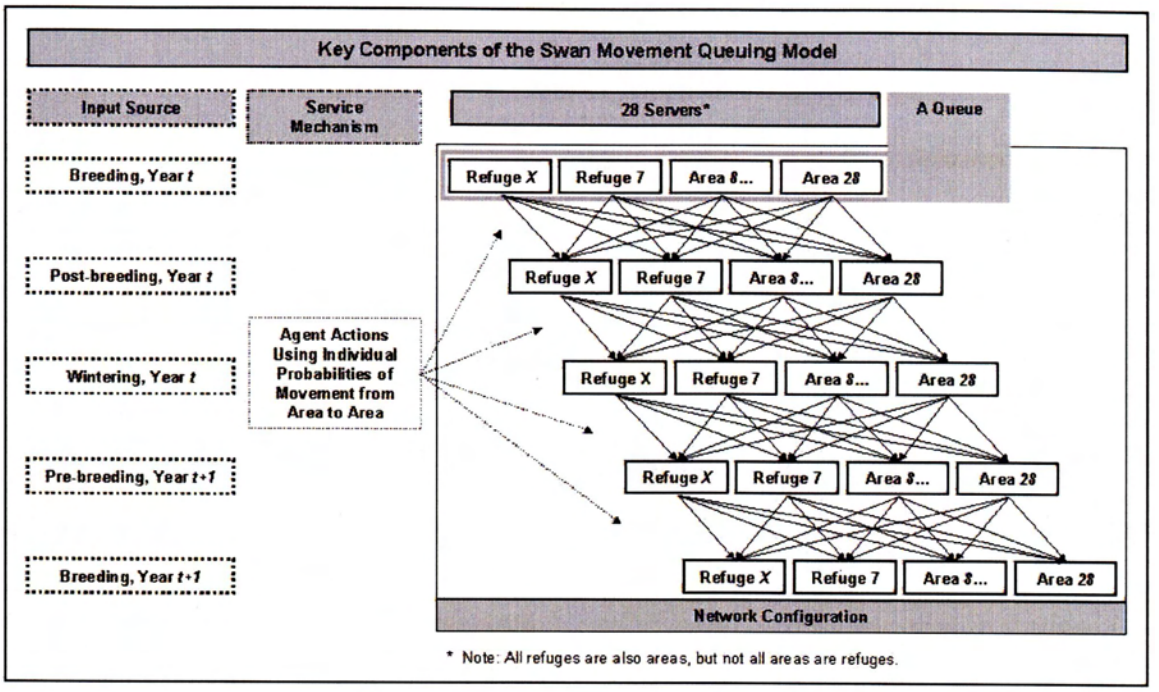

Figure 3-3. Graphical depiction of the D/BB/28 queuing system.

The swan queuing system has a deterministic input source with birds moving, potentially, from area to area at the end of each season. The starting queue is theoretically of infinite size, but is represented by the total number of swans as surveyed in September of a particular year. Those observed values are found in Appendix 2. Although this input source is comprised of September values, it is called a "breeding number" because it represents the size of the population at the end of the breeding queue. Intermediate queues are post-breeding, wintering, and pre-breeding; the final queue is subsequent breeding.

The queue discipline is FIFO (first in, first out) but becomes somewhat inconsequential because the service mechanism is a batch process. I.e., all birds in a queue are processed simultaneously into the next queue and set of areas. And, because the service mechanism operates deterministically, the idle discipline is nonexistent for all

practical, computing purposes. Each server governs, through the movement probability matrix, the number of birds allowed into itself. There are 28 servers in the system, 27 geographic areas and one unknown buffer. The latter handles imprecise situations where the underlying movement likelihoods are undetermined. The network configuration is provided by the algorithm that uses the movement probability matrix to distribute swans among the 27 areas. The service mechanism is embedded within additional problem solving algorithms of the move agent and iteratively steps through the seasons to provide a predicted distribution of swans for the subsequent breeding season. Additional complexity is modeled in the algorithm that is used to adjust the matrix of movement probabilities (see section on the DECAF task, dss_Sim). The amount of interchange among Canadian breeding and U.S. breeding trumpeter swans of the Rocky Mountain Population is unknown, but generally thought to be small. Therefore, the service mechanism tracks the broad breeding queue origin of birds. When entering subsequent intermediate queues, birds are allowed to mix among servers. However, via that tracking function, Canadian birds are not allowed to be serviced by U.S. servers, and vice versa, when entering the subsequent breeding queue. The only exception to this is minor in magnitude, affecting less than 0.5 percent of the entire population. Based on expert opinion, birds are allowed to move from Summer Lake, OR to Okanogan, BC during northward migration. This, along with the movement probabilities, themselves, are primary components of the network configuration.

3.2.2.1 *Knowledge Engineering for the Movement Probabilities*

No quantitative information exists about trumpeter swan movements among seasons. Therefore, values for movement probabilities were gathered by conducting knowledge engineering sessions utilizing three experts working together for two days to arrive at

consensus about those values. Two of the experts were waterfowl and refuge biologists with the U.S. Fish and Wildlife Service and one was a waterfowl biologist with the Idaho Department of Fish and Game. They were asked to provide their best estimates of these values using their own knowledge, informally consulting a database of neck collar resightings which one of them had designed, and using existing seasonal survey data. Data from the neck collar database was not directly used to estimate probabilities because it is thought to be biased, representing birds both marked opportunistically and resighted opportunistically. This data can be used, nonetheless, to verify the existence of certain migratory pathways. Ideally, movement information would have been collected for all Canadian breeding and staging areas. Although a request was made for such information from the Canadian Wildlife Service, the information was not received. Because independently providing such information is not a trivial task for them and expanding the personal level of knowledge engineering I could provide was not feasible, I did not pursue this aspect. Therefore, I limited the movement information to that provided by the U.S. experts.

The experts were unable to provide direct probabilities of movement from area to area; but, they were willing to provide the likelihood that a bird in each of 27 areas would be seen in each area (including the starting area) during the subsequent season. Due to movement among areas within a season, however, the sum of these raw likelihoods for a particular season can exceed one. To overcome this, each likelihood value was weighted by dividing it by the sum of all likelihoods for the season. Such a weighted

value was named a probability of movement from Area x to Area y (although it is not represented by an empirical probability distribution, per se) and is represented as:

$L_{x,y}$ = raw likelihood value, representing expert opinion of the percentage of birds from Area x likely to be seen during the subsequent season in Area y

$M_{x,y}$ = probability of moving from Area x to Area y

$$M_{x,y} = \frac{L_{x,y}}{\sum_{y=1}^{28} L_{x,y}}$$

This does assume that, during any one season, swans will be found in the same proportions among areas at the end of the season as represented by the likelihoods of movement into that area at the beginning of the season. At this time, no data exists to test this assumption. The raw likelihood values are found in Appendix 3.

Table 3-1 and Figure 3-4 lists the names of the servers and their corresponding geographic locations. These were chosen by the experts during the knowledge engineering session as logical groupings of either traditional survey units or areas of management importance.

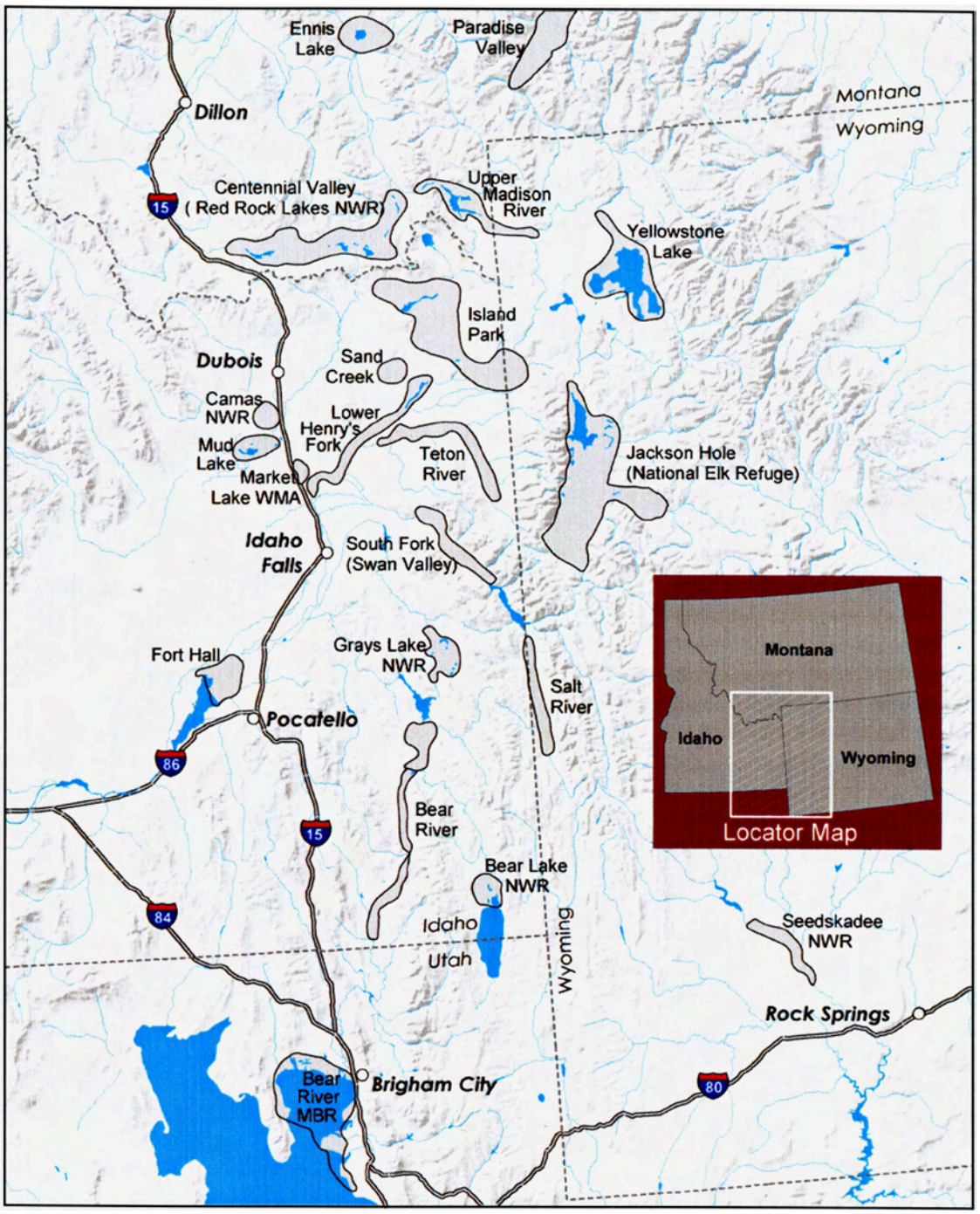

Figure 3-4. Geographic locations of areas (servers) in the Tri-state Region used in the queuing system.

3.2.2.2 *Queuing Model Output*

The output from the queuing model is a predicted number of swans pertaining to 27 areas and the unknown queue for four seasons. Although designed to be transparent to the user in future system augmentations, this information is written to the workstation console window associated with the move agent in the current version of the system for continuing verification and validation purposes. Queuing model output is central to the overall decision support system. Output from the latter and its format is discussed later.

3.2.3 *Service Mechanism: Connecting Agents and Expert Systems*

The queuing system's service mechanism is represented by an algorithm describing how swans move through the flyway and is embedded within the tasks of the agents themselves and their interaction within the multiagent system. Details of the work of the DECAF tasks and actions as they implement the service mechanism are addressed individually in subsequent sections and the Appendices. Here, I describe the high level algorithm that constitutes the service mechanism.

For any transition stage, comprised of an input source and set of servers, swans potentially move from each server to every other server, including possibly staying in place (not moving). In the simplest of cases, i.e., the effect of using expert system input is not utilized; only the matrix of base probabilities of movement ($[M_{x,y}]$) is used to distribute birds to the next queue. The refuge agents parse expert system output to determine whether a server (refuge) has conditions of acceptable or unacceptable quality for swans. When a refuge agent determines that a server (refuge) has conditions of acceptable quality for swans, that server accepts its base probability for that seasonal transition. If that is not the case (and conditions are unacceptable), that server is

allowed to accept only 0.1 of the swans indicated by its base probability. This redistribution value is arbitrary and is set by editing the configuration file. The default value of 0.1 also is arbitrary, but represents the relatively strong philopatry that is thought to exist in trumpeter swans.

Obviously, the birds not accepted by a server must be redistributed, and this is implemented in the following way. The remaining 0.9 of the swans are sent to the server that had the next highest base probability. For example, for server 1, this would be where $M_{1,y}$ is maximum; and the 0.9 of the swans would be sent to server y. Should there be more than one such server y (i.e., there is a tie) the 0.9 of the birds are sent to the server closest in straight line distance to the server not accepting all its swans (in this example, server 1). Because it was felt that there was no ecological, migratory significance among movement between some areas, some areas were grouped. This resulted in assigning the same distance from one area to more than one other area. E.g., a distance of 468 km was assigned to the distance from Flathead Lake to Teton Basin, Swan Valley, Island Park, Lower Henry's Fork, and Jackson Hole. Such grouping results in potential ties in straight line distance at this stage of the algorithm. When this occurs, the 0.9 of the birds are distributed equally among those areas. There has not been any experimental work published that has quantified migration parameters for the Rocky Mountain population of trumpeter swans, and the above algorithms are based on information gleaned from the various knowledge engineering sessions. See Appendix 4 for the distance values and groupings.

How the recommendations from each expert system are used to affect the functioning of servers varies among expert systems. The expert system that assesses breeding

quality provides a numeric rating from 0-100 and affects servers in the breeding queue only. This rating is described within the expert system, itself, as:

> *...a somewhat arbitrary assessment of the degree to which this wetland represents ideal habitat for breeding trumpeter swans, based on a synthesis of all information. On a scale of 0-100, the higher the number, the more confident one can be that satisfactory habitat exists. A zero indicates that at least one habitat component is severely compromised.*

The threshold of breeding habitat rating to be used by the decision support system to determine acceptability in the distribution algorithm is arbitrary and can be set by editing the configuration file. A rating of 50 indicates equal confidence in the breeding habitat being either good or poor. A rating of 60 represents the conceptual value for minimally satisfactory habitat and was therefore used as the default value in the distribution algorithm.

Recommendations for water level management from the montane wetland expert system are used to determine server acceptability in all four queues based on a partial implementation of concepts generated during knowledge engineering sessions as represented in Table 3-2. The system only considers the short term expected consequence on swans when alternate habitat is not available. From the table, a "+" and "o" allow a server to accept swans. A "-" results in a server accepting only 0.1 of the swans and the remainder are redistributed.

Table 3-2. Expected consequences (degree of effect) of water levels on wetlands, swans, and wetland use by swans when water levels are those recommended by the management of montane wetlands expert system.

SEASON	RECOMMENDED WATER LEVEL	TIME SCALE OF EXPECTED CONSEQUENCE	EXPECTED CONSEQUENCE ON...						
			... WETLANDS		... INDIVIDUAL WETLAND USE BY SWANS	... SWANS			
						ALTERNATE HABITAT AVAILABLE		ALTERNATE HABITAT NOT AVAILABLE	
			INDIVIDUAL	TRI-STATE		INDIVIDUAL SWANS	TRI-STATE SEGMENT	INDIVIDUAL SWANS	TRI-STATE SEGMENT
Breeding	Low	Short term	+	+	-	o	o	-	-
		Long term	+	+	+	+	+	+	+
	Medium	Short term	+	+	+	+	+	+	+
		Long term	+	+	+	+	+	+	+
	High - breeding habitat still exists	Short term	+	+	-	+	+	+	+
		Long term	+	+	+	+	+	+	+
	High - breeding habitat eliminated	Long term	+	+	-	o	o	-	-
		Short term	+	+	+	+	+	+	+
Post-breeding	Low	Short term	+	+	-	o	o	-	-
		Long term	+	+	+	+	+	+	+
	Medium	Short term	+	+	+	+	+	+	+
		Long term	+	+	+	+	+	+	+
	High	Short term	+	+	-	o	o	-	-
		Long term	+	+	+	+	+	+	+
Wintering	Low	Short term	+	+	-	o	o	-	-
		Long term	+	+	+	+	+	+	+
	Medium	Short term	+	+	+	+	+	+	+
		Long term	+	+	+	+	+	+	+
	High	Short term	+	+	-	o	o	-	-
		Long term	+	+	+	+	+	+	+
Pre-breeding	Low	Short term	+	+	-	o	o	-	-
		Long term	+	+	+	+	+	+	+
	Medium	Short term	+	+	+	+	+	+	+
		Long term	+	+	+	+	+	+	
	High	Short term	+	+	-	o	o	-	
		Long term	+	+	+	+	+	+	

The expert system for assessing the contribution an area can make towards the flyway management plan for the Rocky Mountain population of trumpeter swans provides an assessment of whether a server (refuge) can, or cannot, make such a contribution for the wintering and breeding queues. Using this expert system output, the decision support system allows a server to accept all its birds, or accept 0.1 of the birds and redistribute the remainder.

It should be pointed out, here, that the expert systems only indirectly affect the functioning of the server mechanism. From a software perspective, the expert systems exist independently of the multiagent system. However, the facilitator agent can ask the user to run an expert system if it is determined to be needed. And, the agents interact to determine the existence of appropriate expert system output, and they parse information in that output for eventual use in the queuing model. In any situation where an expert system is unable to provide a recommendation for any reason, the default movement probability values are accepted.

3.2.4 *System Output*

Final decision support system output consists of two similar ASCII text files, each with the same name but one with a ".txt" extension (Figure 3-5) and the other a ".dat" extension. The latter differs slightly in format so that it can be more easily imported as a data file for the Matched Pairs Multivariate Permutation Test (MVPTMP) statistical analysis program (Mielke and Berry 2001; Mielke et al. 1996). The basic output data are the predicted numbers of swans at each area for the breeding season subsequent to that of the starting year, as generated by the queuing model. These swan data are combined with both system configuration and user preference information for archive

purposes. Output data and information are assembled cooperatively by the facilitator and move agents, with files actually written by the latter.

File: /data5/decaf/dss/out/rihard_sojda_1992_1992_br_test_cv_jh_ip.txt
BREEDING THRESHOLD VALUE: 60
REMAIN IN QUEUE MULTIPLIER: 0.1
IGNORED EXPERT SYSTEMS: None
ACTIVE REFUGE AGENTS: cv jh ip
CANADIAN BIRDS INCLUDED: Yes
SIMULATED NUMBERS USED: No

Refuge Mask (available queue = 1, restricted queue = 0):

Ref	Year		PO	WI	PR	BR
br	1992-1993	1	1	1	1	
ca	1992-1993	1	1	1	1	
cv	1992-1993	1	0	0	0	
gl	1992-1993	1	1	1	1	
ip	1992-1993	1	0	1	0	
jh	1992-1993	1	0	1	0	
um	1992-1993	1	1	1	1	

Simulated Numbers of Swans:
1660 0 0 0 1 0 0 15 0 0 7 10 0 250 4 5 55 4 45 0 5 0 1 13 39 34 0 56 2205

Figure 3-5. Example of final output file from the decision support system.

3.2.5 *Hardware, OS, software, and compilers used*

The decision support system has been implemented on a Sun Ultra 80, dual-processor, 400MHz workstation with Solaris 8, UNIX operating system. DECAF software was version 2.10 and the related PlanEditor was version 3.0, both from the Computer and Information Sciences Department at the University of Delaware. JAVA compilers used were JDK 1.2.2 from Sun Microsystems. C compilers used were version 6 also from Sun Microsystems. Expert systems were developed using Exsys Developer (version 8.0) from Multilogic Inc. on a personal computer with Microsoft Windows 98 operating system. They were then ported for implementation on the workstation using Exsys'

UNIX rulecompiler (version 8.0.) Exsys' Webruntime (version 8.0) and Apache webserver (version 1.3) software were used on the workstation for final implementation of the expert systems. All agents, expert systems, and necessary files and data for the operational decision support system reside on the UNIX workstation; and, the system is run there.

3.3 **Agent Specifics**

In this section, I describe the specifics of the agent tasks that are central to the ecological aspects of distributed problem solving and the queuing system functions. Emphasis is placed on functions central to implementing the concept of intelligent agents and how they fit together as a multiagent system. I do not discuss those agent tasks that are necessary within DECAF but are only system overhead or agent communications and are not related to the specifics of ecological problem solving. Examples are *_Startup* and *_Shutdown* tasks. Also, the nature of KQML message protocol dictates that return-type messages be sent from receiving to sending agents in certain cases. I generally do not delineate such details in this section. Such particulars, as well as those of individual actions within tasks, can be found in Appendix 5; and, the associated .lsp and .plan files necessary for the DECAF implementation can be found in Appendix 6. All messages being sent within or between agents follow KQML specifications.

3.3.1 *The Facilitator (fac) Agent*

The facilitator agent handles interaction with the user and notifies the move (distribution simulator) agent that the user wishes to proceed with a simulation. The facilitator agent also actively listens for knowledge about specific refuges that could be changing. If it is determined that additional knowledge is necessary, the facilitator agent then handles

interaction with the user by requesting that specific expert systems be run. All communication with the user is via text I/O at the console window from which the facilitator agent was started. Figure 3-6 is the DECAF plan file for the facilitator agent.

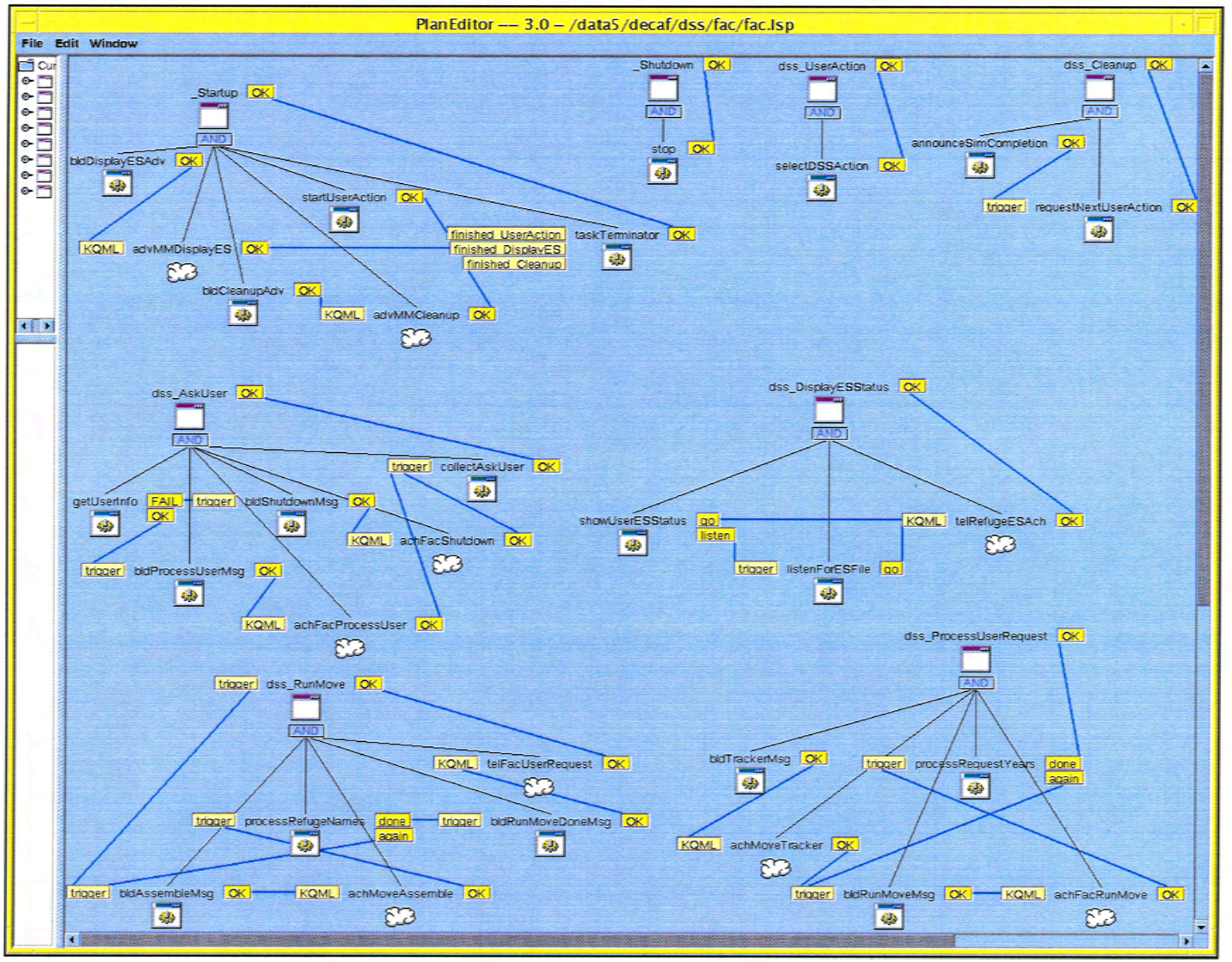

Figure 3-6. The DECAF plan file for the facilitator agent.

3.3.1.1 *dss_UserAction*

This task requests whether the user wishes to proceed with a consultation or quit. If the former choice is made, a local message is sent to dss_AskUser.

3.3.1.2 *dss_AskUser*

This task interactively obtains information from the user about parameters for a consultation, such as user name, start year, etc. It also reads the system configuration file (dss.config) for information about additional parameters (see Appendix 1.) This task sends a message to the local task, *dss_ProcessUserRequest.*

3.3.1.3 *dss_ProcessUserRequest*

This task loops through the number of years requested for the consultation, sending multiple, individual messages pertaining to each year both to the local task, *dss_RunMove*, and to the move agent. The messages include information about the user and the current consultation parameters.

3.3.1.4 *dss_DisplayESStatus*

As a result of an achieve message from a refuge agent, this task requests the user to run a certain expert system(s). As a listening task, it also monitors for specific output files from the expert systems being written to the hard drive of the workstation. Lastly, this task communicates back to the refuge agent(s) that its achieve request has been completed.

3.3.1.5 *dss_RunMove*

The number of refuges activated is monitored. Looping through the list of refuges, separate messages are sent to the move agent requesting that it initiate its activities by assembling refuge information from each refuge agent.

3.3.1.6 *dss_Cleanup*

This task provides information to the user about a just completed consultation and sends a message to *dss_UserAction*, essentially starting another consultation.

3.3.2 *The Refuge Agent*

The refuge agent provides the framework for actual instantiations of this agent type. An agent called "refuge" is never implemented, per se. Refuge agents are activated that have the two letter names as indicated earlier, although they are based on their common DECAF files (refuge.lsp and refuge.plan) and the associated java classes. There may be one to seven such agents active during a consultation. Figure 3-7 is the DECAF plan file for the generic refuge agent. The purpose of each individual refuge agent is to determine the status of its own knowledge as needed for the particular consultation. If they are current, they report this belief to the move agent. If not, this alternate belief is communicated to the facilitator agent so that the user can be prompted to run the needed expert systems.

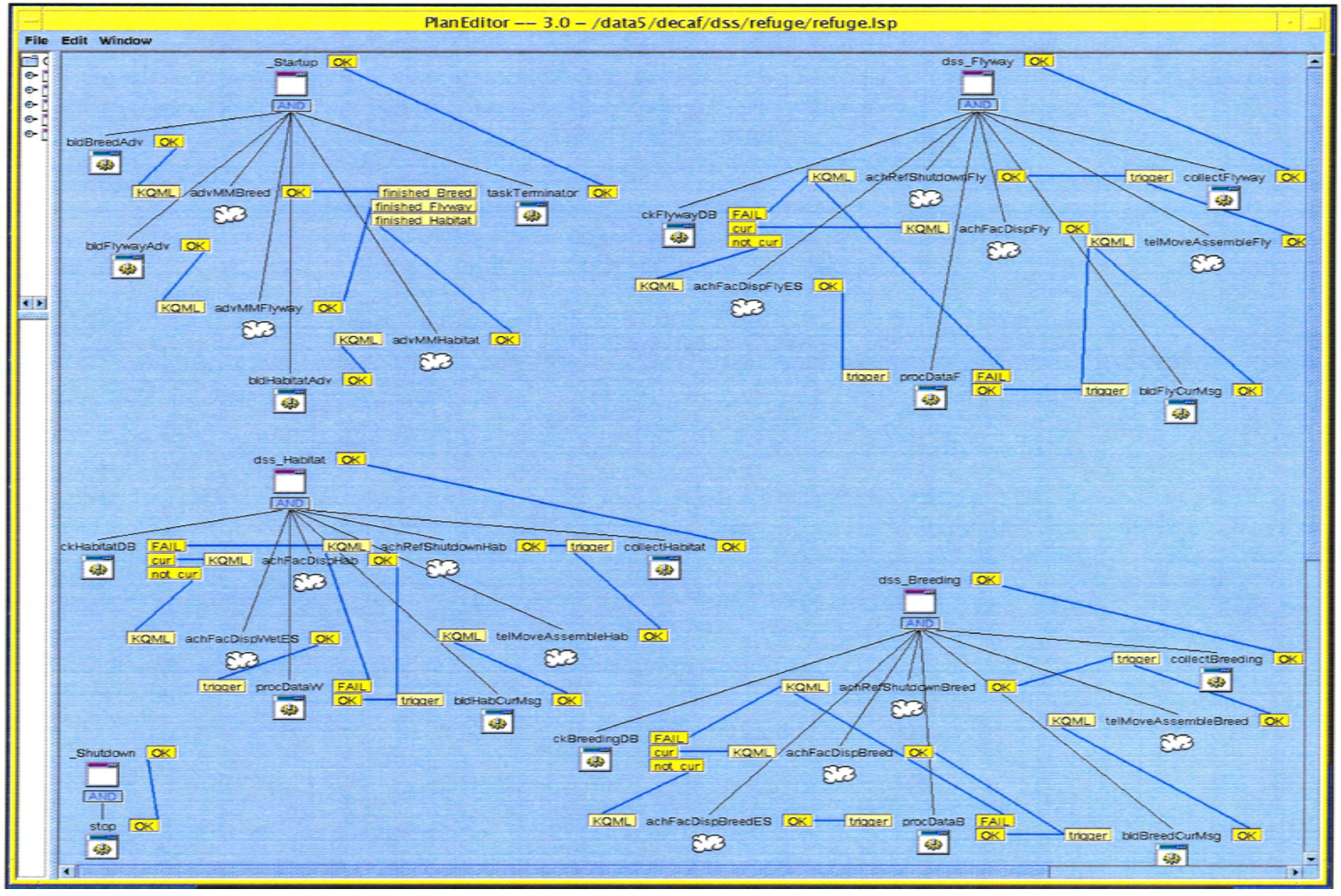

Figure 3-7. The DECAF plan file for the refuge agent.

The refuge agent consists of three primary tasks, each one performing analogous sets of actions but gathering information and updating its beliefs about one of three different segments of knowledge: breeding habitat quality (dss_Breeding), wetland management recommendations (dss_Habitat), or contributions to flyway management (dss_Flyway.) Because each of these primary tasks are identical in their basic functioning, I will discuss them together.

3.3.2.1 *dss_Breeding*, dss_*Habitat*, and *dss_Flyway*

Each task listens for the move agent's need for updated information about ecological knowledge. Upon hearing this message, these tasks check the decision support system's respective databases to determine if any expert systems need to be run. If any do, an achieve message(s) is sent to the facilitator agent, requesting that it, in turn, request the user to run the appropriate expert system(s). It is important to note there can be more than one refuge agent active at this point, but all I/O with the single user is handled via the facilitator agent. Each of the refuge agent's primary tasks then waits, listening for a return message indicating that their respective expert system(s) has been run. Upon hearing that information, the task then parses the expert system output file and updates the appropriate database. In the case of dss_Breeding, the threshold value from the configuration file is compared with the parsed value and determines a server's (refuge's) ability to accept swans. In the case of dss_Flyway, the parsed value is Boolean and is used directly to determine server ability. In the case of dss_Habitat, the parsed value is used in conjunction with rules based on the knowledge represented in Table 3-2 to determine server availability. See the section, "Service Mechanism: Connecting Agents and Expert Systems" for further details about each of these three algorithms. If either the database was current at the start and no expert system runs

were needed, or it has now been made current, the move agent is notified of this updated belief by each, individual task of the refuge agent.

3.3.3 *The Move Agent*

The move agent contains the core of the queuing system and focuses on simulating the distribution of swans. It requests refuge agents to provide information, assembles all necessary system data, runs the queuing model, and writes output files. Message passing between the move agent and the others is routine. Figure 3-8 is the DECAF plan file for the move agent.

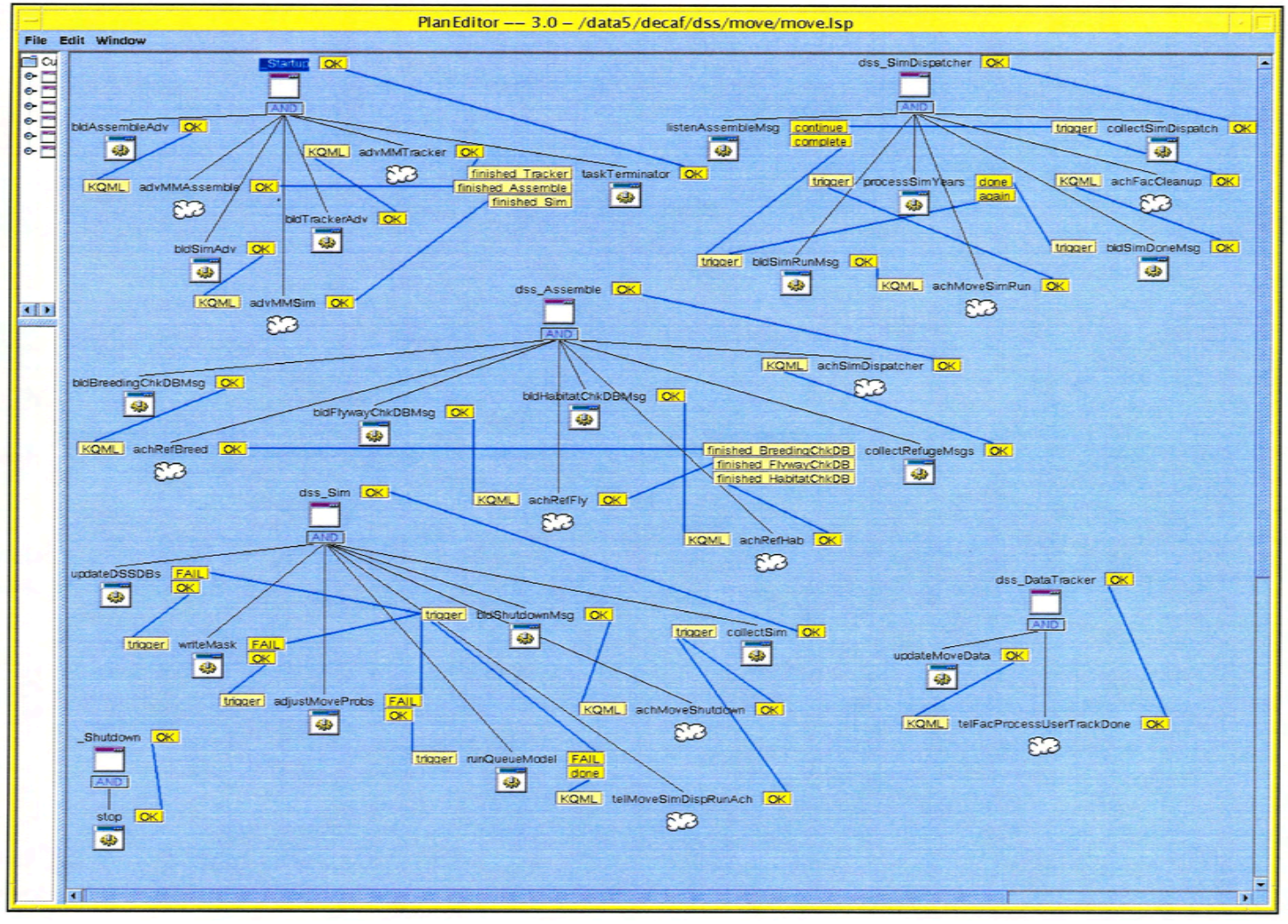

Figure 3-8. The DECAF plan file for the move agent.

3.3.3.1 *dss_DataTracker*

This task maintains the internal information for the move agent about the current consultation as received from the facilitator agent.

3.3.3.2 *dss_Assemble*

This task sends separate messages to each active refuge agent for each piece of ecological information (breeding, wetland, flyway) that may be needed. The task then waits for all messages to return from the refuge agents indicating that all databases are current. It then notifies the local task, *dss_SimDispatcher*, of this belief.

3.3.3.3 *dss_SimDispatcher*

After determining that information has been assembled for all refuges by listening for information internal to the move agent and interacting with the JAVA utility class, this task requests that the task, *dss_Sim*, be run for each year. These runs are requested in one year increments, looping through the entire time interval as originally requested by the user. The task, *dss_SimDispatcher*, creates the final output files, but the only information included at this stage is header information and information describing the current consultation (Figure 3-8.) Actual output data of simulated swan numbers is written as part of a different task, *dss_Sim*. When the task, *dss_SimDispatcher*, hears that all simulations have been run, the facilitator agent is notified of completion. A request also is sent to the facilitator agent to inquire about the user's interest in additional activity.

3.3.3.4 *dss_Sim*

After examining databases which have already been made current by the refuge agents, this task creates a mask file that governs server acceptability (whether particular refuges

are considered as having habitat of acceptable quality) for the different queues. Using the mask, this task adjusts the matrix of base probabilities of movement for the current server (refuge and season). The earlier section, Service Mechanism: Connecting Agents and Expert Systems, provides more details about the algorithm. Next, the central queuing model, itself, is run. This provides the simulated distribution of swans, appending these numbers to the consultation output files created by the task, *dss_SimDispatcher* (Figure 3-8). Each time the task, *dss_Sim*, is executed, the output files for the consultation are appended with the additional, simulated swan numbers. A message is sent internally to notify the task, *dss_SimDispatcher*, that the queuing model has been run for the year requested.

3.4 **The Expert Systems**

Three expert systems have been developed: one for assessing the quality of trumpeter swan breeding habitat that affects servers in the breeding queue; one for recommending water levels in montane, palustrine wetlands that affects servers in all queues; and one for assessing the contribution a particular site can make towards meeting flyway objectives that affects servers in the wintering and breeding queues. Because the expert systems actually write output files to the hard drive of the workstation, access to the working version that is compatible with the overall decision support system is password protected and is accessed currently at http://swan.msu.montana.edu/as/es.html.

I have followed the evolutionary prototyping method of software development recommended by Sprague and Carlson (1982) and Carter et al. (1992). Essentially, this requires developing a prototype system and conducting verification and validation as part of the development process, continually refining the system. The actual process

followed for building the prototype consisted of five steps: (1.) completing a requirements analysis to determine the question to be addressed, (2.) conducting a knowledge engineering workshop to assemble the pieces of pertinent knowledge, (3.) constructing a flowchart of the underlying ecological logic in the system, (4.) encoding that knowledge into digital format, and (5.) making the system available on the World Wide Web. All three expert systems were developed in this manner and followed the approach described by Sojda et al. (in press) for the breeding habitat expert system.

All knowledge engineering sessions were led by two knowledge engineers using standard techniques (Scott et al. 1991) and varied in length from 1-3 days and used 2-4 experts. Notes from those sessions along with further knowledge engineering details can be found at http://swan.msu.montana.edu/cygnet. Knowledge acquired from the experts included both facts and problem solving logic. Such knowledge was encoded with a commercial expert system shell, using typical production rules and organized into decision trees (Figure 3-9.) Such rules are generally of the form:

IF the wetland is generally ice-free during the prelaying period
AND 30% of the wetland has Sago pondweed present
THEN the wetland should be considered good prelaying habitat

Utilities within the development shell were used for verification of logical consistency of each expert system. The utilities automatically do a static check for problems such as incomplete rules and trees and can also dynamically check the system by stochastically simulating runs. Each expert system was checked using 500,000 such simulated runs.

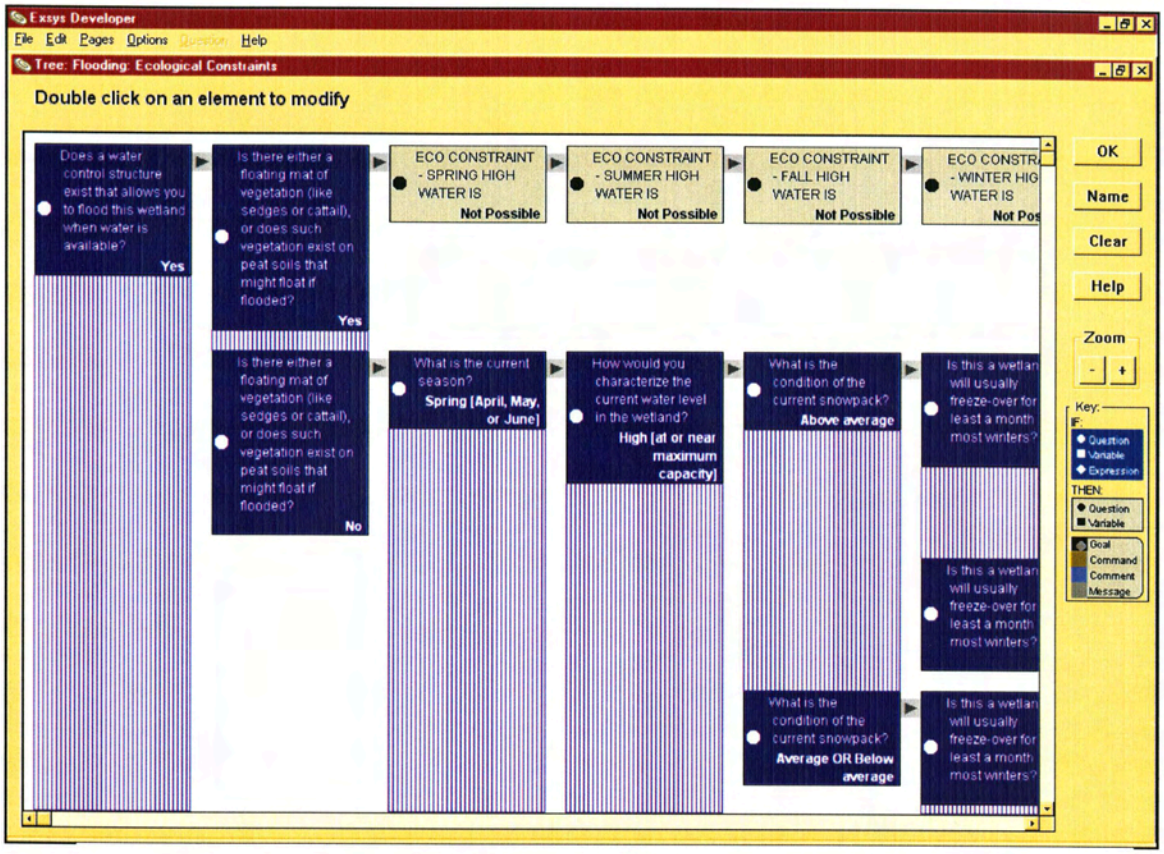

Figure 3-9. Example of a tree structure used to encode knowledge in the wetland management expert system as depicted graphically in the expert system shell.

3.4.1 *Breeding Habitat Needs for Trumpeter Swans*

This expert system helps determine the quality of a wetland as breeding habitat for trumpeter swans based on knowledge of: (1.) wetland depth, (2.) size of the wetland, (3.) growing season length, (4.) food resources for pre-laying birds, (5.) condition of emergent vegetation and other aspects of providing suitable nest sites, and (6.) brood rearing requirements. Knowledge engineering resulted in our categorizing what were determined to be pertinent facts into six decision trees and twenty standalone rules, for a total of 157 total rules. The six trees are: depth and size of the wetland, length of the annual ice free period, prelaying food resources, nest site availability, propensity for nest flooding, and brood habitat. The logical flow of how these decision trees and production

rules interact is found in the flowchart depicted in Figure 3-10. The expert system also uses 18 goals and 21 variables and interacts with the user through 27 potential questions. The actual output value is an integer from 0 to 100; the higher the number, the more confident one can be that satisfactory habitat exists. A zero indicates that at least one habitat component is severely compromised. The output value is a characterization of the degree to which the wetland represents ideal habitat for breeding trumpeter swans.

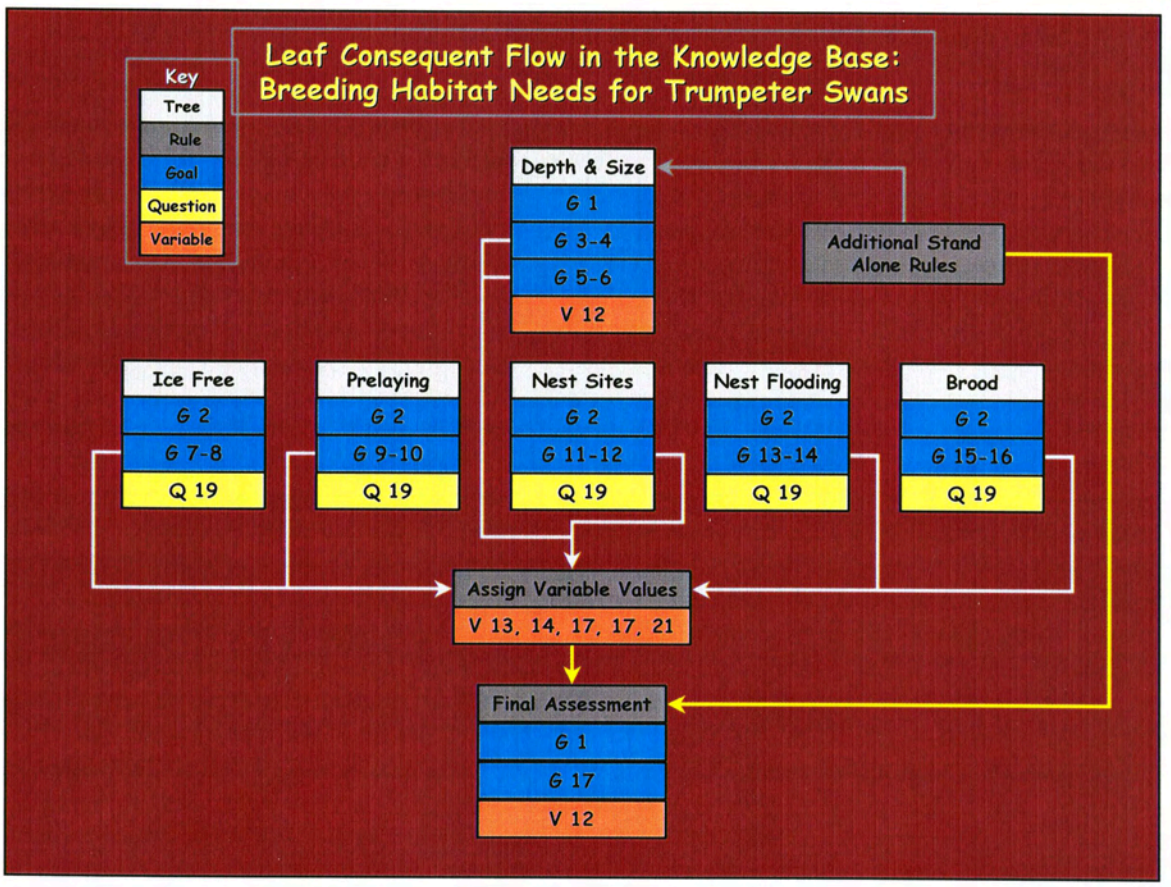

Figure 3-10. Organization of knowledge in the expert system, "Breeding Habitat Needs for Trumpeter Swans."

The primary knowledge engineering session was three days in length. One person, Leigh H. Fredrickson, whose expertise was utilized was a professor from the University of Missouri specializing in waterfowl and wetlands ecology. The second, Todd A. Grant, was a wildlife biologist for J. Clark Salyer National Wildlife Refuge. The third expert, Murray K. Laubhan, was a wetlands research ecologist with the Midcontinent Ecological Science Center of the United States Geological Survey. I was one of the knowledge engineers and the other was David B. Hamilton, an ecologist with the Midcontinent Ecological Science Center of the United States Geological Survey.

3.4.2 *Management of Palustrine Wetlands in the Northern Rockies*

This expert system suggests seasonal water levels within a one year time frame for palustrine emergent wetlands. Actual recommendations are provided as either high, medium, or low. It is intended for areas in the Intermountain West, but may have limited application elsewhere. Recommendations are provided for four seasons and are based on knowledge of: (1.) vegetation, (2.) hydrology, (3.) water quality, (4.) soils, (5.) weather, (6.) wetland dynamics, and (7.) migratory bird management objectives (Figure 3-11). Because it is a planning tool, the system does not attempt to provide recommendations for the current season. Knowledge has been organized into 43 trees and 10 standalone rules, for a total of 1882 rules. In addition, the expert system utilizes 18 goals and 20 variables and interacts with the user through 78 potential questions.

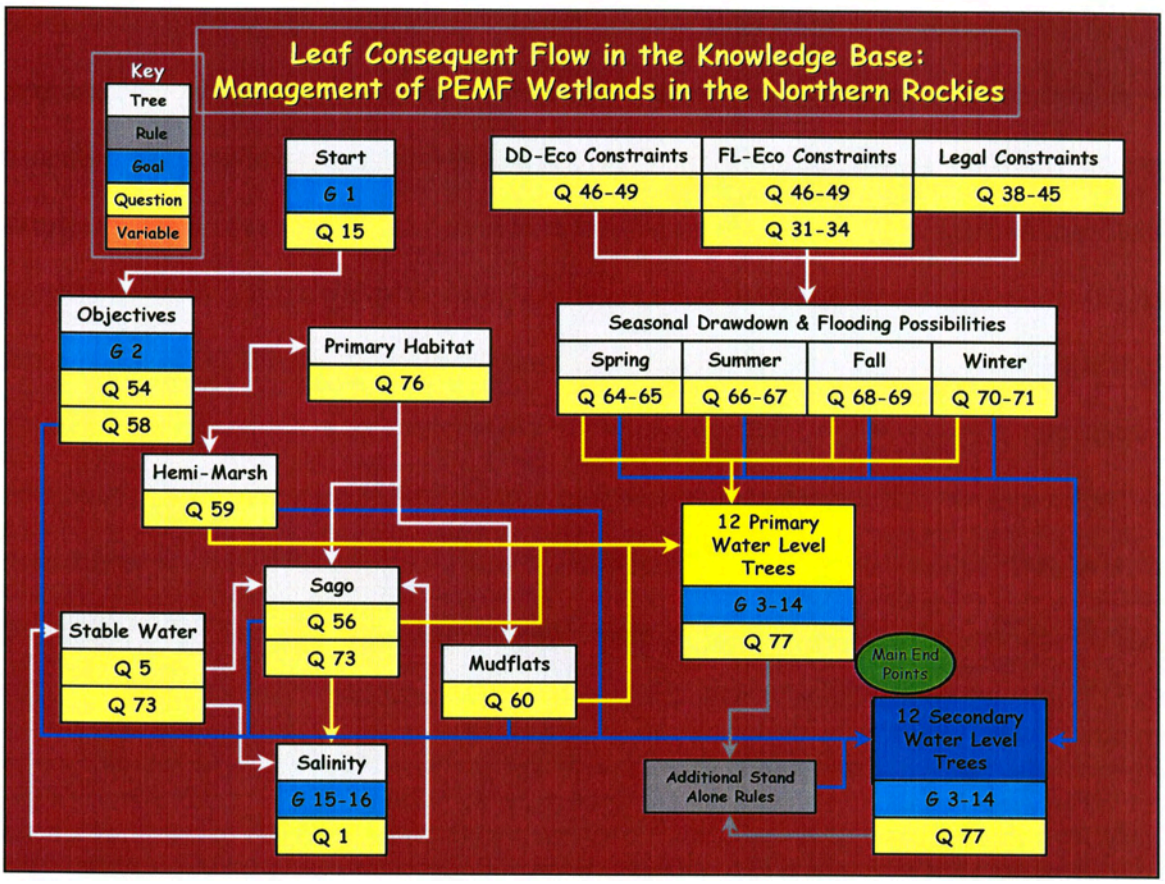

Figure 3-11. Organization of knowledge in the expert system, "Management of Palustrine Wetlands in the Northern Rockies."

The knowledge engineering session was three days in length. Both people whose expertise was utilized were professors specializing in waterfowl and wetlands ecology, each with more than 30 years of such research experience. One, Leigh H. Fredrickson, was from the University of Missouri; the other, John A. Kadlec was from Utah State University. I was one of the knowledge engineers and the other was David B. Hamilton, an ecologist with the Midcontinent Ecological Science Center of the United States Geological Survey.

3.4.3 *Principles of Flyway Management*

This expert system is actually entitled, "Assessing the Contribution an Area Can Make Towards the Flyway Management Plan for the Rocky Mountain Population of Trumpeter Swans." It provides an ecological assessment about whether an area can make a contribution towards breeding and wintering conditions related to population and distribution goals as depicted in the 1998 Flyway Management Plan for the Rocky Mountain Population of Trumpeter Swans (Subcommittee on Rocky Mountain Trumpeter Swans 1998.) This logic is depicted in Figures 3-12 and 3-13. Knowledge has been organized into 3 trees and 1 standalone rule, for a total of 165 rules. In addition, the expert system utilizes 10 goals and 23 variables and interacts with the user through 13 questions.

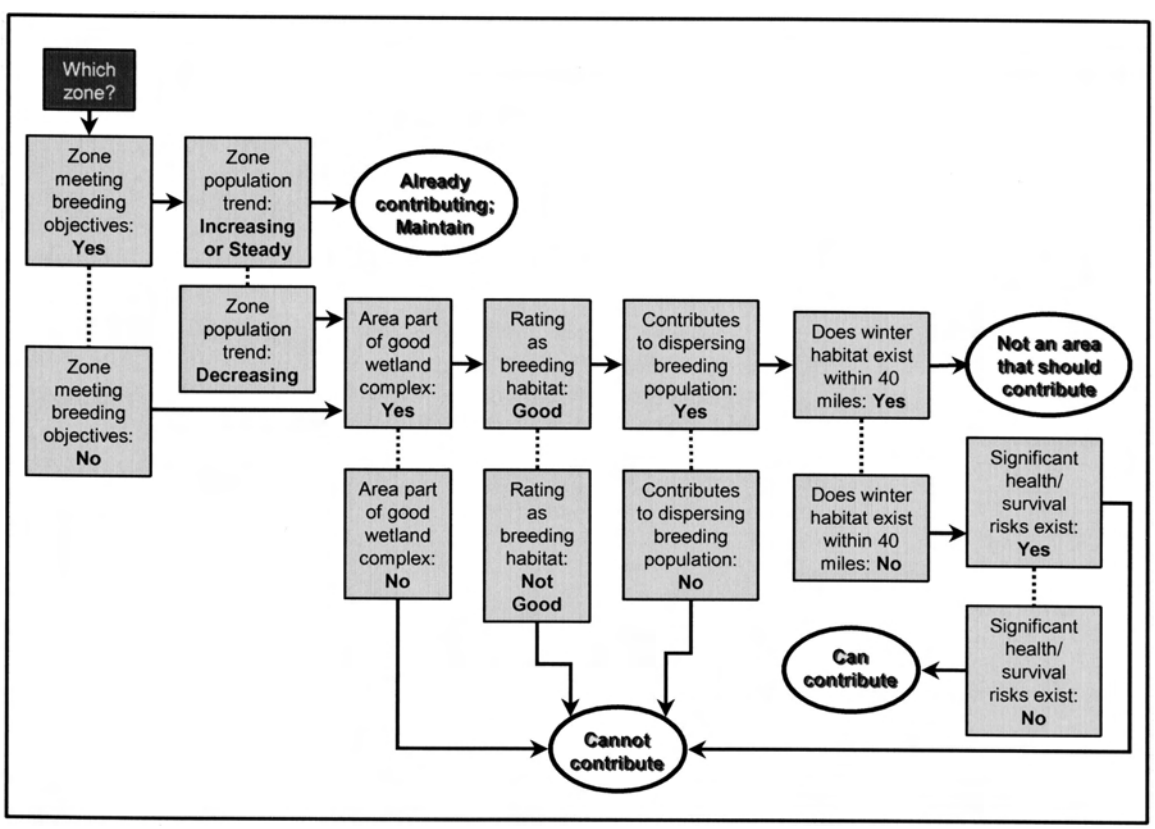

Figure 3-12. A flowchart depicting the knowledge and logic about principles of flyway management related to addressing the question: "Can this area contribute to the Rocky Mountain Population 1998 Plan's breeding population goals?"

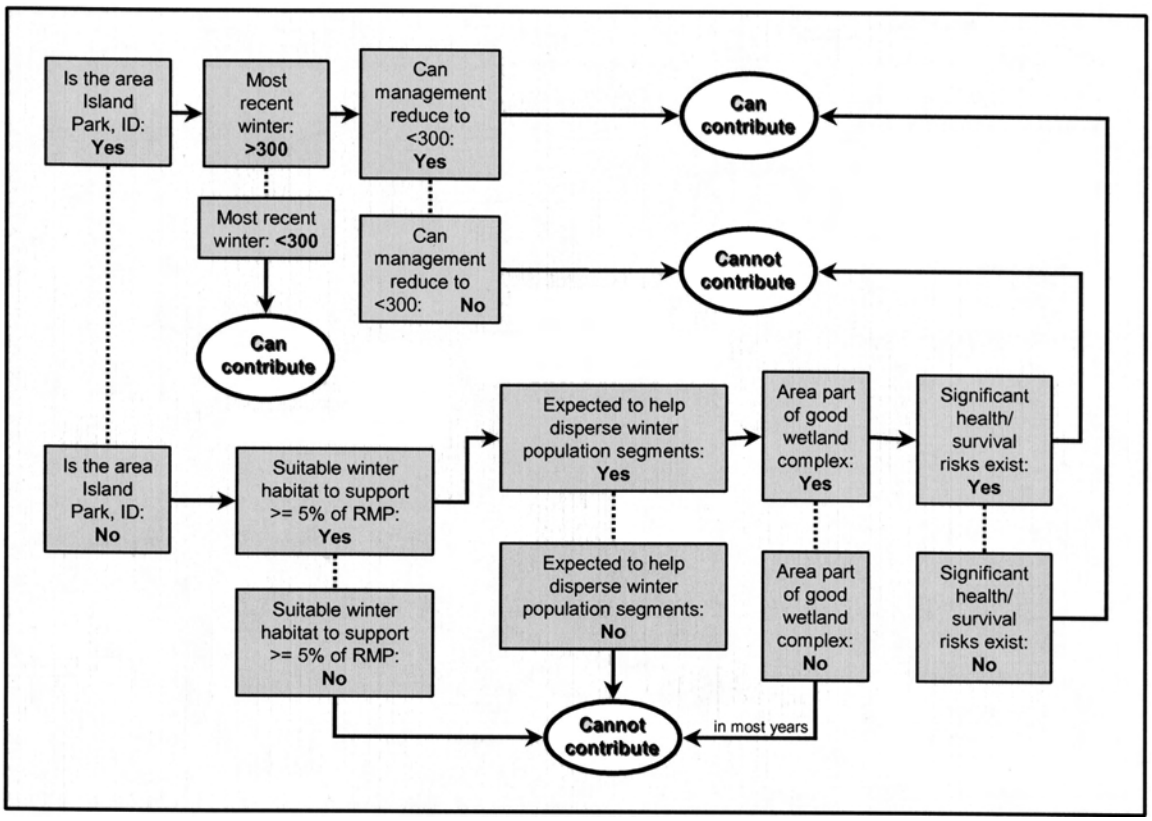

Figure 3-13. A flowchart depicting the knowledge and logic about principles of flyway management related to addressing the question: "Can this area contribute to the Rocky Mountain Population 1998 Plan's wintering population goals?"

This expert system uses the September survey data for the period 1983-2000 along with seven and five year trends in that data. Data are from unpublished reports of the U.S. Fish and Wildlife Service and archived at Red Rock Lakes National Wildlife Refuge. To minimize the variability when examining multiple year trends, only numbers of white birds were analyzed. Exact multiple response permutation procedures (EMRSP) of Mielke and Berry (2001) were used to determine the presence of an autoregressive pattern in multiple seven or five year windows, depending on data availability. For the years 1989-2000, the data were examined for the presence of a pattern over seven years. For the years 1987-1988, the data were examined for the presence of a pattern over five years. A trend was arbitrarily defined as occurring when the p-value was less

than 0.05. In situations where a trend existed, a linear regression analysis was then done using SAS software (SAS Institute, Inc. Cary, N.C. 2000) to determine whether the previously determined pattern represents an increasing or decreasing trend. The value corresponding to the appraisal of a trend over the multiple year period was then assigned to the most recent year in that period. For example, for the period 1986-1992, no trend was detected for the Centennial Valley. The expert system would then use this as the value for 1992.

There were two knowledge engineering sessions. The first focused on flyway management as administered in the current system of Flyway Councils and coordinated by the federal Office of Migratory Bird Management. It was three days in length and involved four people. Three, John E. Cornely, David E. Sharp, and Robert E. Trost, were specialists in migratory bird management from the United States Fish and Wildlife Service; and one, Murray K. Laubhan, was a wetlands research ecologist with the Midcontinent Ecological Science Center of the United States Geological Survey. The second session was slightly different and centered on the ecological nature of how to address the requirements of migratory birds distributed over time and space. Much of the knowledge elicited in the second session has been incorporated in the service mechanism of the queuing system as well as in this expert system. There were two people involved in the second session. One person, Leigh H. Fredrickson, was a professor from the University of Missouri specializing in waterfowl and wetlands ecology; the other, Murray K. Laubhan, was a wetlands research ecologist with the Midcontinent Ecological Science Center of the United States Geological Survey. For both sessions, I was one of the knowledge engineers and the other was David B. Hamilton, an ecologist with the Midcontinent Ecological Science Center of the United States Geological Survey.

3.5 **Conclusions**

The purpose of this project was to build a decision support system by applying multiagent system methods and integrating queuing system theory. My system does, in fact, simulate the movement of swans in time and space, relying on ecological knowledge to do so. Here, I present observation-based conclusions about the framework developed and the methodologies used in the multiagent system. Evidence indicating whether intelligent agents were developed is examined first. Then, an appraisal is made of queuing systems for modelling waterfowl migration. Empirical validation of the system based on the simulations of swan movements, however, is described elsewhere.

3.5.1 *Evidence for a System of Cooperating Intelligent Agents*

Multiagent systems are relatively new and are not yet typically developed with standard ("off-the-shelf") methods and procedures. In addition, such systems have not yet been applied in the field of waterfowl ecology. Because I was attempting to build a multiagent system oriented towards modelling swan movements, it seems of primary importance to show that a system of cooperating agents was, indeed, developed. I will show that the three criteria that fundamentally define an agent have been met in my system: agents must be autonomous, capable of sensing their environment, and able to take action in response to changing conditions. The agents' listening and response capabilities provide the nexus for addressing distributed problems. Furthermore, the perspective of a BDI architecture was adopted, allowing the system to center on a belief structure and the ability to update those beliefs as appropriate.

3.5.1.1 *Autonomy*

At a low level, each agent exists as its own set of programming elements, linked only by the action execution scheduling of DECAF. In my system, each agent is both started and shut down independently of each other. In addition, the persistence of each is not dependent on any external elements aside from hardware, operating system, and basic DECAF process continuance. At a higher, algorithmic view, the agents also are self-directed in that they do not require direct commands from other agents to initiate their own tasks. These characteristics lead me to conclude that my agents do embody the essential agent criteria of autonomy. The latter aspect of being self-directed, however, is somewhat vague when programming DECAF agents; i.e., it is not total autonomy that exists. Certain tasks within an agent may require information from another agent as a trigger, or tasks may be initiated as the result of an achieve performative having been sent. DECAF agents are programmed, from a conceptual perspective, as a series of production rules. The underlying software schedules the execution of tasks and actions partially based on those rules. In a similar vein, the provisions of an action (antecedent of a rule) in one agent can often have its roots in the outcome of an action (consequent of a rule) in a different agent via passing of KQML messages. The development of my multiagent system demonstrates aspects of strong autonomy in agents, but also utilizes the strength of DECAF to build systems of communicating and cooperating agents.

3.5.1.2 *Sensory Capability: Listening*

In general it is important that agents be capable of sensing their environment, but this can take several forms. The decision support system does not have any direct connection to sensors representing incoming data streams from the physical world. The system does, however, have the capability of listening, or sensing, at two levels. First, the nature of DECAF agent communication through message passing hinges on an

agent's ability to both send and receive messages. This is inherent in the standard, "non-local task" for inter-agent communication in DECAF. Related to this is the parsing of "provision cells" at the level of JAVA methods to transfer actual information within and between agents. Together, these make up a kind of low-level listening capability, listening for messages among agents. In my system, such messages are both providing information (tell performatives) and requesting other agents to do something (achieve performatives). In addition, the facilitator agent has an active listening capability for determining when particular files pertaining to ecological knowledge have been written to the hard drive. It should be pointed out, however, these files result from real-world interaction of the user having run the expert system(s). I conclude that my agents do collect information about their environment, but recognize that this is implemented in a low-level, virtual environment. And, I recognize that these are not sophisticated listening facilities or data stream handlers as might be necessary in some other applications.

3.5.1.3 *Response Capability*

As conditions change in the virtual world of the decision support system, each agent responds as appropriate. Such response varies from requesting that the user run an expert system, to sending and responding to messages among agents, to parsing ecological information from newly arrived files, to running the base queuing model once information is complete. It is such response to changing conditions, as perceived by the agents, that allows the queuing system's service mechanism to function. It is also such response that is the heart of agents addressing, cooperatively, the distributed nature of flyway management.

3.5.1.4 *Relationship to BDI Architectures*

As pointed out earlier, Graham (2001) has delineated how the DECAF framework is based on the concept of intention revision and how this has been implemented via the scheduler. Furthermore, agents are programmed as hierarchical task networks. It is the provision of information to populate and update the antecedents of rules governing the networks that is, in fact, setting and updating the agent's beliefs. The connections between listening capability and responding to changing conditions is an effectual implementation of a BDI agent architecture.

3.5.2 *Queuing Systems and Waterfowl Migration*

Queuing systems have not been previously used as a modelling paradigm for waterfowl migration. The intrinsic character of queuing theory is to represent the spatial and temporal distribution of entities and how they move, are placed in queues, and are serviced. The decision support system made use of these characteristics in modelling the movement of swans through their migration corridor. The parallel of queuing system servers and service mechanisms with waterfowl management areas and annual life cycle events made the transfer of the theory to practical application straightforward. Furthermore, the modularity of DECAF agents provided me the opportunity to apply the internal agent tasks and actions to the specific elements of queuing system theory. The nature of agents as independently responding to their environment allowed them to function as ideal entities for implementing the service mechanism in my model of waterfowl migration.

3.5.3 *Future Directions*

An inordinate number of examples of research related to parameterization of the various components of the decision support system could be suggested. These are not

discussed, here, because suggestions for refining model parameters should be based on validation experiments and sensitivity analysis. I will suggest some ways the system, itself, might be improved, however. There also is room for developing and shaping conceptual theory associated with multiagent systems and applying queuing theory as frameworks for ecological modelling.

3.5.3.1 *System Implementation Improvements*

The following is a list of several possible system improvements:

- Use applets for a graphical user interface, especially for system startup parameters.
- Provide automated trend analysis of system output for determining whether virtual population change has indeed occurred in simulated numbers.
- Develop sufficiently robust ecological databases that the expert systems could access and provide recommendations unassisted by the user. Each expert system could be implemented as a stand-alone agent.
- Allow any area, or at least an increased number of areas, to affect the matrix of movement probabilities, should sufficient ecological knowledge be available.
- Provide the facility for the user to suggest alternate management options (override expert system output) so that the effects of such management could be simulated.
- Apply probabilistic methods as a way of quantifying possible movement of swans among areas within a season as an alternative to the relatively simplistic weighting method that I used for calculating overall movement probabilities.

3.5.3.2 *System Theory Development*

Existing DECAF software allows the application of distributed processing, essentially having different instantiations of agents running on different computers. If coupled with the appropriate facilities, this could allow refuge biologists and ecological experts to incorporate real-time advice, knowledge, and data into the system. This would, at least conceptually, move the system even further towards a cooperative distributed problem solving methodology if coupled with common knowledge bases and a common problem solving algorithm. It would be interesting to empirically compare the simulated distributions of swans to see if actual queuing system behavior would significantly change.

Building agents that could utilize all the cross spatial and temporal scale information related to the consequences of water levels on wetlands (Table 3-2) would add additional dimensions to the algorithms governing the service mechanism. Using negotiation among agents as a way of implementing a constraint satisfaction algorithm (Armstrong and Durfee 1997; Pinson et al. 1997) for incorporating such cross-scalar knowledge might begin to mimic some of the ecological complexity involved in the flyway/wetlands/waterfowl domain. As in much of this domain, the knowledge being used is not definitive, but such a model could be used to test which aspects of the knowledge base are the most critical.

Propagating uncertainties associated with pieces of knowledge, as well as actual data where it exists, through any knowledge-based system is hardly a trivial task. Classical methods include certainty factors (Buchanan and Shortliffe 1984), Dempster-Shafer theory (Horvitz et al. 1986; Shafer 1988, 1990), and Bayesian methods (Pearl 1990; Jensen 1996). In fact an entire subdiscipline has developed regarding the understanding of uncertainty in artificial intelligence. Representing the uncertainties in

the decision support system would provide outputs with greater empirical value if those probabilities could be appropriately propagated through such a complex network and assigned to the final simulated distributions. The complexity of models that use ecological knowledge as a basis for simulating the distribution of waterfowl across large landscapes presents important and intricate challenges for methodology development. Difficulties arise not only from a representation standpoint, but also from the need to segregate dependent and independent probabilities, and from the difficulties in identifying causality in complex systems.

Because the use of queuing systems for modelling waterfowl migration is hardly a mature methodology, developing systems using stochastically modified arrival times and stochastically modified processing methods holds promise for improving the degree to which such a system would emulate real world situations. It is not entirely clear, however, that such changes would improve the value of the system to waterfowl managers. Empirical sensitivity analysis regarding arrival times and processing methods could guide preliminary planning for further such development.

CHAPTER 4

EMPIRICAL EVALUATION OF A MULTIAGENT SYSTEM FOR TRUMPTER SWAN MANAGEMENT

4.1 Introduction

A multiagent system was developed to assist waterfowl managers with the management of trumpeter swans (see Section IV). DECAF software (Distributed Environment Centered Agent Framework) was used to construct the agents, allow for their interaction, and to handle user I/O (Graham and Decker 2000; Graham 2001). Within this framework, a queuing system (Dshalalow 1995; Hillier and Lieberman 1995) was integrated that simulates the geographic and temporal distribution of swans. The system utilizes output from expert systems related to ecological aspects of the flyway management of migratory birds, especially trumpeter swans and manipulation of their habitat. These expert systems also were developed as part of the overall decision support system (Sojda and Howe 1999.) Here, I report on verification and validation of the entire decision support system and some of the component modules. The general question to be addressed is whether the system achieved the purposes for which it was intended.

4.1.1 *Verification and Validation Defined*

Wallace and Fujii (1989) define verification and validation of software as the analysis and testing "to determine that it performs its intended functions correctly, to ensure that it performs no unintended functions, and to measure its quality and reliability."

Verification is ensuring that the system is internally complete, coherent, and logical from a modelling and programming perspective. Validation is examining whether the system is realistic and useful to the user or decision maker. Was the system successful at addressing its intended purpose? Were the stated objectives of the project met (Geissman and Schultz 1988)? O'Keefe et al. (1987) state: "Validation means building the right system. Verification means building the system right." I use the term evaluation to encompass both verification and validation, but distinguish between them when used independently. I agree with Adelman (1992) that both should be part of the development process, and evaluators should specifically be part of the development team. There is a plethora of discussions about the semantics of evaluating models, and Johnson (2001) provides an excellent summary related to natural resource management. My work is interdisciplinary, and I choose to accept the above definitions because I feel that they are both the most logical and most widely accepted, especially in the field of artificial intelligence. This delineation is particularly important in my case because my project had not been provided formal, software development requirements and specifications to be achieved.

A rich literature exists on verification and validation of expert systems and other artificial intelligence methods. These fit the general area of decision support systems well, because expert systems are almost always a type of decision support system (Adelman 1992; Bahill 1991; Cohen and Howe1989; Grogono et al. 1991; Gupta 1991; Hamilton et al. 1991; O'Leary 1994). Although interest in the theory and application of intelligent agents and multiagent systems has blossomed in the last decade, no widely accepted methodology pertaining to their evaluation has yet emerged.

4.1.2 *An Overview of Potential Methods for Verification and Validation*

Adelman (1992) hinges successful implementation of decision support and expert systems on incorporating three evaluation procedures: (1.) those that examine the logical consistency of the system algorithms themselves (verification), (2.) those that empirically test the predictive (in my case, ecological) accuracy of the system (validation), and (3.) those that document user satisfaction.

Stuth and Smith (1993) followed the ideas of Eason (1988) and recommend iterative prototyping methods for decision support system development. When using such methods, verification and validation are part of the iterative process of system development. Verification should be performed at the stage prior to any delivery of a working system to users, even if only a prototype system has been developed. General validation might be done at this stage as well, with more detailed efforts performed once an operating system is delivered. If one subscribes to the concept that software development can be a living process, then verification and validation are part and parcel to that living process and need to continue as system refinements and re-deployments continue (Carter et al. 1992; Stuth and Smith 1993).

Sprague and Carlson (1982) recommend that an organization building their first decision support system recognize that it essentially is a research activity, and that evaluation should center on a general, "value analysis". They state that iterative prototyping will ensure a quality product from the managers' perspectives, but recognize the qualitative nature of such evaluation. It is imperative that analytic and quantitative rigor be added (Adelman 1991; Adelman 1992; Andriole 1989; Cohen and Howe 1989) beyond the "soft testimonials" often seen. Sensitivity analysis can be a powerful tool for validation, especially for heuristic-based systems, and for systems where few or no test cases are available for comparison (Bahill 1991; O'Keefe et al. 1987). Another issue suggested by

Rushby (1991) is that it is necessary to show not only how well a system performs, but also to show that it can avoid a catastrophic recommendation. This is important in many natural resource venues because of the great concern for long term, irretrievable ecological changes.

Verification and validation of knowledge-based and other decision support systems are known to be more problematic than in other modelling efforts for many reasons (Gupta 1991). Under some conditions, modelling researchers can test performance against a preselected gold standard. Often in natural resource issues, such a standard does not exist. This is particularly true with near real-time decision support that is expected to predict and guide future scenarios while those scenarios are, in fact, unfolding. Also, not only is it important for a system to handle the most common cases, it ought to be able to deal with extreme events. This latter ability is one characteristic often only found with human experts; but, extreme events are not only common in, but often drive, ecological systems.

It is sometimes possible to test expert system performance against an independent panel of experts (O'Keefe et al. 1987). Two concerns must be addressed, however. First, the panel of experts needed for such an evaluation must not be the same people who will be closely connected to system development itself. To do so would add such confounding effects that no reasonable experimental design is feasible. Second, one of the basic tenets of using decision support systems for complex issues is that such questions can be beyond the capability of single persons to conceptualize and solve (Boland et al. 1992; Brehmer 1991).

Wallace and Fujii (1989) provide a comprehensive matrix of 41 techniques and tools that can be applied to 10 verification and validation issues. Cohen and Howe (1989) take a

slightly different approach, but also discuss evaluation from the perspective of the software development life cycle. They emphasize empirical studies to accomplish such evaluation, whether one is focusing on verification or validation. Specifically for knowledge-based systems, Murrell and Plant (1997) provide a categorization of 145 different automated techniques for testing such systems.

4.2 **A Modelling Perspective**

Models are abstractions of reality. They are representations of some portion or aspects of the real world. There are several models integrated into the decision support system for trumpeter swan management. Each of the expert systems is a digital representation of ecological knowledge within a specific domain. The queuing system is a representation of how swans move among specific areas. The refuge, move, and facilitator agents represent the ecological knowledge and the logic of how swans are distributed in the flyway in response to certain ecological conditions. By combining all these models, the overall system provides a digital representation of one way for waterfowl biologists to evaluate their management actions.

4.2.1 *Purpose of the Multiagent System*

The broad question I have posed is: Are multiagent systems an effective platform for integrating flyway management into site-specific decision support for trumpeter swan management? Within this context, I also was interested in determining the effect that encoded ecological knowledge could have on simulated distribution of swans as represented by a queuing system. The overall purpose of the system is to simulate the effect of sets of specific management actions on swan distributions. The 1998 Management Plan for the Rocky Mountain Population of Trumpeter Swans (Subcommittee on Rocky Mountain Trumpeter Swans 1998) is one such set of

management actions at a flyway level. The expert system, Management of Palustrine Wetlands in the Northern Rockies, also provides a set of management recommendations, but at a site specific level. All evaluation of the system hinges on attempting to determine how well the system addresses the stated purpose. Furthermore, from a theoretical perspective I am interested in determining the value of using intelligent agents for reasoning about ecological questions in multiple scales.

4.2.2 *Why Empirical Evaluation Is Important in the Trumpeter Swan Domain*

It is easy to argue that evaluation of all decision support systems is important. In the case of trumpeter swans, there are ecological and public policy reasons that increase the importance of ensuring that the right system has been built and been built correctly. First, swan numbers have not increased in the Tri-State Area to targeted levels (Subcommittee on Rocky Mountain Trumpeter Swans 1998.) Second, there is management interest in understanding the effect that management at one location might have on the need for management at other locations to optimize use of public wildlife management funds. Third, during 2001, the United States Fish and Wildlife Service found itself embedded in litigation related to both endangered species and migratory bird hunting issues connected to trumpeter swans.

4.2.3 *Why Expert System Validation Was Not Attempted*

Although detailed, empirical, field evaluation of each expert system will be important if they are to be used as stand-alone systems, it was beyond the scope of this project to do so. My undertaking was more focused on developing the algorithms for applying multiagent systems and distributed problem solving methods. As such, the expert systems should be thought of as an input to the multiagent system.

Furthermore, one of the values of expert system technology is that it allows knowledge engineers to make existing expertise available to others. Often such an approach is most valuable in situations where it is not currently feasible to procedurally model causal relationships because (1.) empirical data from scientific experiments are lacking, or (2.) the complexity of the system of interest prevents mathematical descriptions of those causal relationships. It is my opinion that both of these conditions exist for Trumpeter Swans in the Northern Rocky Mountains. Once a detailed understanding of the breeding ecology of these birds is more complete, field experiments to validate this expert system can be designed.

4.3 **Methods**

The decision support system for trumpeter swan management was evaluated at three levels: (1.) verification of individual components, as well as the overall system, was directed at ensuring logical consistency; (2.) soft validation of the expert systems was accomplished through demonstrations to and informal trials by potential users; and (3.) the system was empirically tested using observed swan distribution data. The latter included analyzing the system for its sensitivity to various ecological parameters.

It was decided not to evaluate the system against a team with expertise in flyway management of swans, primarily because it was not feasible to assemble such a panel that was remotely independent of the people used in knowledge engineering. This was true for two related reasons. First, the total number of workers in the domain is small. Second, the cadre of such workers are closely interrelated institutionally and academically.

4.3.1 *Verification*

Logical consistency in each of the expert systems was confirmed as part of the iterative prototyping process (Eason 1988; D'Erchia et al. 2001), and this process was followed throughout the development of the decision support system. A key part of designing each of the individual expert systems was developing flowcharts of the ecological logic and then using those to consult with experts for changes and refinement. Similarly, the "planeditor" facility in DECAF allowed me to develop graphical representations of the logic underlying each individual agent. I used these to consult with specialists in multiagent system design prior to further development and implementation.

Utilities within the expert system development shell were used for verification of logical consistency of each expert system. The utilities automatically do a static check for problems such as incomplete rules and trees and can also dynamically check the system by stochastically simulating runs. For example, if more than one rule tried to set a value for a single-valued variable, an error would be detected by the shell. Similarly, if the consequent portion of a rule was inadvertently not provided, an error would be detected. All such errors were corrected. Each expert system was checked periodically and corrected during development, and the final systems were checked using 500,000 such simulated runs with no problems detected. When running the multiagent system, DECAF has a graphical window for each agent that provides information about how the agent is functioning, including error messages of failed communications among agents. This facility was used for fixing any problems that arose.

4.3.2 *Soft Validation of the Expert Systems*

Demonstrations of each expert system were made at every available opportunity to waterfowl managers, field biologists, migratory bird specialists, and researchers. This included individual meetings, workshops, and telephone consultations where individuals were requested to run actual scenarios and provide comments. All problems uncovered, or concerns and suggestions mentioned, were considered, and the system(s) changed accordingly. In addition, the expert systems have been available in stand-alone fashion on the World Wide Web for the duration of the project, both in prototype and final versions. Few comments were received via this method, but these were similarly used to improve the systems.

4.3.3 *Empirical Testing of the Multiagent System*

No public demonstrations of the entire, completed multiagent system have been done to date for soft validation purposes. Validation efforts have been concentrated on empirical testing of the system. The system, itself, begins by using an observed number of swans at each of 27 geographic areas for the breeding season of a particular year and then simulates the number at each of those areas for the four subsequent seasons, concluding with a simulated number for the breeding season (of the subsequent year). The system always simulates breeding swan numbers in one year increments. It was a comparison of the simulated number for the subsequent year versus the actual observed number for that same year that was the basis of my empirical testing. An actual observed number of swans was available only for the breeding season, and not the other seasons, so analysis was limited to data for that season. Simulated data for the intervening seasons was used only by the service mechanism to generate the concluding number. Observed data were available for 14 contiguous years, beginning

with 1987. Therefore, comparisons of simulated and observed data could be made for the 13 years, 1988-2000.

Although all 27 servers (geographic areas) were always used in the queuing system, seven areas never have had swans during those breeding seasons and were excluded from statistical analysis. In all such cases, the system did not simulate swans in those areas. This was an attempt to ensure that the consistent simulation of no swans where none were expected did not artificially inflate the accuracy and precision of the system during evaluation.

The decision support system uses swan numbers that were collected by the member agencies of the Pacific Flyway Council and informally reported by the United States Fish and Wildlife Service on an annual basis (e.g., Reed 2000). These data and reports are archived in files at Red Rock Lakes National Wildlife Refuge. Data from the unpublished written reports were entered into digital format under contract between the United States Geological Survey and the Environmental Statistics Group at Montana State University. These overall data are currently available via the World Wide Web at: http://swan.msu.montana.edu/cygnet/time_series_project.html.

4.3.3.1 *Data Analysis*

Multivariate Matched-Pairs Permutation Test (MVPTMP) statistical procedures (Mielke and Berry 2001) were used for all empirical analyses. Number of responses was 20, representing the number of areas; and number of blocks was 13, representing the number of years. A pair is represented by two, one-dimensional arrays of 20 responses each. The first of the pair is simulated data, the second is either observed data or simulated data from a run of the system with a different configuration; both arrays represent data corresponding to the same year. All such arrays (years) for a particular

run or the observed data form a 20 by 13 matrix termed a group in MVPTMP analysis. Because this is matched-pairs type procedure, the analysis is always comparing two groups. In such analyses, a small p-value is evidence of similarity between the two groups. In a few cases, groups of six areas rather than 20 were examined to test whether the broader relationships also held for a smaller group of areas of special management concern.

4.3.3.2 *Description of the Experimental Runs*

Null hypotheses were developed to assist with an assessment of the validity of the entire system as well as its components. Subsequently, corresponding runs of the system were made for validation purposes and are described in Table 4-1. I was interested in performance of the system from a flyway perspective over time, not the perspective of individual areas or individual years. Therefore, the multivariate approach provided by MVPTMP seemed particularly appropriate because it allows a statistical look at a problem that has concurrent spatial and temporal components.

Table 4-1. Description of experimental runs of the decision support system used for validation. These represent the adjustment of all major inputs to the system for sensitivity-type analysis.

Run label	Brief description of run	Specific system components that were experimentally altered					
		Number of expert systems	Number of refuges	Movement probability matrix	Breeding habitat quality threshold	Percentage retained in queue	Starting queue numbers
A	base queuing model	0		S			Q
B	default configuration	3	7	S	0.6	0.1	Q
C	default configuration, 1 refuge	3	1	S	0.6	0.1	Q
D	starting queue uses predicted numbers	3	7	S	0.6	0.1	Q'
E	default configuration, 1 expert system	1 [flyway]	7	S	0.6	0.1	Q
F	default configuration, 1 expert system	1 [breeding]	7	S	0.6	0.1	Q
G	default configuration, 1 expert system	1 [wetland]	7	S	0.6	0.1	Q
H	alternate breeding threshold	3	7	S	0.8	0.1	Q
I	alternate breeding threshold	3	7	S	0.4	0.1	Q
J	alternate queue percentage	3	7	S	0.6	0.75	Q
K	alternate queue percentage	3	7	S	0.6	0.5	Q
L	alternate queue percentage	3	7	S	0.6	0.25	Q
M	alternate movement probability matrix	3	7	S_a	0.6	0.1	Q
N	alternate queue percentage & expert system outputs forced to unacceptable	3	7	S	0.6	0.1	Q
P	alternate queue percentage & expert system outputs forced to unacceptable	3	7	S	0.6	0	Q

S: matrix of standard movement probabilities developed from expert opinion
S_a: matrix of alternate movement probabilities. See Section 4.3.3.3 for details.
Q: each year's starting queue uses observed numbers for breeding season of that year
Q': each year's starting queue uses numbers generated by the previous year's run of the queuing mode

4.3.3.3 *Sensitivity Testing*

Much of the sensitivity testing was straightforward. Configuration parameters were simply changed to more and lesser extreme values and output compared. Such parameters changed were: the number of expert systems; the number of refuge servers; the breeding habitat quality thresholds; and the percentage accepted into a queue when it is determined to be unacceptable by the system. All those parameters are set either in the system configuration file or by choosing which refuge agents to activate at run time.

Sensitivity testing for the movement probabilities was quite different. In that case, new movement probability matrices had to be generated to replace those developed during knowledge engineering sessions with waterfowl experts. I followed the now classic methods described by Shortliffe and Buchanan (1984) for sensitivity testing of their certainty factors used in their medical expert systems. They chose several different intervals into which to map their data, using the midpoint of the range as the new value. In my system, movement probabilities range from 0 to 1.0. In the simplest case of mapping to two intervals, 0.25 is the midpoint of the first interval and 0.75 of the second. Next, any value less than or equal to 0.5 in the original probability matrices was replaced with 0.25, and any value greater than 0.5 was replaced with 0.75. The system was then run using these new probability matrices. Using this procedure, I mapped data over 2, 4, 6 and 10 intervals and generated a set of four alternate matrices of movement probabilities for each interval. As the number of intervals decreases (i.e., the width of the interval increases), one would expect that system performance would deteriorate if the system was sensitive to the real movement probabilities.

4.4 **Results**

All verification and soft validation issues were handled as part of the software development process and no known issues remain. Here, I will focus on results from the empirical testing of the multiagent system. Thirty-four different experiments were conducted, representing the adjustment of all major inputs to the system. Table 4-2 provides the results from all MVPTMP analyses. I agree with Mielke (personal communication) that interpretation of p-values in determining whether to reject a null hypothesis should be left to the reader because it is a probabilistic and somewhat subjective process. So, the following are simply my personal examination of those values. I rejected all null hypotheses associated with a p-value less than 0.01, and failed to reject them when the p-value was greater than 0.1. Values between 0.01 and 0.1 were considered equivocal. The basic null hypothesis is that the two groups being tested are dissimilar and is rejected with an accompanying small *P*-value. In an MVPTMP analysis, such a small *P*-value results when the δ_o statistics representing the analysis of paired data matrices are randomly distributed and results from similarity between the two groups. Therefore, rejecting the null hypothesis is evidence of similarity between the two groups. However, failure to reject the null hypothesis is not evidence of dissimilarity.

Table 4-2 . Interpretation of statistical analyses of experimental runs of the decision support system using MVPTMP. Data are for the standard 20 areas unless otherwise noted. Null hypotheses were developed *a priori* to compare output from runs of the system as described in Table 4-1. In the table below, O = observed numbers from 1988-2000. The generic null hypothesis (H_o) is: the δ_o statistics representing the analysis of paired data matrices are randomly distributed. Small differences (similarities) in pairs of data result in a small *P*-value and a rejection of H_o. Interpretations that should result are provided when the decision is made to reject the null hypotheses. Whether to reject the null hypothesis in each instance is left to the discretion of the reader based on their individual interpretation of the p-value provided. No *P*-value is reported when output between the two groups was identical.

Test label	**Groups compared** [a.]	**p-value from MVPTMP**	**Interpretation from rejecting H_o**
1	A & O	.0001	output from base queuing model similar to observed numbers
2	B & A	.0002	output from base queuing model similar to default dss
3A	B & O	.0001	output using all expert systems (3) and activating all (7) refuge agents similar to observed numbers
3B	B & O	.0001	output using all expert systems and activating all refuge servers similar to observed numbers, analyzing output data for only 6 selected areas
4A	C [br] & O	.0001	output with only Bear River MBR as an active agent similar to observed numbers
4B	C [ca] & O	.0002	output with only Camas NWR as an active agent similar to observed numbers
4C	C [cv] & O	.0001	output with only Centennial Valley as an active agent similar to observed numbers
4D	C [gl] & O	.0001	output with only Grays Lake NWR as an active agent similar to observed numbers
4E	C [ip] & O	.0002	output with only Island Park as an active agent similar to observed numbers
4F	C [jh] & O	.0002	output with only Jackson Hole as an active agent similar to observed numbers
4G	C [um] &O	.0001	output with only Upper Madison River as an active agent similar to observed numbers
5A	C [br] & B	.0002	output with only Bear River MBR as an active agent similar to that when 7 refuge agents activated
5B	C [ca] & B	.03	output with only Camas NWR as an active agent similar to that when 7 refuge agents activated
5C	C [cv] & B	.0002	output with only Centennial Valley as an active agent similar to that when 7 refuge agents activated
5D	C [gl] & B	.0002	output with only Grays Lake NWR as an active agent similar to that when 7 refuge agents activated
5E	C [ip] & B	.11	output with only Island Park as an active agent similar to that when 7 refuge agents activated
5F	C [jh] & B	.05	output with only Jackson Hole as an active agent similar to that when 7 refuge agents activated
5G	C [um] & B	.0002	output with only Upper Madison River as an active agent similar to that when 7 refuge agents activated
6A	E & B	-	output using 3 expert systems identical to that with only the flyway expert system
6B	F & B	.0002	output using 3 expert systems similar to that with only the breeding expert system
6C	G & B	.0002	output using 3 expert systems similar to that with only the wetland expert system
7A	H & B	-	output using alternate breeding threshold of 0.4 identical to that using the standard, 0.6
7B	I & B	-	output using alternate breeding threshold of 0.8 identical to that using the standard, 0.6

8A	J & B	.0002	output using alternate percentage retained in unacceptable queues is set to 0.75 similar to default system
8B	K & B	.0002	output using alternate percentage retained in unacceptable queues is set to 0.5 is similar to default system
8C	L & B	.0002	output using alternate percentage retained in unacceptable queues is set to 0.25 similar to default system
9	D & B	.001	output using simulated swan numbers for each seasonal starting queue similar to default system
10A	M [10] & B	.0002	output when movement probabilities are mapped to 10 intervals similar to default system
10B	M [6] & B	.0002	output when movement probabilities are mapped to 6 intervals similar to default system
10C	M [4] & B	.0002	output when movement probabilities are mapped to 4 intervals similar to default system
10D	M [2] & B	.0002	output when movement probabilities are mapped to 2 intervals similar to default system
11	N & B	.0001	output when alternate percentage retained in unacceptable queues is set to 0 and expert system outputs forced to unacceptable is similar to default system
12A	P & B	.0001	output when alternate percentage retained in unacceptable queues is set to 0 and expert system outputs forced to unacceptable is similar to default system, analyzing output data for only 6 selected areas
12B	P & 0	.0008	output when alternate percentage retained in unacceptable queues is set to 0 and expert system outputs forced to unacceptable is similar to observed numbers

[a.] br = Bear River Migratory Bird Refuge; ca = Camas National Wildlife Refuge; cv = Centennial Valley and Red Rock Lakes National Wildlife Refuge; gl = Grays Lake National Wildlife Refuge; ip = Island Park and Harrimann State Park; jh = Jackson Hole and the National Elk Refuge; um = Upper Madison River

Although the results of the MVPTMP analyses provide a mathematical representation of a multivariate comparison, it is difficult to graphically depict comparisons of different groups of spatial data over time because of the inherent multidimensional structure. To partially handle this, I have chosen either to plot data as departures between groups, or to plot only select portions of the data. In Figures 4-1 and 4-2, the departure of the experimental data from the observed data in terms of number of swans has been plotted. Overall, the evidence is strong that the base model mimics the observed pattern of swan distribution over time (Table 4-2, Test 1; Figure 4-3) , as does the system run with the default configuration (Table 4-2, Test 3A; Figure 4-1). Almost all other experimental runs of the model show the same pattern. The scale of the y-axis in Figure 4-2 has been expanded to show the data for only six selected areas of special

management interest. It is noted that in spite of apparent disparity of some areas from observed numbers, e.g., Canada and Yellowstone Lake, there is no empirical difference from a flyway perspective as analyzed using MVPTMP (Table 4-2, Tests 3A-B). These two areas were ones for which the experts felt they had the least confidence in their information, and I was unable to secure additional information for these areas. Figure 4-3 depicts the differences between simulated numbers and the base queuing model, i.e., the decision support system run without the outputs from the expert systems. Figure 4-4, in turn, shows the similarity between the outputs from the base queuing model and the default system, i.e., run with the expert systems.

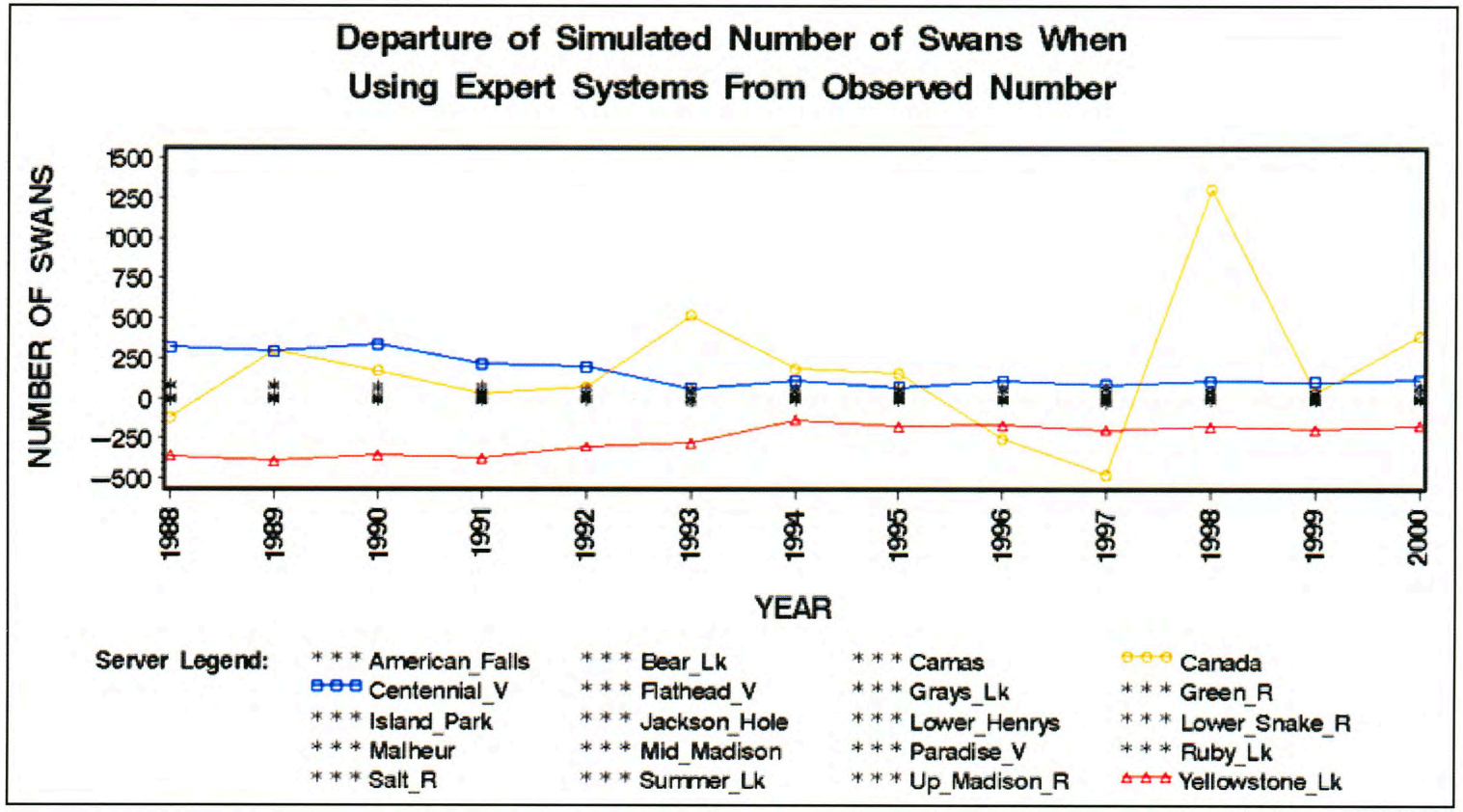

Figure 4-1. Plot of the difference in number of swans between the observed number and the simulated number when using expert systems (default system: Run B from Table 4-1). Simulated numbers are from the complete system run with all refuge servers activated. Those severs with most extreme differences are shown by lines for emphasis. During this time, the total population varied between 1115 and 3471 birds.

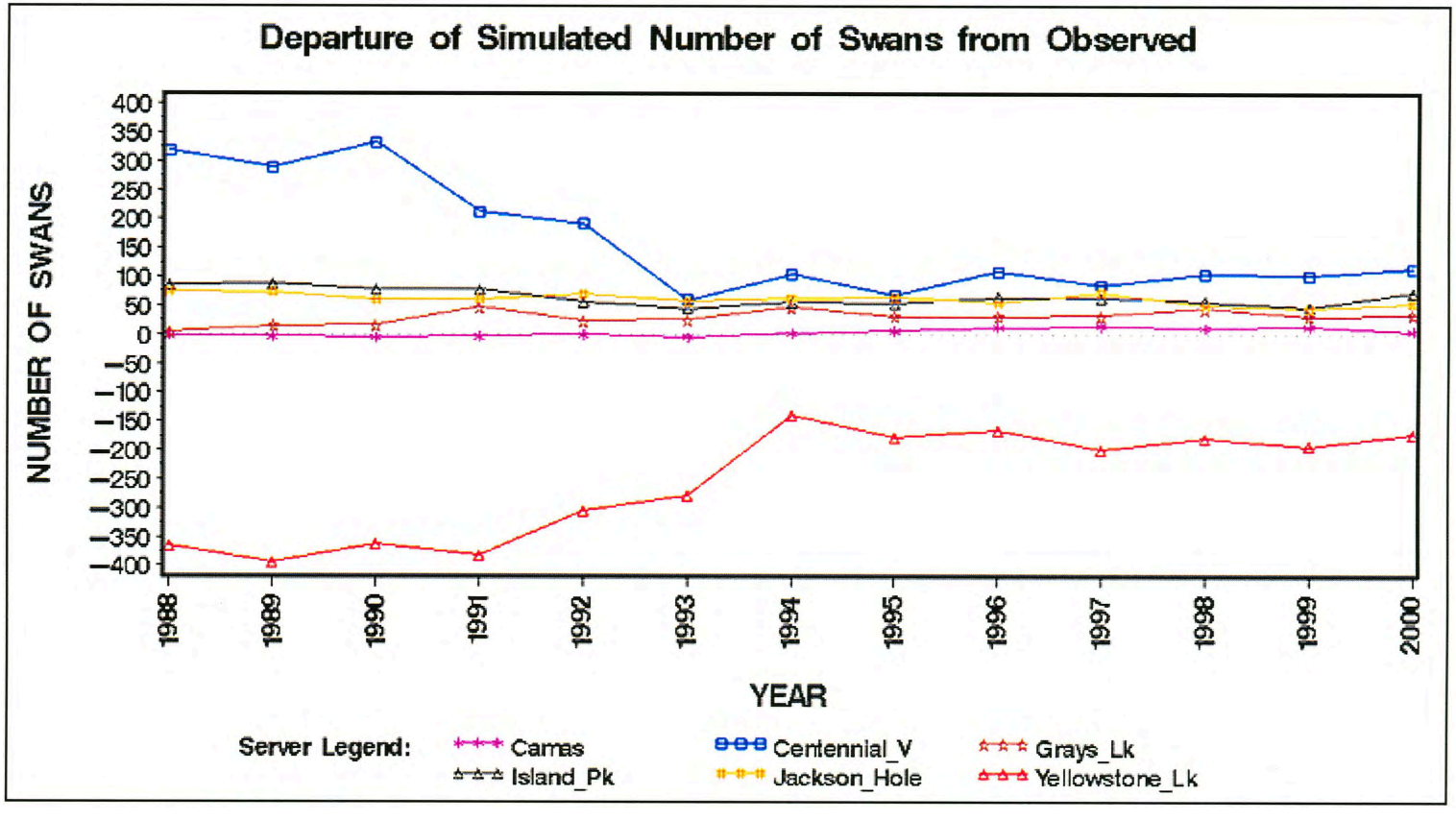

Figure 4-2. Plot of the difference in number of swans between the observed number and the simulated number for six selected servers. Simulated numbers are from the complete system run with all refuge servers activated and input from all expert systems (Run B from Table 4-1). During this time, the total population varied between 1115 and 3471 birds.

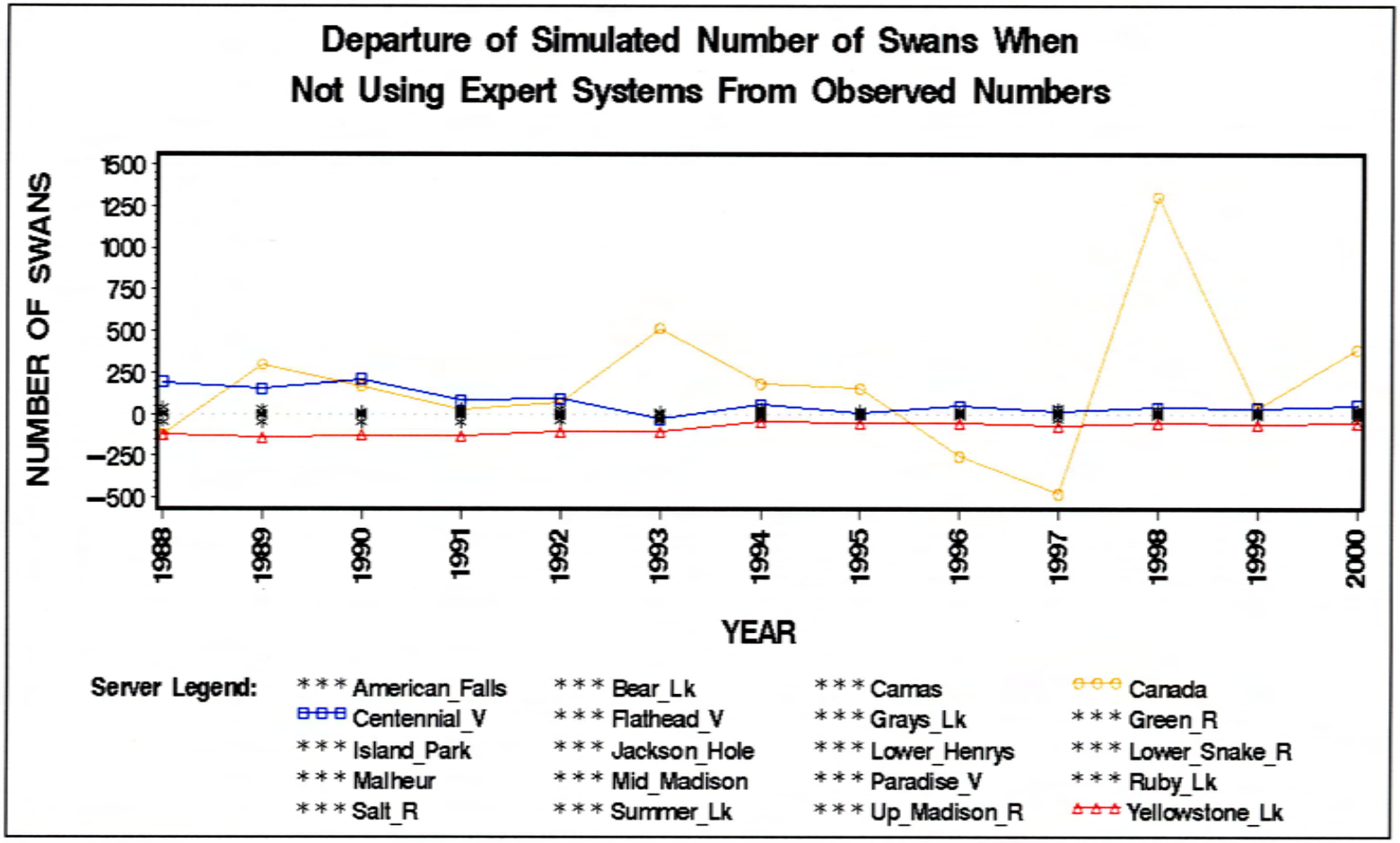

Figure 4-3. Plot of the difference in number of swans between the observed number and the simulated number when not using expert systems (base queuing model: Run A from Table 4-1). Simulated numbers are from the complete system run with all refuge servers activated. Those severs with most extreme differences are shown by lines for emphasis. During this time, the total population varied between 1115 and 3471 birds.

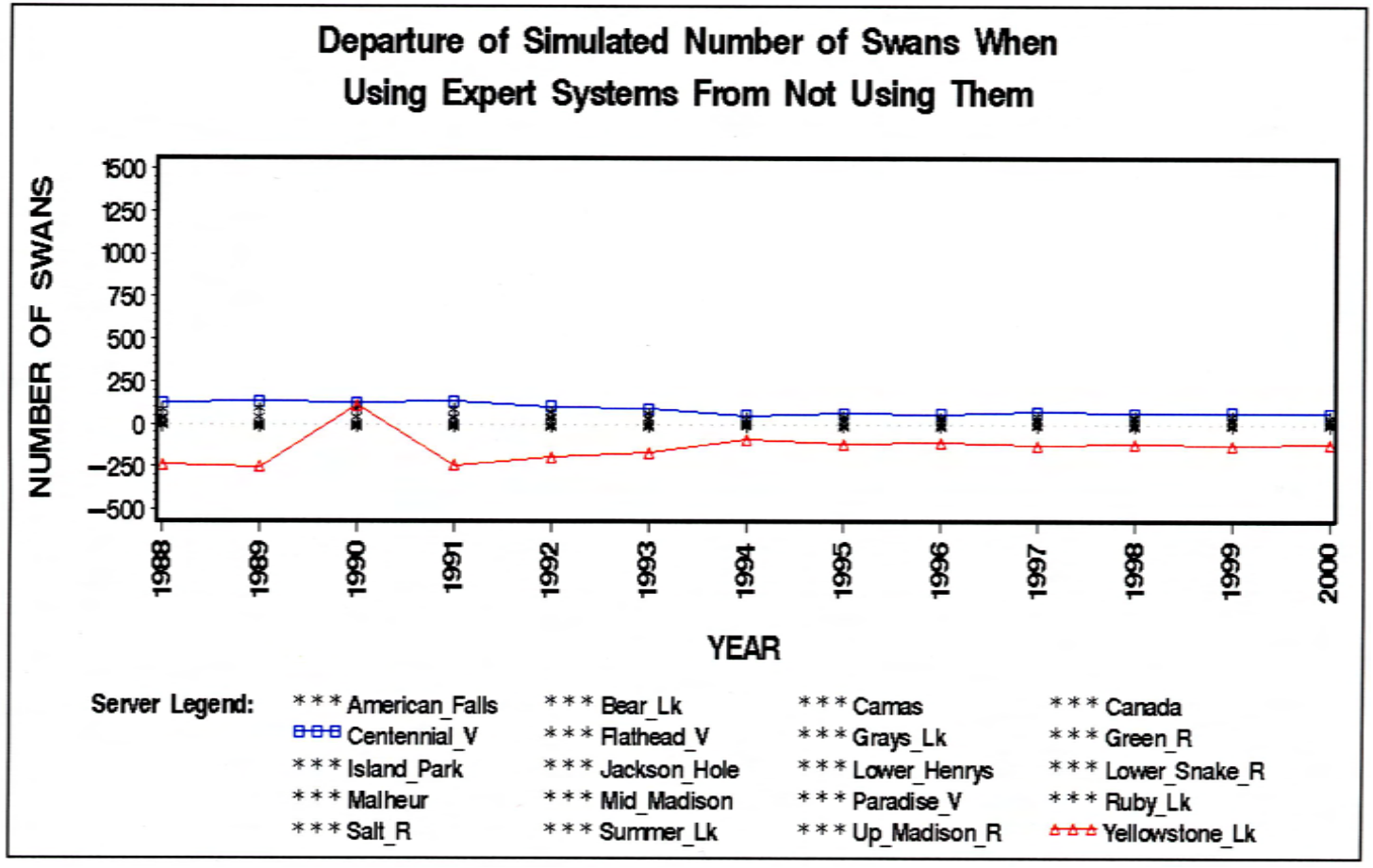

Figure 4-4. Plot of the difference in the simulated number of swans between the base queuing model (not using expert systems: Run A from Table 4-1) and the default system (Run B from Table 4-1). Both data sets are from the system run with all refuge agents activated. Those severs with most extreme differences are shown by lines for emphasis. The y-axis retains the scale of that from Figures 4-1 and 4-3 for comparison. During this time, the total population varied between 1115 and 3471 birds.

When the system is run with only one refuge agent active, the distribution of swans over time compared to the observed numbers did not show significant disparities (Table 4-2, Tests 4A-G). Figures 4-5 thru 4-7 provide examples of typical distributions of swans for selected years for such system runs. The results from years 1988, 1994, and 2000 were arbitrarily selected to cover the range of years in the tests. When these runs of the system were compared to running the system with all agents active (rather than comparing against observed numbers), the evidence for the system not being sensitive to any one particular refuge agent was not as consistently strong (Table 4-2, Tests 5A-G). However, not rejecting the null hypothesis of similarity for Island Park runs, for example, does not prove that dissimilarity exists in that case, only that the evidence is weak for proving similarity. The Island Park relationship has been plotted in Figure 4-8.

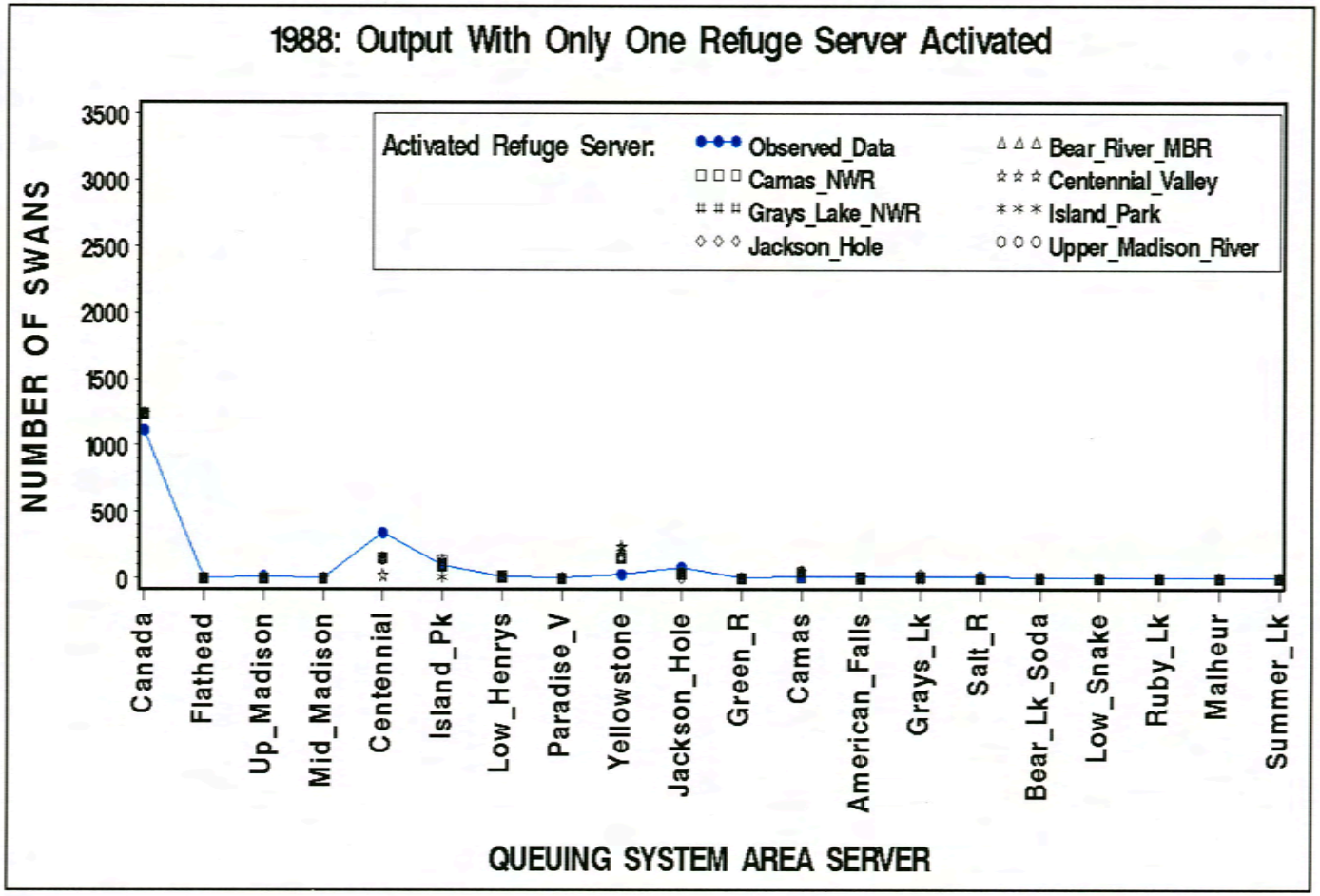

Figure 4-5. Plot for 1988 that shows both the observed number of swans and the output from seven individual runs of the system, each time with only one particular refuge server (agent) having been activated (Run C from Table 4-1). The appearance of few data points results from the combination of nearly identical data from different runs being plotted on top one another and the required scale to plot data from the server, "Canada". The total population in 1988 was 1115 birds.

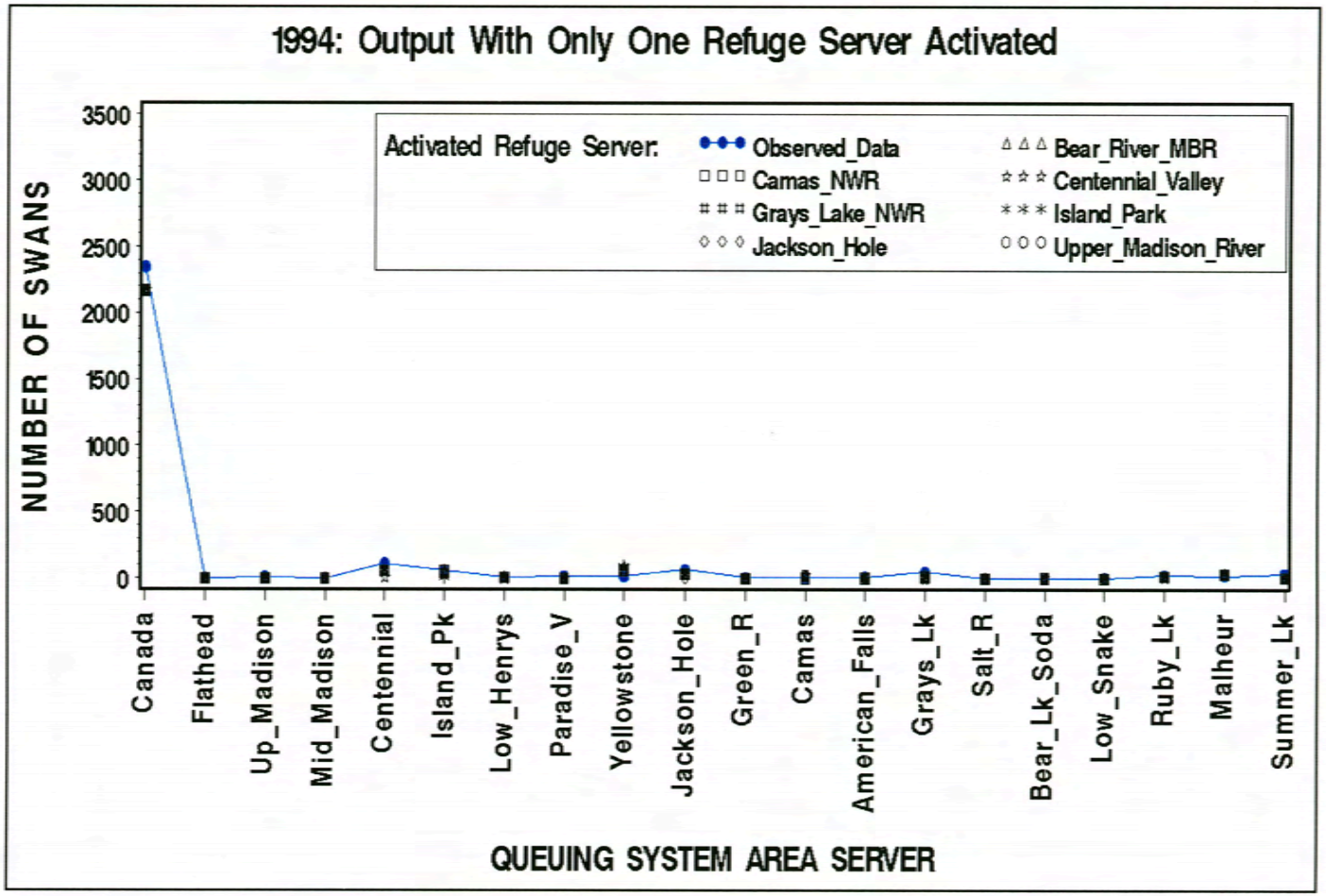

Figure 4-6. Plot for 1994 that shows both the observed number of swans and the output from seven individual runs of the system, each time with only one particular refuge server (agent) having been activated (Run C from Table 4-1). The appearance of few data points results from the combination of nearly identical data from different runs being plotted on top one another and the required scale to plot data from the server, "Canada". **The total population in 1994 was 2349 birds.**

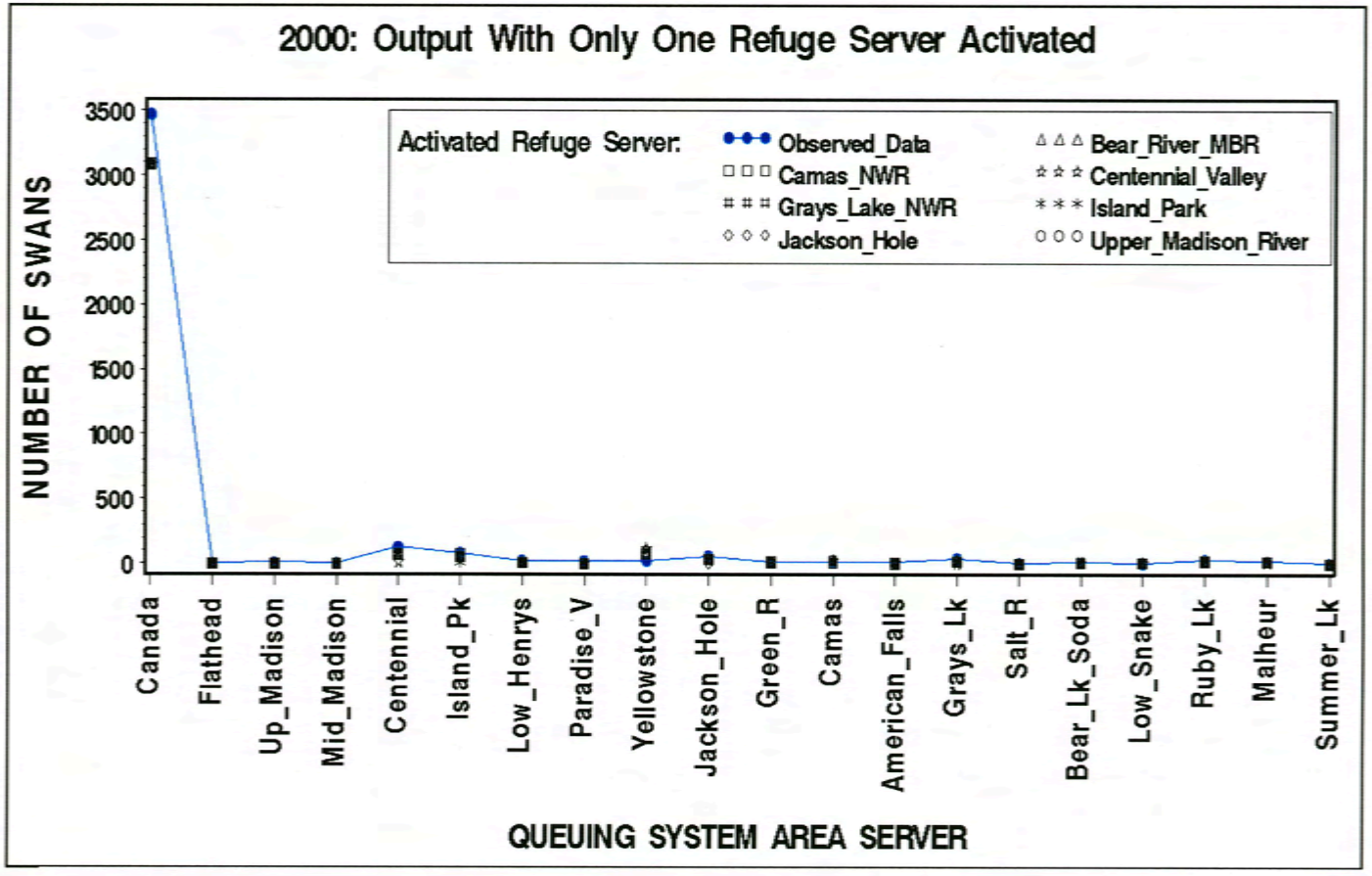

Figure 4-7. Plot for 2000 that shows both the observed number of swans and the output from seven individual runs of the system, each time with only one particular refuge server (agent) having been activated (Run C from Table 4-1). The appearance of few data points results from the combination of nearly identical data from different runs being plotted on top one another and the required scale to plot data from the server, "Canada". **The total population in 2000 was 3471 birds.**

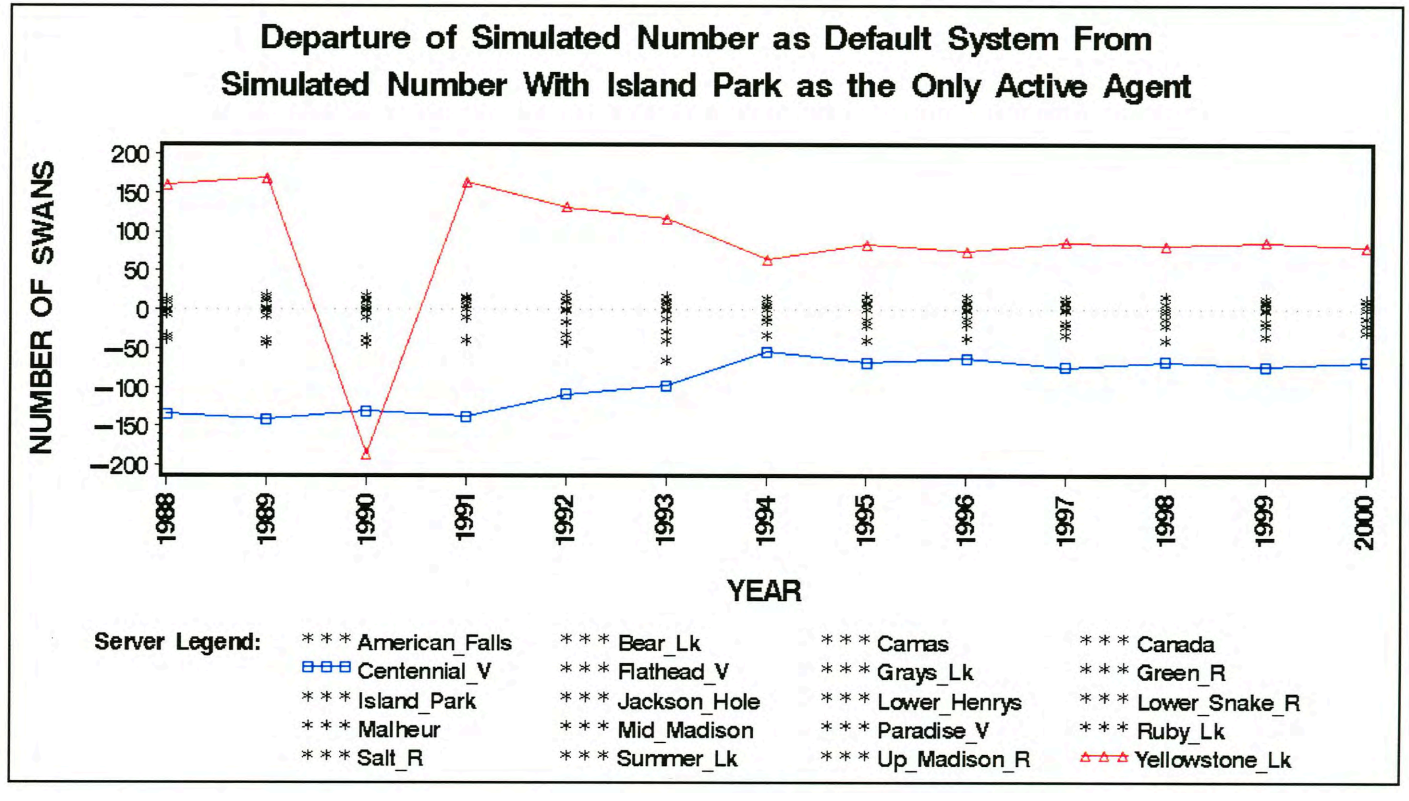

Figure 4-8. Plot of the difference in the simulated number of swans between running the system with only the Island Park agent active (Run C from Table 4-1) and the default system (Run B from Table 4-1). Both data sets are from the system run with corresponding expert systems activated. Those severs with most extreme differences are shown by lines for emphasis. During this time, the total population varied between 1115 and 3471 birds.

The refuge agents manage, listen for, and access expert system output in combination with the move agent to implement the server mechanism of the queuing system. This implementation was an attempt to infuse ecological information along with a flyway perspective especially as represented in the overall goals of the1998 Management Plan for the Rocky Mountain Population of Trumpeter Swans (Subcommittee on Rocky Mountain Trumpeter Swans 1998). The expert system information had no effect on the simulated distribution of swans (Table 4-2, Tests 6A-C). This was true if all three expert systems had been activated in combination, or if they had been activated individually. Changing values for the breeding habitat quality threshold or for the alternate percentage retained in unacceptable queues was another way to alter the service mechanism. However, such changes did not affect the simulated distribution of swans when compared to the system run with the default configuration (Table 4-2, Tests 7-8).

Because all aspects of the system seemed so insensitive to the expert systems, I conducted some runs where the expert system output was forced to only values that would make all refuge servers unacceptable for all seasons and all years (Table 4-1, run N). In an attempt to create an artificial run with even more extreme conditions, I also did not allow these now unacceptable servers to accept any swans; all had to be redistributed (Table 4-1, run P). The simulated numbers from these runs also showed similarity to both the observed numbers and default system number (Table 4-2, Tests 11-12).

The matrices of probabilities representing movement from season to season are the foundation of the service mechanism, the algorithm for which is principally organized within the move agent. And, that is why I conducted the experiments to alter those probabilities. The statistical analysis provided strong evidence that there was similarity in simulated swan distributions using the artificially mapped probabilities when compared

to output from the system run with the probabilities provided by the experts (Table 4-2, Tests 10A-D). This was even true when the movement probabilities were mapped to only two intervals. Even though the service mechanism is based on the movement probabilities, the system was not sensitive to the actual values provided by the experts. Figures 7 and 8 depict the close agreement in simulated swan numbers as the movement probabilities are altered. Although not shown here, the plots of swan distributions corresponding to the mapping of four and six intervals were similar to those shown for two and ten intervals. It is suspected that the deviation for the Yellowstone Lake numbers represents the lack of confidence in those original associated probabilities as provided by the experts and potentially carried through in the mapping. It is not known why that deviation apparently does not exist for 1990. I speculate that the probabilities provided by the experts may actually apply to 1990, but not other years.

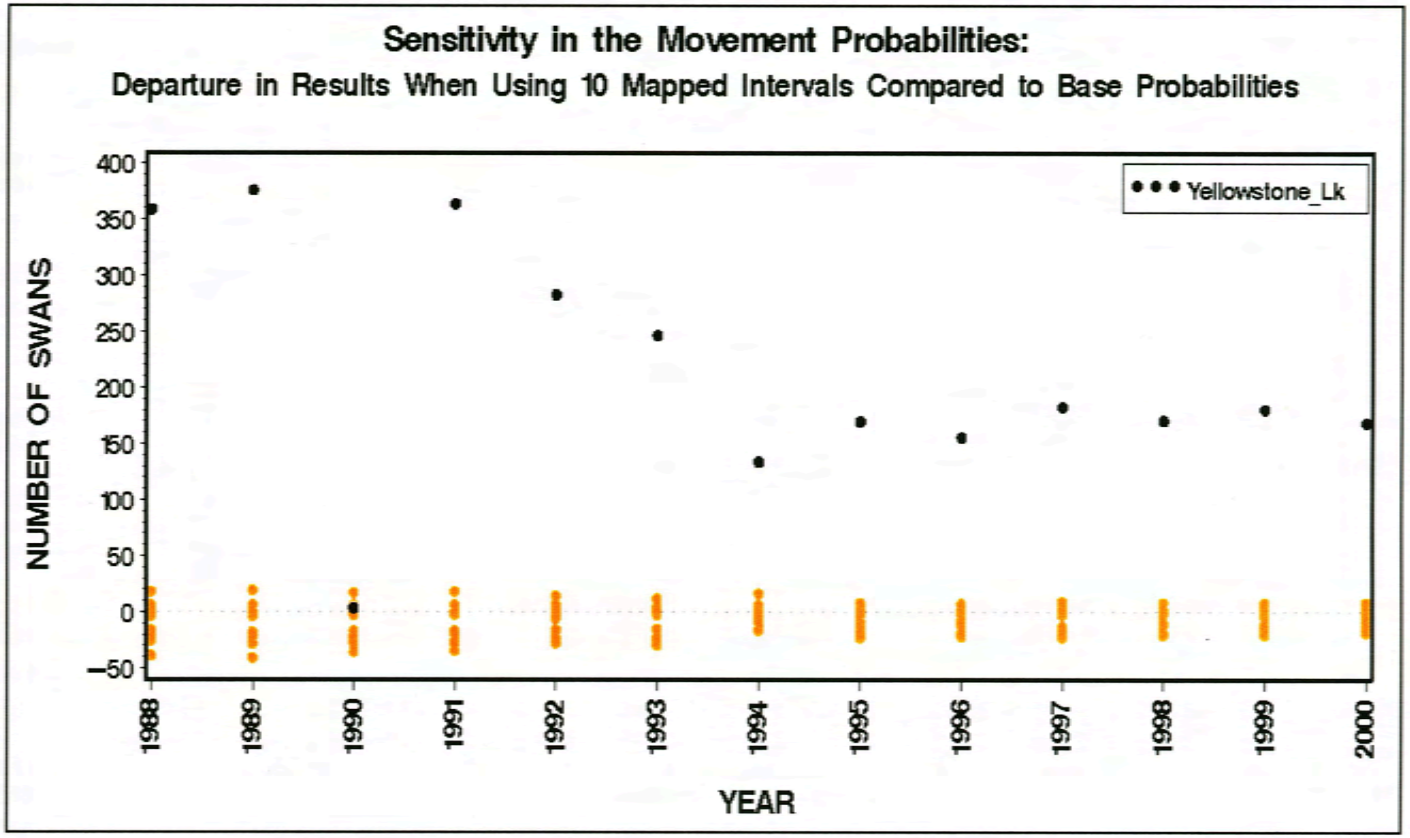

Figure 4-9. Plot of the difference in number of swans between the observed number and the simulated number when the base movement probabilities have been mapped to 10 intervals. Simulated numbers are from the complete system run with all refuge servers activated and input from all expert systems (Run M from-Table 4-1). During this time, the total population varied between 1115 and 3471 birds.

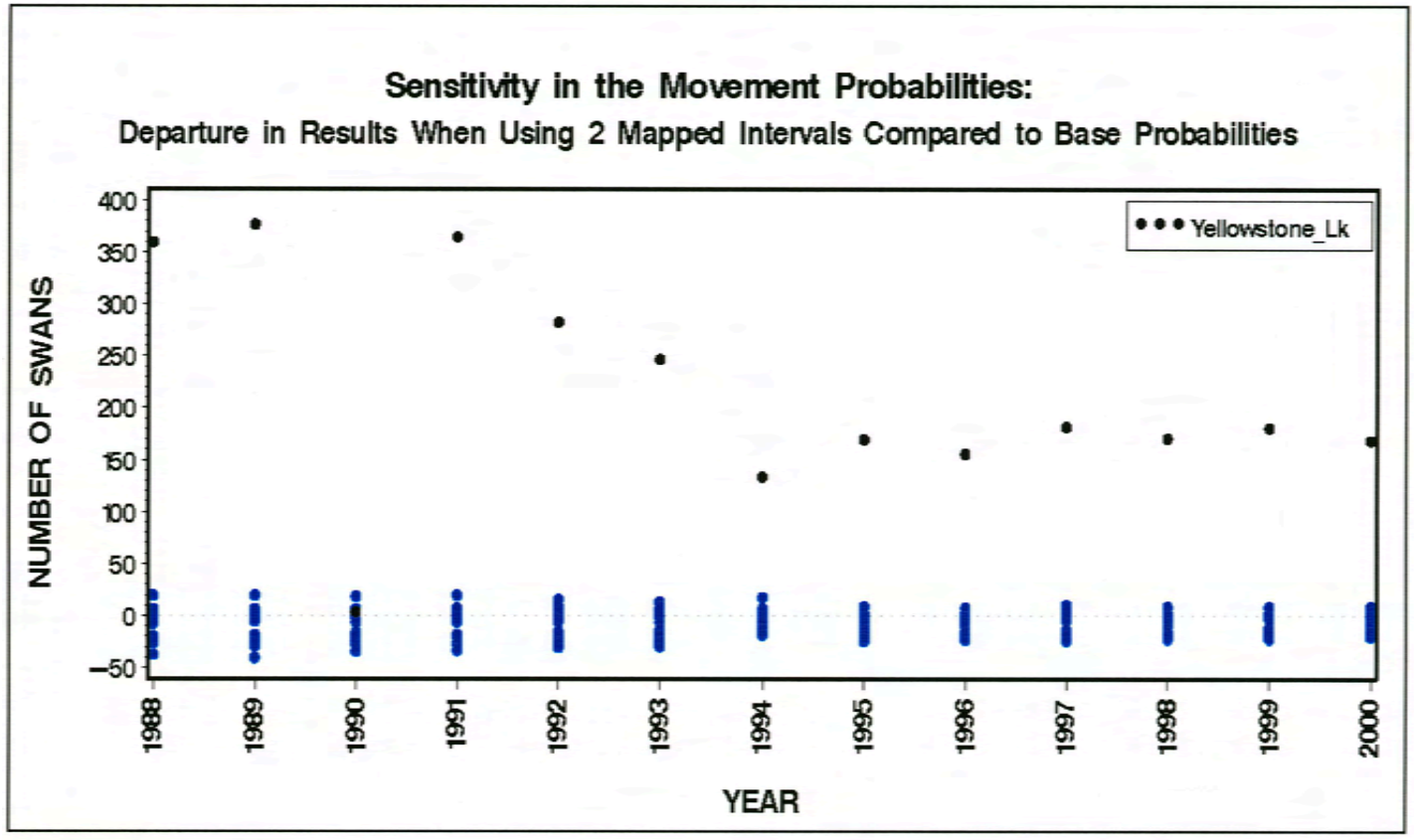

Figure 4-10. Plot of the difference in number of swans between the observed number and the simulated number when the base movement probabilities have been mapped to 10 intervals. Simulated numbers are from the complete system run with all refuge servers activated and input from all expert systems (Run M from-Table 4-1). During this time, the total population varied between 1115 and 3471 birds.

4.5 **Discussion**

The totality of the decision support system rests on a foundation of intelligent agents being used to implement and employ a queuing system to simulate the distribution of swans. Outputs from three expert systems are used to provide inputs to the base queuing model. There could have been other ways to provide such input. I chose expert systems because they allowed a unique way to provide ecological expertise to the system, yet allowed on-site managers and biologists to provide their unique interpretation of their own situations. The expert systems act as a type of filter that accepts knowledge about on-site conditions, but then makes recommendations using that information about the best approach for providing habitat for trumpeter swans, always from a flyway perspective. The idea behind encoding the key ecological knowledge is that the system can be run as a model that is simulating the best management possible for seven key geographic areas. If one accepts the notion that the system, as it is run in its default configuration, represents close to ideal ecological conditions, then one can assess the similarity between observed distributions of swans with those simulated by the system. If these distributions are similar, two conclusions are plausible. One, the actual conditions in the flyway from 1987-2000 represented ideal management. Or two, the concepts represented by the 1998 Management Plan for the Rocky Mountain Population of Trumpeter Swans (Subcommittee on Rocky Mountain Trumpeter Swans 1998) would not have changed the distribution of swans significantly from what had been observed. I easily assume that the experts would not have developed the 1998 Plan if they had not felt that improving the management of trumpeter swans was possible and therefore, albeit subjectively, place little credence on the first conclusion. I then conclude that the 1998 Plan could not have affected the distribution of swans even if implemented completely. It is important to recognize that my conclusion

has been drawn in retrospect, and does not imply criticism of the managers in developing the 1998 Plan.

Any and all inferences drawn from using the decision support system hinge on trusting the base queuing model that underlies it. The empirical evidence that I gathered on its performance leads me to conclude that the base queuing model does accurately simulate swan distributions in the flyway. Although I did not empirically test them, I also feel that the MVPTMP procedures of Mielke and Berry (2001) were an effective approach for grasping a multidimensional view of the effect of management actions on the flyway distributions of waterfowl over time.

It is puzzling why the multiagent system seemed so insensitive to all parameters tested. It is possible that the system was simply never taken to thresholds where the various parameters altered would result in system performance degradation. However, it is difficult for me conceptualize additional experiments of this nature beyond those employed without requiring major restructuring of the move agent, and to a smaller degree, the facilitator agent. One explanation that I proffer (but have not tested) is that the base queuing model is particularly effective at representing the actual ecological diversity in the world of Rocky Mountain trumpeter swans, both spatial and temporally. The queuing system has been based on substantial expertise gathered using formal knowledge engineering procedures. It seems that if the model has effectively encoded and utilized that expertise, embedded within that knowledge are many aspects related to the importance of diversity in the flyway ecology of waterfowl. Certainly ecological diversity is not a new principle. I do infer that my decision support system provides evidence that management actions at one, two, or even seven important trumpeter swan habitats cannot have a substantial effect on the temporal and spatial distribution of swans in the flyway, at least within the simulations of my system. It would be interesting

to restructure the decision support system to allow for greater than seven active refuge agents. This would allow further empirical exploration about whether management thresholds might exist when the total number of sites is greater than seven. There have been several attempts by waterfowl managers to move swans, and no major changes in swan distribution have been reported. However, I am aware of no experiments that were designed to evaluate those attempts. Such anecdotes might be interpreted as weak evidence corroborating my analytical findings, however.

I also note that my system is possibly limited by the actual knowledge of the experts, the knowledge engineering efforts, and the temporal and geographic scope of the underlying data and knowledge. It is also not known if dependencies might result from having developed the movement probability knowledge from a subset of experts who collected and assembled the actual observed data. However, there was no other way to have collected either set of data or knowledge, nor does another independent flyway exist in which to test the system to statistically isolate any potential dependencies.

Twenty-seven areas sharing birds amongst each other, even if within somewhat structured migration parameters, across four seasonal changes, becomes a massively interconnected, multi-dimensional network. I obviously was able to represent this network algorithmically, but did not attempt to represent it mathematically. I do wonder whether a deeper understanding of the algebraic relationships, and any emergent behaviors from the complexity of the network, might not further elucidate a small aspect of ecological diversity as related to flyway management and migration of waterfowl. Alternatively, such a theoretical approach might elucidate thresholds where the service mechanism that I implemented begins to break down.

4.6 **Conclusions**

Validation is the process of determining whether the stated purpose of the system was achieved. I conclude that multiagent systems can be an effective platform for modelling the movement of waterfowl in a flyway perspective. Although the technology allowed me to integrate site-specific and flyway perspectives into the model, local management actions might not actually affect flyway swan distributions. The algorithmic complexity of the system may possibly be mimicking natural diversity of the flyway, its swan habitats, and their management. Because models are abstractions of reality, it is inherent that they will have shortcomings from not being able to accurately represent all knowledge, logical relationships, and probabilistic intricacies.

4.6.1 ***In Terms of Waterfowl Ecology and Management...***

- Ecological knowledge has been encoded into the overall system in many places. Doing so allowed the queuing system to effectively simulate the distribution of swans, mimicking observed numbers well.
- The 1998 Management Plan for the Rocky Mountain Population of Trumpeter Swans, as written, could not have affected the flyway distribution of swans.
- The distribution of swans in the flyway could be effectively simulated, and this is a strategic precursor when needing to predict the reestablishment of traditional migratory pathways. Documenting the development of such pathways would be critical; and applying MVPTMP as a multivariate tool was a new approach for comparing flyway distributions of waterfowl.
- Although the system could simulate swan distributions, the types of management actions included in the system seemed to have no effect on those distributions. This is possibly explained by the natural diversity of the

system being modeled. Alternatively, it could result from an ineffective implementation stemming from a lack of knowledge about the effects of management actions.

4.6.2 ***In Terms of Multiagent Systems...***

- DECAF worked well as a multiagent platform for an ecological queuing system.
- Agents communicated effectively to provide simulated distributions of trumpeter swans, and accessed ecological knowledge successfully.
- Although my agents did not encompass all the ecological and mathematical knowledge necessary to intricately model flyway management of swans, they did seem capable of cooperating with each other to reason in multiple spatial and temporal scales.
- The multiagent framework worked well to integrate output (knowledge) from individual expert systems, ensured that that output remained attributable to individual users, and was successful at communicating to the user when additional expert system knowledge was necessary.

4.6.3 ***Future Directions***

There are many potential future directions that would be fruitful for research in the use of multiagent systems for flyway management of waterfowl. They are as varied as the questions that managers face. I believe that multiagent systems are ideal platforms for modelling waterfowl movements that integrate a flyway and local perspective. It is a domain that is inherently represented as a distributed problem(s), and the technology evolved from algorithms that provided the capability for complex and distributed

problems. The work I have conducted leads toward the need for determining what other factors might allow a deeper understanding of the effects of management actions on the flyway distribution of waterfowl. Knowing those would allow the more refined development of algorithms for effective decision support systems via collaboration by intelligent agents. This would result from more robust knowledge in their belief systems, more detailed relationships in updating their intentions, and deeper communication strategies for addressing distributed problems. A companion thrust could be to add probabilistic sophistication to the service mechanism by representing the uncertainty in the movement probabilities and propagating that uncertainty into the simulated distributions.

CHAPTER 5

ARTIFICIAL SWANS, ARTIFICIAL MARSHES, AND ARTIFICIAL INTELLIGENCE: SUMMARY, CONCLUSIONS, AND REFLECTIONS

5.1 What Was Accomplished

When starting this research, doing work that merged the field of waterfowl and wetland management with the field of artificial intelligence intrigued me. My goal was to develop and empirically test a tool that would help refuge managers be able to think about their local marshes from a flyway perspective. I already knew that managing natural resources is difficult. I did not expect to make it easy nor provide a cookbook. I did hope that the tools I would provide would allow people, myself included, to conceptualize problems that were both small-scale and site-specific simultaneously. These were problems that intertwine time as a critical component and include yesterday's events and choices as constraints to what we should do now. Such problems often have more intricacies than one person can perceive, visualize, and solve. There seemed to be a possibility of applying new artificial intelligence methodologies to such problems. The ability to encode the knowledge and problem solving abilities of experts so that it might be more widely utilized seemed useful. And, I recognized that logically connecting this knowledge and metaknowledge in multiple dimensions was the venue of artificial intelligence. However, these were methodologies based on theories and technologies more rapidly developing than I had certainly envisioned. What started out as cutting-edge in 1991 was nearly obsolete by 2001; and the project evolved in many technological ways. But, the goal remained the same. What I hopefully have

contributed emanates from an innovative, interdisciplinary, conceptual synthesis, and subsequently from empirical evaluation of the resulting methodology.

System and methodology development appeared at least partly successful. For the first time, digital swans could be managed in digital marshes. Expert systems are available on the World Wide Web that could provide assessments of trumpeter swan breeding habitat, make recommendations about managing semipermanent wetlands, and gauge the value of an area to the flyway management of trumpeter swans. Plus, queuing theory had been applied to modelling waterfowl migration for the first time, albeit a digital migration. That migration was simulated over time by creating intelligent agents linked together by an artificial intelligence based framework, DECAF, that had the capability to deal with such distributed problems (but had not yet been applied to many real situations). These agents were autonomous, had some sensory capability, and could respond to changing conditions. The agents effectively used KQML protocols to send and receive messages among each other.

Although system development had charted brand new interdisciplinary territory, the ultimate quest was to know whether it worked. Empirically testing multiagent systems, much like ecological models, is never straightforward because they are inherently complex, can exhibit emergent behaviors, and often have a heuristic foundation due to incomplete knowledge. Because trumpeter swans have been of great management interest, especially by the U.S. Fish and Wildlife Service, there is reasonably good, long term data about their numbers throughout the Northern Rocky Mountains. This provided a unique opportunity to empirically test the decision support system with experimental runs against a backdrop of observed numbers across 20 areas for the period 1988-2000. It was encouraging to see the base queuing model perform so well as judged by matched-pairs multivariate permutation methods. It was more puzzling to be unable to

adequately explain why the system seemed so insensitive to various parameters and configuration changes. Nonetheless, analysis of the various experimental simulations provide evidence that local management at single wetlands or small groups of them, even when they comprise critical habitats locally, has little effect on the flyway distribution of trumpeter swans over time. I suspect that the algorithms underlying the multiagent system, particularly those composing the service mechanism, perform reasonably well in representing the natural ecological diversity of swans and their use of habitats in the flyway. This likely stems from these algorithms and knowledge having been based on both effective knowledge engineering and the cooperation of true experts. Empirical testing of simulations provides some evidence that the 1998 Flyway Management Plan for the Rocky Mountain Population of Trumpeter Swans might only have had a small chance to affect the distribution of swans. One might also interpret this to mean that the Plan effectively avoided distributions that would have resulted in population collapse. My work did not involve population modelling or estimation, and that would be a necessary component to assess these possibilities.

5.2 What the Future Offers

Foremost, I emphasize that the decision support system for trumpeter swan management is a model, which by definition, is an abstraction of reality. But models can be important because they allow us to perturb systems, run experimental simulations to test management scenarios, and explore knowledge bases. However, all models can only be based on information that exists at the time of development. Therein lies the future.

5.2.1 *The Flyway Distribution of Waterfowl Via Multiagent Systems*

The system could be enhanced to predict the number of swans at specific sites. More accurate information on movement probabilities would be necessary, as would measures of precision. This information needs to be gathered through experimental field studies on waterfowl migration and movements. But, that is only part of the picture. Research to quantify the expected consequences of water levels on wetlands, swans, and wetland use by swans (Table 3-2) would allow probabilistic modelling in spatial and temporal dimensions. Understanding the associated uncertainties would allow further melding of artificial intelligence methodologies, using decision trees and Bayesian belief networks, into models of waterfowl management with both flyway and local perspectives. The end result would be the kind of modelling that melds science and management (Williams 1997; Clark and Schmitz 2001), and that might integrate qualitative and quantitative ecological information (Rykiel 1989; Starfield 1989). Such work must continue to build better estimates of fundamental ecological relationships as described by Johnson (1999). As we improve our models of waterfowl distribution and movement, we will likewise continue to improve our ability to evaluate management actions.

Even without additional ecological knowledge, my current system could be improved in at least four ways. First, the service mechanism in the queuing system could be made into a probabilistic mechanism using the heretofore unused estimates of precision provided by the experts regarding the movement probabilities. This would potentially allow confidence intervals (albeit based on heuristics) to be placed on the simulated distributions, or alternatively allow probability distributions to be developed and used. Second, ways of presenting the multivariate results of distributions changing over time (trend analysis) to the user would be advantageous in explaining the results. This is no trivial task, but the modular nature of DECAF should allow additional agents to

communicate to accomplish this. Third, the system could more closely fit accepted definitions of decision support (Adelman 1992; D'Erchia et al. 2001; Sprague and Carlson 1982). This would include increased emphasis on user interactions, particularly in the areas of problem definition and scenario development, as well as in the presentation of results and their concurrent statistical analysis. Along with this, online use of the system is a needed feature, including multiple user collaboration. Finally, in a more futuristic development, agents could be built to gather information from ecological sensors to judge habitat conditions, check on the status of the migration of radioed swans, and communicate with biologists in the field about current conditions. The question to be asked with any of these four improvements is whether they would cause the simulated distribution of swans to be more accurate or precise, or have some other demonstrable benefit to swan management. New system developments could likely be accompanied by a newly articulated purpose(s). One possibility would be to propose and test new management scenarios.

A working decision support system has been built and empirically evaluated. I encourage the continued modelling of feathered swans with digital ones and support the additional blurring of the distinction to further the use of adaptive resource management. Now, it is time to explore other ways to apply artificial intelligence in the multidimensional modelling of waterfowl migration as decision support for wildlife managers. It is also time to put on the hip boots for additional ecological field work.

CHAPTER 6

LITERATURE CITED

Adelman, L. 1991. Experiments, quasi-experiments, and case studies: a review of empirical methods for evaluating decision support systems. IEEE Transactions on Systems, Man, and Cybernetics 21(2):293-301.

Adelman, L. 1992. Evaluating decision support and expert systems. John Wiley and Sons. New York, New York.

Andriole, S. J. 1989. Handbook of decision support systems. TAB Professional and Reference Books. Blue Ridge Summit, Pennsylvania. 248 pages

Armstrong, A., and E. Durfee. 1997. Dynamic prioritization of complex agents in distributed constraint satisfaction problems. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence 15:620-625.

Bahill, T. A. 1991. Verifying and validating personal computer-based expert systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey. 205 pages.

Batabyal, A. A. 1996. The queuing theoretic approach to groundwater management. Ecological Modelling 85:219-227.

Blankenhorn, W. U. and T. Caraco. 1992. Social subordinance and a resource queue. The American Naturalist 139:442-449.

Boland, R. J., A. K. Mahewshwari, D. Te'eni, D. G. Schwartz, and R. V. Tenkasi. 1992. Sharing perspectives in distributed decision making. pages 306-313, *in*: Proceedings of the Conference on Computer-Supported Cooperative Work. Association for Computing Machinery. New York, New York.

Brehmer, B. 1991. Distributed decision making: some notes on the literature. pages 3-14 *in:* Rasmussen, J., B. Brehmer, and J. Leplat, eds. Distributed decision making: cognitive models for cooperative work. John Wiley and Sons. Chichester, England.

Burd, M. 1996. Server system and queuing models of leaf harvesting by leaf-cutting ants. The American Naturalist 148:613-629.

Carter, G. M., M. P. Murray, R. G. Walker, and W. E. Walker. 1992. Building organizational decision support systems. Economic Press Inc. San Diego, California. 358 pages.

Carver, N., Z. Cvetanovic, and V. Lesser. 1991. Sophisticated cooperation in FA/C distributed problem solving systems. pages 191-198 *in*; Proceedings of the Ninth National Conference on Artificial Intelligence. AAAI Press. Menlo Park, California.

Carver, N., and Lesser, V. 1992. The evolution of blackboard control architectures. PSCI Technical Report 92-71. Department of Computer Science, University of Massachusetts. Amherst, Massachusetts.

Chang, C. 1997. Using computer simulation to manage crowding in parks: a study. Landscape and Urban Planning 37(3-4):147-161.

Clark, W. R., R. A. Schmitz. When modelers and field biologists interact: progress in resource science. pages 197-208 *in*: Shenk, T. M., and A. B. Franklin, eds. Modelling in Natural Resource Management: Development, Interpretation, and Application. Island Press. Washington, DC.

Cohen, Y. 1984. A simplified approach to the queuing model of white-tailed deer harvest. Journal of Wildlife Management 48(1):271-275.

Cohen, P. R., and A. E. Howe. 1989. Toward AI research methodology: three case studies in evaluation. IEEE Transactions on Systems, Man, and Cybernetics 19(3):634-646.

Corkill, D. D. 1991. Blackboard systems. AI Expert 6(9):40-47.

Davis, J. R. and J. L. Clark. 1989. A selective bibliography of expert systems in natural resource management. AI Applications 30:1-17.

Decker, K., A. Pannu, K. Sycara, and M. Williamson. 1997. Designing behaviors for information agents. Pages 404-413 *in*: Proceedings of the First International Conference on Autonomous Agents. Marina del Rey, California

D'Erchia, F., C. Korschgen, M. Nyquist, R. Root, R. Sojda, P. Stine. 2001. A framework for ecological decision support systems: building the right systems and building the systems right. U.S. Geological Survey, Biological Resources Division, Information and Technology Report USGS/BRD/ITR-2001-0002. 50 pages.

Dshalalow, J. H. 1995. An anthology of classical queuing models. pages 1-42 *in*: Dshalalow, J. H., ed. Advances in queuing: theory, methods, and open problems. CRC Press. Boca Raton, Florida.

Durfee, E. H. and V. R. Lesser. 1991. Partial global planning: a coordination framework for distributed hypothesis formation. IEEE Transactions on Systems, Man, and Cybernetics 21:1167-1183.

Durfee, E. H. and J. S. Rosenschein. 1994. Distributing problem solving and multi-agent systems: comparisons and examples. pages 94-104 *in*: Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop.

Durfee, E. H., V. R. Lesser, and D. D. Corkill. 1989. Cooperative distributed problem solving. pages 84-147 *in*: Barr, A., P. R. Cohen, and E. A. Feigenbaum, eds. The Handbook of Artificial Intelligence, Vol. IV. Addison-Wesley. Reading, Massachusetts.

Eason, K. 1988. Information technology and organizational change. Taylor and Francis Publishing. London, United Kingdom. 247 pages.

Ens, B. J., Weissing. F. J., and Drent, R. H. 1995. The despotic distribution and deferred maturity: two sides of the same coin. The American Naturalist 146:625-650.

Erol, K., Hendler, J., and Nau, D. S. 1994. Complexity results for HTN planning. Technical Report 94-44. Computer Science Department, University of Maryland. College Park, MD. 30 pages.

Geissman, J. R., and R. D. Schultz. 1988. Verification and validation of expert systems. AI Expert 3(2):26-33.

Graham, J. R. 2001. Real-time scheduling in distributed multi agent systems. PhD Dissertation. University of Delaware. Newark, Delaware. 166 pages.

Graham, J. R. and Decker, K. S. 2000. Towards a distributed, environment-centered agent framework. pages 290-304 *in*: Jennings, Nicholas R. and Y. Lesperance, eds. Proceedings of the Sixth International Workshop on Agent, Theories, Architectures, and Languages (ATAL-99). Springer-Verlag. Berlin, Germany.

Graham, J. R., D. McHugh, M. Mersic, F. McGreary, M. V. Windley, D. Cleaver, and K. S. Decker. 2001. Tools for developing and monitoring agents in distributed multiagent systems. pages 12-27 *in*: Lecture Notes in Computer Science No. 1887: Fourth International Conference on Autonomous Agents. Springer-Verlag. Berlin, Germany.

Graham, I., and P. L. Jones. 1988. Expert systems: knowledge, uncertainty, and decision. Chapman and Hall. New York, New York. 363 pages.

Grogono, P., A. Batarekh, A. Preece, R. Shinghal, and C. Suen. 1991. Expert system evaluation: a selected bibliography. Expert Systems 8(4):227-239.

Gross, D. and C. M. Harris 1974. Fundamentals of queuing theory. John Wiley and Sons, Inc. New York, New York. 556 pages.

Gupta, U. 1991. Validating and verifying knowledge-based systems. IEEE Computer Society Press. Washington, DC. 423 pages.

Hamilton, D., and K. Kelley. 1991. State-of-the-art practice in knowledge-based system verification and validation. Expert Systems With Applications 3:403-410.

Hillier, F. S. and G. J. Lieberman. 1995. Introduction to operations research. McGraw-Hill, Inc. New York, New York. 998 pages.

Holling, C. S. 1978. Adaptive environmental assessment and management. John Wiley and Sons. New York, New York.

Hollnagel, E. 1991. The pragmatic and the academic view on expert systems. Expert Systems With Applications 3:179-185.

Horvitz, E. J., D. E. Heckerman, and C. P. Langlotz. 1986. A framework for comparing alternative formalisms for plausible reasoning. Science 210-214.

Huhns, M. N., and L. M. Stephens. 1999. Multiagent systems and societies of agents. pages 79-120 *in*: Weiss, G., ed. Multiagent systems. MIT Press. Cambridge, Massachusetts.

Hushon, J. M. 1990. Overview of environmental expert systems. pages 1-24 *in*: Hushon, J. M., ed. Expert systems for environmental applications. American Chemical Society. Washington, DC.

Jacobs, D. and K. R. Dixon. 1982. A queuing model of white-tailed deer harvest. Journal of Wildlife Management 46(2):325-332.

Jensen, F. V. 1996. An introduction to Bayesian belief networks. Springer-Verlag. New York, New York. 178 pages.

Johnson, D. H. 1999. The insignificance of statistical significance testing. Journal of Wildlife Management 63(3):763-772.

Johnson, D. H. 2001. Validating and evaluating models. pages 105-119 *in*: Shenk, T. M., and A. B. Franklin, eds. Modelling in Natural Resource Management: Development, Interpretation, and Application. Island Press. Washington, DC.

Kendall, D. G. 1953. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. Annals of Mathematical Statistics 24(3):338-354.

Kleinrock, L. 1975. Queuing systems volume I: theory. John Wiley and Sons. New York, New York. 417 pages.

Kokko, H., J. Lindstrom, R. V. Alatalo, and P. T. Rintamaki. 1998. Queuing for territory positions in the lekking black grouse (*Tetrao tetrix*). Behavioral Ecology 9(4):376-383.

Labrou, Y. and T. Finin 1997. A proposal for a new KQML specification. Technical Report TR CS-97-03. Computer Science and Electrical Engineering Department, University of Maryland Baltimore County. Baltimore, Maryland. 42 pages.

Maitre, B., and Laasri, H. 1990. Cooperating expert problem-solving in blackboard systems: ATOME case study. pages 251-263 *in*: Y. Demazeau, ed. Decentralized A.I.: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World. England: Elsevier Science Publishers. Cambridge, England.

Maniak, U. and W. Trau. 1971. Optimal operations of reservoirs in the Harz Mountains. pages 901-907 *in*: Mathematical Models in Hydrology: Proceedings of the Warsaw Symposium. Paris, France.

Mielke, P. W. and K. J. Berry. 2001. Permutation methods: a distance function approach. Springer-Verlag. New York, New York. 352 pages.

Mielke, P. W., K. J. Berry, and C. O. Neidt. 1996. A permutation test for multivariate matched-pair analyses: comparisons with Hotelling's multivariate matched-pair T^2 test. Psychological Reports 78:1003-1008.

Murrell, S. and R. T. Plant. 1997. A survey of tools for the validation and verification of knowledge-based systems: 1985-1995. Decision Support Systems 21:307-323.

Nii, H. P. 1986a. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. AI Magazine 7(2):38-53.

Nii, H. P. 1986b. Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective. AI Magazine 7(3):82-106.

O'Keefe, R. M., O. Balci, and E. P. Smith. 1987. Validating expert system performance. IEEE Expert 2(4):81-90.

O'Leary, D. E. 1994. Verification and validation of intelligent systems: five years of AAAI workshops. International Journal of Intelligent Systems 9:653-657.

Pearl, J. 1990. The Bayesian approach. pages 339-344 *in*: Shafer, G., and J. Pearl, eds. Readings in uncertain reasoning. Morgan Kaufmann Publishers. San Mateo, California.

Pinson, S. D., J. A. Louca, and P. Moraitis. 1997. A distributed decision support system for strategic planning. Decision Support Systems 20:35-51.

Rao, A. S. and M. P. Georgeff. 1991. Modeling rational agents within a BDI-architecture. pages 1-18 *in*: Allen, J., R. Fikes, and E. Sandewall, Principles of knowledge representation and reasoning: proceedings of the second international conference . Morgan Kaufmann Publishers. San Mateo, California.

Rao, A. S. and M. P. Georgeff. 1995. BDI agents: from theory to practice. pages 312-319 *in*: Proceedings of the First International Conference on Multiagent Systems. AAAI Press. Menlo Park, California.

Reed, T. 2000. 2000 Fall trumpeter swan survey. Unpublished report. U.S. Fish and Wildlife Service. Lakeview, Montana. 28 pages.

Rushby, J. 1988. Validation and testing of knowledge-based systems: how bad can it get? pages 77-83 *in*: Gupta, U., ed. Validating and Verifying Knowledge-Based Systems. IEEE Computer Society Press. Los Alamitos, California.

Russell, S. J. and P. Norvig. 1995. Artificial intelligence: a modern approach. Prentice Hall. Englewood Cliffs, New Jersey. 932 pages.

Rykiel, E. J. 1989. Artificial intelligence and expert systems in ecology and natural resource management. Ecological Modelling 46:3-8.

Scott, A. C., J. E. Clayton, and E. L. Gibson. 1991. A practical guide to knowledge acquisition. Addison-Wesley Publishing Company. Reading, Massachusetts. 509 pages.

Shafer, G. 1988. Probability judgment in artificial intelligence. pages 127-135 *in*: Kanal, L. N. and J. F. Lemmer, eds. Uncertainty in artificial intelligence: proceedings of the fourth conference. Elsevier Science Publishers. B.V. Amsterdam, The Netherlands.

Shafer, G. 1990. Belief functions. pages 473-481 *in*: Shafer, G., and J. Pearl ,eds. Readings in uncertain reasoning. Morgan Kaufmann Publishers. San Mateo, California.

Shortliffe, E. H., and B. G. Buchanan. 1984. A model of inexact reasoning in medicine. pages 233-262 *in*: Buchanan, B. G., and E. H. Shortliffe, eds. Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project. Addison-Wesley Publishing Company. Reading, Massachusetts.

Sojda, R. S., D. J. Dean, and A. E. Howe. 1994. A decision support system for wetland management on national wildlife refuges. AI Applications 8(2):44-50.

Sojda, R. S., and A. E. Howe. 1999. Applying cooperative distributed problem solving methods to trumpeter swan management. pages 63-67 *in* Cortes, U. and M. Sanchez-Marre, eds. Environmental Decision Support Systems and Artificial Intelligence. American Association for Artificial Intelligence Technical Report WS-99-07. AAAI Press. Menlo Park, California.

Sojda, R. S., J. E. Cornely, and A. E. Howe. in press. Development of an expert system for assessing trumpeter swan breeding habitat in the Northern Rocky Mountains. Waterbirds (in press).

Sprague, R. H. Jr. and E. D. Carlson. 1982. Building effective decision support systems. Prentice-Hall. Englewood Cliffs, New Jersey. 329 pages.

Starfield, A. M., B. P. Farm, and R. H. Taylor. 1989. A rule-based ecological model for the management of an estuarine lake. Ecological Modelling 46:107-119.

Stuth, J. W. and M. S. Smith. 1993. Decision support for grazing lands: an overview. pages 1-35 in J.W. Stuth and B.G. Lyons, eds. Decision support systems for the management of grazing lands. Man and the Biosphere Series Volume 11: Papers from the International Conference on Decision Support Systems for Resource Management. The Parthenon Publishing Group. Pearl River, New York

Subcommittee on Rocky Mountain Trumpeter Swans. 1998. Pacific Flyway management plan for the Rocky Mountain population of trumpeter swans. Unpublished report. Pacific Flyway Study Committee. Portland, Oregon.

Wallace, D. R. and R. U. Fujii. 1989. Software verification and validation: an overview. IEEE Software 6(3):10-17.

Walters, C. 1986. Adaptive management of renewable resources. Macmillan Publishing Co. New York, New York. 374 pages.

Weiss, G. 1999. Prologue: multiagent systems and distributed artificial intelligence. pages 1-23 *in*: Weiss, G., ed. Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press. Cambridge, Massachusetts.

White, G. C., L. H. Carpenter, and D. R. Anderson. 1985. Application of expert systems in wildlife. Transactions of the North American Wildlife and Natural Resources Conference 50:363-366.

Williams, B. K. 1997. Logic and science in wildlife biology. Journal of Wildlife Management 61(4): 1007-1015.

Wooldridge, M. and N. R. Jennings. 1995. Intelligent agents: theory and practice. Knowledge Engineering Review 10(2):115-152.

Wooldridge, M. 1999. Intelligent agents. pages 27-77 *in*: Weiss, G., ed. Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press. Cambridge, Massachusetts.

APPENDIX 1

THE DEFAULT CONFIGURATION FILE [dss.config] FOR THE DECISION SUPPORT SYSTEM FOR TRUMPETER SWAN MANAGEMENT

The following text file, dss.config, resides in the directory, .../decaf/dss/data/:

```
#dss configuration

#number between 0 - 100
BREEDING_THRESHOLD_VALUE = 60

#number between 0 - 1
REMAIN_IN_QUEUE_MULTIPLIER = .1

#ignore Breeding expert system output (yes/no)
IGNORE_BREEDING_EXPERT_SYSTEM = no

#ignore Flyway expert system output (yes/no)
IGNORE_FLYWAY_EXPERT_SYSTEM = no

#ignore Wetland expert system output (yes/no)
IGNORE_WETLAND_EXPERT_SYSTEM = no
```

APPENDIX 2

THE OBSERVED NUMBERS OF SWANS AS USED IN THE QUEUING SYSTEM

This table provides the total, observed number of swans (September) represented in each queuing system server. Actual totals are calculated during system execution (by the program ../decaf/dss/move/dss_move.c) from the raw data contained in the files ../decaf/dss/data/Fall_<year>.txt .

YEAR	SERVER[1]																											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1971	0	0	0	0	2	1	0	337	0	0	69	5	0	35	44	0	0	0	0	0	0	0	0	0	0	0	0	0
1974	124	0	0	0	2	0	0	343	0	0	88	0	0	54	50	0	0	0	0	0	0	0	0	0	0	0	0	0
1977	278	0	0	0	0	0	0	329	0	0	61	0	0	55	34	0	12	0	0	0	0	0	0	0	0	0	0	0
1979	0	0	0	0	3	0	0	380	0	0	60	2	0	41	43	0	11	0	0	0	0	0	0	0	0	0	0	0
1980	703	0	0	0	6	0	0	313	0	0	70	7	0	40	40	0	6	0	0	0	0	0	0	0	0	0	0	0
1983	845	0	0	0	1	0	0	254	0	0	96	4	0	35	47	0	7	0	0	0	0	0	0	0	0	0	0	0
1984	945	0	0	0	3	2	0	269	0	0	96	1	2	31	0	0	10	0	0	0	0	0	0	0	0	0	0	0
1985	1038	0	0	0	0	0	0	291	0	0	104	7	0	0	24	0	3	0	0	0	0	0	0	0	0	0	0	0
1986	1019	0	0	0	1	0	0	201	0	0	98	3	0	36	45	0	2	0	0	0	0	0	2	0	0	0	0	0
1987	1241	0	0	0	0	0	0	343	0	0	0	6	0	0	0	0	6	0	0	4	0	0	0	0	0	0	0	0
1988	1115	0	0	0	4	1	0	340	0	0	86	13	0	43	78	0	7	9	6	10	0	0	0	0	0	0	0	0
1989	1409	0	0	0	3	2	0	312	0	0	90	6	0	30	77	0	6	5	17	10	0	0	1	0	0	0	0	0
1990	1574	0	0	0	0	0	0	353	1	0	78	12	0	28	64	0	3	0	19	5	0	0	0	0	0	0	0	0
1991	1598	0	0	0	2	0	0	234	0	0	75	17	7	33	64	0	5	0	50	0	0	0	14	23	36	2	0	0
1992	1660	0	0	0	2	1	0	210	0	0	55	8	11	0	74	5	6	21	26	9	4	0	0	13	39	36	0	0
1993	2172	0	0	0	0	0	0	76	0	0	47	10	9	0	60	0	2	12	28	4	0	0	0	13	34	9	0	0
1994	2349	0	0	0	0	0	0	113	0	0	52	10	15	35	65	8	5	12	49	1	0	0	4	24	22	32	0	0
1995	2498	0	0	0	0	0	0	80	0	0	54	21	24	34	67	0	11	6	34	3	2	0	11	20	14	23	0	0
1996	2240	0	0	0	0	3	0	119	0	0	64	9	12	21	58	3	16	6	34	2	16	0	5	21	22	33	0	0
1997	1756	0	0	6	0	1	0	98	0	0	62	13	25	18	76	10	19	4	35	2	0	0	2	23	23	2	0	0
1998	3058	0	0	0	0	0	0	116	0	0	60	14	20	23	53	6	16	7	47	4	5	0	0	16	27	5	0	0
1999	3088	0	0	0	0	3	0	114	0	0	45	17	21	20	48	11	18	3	34	6	3	0	8	19	14	1	0	0
2000	3471	0	0	0	5	2	0	126	0	0	79	21	15	18	56	8	10	7	37	3	7	0	0	28	15	3	0	0

[1] Server names as used in the queuing system are found on the following page.

Server names as used in the queuing system. These can also be found in the file ../decaf/dss/move/dss_move.c :

1	Canada
2	Okanagan
3	Freezeout Lake
4	Flathead Valley
5	Upper Madison
6	Mid-Madison
7	Lower Madison
8	Centennial Valley
9	Teton Basin
10	Swan Valley
11	Island Park
12	Lower Henry's Fork
13	Paradise Valley
14	Yellowstone Lake
15	Jackson Hole
16	Green River
17	Camas NWR
18	American Falls
19	Grays Lake NWR
20	Salt River
21	Bear Lk/Soda Sp
22	Bear River MBR
23	Lower Snake River
24	Ruby Lake NWR
25	Malheur NWR
26	Summer Lake WMA
27	Central Valley
28	Unknown

APPENDIX 3

THE RAW VALUES FOR LIKELIHOOD OF MOVEMENT BETWEEN SEASONS

The following matrices are interpreted as the likelihood of movement from server (area) j to server i where a cell in the matrix is represented as row i, column j. Server names are in the following order:

1 Canada
2 Okanagan
3 Freezeout Lake
4 Flathead Valley
5 Upper Madison
6 Mid-Madison
7 Lower Madison
8 Centennial Valley
9 Teton Basin
10 Swan Valley
11 Island Park
12 Lower Henry's Fork
13 Paradise Valley
14 Yellowstone Lake
15 Jackson Hole
16 Green River
17 Camas NWR
18 American Falls
19 Grays Lake NWR
20 Salt River
21 Bear Lk/Soda Sp
22 Bear River MBR
23 Lower Snake River
24 Ruby Lake NWR
25 Malheur NWR
26 Summer Lake WMA
27 Central Valley
28 Unknown

Breeding to Postbreeding

0	5	0	0	20	5	5	10	5	0	75	0	0	5	5	0	2	2	0	0	5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	5	0	0	100	10	0	50	0	0	50	5	0	5	5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	5	5	0	10	10	0	100	0	0	5	5	0	5	15	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	95	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	10	0	0	5	0	0	5	0	0	55	25	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	5	0	0	0	95	0	0	0	0	35	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	15	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	100	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	80	40	50	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Postbreeding to Wintering

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
0	0	0	0	50	50	0	0	15	15	75	20	0	5	2	0	0	10	0	0	0	1	2	0	0	0	5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	100	0	0	0	20	20	95	20	0	10	2	0	0	10	0	5	0	1	2	0	0	0	5	0
0	0	0	0	90	0	0	0	25	20	95	35	0	0	2	0	0	10	0	0	0	1	2	0	0	0	5	0
0	0	0	0	80	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	15	5	90	35	0	5	0	0	0	5	0	0	0	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	100	25	60	50	0	0	5	0	0	5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	20	20	100	40	0	10	2	0	0	10	0	0	5	0	2	0	0	0	5	0
0	0	0	0	0	0	0	0	0	0	33	33	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	30	0	0	0	15	5	50	15	0	100	5	0	0	5	0	5	0	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	10	0	10	5	0	0	100	0	0	0	0	25	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	35	30	60	75	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	15	0	55	20	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	25	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	75	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Wintering to Prebreeding

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	75	0	100	75	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	75	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	80	0	80	80	0	10	0	0	80	5	0	0	5	0	5	0	0	0	0	0	0	0	0	0	0	0
0	0	60	0	60	60	0	15	0	0	60	10	0	0	3	0	10	0	0	0	0	0	0	0	0	0	0	0
0	0	75	0	75	75	0	15	0	0	100	3	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
0	0	80	0	80	80	0	5	0	0	85	5	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	70	0	70	70	0	2	0	0	30	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	20	0	20	20	0	2	0	0	20	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	50	0	50	50	0	10	0	0	50	15	0	0	0	0	15	100	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	5	0	5	5	0	0	0	0	5	0	0	0	50	0	0	0	0	100	0	0	0	0	0	0	0	0
0	0	5	0	5	0	0	5	0	0	5	0	0	0	0	0	0	5	0	0	100	0	0	0	0	0	0	0
0	0	5	0	5	5	0	5	0	0	5	5	0	0	0	0	5	5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Prebreeding to Breeding

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	45	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	2	0	0	10	0	0	5	0	0	5	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	5	0	0	5	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	5	65	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	5	0	0	2	0	0	5	0	0	4	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	5	0	0	5	0	0	0	0	0	5	5	5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	40	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	50	0	50	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

APPENDIX 4

STRAIGHTLINE DISTANCES BETWEEN AREAS AND GROUPINGS OF AREAS

Table of distances (km) between Rocky Mountain population of trumpeter swan September survey areas. Areas delineating rows and columns follow the order in the subsequent table, "explanation of actual location of survey areas".

0	594	600	549	1323	1176	1176	1323	1323	1323	1323	1323	1323	1323	1323	1630	1323	1323	1323	1323	1601	1601	1397	1698	1328	1384	1765
594	0	597	457	867	867	867	867	867	867	867	867	867	867	867	1165	867	867	867	867	1091	1091	829	1126	735	790	1166
600	597	0	156	325	251	209	332	399	399	399	399	274	377	399	669	399	522	556	556	556	685	603	881	727	866	1221
549	457	156	0	412	330	302	398	468	468	468	468	381	476	468	756	468	558	614	614	614	725	568	868	638	765	1136
1323	867	325	412	0	88	117	65	93	141	49	107	90	70	129	336	128	221	188	207	275	367	418	627	647	798	1083
1176	867	251	330	88	0	45	86	170	218	136	164	82	147	213	419	172	275	261	286	339	436	424	666	629	779	1086
1176	867	209	302	117	45	0	131	193	193	193	193	79	176	193	463	193	329	352	352	352	482	464	710	658	807	1121
1323	867	332	398	65	86	131	0	92	136	63	170	135	124	149	351	87	189	180	208	253	349	369	588	597	749	1035
1323	867	399	468	93	170	193	92	0	50	40	43	175	99	60	256	84	146	96	117	173	277	343	553	614	764	1030
1323	867	399	468	141	218	193	136	50	0	90	60	225	140	66	219	92	110	47	78	126	229	343	553	614	764	1030
1323	867	399	468	49	136	193	63	40	90	0	59	142	82	90	295	85	172	132	154	222	313	377	553	614	764	1030
1323	867	399	468	107	164	193	170	43	60	59	0	191	131	94	276	44	116	99	132	169	271	343	553	614	764	1030
1323	867	274	381	90	82	79	135	175	225	142	191	0	109	198	397	209	308	268	287	353	451	480	701	697	848	1144
1323	867	377	476	70	147	176	124	99	140	82	131	109	0	94	292	167	242	178	183	256	359	463	642	706	857	1123
1323	867	399	468	129	213	193	149	60	66	90	94	198	94	0	209	135	169	91	90	162	266	407	553	614	764	1030
1630	1165	669	756	336	419	463	351	256	219	295	276	397	292	209	0	307	262	188	148	165	199	500	516	773	907	1080
1323	867	399	468	128	172	193	87	84	92	85	44	209	167	135	307	0	104	122	159	187	270	295	553	614	764	1030
1323	867	522	558	221	275	329	189	146	110	172	116	308	242	169	262	104	0	94	133	107	171	244	393	511	656	883
1323	867	556	614	188	261	352	180	96	47	132	99	268	178	91	188	122	94	0	44	84	182	336	429	606	747	956
1323	867	556	614	207	286	352	208	117	78	154	132	287	183	90	148	159	133	44	0	83	178	336	429	606	747	956
1601	1091	556	614	275	339	352	253	173	126	222	169	353	256	162	165	187	107	84	83	0	99	336	429	606	747	956
1601	1091	685	725	367	436	482	349	277	229	313	271	451	359	266	199	270	171	182	178	99	0	334	324	599	727	882
1397	829	603	568	418	424	464	369	343	343	377	343	480	463	407	500	295	244	336	336	336	334	0	306	272	412	661
1698	1126	881	868	627	666	710	588	553	553	553	553	701	642	553	516	553	393	429	429	429	324	306	0	454	531	576
1328	735	727	638	647	629	658	597	614	614	614	614	697	706	614	773	614	511	606	606	606	599	272	454	0	152	495
1384	790	866	765	798	779	807	749	764	764	764	764	848	857	764	907	764	656	747	747	747	727	412	531	152	0	386
1765	1166	1221	1136	1083	1086	1121	1035	1030	1030	1030	1030	1144	1123	1030	1080	1030	883	956	956	956	882	661	576	495	386	0

Explanation of actual location of survey areas.

CODE	ACTUAL NAME	SPECIFIC LOCATION USED FOR DISTANCES
Canada	Canada	Grande Prairie, AB
okanag	Okanagan	Kelowna, BC
freeze	Freezeout Lake	Fairfield, MT
flathe	Flathead Lake	approximate center of Flathead Lake, MT
uppmad	Upper Madison River and Hebgen Lake	East end of Hebgen Lake
ennisl	Ennis Lake and Mid-Madison River	Ennis Lake
lowmad	Lower Madison River	10km due south of Interstate 90
centen	Centennial Valley	approximately in the Swan Lake area of Red Rock Lakes NWR
tetonb	Teton Basin	approximately 25km ENE of Teton, ID
swanva	Swan Valley	Swan Valley, ID
island	Island Park	Big Bend area of the Henry's Fork in Harriman State Park, ID
lowhen	Lower Henry's Fork River, Market Lake, and Sand Creek	St. Anthony, ID
paradi	Paradise Valley	approximately 4km South of Emigrant, MT
yellow	Yellowstone Lake and Nearby River	approximate center of Yellowstone Lake
jackso	Jackson Hole	Flat Creek marshes in National Elk Refuge
greenr	Green River	Seedskadee NWR
camas	Camas NWR and Mud Lake	midway between Camas NWR And Mud Lake
americ	American Falls Reservoir	approximate center of the North one-third of reservoir
graysl	Grays Lake NWR	approximate center of Grays Lake
saltr	Salt River	near Thayne, WY
bearba	Bear River Basin, Bear Lake NWR, and Soda Springs	Soda Springs, ID
bearmb	Bear River Migratory Bird Refuge	near mouth of the Bear River
lowsna	Lower Snake River	Brunneau Dunes State Park, 27km South of Mountain Home, ID
rubyl	Ruby Lake NWR	approximately 6km North of the South refuge boundary
malheu	Malheur NWR	midway between Harney and Malheur Lakes
summer	Summer Lake	approximate center of Summer Lake, OR
centra	Central Valley of California	Willows, CA

NOTES:

- Measurements from Grande Prairie and Kelowna were taken to Island Park and applied to areas in the core tri-state area, except for Bear River Basin and Bear River MBR.
- Measurements from Flathead, Freezeout, Lower Madison, Central Valley, Summer Lake, Malheur, Ruby Lake were taken to the following groupings:
  - to Soda Springs, ID for Grays Lake, Salt River, and Bear River Basin;
  - to Ashton, ID for Island Park, Lower Henry's Fork, Camas, Teton Basin, Swan Valley, and Jackson Hole
- Measurements from Lower Snake River were taken to the following groupings:
  - to Soda Springs, ID for Grays Lake, Salt River, and Bear River Basin;
  - to Teton, ID for Lower Henry's Fork, Teton Basin, and Swan Valley

APPENDIX 5

DOCUMENTATION FOR THE AGENTS, TASKS, AND ACTIONS

Facilitator Agent: plan file overview
Facilitator Agent

_Startup	
bldDisplayESAdv	Builds the advertise KQML message addressed to the Matchmaker agent about the dss_DisplayESStatus task.
advMMDisplayES	Requests that the advertise KQML message built by the bldDisplayESAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
bldCleanupAdv	Builds the advertise KQML message addressed to Matchmaker about the dss_Cleanup task.
advMMCleanup	Requests that the advertise KQML message built by the bldCleanupAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
startUserAction	Starts the dss_UserAction task via a local achieve message.
taskTerminator	Waits for the three OK outcomes from the other actions of this task, and returns the OK outcome notifying DECAF that _Startup is complete.
_Shutdown	
stop	Prints a short status message to the system console.
dss_UserAction	
selectDSSAction	Presents two choices to the user via the system console: (1) start a new Decision Support System consultation or (2) quit. If option 1 is chosen, a local achieve KQML message is sent to dss_AskUser task, else if option 2 is selected, a local achieve KQML message is sent to the _Shutdown task.

dss_Cleanup	
announceSimCompletion	Prints a descriptive summary of the just-completed simulation that consists of user name, simulation description, start and end years, refuge agents that were active during the consultation, expert system(s) that did not participate in the run, whether simulated or the actual observed swan numbers were used, and whether Canadian bird numbers were included in the input data.
requestNextUserAction	Sends a local achieve KQML message to the dss_UserAction task.
dss_AskUser	
getUserInfo	Interactively obtains the user information needed by the Decision Support System to start a new consultation. This includes user's name, simulation description, start and end years, the names of refuge agents that should be active during the simulation, whether the system should use simulated swan numbers, and if input data should include Canadian bird numbers. Also reads the local DSS configuration file to decide what expert systems should be ignored during the simulation. When an expert system is *ignored*, the DSS database file that corresponds to that expert system is modified to prevent changes in the queuing model based on the information contained in the database. Unless the configuration file disabled all expert systems, the user will be given an option to clear out the databases for the active expert systems. When a database is *cleared* DSS will reinitialize it later using the output of the corresponding expert system. Because this action uses user-provided information from both the console and the configuration file, an unforeseen input or output errors may occur. When the action detects an error it will ask the user whether to restart the system. If the answer is "No" the action will return a FAIL outcome, which will activate the bldShutdownMsg task. Otherwise the action will attempt to collect the needed information again. User-provided information is stored in a Java utility class internal to the Facilitator agent.
bldProcessUserMsg	Builds a local achieve KQML message requesting the dss_ProcessUserRequest task.
achFacProcessUser	Requests that the achieve KQML message built by the bldProcessUserMsg action be sent to the Facilitator agent. No response message is expected, and task will continue its execution as soon as the message is sent.

bldShutdownMsg	Builds a local achieve KQML message requesting the Shutdown task.
achFacShutdown	Requests that the achieve KQML message built by the bldShutdownMsg action be sent to the Facilitator agent. No response message is expected, and task will continue its execution as soon as the message is sent.
collectAskUser	Waits for the OK outcome from one of the other actions of this task, and then returns the OK outcome informing DECAF that dss_AskUser task has finished.
dss_DisplayESStatus	
showUserESStatus	Prints to the console a request from a refuge agent informing the user that a particular expert system needs to be run. No message is printed if the refuge agent says that the database is current, or if the expert system output file already exists.
listenForESFile	Monitors a specific directory on disk to see if the expert system output exists. The action continues to wait until the file is detected, and the user is then notified that the file has been found, and the task continues its execution. The current pause between file system checks is one second.
telRefugeESAch	Requests that the tell KQML message be sent as a response to the refuge agent, telling the refuge agent that a previous achieve request has been successfully satisfied. No response message is expected, and task will continue its execution as soon as the message is sent.
dss_RunMove	
bldAssembleMsg	Builds an achieve message addressed to the Move agent, requesting that the dss_Assemble task be initiated when possible.
achMoveAssemble	Requests that the achieve KQML message built by the bldAssembleMsg action be sent to the Move agent. No response message is expected, and task will continue its execution as soon as the message is sent.
processRefugeNames	Ensures that each refuge agent requested by the user has been notified of the new request. This is done by building an achieve message requesting the dss_Assemble task of the Move agent. The name of the refuge agent to be notified is included in the content field of the KQML message. This action uses the Java utility class FacilitatorData internal to the Facilitator agent to carry out the processing of refuge agent names specified by the user.

bldRunMoveDoneMsg	Builds a local tell message in response to a local achieve message from the dss_ProcessUserRequest task saying that the previous request was successfully satisfied.
telFacUserRequest	Requests that the tell KQML message built by the bldRunMoveDoneMsg action be sent to the Facilitator agent. No response message is expected, and task will continue its execution as soon as the message is sent.
dss_ProcessUserRequest	
bldTrackerMsg	Builds an achieve KQML message addressed to the Move agent requesting the dss_DataTracker task. This message contains a subset of user-provided information that the Move agent requires. The subset consists of the user's name, start and end years of the simulation, active refuges, the use of Canadian bird numbers, user-provided simulation description, as well as the number of times the dss_Assemble task will be requested by the Facilitator agent.
achMoveTracker	Requests that the achieve KQML message built by the bldTrackerMsg action be sent to the Move agent. A response message is expected, and the task will pause its execution until a reply is received.
bldRunMoveMsg	Builds a local achieve KQML message requesting dss_RunMove task.
achFacRunMove	Requests that the achieve KQML message built by the bldRunMoveMsg action be sent to the Facilitator agent. A response message is expected, and the task will pause its execution until a reply is received.
processRequestYears	Ensures that refuge agent(s) of the requested simulation have been activated for each year of the simulation. This is implemented by sending a local achieve message requesting the dss_RunMove task and including the year for which the refuge agent(s) should be run. Year accounting is done by the Java utility class FacilitatorData internal to the Facilitator agent.

Move Agent: plan file overview
Move Agent

_Startup	
bldAssembleAdv	Builds the advertise KQML message addressed to the Matchmaker agent about the dss_ task.
advMMAssemble	Requests that the advertise KQML message built by the bldAssembleAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.

bldSimAdv	Builds the advertise KQML message addressed to Matchmaker about the dss_Sim task.
advMMSim	Requests that the advertise KQML message built by the bldSimAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
bldTrackerAdv	Builds the advertise KQML message addressed to Matchmaker about the dss_DataTracker task.
advMMTracker	Requests that the advertise KQML message built by the bldTrackerAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
taskTerminator	Waits for the three OK outcomes from the other actions of this task, and returns the OK outcome notifying DECAF that _Startup is complete.
_Shutdown	
stop	Prints a short status message to the system console.
dss_DataTracker	
updateMoveData	Uses the simulation information sent by the Facilitator agent to update the data stored in the Java utility class internal to the Move agent. Upon successful update constructs a tell KQML message addressed to the Facilitator agent in response to a previous achieve message.
tellFacProcessUserTrackDone	Requests that the tell KQML message built by the updateMoveData action be sent to the Facilitator agent. A response message is not expected, and the task will continue its execution as soon as the message is sent.
dss_Assemble	
bldBreedingChkDBMsg	Builds an achieve KQML message addressed to the Refuge agent requesting the dss_Breeding task.
achRefBreed	Requests that the achieve KQML message built by the bldBreedingChkDBMsg action be sent to the Refuge agent. A response message is expected, and the task will pause the execution of the "Breeding" path until a reply is received.
bldFlywayChkDBMsg	Builds an achieve KQML message addressed to the Refuge agent requesting the dss_Flyway task.

achRefFly	Requests that the achieve KQML message built by the bldFlywayChkDBMsg action be sent to the Refuge agent. A response message is expected, and the task will pause the execution of the "Flyway" path until a reply is received.
bldHabitatChkDBMsg	Builds an achieve KQML message addressed to the Refuge agent requesting the dss_Habitat task.
achRefHab	Requests that the achieve KQML message built by the bldHabitatChkDBMsg action be sent to the Refuge agent. A response message is expected, and the task will pause the execution of the "Habitat" path until a reply is received.
collectRefugeMsgs	Waits for the three OK outcomes from the other actions of this task. When the other three actions provide the OK outcomes it means that the refuge agent has successfully achieved its objectives and dss_Assemble can signal the dss_SimDispatcher task that one refuge is ready for one year of the DSS simulation run. A local achieve KQML message requesting dss_SimDispatcher task is therefore constructed by this action.
achSimDispatcher	Requests that the achieve KQML message built by the collectRefugeMsgs action be sent to the Move agent. A response message is not expected, and the task will continue its execution as soon as the message is sent.
dss_SimDispatcher	
listenAssembleMsg	Waits for local achieve KQML messages from the dss_Assemble task. Upon receipt of each message the internal Java utility class is notified of the new message. The utility class keeps track of the received messages and determines the outcome of this action. If more messages are expected from dss_Assemble then this action will produce the continue outcome. If all expected messages have been received, then the action returns the complete outcome.
bldSimRunMsg	Builds a local achieve KQML message requesting the dss_Sim task.
achMoveSimRun	Requests that the achieve KQML message built by the bldSimRunMsg action be sent to the Move agent. A response message is expected and the task will pause its execution until the response is received.
processSimYears	Ensures that the dss_Sim task is called for every year of the user-requested simulation. This is implemented with the help of the Java utility class TrackerData internal to the Move agent. The utility class increments the simulation start year each time this action is invoked until the end year is reached.

bldSimDoneMsg	Builds an achieve KQML messages addressed to the Facilitator agent requesting the dss_Cleanup task.
achFacCleanup	Requests that the achieve KQML message built by the bldSimDoneMsg action be sent to the Facilitator agent. A response message is not expected and the task will continue its execution as soon as the message is sent.
collectSimDispatch	Waits for an outcome from one of the actions of this task, and returns the OK outcome notifying DECAF that task dss_SimDispatcher is complete.
dss_Sim	
updateDSSDBs	Invokes the C-code native function to check the temporary output directory for database update files. If these files exist, the new information will be imported into the DSS databases. At most three update files can exist per user, per refuge, per year, corresponding to the three DSS databases. Because this action needs to perform several input and output operations on disk, an unforeseen error may occur. In this case the action will return the FAIL outcome. Otherwise the outcome OK is returned.
writeMask	Invokes the C-code native function to create a refuge availability mask. The mask is represented as a matrix with refuge names as rows and seasons as columns. A zero value (0) in the mask denotes that the refuge is "restricted" during the season, which implies that the queuing model will redistribute some percentage of swans from this refuge queue into other queues. The exact percent value of the redistributed amount is specified in the DSS configuration file. If a value of one (1) is specified in the mask file for a refuge during a season then the refuge is "unrestricted" and the queuing model will not reduce the expert-suggested bird movement probabilities for this refuge during the season. Because this action needs to perform several input and output operations on disk, an unforeseen error may occur. In this case the action will return the FAIL outcome. Otherwise the outcome OK is returned.
adjustMoveProbs	Invokes the C-code native function to create the movement probability matrix based on the original expert-suggested migration probabilities and the mask file created by the writeMask action. The expert-suggested probabilities were obtained from knowledge engineering sessions and have been stored in the native function code at compilation time as four (4) two-dimensional arrays.

	The probability adjustment algorithm reads the mask file, and if a value of one (1) is present for the refuge and season combination then the original migration probability value is carried over into the final movement probability matrix. If a value of zero (0) is present for the refuge and season then the REMAIN_IN_QUEUE_MULTIPLIER value is read from the DSS configuration file and is then multiplied by the expert-suggested migration probability value. The resulting probability value is stored in the movement probability matrix for the current refuge and season. The remaining percentage P_r = (1-REMAIN_IN_QUEUE_MULTIPLIER)*ORIGINAL_VALUE is added to the refuge with the maximum probability value. In case there is more than one refuge with the maximum probability value, P_r is distributed to the geographically closest refuge with the maximum probability value. And if there is a tie between refuge distances, the P_r is distributed equally between the closest refuges with the maximum probability values. Because this action needs to perform several input and output operations on disk, an unforeseen error may occur. In this case the action will return the FAIL outcome. Otherwise the outcome OK is returned.
runQueueModel	Invokes the C-code native function to carry out the simulation of one year of swan migration. The function first performs the weighting of the adjusted movement probabilities in order to normalize them with respect to bird movement during a season. Migration probability matrix is a 28 by 28 cell matrix with each cell corresponding to one of the refuges. The last cell represents the "Unknown" queue, which was introduced to prevent the loss of birds due to difficulty in predicting what movement probabilities will be used during the simulation. The Unknown queue is treated in the same manner as regular refuges, except the transition probability from the Unknown queue to other queues is always zero (0). The probability normalization process guarantees that the sum of every row in the matrix is always one (1). The queuing model makes the assumption that all Canadian birds that winter in the U.S. return to Canada to breed. The queuing model separates the U.S. and Canadian birds into two (2) distinct starting vectors and two independent simulations are performed. To obtain the final output the resulting vectors are combined after the simulation is complete. Because it is known that no U.S. birds winter in Canada, the queuing model

	implementation explicitly sets the probability of U.S. birds migrating to Canada to zero (0). Note that if this adjustment results in an all-zero row in the migration probability matrix, the movement probability of birds into the Unknown queue will be set to one (1) for this row in the matrix in order to prevent the loss of any birds. Once the weighted probabilities are obtained the simulation runs for each season, multiplying the migration probability matrix for each of the seasons by the starting vector of that season to obtain the starting vector for the next season. Mathematically, the simulation calculation can be expressed as follows: $M_{0,1} * S_0 = S_1$ where $M_{0,1}$ is the movement probability matrix from season 0 to season 1 and S_0 is the starting vector with swan numbers for season 0. S_1 is then the resulting starting vector with swan numbers for the next season. The simulation always starts with the observed number of swans surveyed during the end of the breeding September season, and ends with the predicted bird numbers at the start of the next breeding season. The September survey numbers are read from the data files on disk each time a simulation is run. Because this action needs to perform several input and output operations on disk, an unforeseen error may occur. In this case the action will return the FAIL outcome. Otherwise the action builds a local tell KQML message in response to a previous achieve KQML message from the dss_SimDispatcher task, and returns the OK outcome.
telMoveSimDispRunAch	Requests that the tell KQML message built by the runQueueModel action be sent to the Move agent. A response message is not expected, and the task will continue its execution as soon as the message is sent.
bldShutdownMsg	Builds a local achieve KQML message requesting the _Shutdown task.
achMoveShutdown	Requests that the achieve KQML message built by the bldShutdownMsg action be sent to the Move agent. A response message is not expected, and the task will continue its execution as soon as the message is sent.
collectSim	Waits for the OK outcome from one of the actions of this task and returns the OK outcome notifying DECAF that task dss_Sim is complete.

Refuge Agent: plan file overview
Refuge Agent

_Startup	
bldBreedAdv	Builds the advertise KQML message addressed to the Matchmaker agent about the dss_Breeding task.
advMMBreed	Requests that the advertise KQML message built by the bldBreedAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
bldFlywayAdv	Builds the advertise KQML message addressed to Matchmaker about the dss_Flyway task.
advMMFlyway	Requests that the advertise KQML message built by the bldFlywayAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
bldHabitatAdv	Builds the advertise KQML message addressed to Matchmaker about the dss_Habitat task.
advMMHabitat	Requests that the advertise KQML message built by the bldHabitatAdv action be sent to the Matchmaker agent. No response message is expected, and task will continue its execution as soon as the message is sent.
taskTerminator	Waits for the three OK outcomes from the other actions of this task, and returns the OK outcome notifying DECAF that _Startup is complete.
_Shutdown	
stop	Prints a short status message to the system console.
dss_Breeding	
ckBreedingDB	Checks the DSS Breeding database whether the data corresponding to the current year, season, and refuge needs to be obtained from the expert system. Necessity of the update is indicated by the sentinel value "-1" in the breeding.dat database file. This action can return one of three possible outcomes: FAIL if the file read operation fails, cur if the database file does not contain the "-1" value, and not_cur if the database does contain the "-1" value for this year, season, and refuge.

	Depending on the outcome value the action will build an appropriate KQML message: the FAIL outcome is associated with an achieve message to the _Shutdown task of the Refuge agent, the cur and not_cur outcomes are associated with an achieve message to the dss_DisplayESStatus action of the Facilitator agent with different values of the :content KQML field.
achFacDispBreedES	Requests that the achieve KQML message built by the ckBreedingDB action be sent to the Facilitator agent. Response message is expected and the task will pause its execution until the response from the Facilitator agent is received.
achFacDispBreed	Requests that the achieve KQML message built by the ckBreedingDB action be sent to the Facilitator agent. Response message is expected and the task will pause its execution until the response from the Facilitator agent is received.
procDataB	Parses the output of the breeding expert system and extracts the integer representation of the breeding habitat quality. This value is then written to a binary file in a temporary directory preceded by the four-digit representation of next year. If the parsing process terminates normally this action will return the OK outcome. Otherwise the action returns the FAIL outcome.
bldBreedCurMsg	Builds the tell KQML message addressed to the dss_Assemble task of the move agent. This message is the response to a previous achieve message from the achRefBreed action.
telMoveAssembleBreed	Requests that the tell KQML message built by the bldBreedCurMsg action be sent to the Move agent. No response message is expected, and task will continue its execution as soon as the message is sent.
achRefShutdownBreed	Requests that the achieve KQML message built by either the ckBreedingDB or procDataB actions be sent to the Refuge agent. No response message is expected, and task will continue its execution as soon as the message is sent.
collectBreeding	Waits for the OK outcome from one of the other actions of this task, and then returns the OK outcome informing DECAF that dss_Breeding task has finished.
dss_Flyway	
ckFlywayDB	Checks the DSS Flyway database whether the data corresponding to the current year, season, and refuge needs to be obtained from the expert system. Necessity of the update is indicated by the sentinel value "-1" in the flyway.dat database file.

	This action can return one of three possible outcomes: FAIL if the file read operation fails, cur if the database file does not contain the "-1" value, and not_cur if the database does contain the "-1" value for this year, season, and refuge. Depending on the outcome value the action will build an appropriate KQML message: the FAIL outcome is associated with an achieve message to the _Shutdown task of the Refuge agent, the cur and not_cur outcomes are associated with an achieve message to the dss_DisplayESStatus action of the Facilitator agent with different values of the :content KQML field.
achFacDispFlyES	Requests that the achieve KQML message built by the ckFlywayDB action be sent to the Facilitator agent. Response message is expected and the task will pause its execution until the response from the Facilitator agent is received.
achFacDispFly	Requests that the achieve KQML message built by the ckFlywayDB action be sent to the Facilitator agent. Response message is expected and the task will pause its execution until the response from the Facilitator agent is received.
procDataF	Parses the output of the flyway expert system and creates an integer representation of the expert system's assessment for the breeding and wintering seasons. There are three possible integers that this action assigns to the expert system's assessment: "999", "1", and "0". The first value of "999" is used in cases when the expert system did not have enough information to make a recommendation. Values "999" and "1" are treated alike in by movement probability adjustment algorithm in the queuing model. The two values corresponding to the breeding and wintering seasons are then written to a binary file in a temporary directory preceded by the four-digit representation of next year for the breeding season, and the current year for the wintering season. If the parsing process terminates normally this action will return the OK outcome. Otherwise the action returns the FAIL outcome.
bldFlyCurMsg	Builds the tell KQML message addressed to the dss_Assemble task of the move agent. This message is the response to a previous achieve message from the achRefFly action.
telMoveAssembleFly	Requests that the tell KQML message built by the bldFlyCurMsg action be sent to the Move agent. No response message is expected, and task will continue its execution as soon as the message is sent.

achRefShutdownFly	Requests that the achieve KQML message built by either the ckFlywayDB or procDataF actions be sent to the Refuge agent. No response message is expected, and task will continue its execution as soon as the message is sent.
collectFlyway	Waits for the OK outcome from one of the other actions of this task, and then returns the OK outcome informing DECAF that dss_Flyway task has finished.
dss_Habitat	
ckHabitatDB	Checks the DSS Habitat database whether the data corresponding to the current year, season, and refuge needs to be obtained from the expert system. Necessity of the update is indicated by the sentinel value "-1" in the habitat.dat database file. This action can return one of three possible outcomes: FAIL if the file read operation fails, cur if the database file does not contain the "-1" value, and not_cur if the database does contain the "-1" value for this year, season, and refuge. Depending on the outcome value the action will build an appropriate KQML message: the FAIL outcome is associated with an achieve message to the _Shutdown task of the Refuge agent, the cur and not_cur outcomes are associated with an achieve message to the dss_DisplayESStatus action of the Facilitator agent with different values of the :content KQML field.
achFacDispWetES	Requests that the achieve KQML message built by the ckHabitatDB action be sent to the Facilitator agent. Response message is expected and the task will pause its execution until the response from the Facilitator agent is received.
achFacDispHab	Requests that the achieve KQML message built by the ckHabitatDB action be sent to the Facilitator agent. Response message is expected and the task will pause its execution until the response from the Facilitator agent is received.
procDataW	Parses the output of the wetland expert system and extracts an integer representation of the wetland habitat quality for current year's postbreeding and wintering seasons, and next year's prebreeding and breeding seasons. There are four possible values: "999", "3", "2", and "1". The "999" value corresponds to a case when the expert system could not make a recommendation, and "3", "2", "1" correspond to "high", "medium", and "low" water levels respectively. The values are saved to a binary file in a temporary directory preceded by the four-

	digit representation of the year that corresponds to the season: the current year for postbreeding and wintering seasons, and the next year for prebreeding and breeding seasons.. If the parsing process terminates normally this action will return the OK outcome. Otherwise the action returns the FAIL outcome.
bldHabitatCurMsg	Builds the tell KQML message addressed to the dss_Assemble task of the move agent. This message is the response to a previous achieve message from the achRefHab action.
telMoveAssembleHab	Requests that the tell KQML message built by the bldHabCurMsg action be sent to the Move agent. No response message is expected, and task will continue its execution as soon as the message is sent.
achRefShutdownHab	Requests that the achieve KQML message built by either the ckHabitatDB or procDataW actions be sent to the Refuge agent. No response message is expected, and task will continue its execution as soon as the message is sent.
collectHabitat	Waits for the OK outcome from one of the other actions of this task, and then returns the OK outcome informing DECAF that dss_Habitat task has finished.

APPENDIX 6

EACH AGENT'S .lsp FILE AS REQUIRED BY DECAF

../decaf/dss/fac/fac.lsp

```
(defaction
   :name ("bldShutdownMsg")
   :parent ("dss_AskUser")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("selectDSSAction.OK")
   :to ("dss_UserAction.OK")
)


(defaction
   :name ("requestNextUserAction")
   :parent ("dss_Cleanup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("advMMCleanup")
   :parent ("_Startup")
   :parameters ("NOPROV")
```

```
   :provisions ("KQML")
   :outcomes ("OK")
)


(NLT
   :name ("advMMDisplayES")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(defaction
   :name ("collectAskUser")
   :parent ("dss_AskUser")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("taskTerminator.OK")
   :to ("_Startup.OK")
)


(deftask
   :name ("dss_Cleanup")
   :parent ("NONE")
   :children ("announceSimCompletion"
              "requestNextUserAction")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("bldCleanupAdv.OK")
```

```
   :to ("advMMCleanup.KQML")
)


(deftask
   :name ("dss_DisplayESStatus")
   :parent ("NONE")
   :children ("showUserESStatus"
              "telRefugeESAch"
              "listenForESFile")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                   )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("listenForESFile")
   :parent ("dss_DisplayESStatus")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("go" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                   )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("telFacUserRequest")
   :parent ("dss_RunMove")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("achFacProcessUser.OK")
   :to ("collectAskUser.trigger")
)


(provides
   :from ("processRefugeNames.done")
```

```
   :to ("bldRunMoveDoneMsg.trigger")
)


(provides
   :from ("telFacUserRequest.OK")
   :to ("dss_RunMove.OK")
)


(NLT
   :name ("achMoveAssemble")
   :parent ("dss_RunMove")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("achFacRunMove.OK")
   :to ("processRequestYears.trigger")
)


(defaction
   :name ("announceSimCompletion")
   :parent ("dss_Cleanup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("processRequestYears")
   :parent ("dss_ProcessUserRequest")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("done" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  "again" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
```

```
   :caf ("AND")
)


(defaction
   :name ("getUserInfo")
   :parent ("dss_AskUser")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
              )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("bldRunMoveMsg.OK")
   :to ("achFacRunMove.KQML")
)


(provides
   :from ("processRequestYears.done")
   :to ("dss_ProcessUserRequest.OK")
)


(provides
   :from ("telRefugeESAch.OK")
   :to ("dss_DisplayESStatus.OK")
)


(provides
   :from ("achMoveAssemble.OK")
   :to ("processRefugeNames.trigger")
)


(provides
   :from ("startUserAction.OK")
   :to ("taskTerminator.finished_UserAction")
)


(NLT
   :name ("achMoveTracker")
```

```
   :parent ("dss_ProcessUserRequest")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(defaction
   :name ("bldDisplayESAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("bldTrackerMsg")
   :parent ("dss_ProcessUserRequest")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("processRefugeNames")
   :parent ("dss_RunMove")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("done" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "again" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)
```

```
(NLT
   :name ("achFacShutdown")
   :parent ("dss_AskUser")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("bldRunMoveDoneMsg.OK")
   :to ("telFacUserRequest.KQML")
)


(defaction
   :name ("bldRunMoveDoneMsg")
   :parent ("dss_RunMove")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("bldTrackerMsg.OK")
   :to ("achMoveTracker.KQML")
)


(NLT
   :name ("achFacProcessUser")
   :parent ("dss_AskUser")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(deftask
   :name ("_Startup")
   :parent ("NONE")
   :children ("taskTerminator"
              "bldDisplayESAdv"
              "advMMDisplayES"
              "bldCleanupAdv"
```

```
            "advMMCleanup"
            "startUserAction")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(deftask
   :name ("dss_ProcessUserRequest")
   :parent ("NONE")
   :children ("bldRunMoveMsg"
            "achFacRunMove"
            "processRequestYears"
            "bldTrackerMsg"
            "achMoveTracker")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(deftask
   :name ("dss_UserAction")
   :parent ("NONE")
   :children ("selectDSSAction")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("listenForESFile.go")
   :to ("telRefugeESAch.KQML")
)
```

```
(provides
   :from ("bldShutdownMsg.OK")
   :to ("achFacShutdown.KQML")
)


(provides
   :from ("showUserESStatus.go")
   :to ("telRefugeESAch.KQML")
)


(provides
   :from ("achMoveTracker.OK")
   :to ("bldRunMoveMsg.trigger")
)


(provides
   :from ("showUserESStatus.listen")
   :to ("listenForESFile.trigger")
)


(provides
   :from ("getUserInfo.FAIL")
   :to ("bldShutdownMsg.trigger")
)


(defaction
   :name ("stop")
   :parent ("_Shutdown")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("requestNextUserAction.OK")
   :to ("dss_Cleanup.OK")
)


(defaction
   :name ("bldCleanupAdv")
```

```
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("bldAssembleMsg.OK")
   :to ("achMoveAssemble.KQML")
)


(provides
   :from ("advMMCleanup.OK")
   :to ("taskTerminator.finished_Cleanup")
)


(provides
   :from ("dss_RunMove.trigger")
   :to ("bldAssembleMsg.trigger")
)


(provides
   :from ("getUserInfo.OK")
   :to ("bldProcessUserMsg.trigger")
)


(deftask
   :name ("dss_AskUser")
   :parent ("NONE")
   :children ("getUserInfo"
              "collectAskUser"
              "bldShutdownMsg"
              "achFacShutdown"
              "achFacProcessUser"
              "bldProcessUserMsg")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
```

```
   :caf ("AND")
)


(provides
   :from ("stop.OK")
   :to ("_Shutdown.OK")
)


(deftask
   :name ("_Shutdown")
   :parent ("NONE")
   :children ("stop")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("taskTerminator")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("finished_UserAction"
            "finished_DisplayES"
            "finished_Cleanup")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("bldProcessUserMsg.OK")
   :to ("achFacProcessUser.KQML")
)


(provides
   :from ("bldDisplayESAdv.OK")
   :to ("advMMDisplayES.KQML")
)
```

```
(defaction
   :name ("bldAssembleMsg")
   :parent ("dss_RunMove")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("telRefugeESAch")
   :parent ("dss_DisplayESStatus")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(defaction
   :name ("startUserAction")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("announceSimCompletion.OK")
   :to ("requestNextUserAction.trigger")
)


(provides
   :from ("collectAskUser.OK")
   :to ("dss_AskUser.OK")
)


(NLT
```

```
   :name ("achFacRunMove")
   :parent ("dss_ProcessUserRequest")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(deftask
   :name ("dss_RunMove")
   :parent ("NONE")
   :children ("bldAssembleMsg"
              "achMoveAssemble"
              "telFacUserRequest"
              "bldRunMoveDoneMsg"
              "processRefugeNames")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("processRequestYears.again")
   :to ("bldRunMoveMsg.trigger")
)


(defaction
   :name ("bldProcessUserMsg")
   :parent ("dss_AskUser")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("showUserESStatus")
   :parent ("dss_DisplayESStatus")
   :children ("NONE")
   :parameters ("NOPROV")
```

```
   :provisions ("NOPROV")
   :outcomes ("go" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  "listen" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("achFacShutdown.OK")
   :to ("collectAskUser.trigger")
)


(defaction
   :name ("bldRunMoveMsg")
   :parent ("dss_ProcessUserRequest")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("processRefugeNames.again")
   :to ("bldAssembleMsg.trigger")
)


(provides
   :from ("advMMDisplayES.OK")
   :to ("taskTerminator.finished_DisplayES")
)


(defaction
   :name ("selectDSSAction")
   :parent ("dss_UserAction")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
```

```
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(utility_function
  (label Information_Gather)
  (goodness_quality_slider 1.0)
  (goodness_cost_slider 0.0)
  (goodness_duration_slider 0.0)
  (threshold_quality_slider 0.0)
  (threshold_cost_slider 0.0)
  (threshold_duration_slider 1.0)
  (quality_threshold 0)
  (cost_limit 5)
  (duration_limit 600)
  (uncertainty_quality_slider 0.2)
  (uncertainty_cost_slider 0.7)
  (uncertainty_duration_slider 0.1)
  (threshold_certainty_quality_slider .1)
  (threshold_certainty_cost_slider   .2)
  (threshold_certainty_duration_slider   .3)
  (quality_certainty_threshold  1)
  (cost_certainty_threshold  2)
  (duration_certainty_threshold  3)
  (meta_goodness_slider 1.0)
  (meta_threshold_slider 0.0)
  (meta_uncertainty_slider 0.0)
  (meta_uncertainty_threshold_slider 0.0)
)
```

../decaf/dss/refuge/refuge.lsp

```
(NLT
   :name ("achFacDispBreed")
   :parent ("dss_Breeding")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("achRefShutdownHab.OK")
   :to ("collectHabitat.trigger")
)


(deftask
   :name ("_Shutdown")
   :parent ("NONE")
```

```
   :children ("stop")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("collectBreeding")
   :parent ("dss_Breeding")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(deftask
   :name ("dss_Breeding")
   :parent ("NONE")
   :children ("procDataB"
              "collectBreeding"
              "bldBreedCurMsg"
              "achRefShutdownBreed"
              "achFacDispBreedES"
              "telMoveAssembleBreed"
              "ckBreedingDB"
              "achFacDispBreed")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("achFacDispBreedES")
   :parent ("dss_Breeding")
   :parameters ("NOPROV")
```

```
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("achFacDispFlyES.OK")
   :to ("procDataF.trigger")
)


(provides
   :from ("collectBreeding.OK")
   :to ("dss_Breeding.OK")
)


(provides
   :from ("procDataB.FAIL")
   :to ("achRefShutdownBreed.KQML")
)


(defaction
   :name ("ckBreedingDB")
   :parent ("dss_Breeding")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "cur" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "not_cur" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
               )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("collectFlyway")
   :parent ("dss_Flyway")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
               )
   :deadline ("0")
   :earliest_start_time ("0")
```

```
   :caf ("AND")
)


(NLT
   :name ("achFacDispWetES")
   :parent ("dss_Habitat")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(defaction
   :name ("bldFlyCurMsg")
   :parent ("dss_Flyway")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("bldHabCurMsg")
   :parent ("dss_Habitat")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("bldHabitatAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
```

```
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("ckFlywayDB")
   :parent ("dss_Flyway")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "cur" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "not_cur" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
              )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("achFacDispHab")
   :parent ("dss_Habitat")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("telMoveAssembleFly.OK")
   :to ("collectFlyway.trigger")
)


(provides
   :from ("achRefShutdownFly.OK")
   :to ("collectFlyway.trigger")
)


(NLT
   :name ("advMMBreed")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)
```

```
(provides
   :from ("advMMBreed.OK")
   :to ("taskTerminator.finished_Breed")
)


(provides
   :from ("bldHabitatAdv.OK")
   :to ("advMMHabitat.KQML")
)


(NLT
   :name ("advMMHabitat")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("bldBreedCurMsg.OK")
   :to ("telMoveAssembleBreed.KQML")
)


(provides
   :from ("stop.OK")
   :to ("_Shutdown.OK")
)


(provides
   :from ("achFacDispBreed.OK")
   :to ("bldBreedCurMsg.trigger")
)


(provides
   :from ("procDataB.OK")
   :to ("bldBreedCurMsg.trigger")
)


(provides
   :from ("ckFlywayDB.not_cur")
   :to ("achFacDispFlyES.KQML")
)


(provides
```

```
   :from ("taskTerminator.OK")
   :to ("_Startup.OK")
)


(provides
   :from ("ckFlywayDB.cur")
   :to ("achFacDispFly.KQML")
)


(NLT
   :name ("achFacDispFlyES")
   :parent ("dss_Flyway")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(NLT
   :name ("achRefShutdownFly")
   :parent ("dss_Flyway")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("bldFlywayAdv.OK")
   :to ("advMMFlyway.KQML")
)


(provides
   :from ("ckHabitatDB.not_cur")
   :to ("achFacDispWetES.KQML")
)


(provides
   :from ("advMMFlyway.OK")
   :to ("taskTerminator.finished_Flyway")
)


(NLT
   :name ("advMMFlyway")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
```

```
)

(NLT
   :name ("achRefShutdownBreed")
   :parent ("dss_Breeding")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)

(provides
   :from ("advMMHabitat.OK")
   :to ("taskTerminator.finished_Habitat")
)

(provides
   :from ("achFacDispWetES.OK")
   :to ("procDataW.trigger")
)

(deftask
   :name ("dss_Habitat")
   :parent ("NONE")
   :children ("procDataW"
              "collectHabitat"
              "bldHabCurMsg"
              "ckHabitatDB"
              "achFacDispWetES"
              "achRefShutdownHab"
              "telMoveAssembleHab"
              "achFacDispHab")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)

(provides
   :from ("procDataW.OK")
   :to ("bldHabCurMsg.trigger")
)

(provides
```

```
   :from ("collectFlyway.OK")
   :to ("dss_Flyway.OK")
)


(provides
   :from ("ckHabitatDB.cur")
   :to ("achFacDispHab.KQML")
)


(defaction
   :name ("bldFlywayAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("ckHabitatDB.FAIL")
   :to ("achRefShutdownHab.KQML")
)


(defaction
   :name ("bldBreedCurMsg")
   :parent ("dss_Breeding")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("collectHabitat")
   :parent ("dss_Habitat")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
```

```
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("telMoveAssembleFly")
   :parent ("dss_Flyway")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(deftask
   :name ("dss_Flyway")
   :parent ("NONE")
   :children ("procDataF"
              "collectFlyway"
              "bldFlyCurMsg"
              "ckFlywayDB"
              "achRefShutdownFly"
              "telMoveAssembleFly"
              "achFacDispFlyES"
              "achFacDispFly")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("telMoveAssembleBreed")
   :parent ("dss_Breeding")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("telMoveAssembleBreed.OK")
   :to ("collectBreeding.trigger")
)
```

```
(provides
   :from ("bldHabCurMsg.OK")
   :to ("telMoveAssembleHab.KQML")
)


(provides
   :from ("bldBreedAdv.OK")
   :to ("advMMBreed.KQML")
)


(defaction
   :name ("stop")
   :parent ("_Shutdown")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("achRefShutdownHab")
   :parent ("dss_Habitat")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("procDataW.FAIL")
   :to ("achRefShutdownHab.KQML")
)


(provides
   :from ("achFacDispFly.OK")
   :to ("bldFlyCurMsg.trigger")
)


(provides
   :from ("achFacDispBreedES.OK")
   :to ("procDataB.trigger")
```

```
)


(provides
   :from ("achFacDispHab.OK")
   :to ("bldHabCurMsg.trigger")
)


(defaction
   :name ("bldBreedAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("procDataW")
   :parent ("dss_Habitat")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("ckBreedingDB.FAIL")
   :to ("achRefShutdownBreed.KQML")
)


(provides
   :from ("bldFlyCurMsg.OK")
   :to ("telMoveAssembleFly.KQML")
)
```

```
(provides
   :from ("telMoveAssembleHab.OK")
   :to ("collectHabitat.trigger")
)


(provides
   :from ("ckFlywayDB.FAIL")
   :to ("achRefShutdownFly.KQML")
)


(deftask
   :name ("_Startup")
   :parent ("NONE")
   :children ("bldBreedAdv"
              "advMMBreed"
              "bldFlywayAdv"
              "advMMFlyway"
              "bldHabitatAdv"
              "advMMHabitat"
              "taskTerminator")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("achFacDispFly")
   :parent ("dss_Flyway")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("ckBreedingDB.not_cur")
   :to ("achFacDispBreedES.KQML")
)


(defaction
   :name ("procDataF")
   :parent ("dss_Flyway")
   :children ("NONE")
   :parameters ("NOPROV")
```

```
   :provisions ("trigger")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("ckBreedingDB.cur")
   :to ("achFacDispBreed.KQML")
)


(defaction
   :name ("ckHabitatDB")
   :parent ("dss_Habitat")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "cur" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "not_cur" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("procDataB")
   :parent ("dss_Breeding")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)
```

```
(provides
   :from ("collectHabitat.OK")
   :to ("dss_Habitat.OK")
)


(provides
   :from ("achRefShutdownBreed.OK")
   :to ("collectBreeding.trigger")
)


(provides
   :from ("procDataF.OK")
   :to ("bldFlyCurMsg.trigger")
)


(NLT
   :name ("telMoveAssembleHab")
   :parent ("dss_Habitat")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("procDataF.FAIL")
   :to ("achRefShutdownFly.KQML")
)


(defaction
   :name ("taskTerminator")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("finished_Breed"
            "finished_Flyway"
            "finished_Habitat")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(utility_function
  (label Information_Gather)
```

```
  (goodness_quality_slider 1.0)
  (goodness_cost_slider 0.0)
  (goodness_duration_slider 0.0)
  (threshold_quality_slider 0.0)
  (threshold_cost_slider 0.0)
  (threshold_duration_slider 1.0)
  (quality_threshold 0)
  (cost_limit 5)
  (duration_limit 600)
  (uncertainty_quality_slider 0.2)
  (uncertainty_cost_slider 0.7)
  (uncertainty_duration_slider 0.1)
  (threshold_certainty_quality_slider .1)
  (threshold_certainty_cost_slider   .2)
  (threshold_certainty_duration_slider   .3)
  (quality_certainty_threshold  1)
  (cost_certainty_threshold  2)
  (duration_certainty_threshold  3)
  (meta_goodness_slider 1.0)
  (meta_threshold_slider 0.0)
  (meta_uncertainty_slider 0.0)
  (meta_uncertainty_threshold_slider 0.0)
)
```

../decaf/dss/move/move.lsp

```
(deftask
   :name ("_Shutdown")
   :parent ("NONE")
   :children ("stop")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("achMoveShutdown.OK")
   :to ("collectSim.trigger")
)


(provides
   :from ("achRefHab.OK")
   :to ("collectRefugeMsgs.finished_HabitatChkDB")
)
```

```
(NLT
   :name ("telFacProcessUserTrackDone")
   :parent ("dss_DataTracker")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("bldBreedingChkDBMsg.OK")
   :to ("achRefBreed.KQML")
)


(provides
   :from ("advMMTracker.OK")
   :to ("taskTerminator.finished_Tracker")
)


(provides
   :from ("bldFlywayChkDBMsg.OK")
   :to ("achRefFly.KQML")
)


(defaction
   :name ("bldSimRunMsg")
   :parent ("dss_SimDispatcher")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("achMoveSimRun.OK")
   :to ("processSimYears.trigger")
)


(provides
   :from ("bldSimAdv.OK")
   :to ("advMMSim.KQML")
)
```

```
(provides
   :from ("listenAssembleMsg.continue")
   :to ("collectSimDispatch.trigger")
)


(provides
   :from ("updateDSSDBs.OK")
   :to ("writeMask.trigger")
)


(provides
   :from ("updateMoveData.OK")
   :to ("telFacProcessUserTrackDone.KQML")
)


(NLT
   :name ("advMMTracker")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(deftask
   :name ("dss_DataTracker")
   :parent ("NONE")
   :children ("updateMoveData"
              "telFacProcessUserTrackDone")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                    )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("telMoveSimDispRunAch")
   :parent ("dss_Sim")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)
```

```
(provides
   :from ("bldHabitatChkDBMsg.OK")
   :to ("achRefHab.KQML")
)


(NLT
   :name ("advMMAssemble")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("bldTrackerAdv.OK")
   :to ("advMMTracker.KQML")
)


(defaction
   :name ("writeMask")
   :parent ("dss_Sim")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("updateMoveData")
   :parent ("dss_DataTracker")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)
```

```
(defaction
   :name ("processSimYears")
   :parent ("dss_SimDispatcher")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("done" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  "again" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("collectSim")
   :parent ("dss_Sim")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("achRefBreed.OK")
   :to ("collectRefugeMsgs.finished_BreedingChkDB")
)


(defaction
   :name ("adjustMoveProbs")
   :parent ("dss_Sim")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)
```

```
(provides
   :from ("achRefFly.OK")
   :to ("collectRefugeMsgs.finished_FlywayChkDB")
)


(provides
   :from ("dss_Sim.OK")
   :to ("collectSim.OK")
)


(provides
   :from ("stop.OK")
   :to ("_Shutdown.OK")
)


(NLT
   :name ("achSimDispatcher")
   :parent ("dss_Assemble")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(provides
   :from ("achFacCleanup.OK")
   :to ("collectSimDispatch.trigger")
)


(NLT
   :name ("achRefHab")
   :parent ("dss_Assemble")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(defaction
   :name ("bldSimAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
```

```
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("achMoveSimRun")
   :parent ("dss_SimDispatcher")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(defaction
   :name ("bldBreedingChkDBMsg")
   :parent ("dss_Assemble")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("bldAssembleAdv.OK")
   :to ("advMMAssemble.KQML")
)


(provides
   :from ("taskTerminator.OK")
   :to ("_Startup.OK")
)


(defaction
   :name ("updateDSSDBs")
   :parent ("dss_Sim")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
```

```
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("processSimYears.again")
   :to ("bldSimRunMsg.trigger")
)


(provides
   :from ("bldSimRunMsg.OK")
   :to ("achMoveSimRun.KQML")
)


(defaction
   :name ("bldTrackerAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("bldSimDoneMsg")
   :parent ("dss_SimDispatcher")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("advMMSim.OK")
   :to ("taskTerminator.finished_Sim")
)
```

```
(defaction
   :name ("bldShutdownMsg")
   :parent ("dss_Sim")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("listenAssembleMsg")
   :parent ("dss_SimDispatcher")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("continue" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                  "complete" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("bldFlywayChkDBMsg")
   :parent ("dss_Assemble")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("adjustMoveProbs.FAIL")
   :to ("bldShutdownMsg.trigger")
)
```

```
(provides
   :from ("bldShutdownMsg.OK")
   :to ("achMoveShutdown.KQML")
)


(provides
   :from ("bldSimDoneMsg.OK")
   :to ("achFacCleanup.KQML")
)


(defaction
   :name ("bldHabitatChkDBMsg")
   :parent ("dss_Assemble")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("writeMask.OK")
   :to ("adjustMoveProbs.trigger")
)


(deftask
   :name ("dss_Sim")
   :parent ("NONE")
   :children ("writeMask"
              "updateDSSDBs"
              "adjustMoveProbs"
              "runQueueModel"
              "collectSim"
              "bldShutdownMsg"
              "achMoveShutdown"
              "telMoveSimDispRunAch")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
```

```
)

(NLT
   :name ("achRefBreed")
   :parent ("dss_Assemble")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)

(NLT
   :name ("achMoveShutdown")
   :parent ("dss_Sim")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)

(provides
   :from ("telMoveSimDispRunAch.OK")
   :to ("collectSim.trigger")
)

(NLT
   :name ("advMMSim")
   :parent ("_Startup")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)

(defaction
   :name ("collectSimDispatch")
   :parent ("dss_SimDispatcher")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)

(provides
   :from ("runQueueModel.done")
```

```
   :to ("telMoveSimDispRunAch.KQML")
)


(defaction
   :name ("stop")
   :parent ("_Shutdown")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("collectRefugeMsgs.OK")
   :to ("achSimDispatcher.KQML")
)


(provides
   :from ("adjustMoveProbs.OK")
   :to ("runQueueModel.trigger")
)


(provides
   :from ("telFacProcessUserTrackDone.OK")
   :to ("dss_DataTracker.OK")
)


(provides
   :from ("listenAssembleMsg.complete")
   :to ("bldSimRunMsg.trigger")
)


(deftask
   :name ("dss_Assemble")
   :parent ("NONE")
   :children ("bldFlywayChkDBMsg"
              "collectRefugeMsgs"
              "bldHabitatChkDBMsg"
              "achRefBreed"
              "achRefFly"
              "achRefHab"
              "bldBreedingChkDBMsg"
```

```
            "achSimDispatcher")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("collectSimDispatch.OK")
   :to ("dss_SimDispatcher.OK")
)


(defaction
   :name ("runQueueModel")
   :parent ("dss_Sim")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("trigger")
   :outcomes ("FAIL" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                "done" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE"
:density "NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(deftask
   :name ("_Startup")
   :parent ("NONE")
   :children ("bldAssembleAdv"
            "bldSimAdv"
            "advMMAssemble"
            "advMMSim"
            "taskTerminator"
            "bldTrackerAdv"
            "advMMTracker")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
```

```
)


(defaction
   :name ("bldAssembleAdv")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(provides
   :from ("writeMask.FAIL")
   :to ("bldShutdownMsg.trigger")
)


(provides
   :from ("processSimYears.done")
   :to ("bldSimDoneMsg.trigger")
)


(provides
   :from ("achSimDispatcher.OK")
   :to ("dss_Assemble.OK")
)


(provides
   :from ("updateDSSDBs.FAIL")
   :to ("bldShutdownMsg.trigger")
)


(provides
   :from ("runQueueModel.FAIL")
   :to ("bldShutdownMsg.trigger")
)


(provides
   :from ("advMMAssemble.OK")
   :to ("taskTerminator.finished_Assemble")
)
```

```
(deftask
   :name ("dss_SimDispatcher")
   :parent ("NONE")
   :children ("listenAssembleMsg"
              "collectSimDispatch"
              "processSimYears"
              "bldSimRunMsg"
              "achMoveSimRun"
              "bldSimDoneMsg"
              "achFacCleanup")
   :parameters ("NOPROV")
   :provisions ("NOPROV")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(defaction
   :name ("collectRefugeMsgs")
   :parent ("dss_Assemble")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("finished_BreedingChkDB"
              "finished_FlywayChkDB"
              "finished_HabitatChkDB")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(NLT
   :name ("achRefFly")
   :parent ("dss_Assemble")
   :parameters ("NOPROV")
   :provisions ("KQML")
   :outcomes ("OK")
)


(NLT
   :name ("achFacCleanup")
   :parent ("dss_SimDispatcher")
   :parameters ("NOPROV")
   :provisions ("KQML")
```

```
   :outcomes ("OK")
)


(defaction
   :name ("taskTerminator")
   :parent ("_Startup")
   :children ("NONE")
   :parameters ("NOPROV")
   :provisions ("finished_Tracker"
            "finished_Assemble"
            "finished_Sim")
   :outcomes ("OK" :behavior_profile (:cost "NONE" :quality "NONE" :duration "NONE" :density
"NONE")
                  )
   :deadline ("0")
   :earliest_start_time ("0")
   :caf ("AND")
)


(utility_function
  (label Information_Gather)
  (goodness_quality_slider 1.0)
  (goodness_cost_slider 0.0)
  (goodness_duration_slider 0.0)
  (threshold_quality_slider 0.0)
  (threshold_cost_slider 0.0)
  (threshold_duration_slider 1.0)
  (quality_threshold 0)
  (cost_limit 5)
  (duration_limit 600)
  (uncertainty_quality_slider 0.2)
  (uncertainty_cost_slider 0.7)
  (uncertainty_duration_slider 0.1)
  (threshold_certainty_quality_slider .1)
  (threshold_certainty_cost_slider   .2)
  (threshold_certainty_duration_slider   .3)
  (quality_certainty_threshold  1)
  (cost_certainty_threshold  2)
  (duration_certainty_threshold  3)
  (meta_goodness_slider 1.0)
  (meta_threshold_slider 0.0)
  (meta_uncertainty_slider 0.0)
  (meta_uncertainty_threshold_slider 0.0)
)
```