JIN WANG
External Heterogeneous Information Source Management Agents
(Under the direction of DON POTTER)

NED-2 is a robust, full service Intelligent Information System designed to provide decision support for forest ecosystem management. Integrating growth and yield models into NED is one of the important elements in developing NED. NED-2 uses external heterogeneous information source management (EHISM) agents to permit communication between external sources and NED-2. FVS is the first simulation model integrated in NED-2. A meta-knowledge base is developed, so the simulation agent can use it to set up and execute the FVS models. This thesis will briefly describe the NED-2 agent-based blackboard architecture, and discuss the design issues explaining the integration of NED-2 and FVS.

INDEX WORDS: Intelligent information system, Blackboard architecture, Agents,
Meta- knowledge, FVS, PROLOG

EXTERNAL HETEROGENEOUS INFORMATION SOURCE MANAGEMENT AGENTS

By

JIN WANG

B.S., Tianjin University, P.R. China, 1983

M.S., Tianjin University, P.R. China, 1987

A Thesis Submitted to the Graduate Faculty of The University of Georgia

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2002

EXTERNAL HETEROGENEOUS INFORMATION SOURCE MANAGEMENT AGENTS

By

JIN WANG

Approved:

Major Professor: Don Potter

Committee: Donald Nute
Charles Cross

Electronic Version Approved:

Gordhan L. Patel
Dean of the Graduate School
The University of Georgia
August 2002

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to Dr. Don Potter, my major professor, for his constant support, concern, and encouragement and invaluable advice throughout this challenging thesis project. I wish to thank Dr. Donald Nute, my committee member, for his constructive suggestions and help. I also wish to thank Dr. Charles Cross, for his advice and serving on my committee.

I thank all my friends and classmates, especially Frederick W. Maier, Yunxiu Xu, and Yali Zhao, for their help to this thesis.

I thank Dr. H. Michael Rauscher, Dr. Mark J. Twery, Scott Thomasma and Pete Knopp, members of the Forest Service, for their help and kindness. I thank the USDA Forest Service for supporting me.

I thank my parents, my husband Ruihua Liu, and my daughter Xin Xin, for their continuous support and encouragement.

This research was supported by funds provided by the U. S. Department of Agriculture, Forest Service, Northeastern Research Station.

TABLE OF CONTENTS

Page

CHAPTER 1

INTRODUCTION

This thesis discusses the design issues of external heterogeneous information source management (EHISM) agents in the NED-2 project, and describes the implementation of the forest vegetation simulation agent.

1.1 NED Project

NED is a robust, full service Intelligent Information System for the sustainable management of forest lands. It is designed to help owners and forest managers plan and achieve wildlife, ecology, water, landscape, and timber production goals (Nute et al. 1999).

NED is an acronym for "Northeast Decision Model". The NED project began in 1987 among researchers within the Northeastern Forest Experiment Station (Twery et al. 2000). One of the objectives the researchers put forward was to develop a computer system that would integrate all the previously independently produced growth and yield models developed by scientists within that station. A primary objective of NED was to develop a single, easy-to-use system to provide summary information and expert prescriptions for any forest type in the northeastern United States (Twery et al. 2000). Now its original applicable scope has expanded. NED-1, the first version of NED, has

functionalities of data acquisition, goal selection, and goal satisfaction analysis (Nute et al. 1999). NED-2, the current version under development, will simulate user's treatment plans by using growth and yield models. The results of the simulation and goal satisfaction analysis are displayed with several visualization tools, such as on-screen displays, hypertext reports, and geographical information systems. Based on a review of these analyses, the user can make a final decision upon the management plan (Nute et al 2002).

Integrating growth and yield models into NED is one of the important elements in developing the NED project. In order to achieve a desired goal, a user may construct several plans and compare the results of each plan. The growth and yield models are very useful tools for plan simulation. They give a projection of conditions into the future under alternative management options.

Some existing simulation models, such as SILVAH and the many variants of FVS, have been or will be integrated into NED. FVS (Teck et al. 1996) is a Forest Vegetation Simulator that projects the growth of forest stands under a variety of conditions. SILVAH (Marquis et al. 1992), for SILViculture of Allegheny Hardwoods prescription system, helps forest managers make treatment decisions. Usually, these models were developed independently, which often causes the problem of heterogeneity. They have been developed using different languages, they run on different hardware systems, they have different input and output formats, and different control code must be invoked for each. They are typically monolithic, stand-alone legacy systems. It is very demanding to deal with heterogeneous components and integrate them into NED.

1.2 NED-2 Agent-based Blackboard Architecture

For many years, the blackboard architecture (Craig 1995, Newell) has been successfully used in different application areas. This approach includes a group of experts, each of them having knowledge in a specific field. They are asked to solve a complex problem that cannot be done alone by any of them. The experts cooperate with each other through a blackboard. When an expert is solving a part of a whole problem and writing the result or a request on the blackboard, no other experts are allowed to write there at the same time.

NED-2 is a blackboard system with semi-autonomous intelligent agents. Its blackboard integrates a Microsoft Access database and a set of Prolog clauses. Inventory data and other information are stored in the database. NED-2 includes a user interface, databases, simulators, knowledge bases, hypertext documents, geographical information systems and visualization tools. Simulators and other external modules are integrated into NED-2 via their intelligent agents (Figure 1) (Nute et al. 2002).

In NED-2, each agent makes a contribution to the problem-solving process. Agents communicate with each other through the NED blackboard. Tasks that need to be done are posted on the blackboard. Agents also post most of the intermediate results of their activities on the blackboard. Agents watch the blackboard continually. The information on the blackboard will prompt an agent to do some work. If an agent performs some task listed on the blackboard, it will erase that task from the task list. An agent may place a report that the task has been performed on the blackboard after completing the task. If an agent begins a task, then discovers that something needs to be

done before the task can be carried out but that is beyond its capability, it can put the new task on the blackboard and wait until another agent performs it before completing its original task.

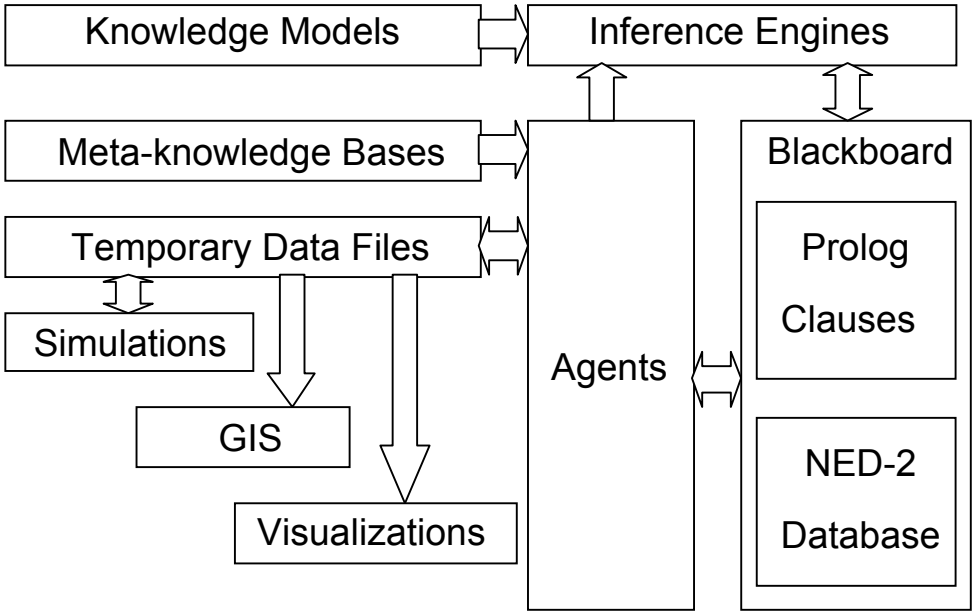

Figure 1: The NED-2 architecture

1.3 External Heterogeneous Information Source Management Agents

NED-2 uses external heterogeneous information source management (EHISM) agents to integrate various resources. In order to add new functionalities to the NED system, NED-2 needs to integrate additional growth and yield models or other autonomous, heterogeneous information sources. Since each source is designed to perform a particular task in a specialized environment, we must deal with these

heterogeneous environments. First, the data format of the information source may need to be translated in order to communicate with the other sources. Second, the information source that is developed in a different programming environment should be invoked by appropriate control codes. We might also need to consider how to perform transparent processing so that the users can just focus on their tasks without having to consider where the results will be gotten.

In NED-2, the earlier framework of integrating external sources was based on a wrapper approach (Wang et al. 2002). Each heterogeneous information source was integrated via a customized wrapper. The wrapper provides a communicating layer between the NED system and the external heterogeneous model. The main disadvantage of the wrapper approach is that we must develop a specific wrapper for each source even though some sources need similar execution procedures. That is inconvenient for developing and maintaining software.

The NED EHISM agents approach overcomes this shortcoming. An EHISM agent is a multi-source controller. Each EHISM agent has a meta-knowledge base. By consulting the meta-knowledge base, the EHISM agent knows when and how to access related sources. Thus, integration of an additional information source becomes easier and faster. One of the EHISM agents in NED-2 has been developed. It is called the simulation agent and it intelligently integrates various forest simulators. The first simulator integrated into NED-2 is FVS. It is what this thesis will focus on.

The background of this thesis has already been provided. Next, the design issues of NED-2 EHISM agents will be discussed, and the implementation of the forest simulation agent will be described in detail. The second chapter will briefly describe the

theory of Intelligent Information Systems (IIS), and give a description of several existing IIS architectures. The third chapter will discuss the design issues of NED-2 EHISM agents, including the behaviors of the EHISM agents and the content of a meta-knowledge base. The fourth chapter will describe how the simulation agent for the FVS simulator is implemented in detail. The last chapter will give conclusions and discuss future direction the NED project might take.

CHAPTER 2

INTEGRATING EXTERNAL HETEROGENEOUS INFORMATION SOURCES

This chapter will provide an overview of Intelligent Information Systems theory and a brief description of its several architectures.

2.1 Intelligent Information System

We are witnessing a rapid increase of information sources. They could (1) be multimedia (video, sound, images, and text), (2) be stored in diverse formats (structured formats, like databases; non-structured formats, like flat files; or semi- structured formats, like HTML files), (3) have different data meanings across sources (e.g., grade point average might be based on a 4.0 system in one database and a 10.0 system in another), (4) differ in temporal and spatial dimension, and/or (5) be application programs. A large number of diverse information sources bring us great power. However, people have to deal with an important issue, that is, how to efficiently make desired information sources available in a unified information system since each information component is usually autonomous. There are several kinds of autonomy ([Ozsu 99]).

- Design autonomy. An information source is independent from others in its design, which means it has an independent data model both in semantic and syntactic aspects, and in its design change, which could happen at any time.

- Communication autonomy. An information source can communicate with an other information system independently. It can be added to or removed from the system at any time.
- Execution autonomy. An information source works independently to execute and schedule all incoming requests.

An Intelligent Information System (IIS) is an integrated information system that provides uniform access to multiple autonomous information sources.

One of the main tasks of an Intelligent Information System is to resolve any heterogeneity between the sources. There are several facets to the heterogeneity of information in systems (Bird 1993): syntactic, control, and semantic. Syntactic heterogeneity exists if knowledge and data are represented using different knowledge representation formats and data definition formats. Control heterogeneity comes from the many reasoning mechanisms for intelligent systems including induction, deduction, analogy, case-based reasoning, etc. Semantic heterogeneity denotes the variety of the meaning and interpretation of knowledge and data.

Transparency is considered as the ideal goal of integration. A perfectly intelligent information system would let users communicate with only one central, locally running, and homogeneous information system so that the users can just focus on their tasks without having to consider where the results will be gotten and which query process will be involved, such as a data retrieval, an inference, a computational method, a problem solving module, or a combination of these.

An IIS is viewed as "composed of a unified knowledge base, database, and model base" (Potter et al 2000). The knowledge base contains domain knowledge and meta-

knowledge. It describes the existing sources' functionality, their relationships, and the details for access. The model base includes various models; decision models support effective decision-making for user queries, simulation models project the future to help users evaluate alternative decisions, and visualization models create visual displays for different scenarios. The use of artificial intelligence techniques, such as agents, knowledge representation, planning, and reasoning, enables the information system to perform information support tasks intelligently.

2.2 Several Integrating Approaches

There are many existing approaches that provide intelligent integration of information, such as a federated database approach (Heimbigner and McLeod 1985; Sheth and Larson 1990), a hierarchical mediator approach (Wiederhold 1992; Roth and Scharz 1997), and an ontology-based semantic approach, (Levy et al. 1996; Cheung et al. 1996). In this section, each of these systems will be discussed briefly.

2.2.1 Federated Database Approach

In a federated database system, resources are created, administered, and enhanced independently. The system provides unified access to other information sources. Each individual database has a global abstraction while keeping autonomy of each component. There are two types of federated database systems: tightly coupled and loosely coupled. In the tightly coupled federated database, the integration of different sources is done in advance, and the user can only query or update the databases in a predefined way. In the

loosely coupled federated database, the integration is more dynamic, and end-users can interact with the individual component databases and create their own federated schema.

The shortcomings of the federated database approach are that they are not very active or scalable, and they focus on only structured information sources like databases.

2.2.2 Hierarchical Mediator Approach

Similar to the federated database approach, the hierarchical mediator approach is a structural approach. A hierarchical mediator system involves decomposing query creators (called mediators) and information providers (called wrappers). It has better semantic-level services and interoperability.

The mediator plays a major role in translating and decoding user global queries into multiple local queries that can be supported by the individual information resources. Intelligent as it is, it knows how the query is best covered by the candidate data sources. In general, the mediator decomposes complex queries to an appropriate level, executes the sub-queries on its data sources, collects the results and returns them back to the higher-level mediator. The mediator can use other mediators and/or wrappers as data sources.

The wrappers manage the external heterogeneous sources, which are wrapped to provide an appropriate interface that affords transparency in communication, data representation, and system environment. The function of a wrapper is to accept the caller's queries and data, convert them into the target-source format, and then pass them on to the target-source for execution. After execution, the results are captured by the wrapper that will transfer them into the format of the caller.

Some "classic projects" with a hierarchical mediator architecture are Garlic (Haas et al. 1997), TSIMMIS (Papakonstatniou et al. 1996), HERMES (Adali and Emery, 1995), and COIN (Bressan et al. 1998). Garlic is capable of integrating different data sources. TSIMMIS focuses on the integration of semi-structured and unstructured data sources. HERMES emphasizes integrating knowledge bases and reasoning systems. COIN is mainly used in querying distributed information sources.

The disadvantage of the hierarchical mediator approach is that each system can only respond to limited user queries since its mediator-wrapper structure is fixed. If new information sources need to be integrated into the system, the corresponding wrappers (and mediators) must be created, and all the related higher-level mediators must be modified too.

2.2.3 Ontology-based Semantic Approach

Unlike the hierarchical mediator approach, the ontology-based semantic approach separates the source descriptions from the source schema. An ontology-based semantic system is a knowledge representation and reasoning system. It uses a common data model to describe the information sources, like their contents, constraints, and capabilities. There are several different architectures, such as the description logic-based architecture, and ontology agent-based architecture.

In a description logic-based system, the contents and properties of information sources are represented based on description logic and rules using a powerful description language. The descriptions and relations of information sources are stored in a knowledge base. The integration is performed by the query planner. When processing a user query,

the planner uses the knowledge base to find sources relevant to the query, reformulates the query, executes the sub-queries, and re-assembles the results. Some "classic projects" with a description logic-based architecture are the Information Manifold (Levy et al. 1996), and SIMS (Arens et al. 1996). Information Manifold focuses on integrating web-based data sources. SIMS can integrate databases and application programs.

Some integration systems use an ontology agent-based architecture. InfoSleuth (Bayardo et al. 1997), for example, uses multiple ontologies to represent data in the same information source. Some ontologies describe the knowledge about the relationships of the data stored by resources. Some ontologies describe agents' knowledge and their relationships. The multiple ontologies provide agents with capturing and reasoning about information content, identifying the relevance of an information source, and specifying the agent infrastructure. The broker agent accepts the user query, interacts with the task execution agent, who will decompose queries, route queries to the appropriate resource agent, and reconstructs the results. The broker agent stores and receives information from all agents and returns the final result to the user. InfoSleuth focuses on integrating information available both in corporate networks and in external networks.

Another typical ontology-based system is OBSERVER (Mena and Illarramendi, 2001). In OBSERVER, the contents of each data source are described by one or more ontologies expressed in a system based on description logics. The system uses ontological inferences to determine relevant sources and translates description logic expressions to the local query languages of the relevant data sources. The goal of OBSERVER is to support integration of various local and remote information sources.

The ontology-based semantic approach provides a very good way for semantic integration while maintaining the autonomy of an individual information source. Because of the independence between the system framework and the component schema and the explicit description of their relationships, the system can evolve easily.

CHAPTER 3

THE NED-2 EHISM AGENT

As mentioned, the earlier framework of integrating external sources into NED was based on a wrapper layer that provides communication between the NED system and the external heterogeneous model. Like the wrapper in a hierarchical mediator approach, each wrapper in the NED-2 system was designed for a specific model. The wrapper cannot be shared by different sources.

The new NED version adopts the EHISM agent as the intelligent integrating manager. As a multi-source controller, an EHISM agent relates a single information source or a set of sources that have similar functionalities and invoking procedure. Each EHISM agent has a meta-knowledge base. Meta-knowledge is knowledge about knowledge. An EHISM agent will consult the meta-knowledge base to know when and how to use a knowledge source or a knowledge tool. Thus, all procedural knowledge for calling external sources is written into the EHISM agent and specific knowledge related to the individual source is stored in the declarative meta-knowledge base.

In NED-2, Prolog, a high-level logic programming language, provides the primary implementation platform for agents, knowledge bases, and inference engines. The user interface is implemented in Microsoft Visual C++. The databases are implemented in Microsoft Access. To perform its tasks, an agent may need to retrieve and update the core data on the blackboard. The Prodata LPA Prolog interface is used to

implement the database access. The Prodata interface provides a tight coupling between LPA Prolog for Windows and all Database Management Systems (DBMSs) that support a sufficient level of Open Database Connectivity (ODBC) compliance to be used with Microsoft ODBC 2. Prodata allows database tables to be accessed from Prolog as though they existed within Prolog's environment as unit ground clauses (facts). All routine database functions such as creating tables, updating/retrieving records, and/or whatever may be achieved via normal Structured Query Language (SQL) commands. In NED-2, instead of using Prodata directly, another method is implemented to access databases also (Maier 2002). It uses various PROLOG predicates that are built on top of the normal Prodata routines to allow easier communication with databases. The test results show that it is much more efficient than Prodata. Currently, the EHISM agent uses Prodata to access the databases. When an EHISM agent needs to get something from the blackboard or put something onto the blackboard, it calls a particular predicate designed to retrieve or update information. If information is already present as facts on the blackboard, it is easily accessed or changed. However, if information is stored in the core database, a Prodata SQL query will be constructed to select, update, add, or delete the data values in the database.

3.1 An Overview for the NED-2 EHISM Agent

The main functionalities of an EHISM agent are deciding which source will be called, translating data between the NED format and the format of the external source,

and invoking the external sources. Figure 2 shows an abstract view of the NED-2 EHISM agents and their environment.

For an EHISM agent, the following steps describe the process of invoking an external source.

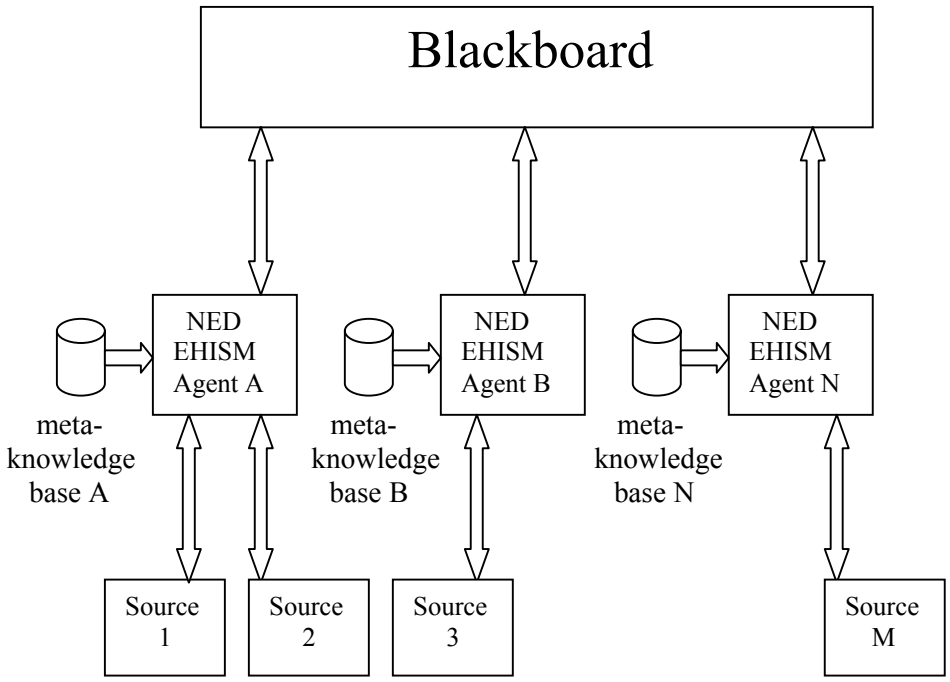

Figure 2: The NED-2 EHISM agents and their environment.

Step 1: Lets user pick an external source or recommends to the user which source is appropriate if the agent manages a set of external sources.

Step 2: Checks whether or not all the conditions that are needed for the external source to function are satisfied.

Step 3: Gets input data from the blackboard, then converts them into the format of the external source.

Step 4: Calls the external source.

Step 5: Transforms the output of the external source into the NED format, and then puts them back on the blackboard.

Since each source is designed to perform a particular task, different sources require different formats for input files and generate output files in different formats. They also require different control codes. Information about these requirements is provided by a meta-knowledge base.

3.2 Meta-knowledge Base

The meta-knowledge base of the EHISM agent stores the declarative knowledge for each external source. It includes answers to the following questions.

- What kind of environment the source is suitable for?

  For example, the simulator FVS northeast variant will give a very good simulation result for simulating a forest in the northeastern U.S., but it is not suitable for a forest in Georgia.

- What conditions must be satisfied for the model to function?

  For example, besides the tree level data, the FVS simulator needs data at the stand level, especially the value of stand site species, stand site index, and stand year of origin.

- What commands or control codes are needed to control execution of a simulation?

The source may be called through its interface, by a command, or by a set of commands.

- What kinds of input and output formats does the external source use, and what is the relationship between the formats used by the external sources and the native NED data format?

  The data file of the external source or caller could be a flat file, a database, or an HTML file. For NED-2, the core database on the blackboard will provide the data for external sources. The input and output formats of the FVS simulator are flat files. The meta-knowledge will provide the relationship between the NED and the external source data formats.

- What functions can be used to translate between the NED and the external source data formats?

  For example, the format of the FVS tree record input file is

```
tree_records_format ((fvs,_),columns,[
        (cluster,[increment(1),pad(front,4,'0')]),
        (tree_id,[increment(1),pad(front,3,'0')]),
        (stems_per,[places(0),pad(front,6,' ')]),
        (tree_alive,[translate([(0,8),(1,1)])]),
        (species,[species_code(ned2,alpha),pad(back,3,' ')]),
        (dbh,[places(1),pad(front,4,' ')])
        ]).
```

So, a set of functions, such as the tool for increasing the value of a variable (*increment /1*), the function for attaching leading or trailing place-holder symbols to a variable (*pad /3*), the function for translating trees' species codes (*species_code /2*), will be provided by the meta-knowledge base.

3.3 The Advantage of the NED-2 EHISM Agents

The integration architecture of the NED-2 EHISM agents is similar to the ontology-based semantic approach that separates the source description from the source schema. The NED-2 EHISM agent stores the source description in its meta-knowledge base. The most important advantage of the EHISM agent is that it makes the integration of external heterogeneous sources easier, faster, and more flexible. In the earlier version of NED, a mediator-wrapper architecture is adopted to integrate external source. We need to build a wrapper for each source. The wrapper cannot be shared by different sources even though several sources have similar invoking processes and data requirements. With the EHISM agent architecture, when a new similar source needs to be integrated into the NED-2 system, we just focus on adding specific knowledge that relates the new source into the meta-knowledge base. This method allows the NED software to have a succinct program structure. Also, it will be more convenient for the evolution and manipulation of the NED software.

CHAPTER 4

THE NED-2 SIMULATION AGENT AND FVS

The simulation agent is one of the EHISM agents in NED-2. It is designed to manage a set of growth and yield models. FVS is the first simulation model integrated in NED-2.

4.1 Background on FVS

FVS is a family of forest growth simulation models. FVS uses common forest inventory data and a simulation model to project the growth of forest stands under a variety of conditions. It started in the 1970s and is used in the U.S. forest management field. There are now more than 20 variants in FVS. Variants of FVS provide growth and yield models for specific geographic regions of the United States. FVS is written in the FORTRAN language and runs on PC and UNIX workstations (Teck et al 1996).

FVS requires two types of input files to run: a tree record file (*.fvs), which is an inventory of the stand the user wants to model, and a keyword file (*.key), which provides FVS with commands that control the simulation. The most important FVS output file for NED is the tree list output file (*.trl), which contains simulation results in tree level detail.

There are two ways to run FVS. One is to run FVS under its native DOS environment. Users need to create the two input files by themselves first. The other way is by using SUPPOSE, a graphical user interface for FVS, to create the key file, and then run FVS. SUPPOSE needs three input files: tree record file (*.fvs), suppose stand list file (*.slf), and suppose locations file (*.loc). Unlike NED-1 which uses SUPPOSE to run FVS, in NED-2, we completely by-pass the SUPPOSE interface. The simulation agent will create FVS input files and run FVS. The data that are included in the input files will be retrieved from the NED-2 database and treatment plan that are on the blackboard.

4.2 FVS, Simulation Agent and NED-2

The first two FVS variants integrated with NED-2 are the Northeast variant (NE) and Southern variant (SN). The simulation agent is responsible for managing these models.

To simulate growth and yield of forest stands, FVS needs stand inventory information and user management plans. The inventory data are stored in the core database that is on the blackboard. The NED user develops management plans by using a drag-and-drop planning screen. Treatments are selected and dropped onto individual stands. Management plans are stored in the core database. A set of default treatment parameters and user specified treatment parameters are stored in the treatment parameter database that is on the blackboard also. The simulation agent will retrieve the information it requires from the core database and treatment parameter database, create the FVS keyword input file and FVS tree record input file, run FVS, and then convert the FVS tree

list output file into the NED-2 format and insert the results back into the NED-2 database. Figure 3 shows NED/FVS integration.

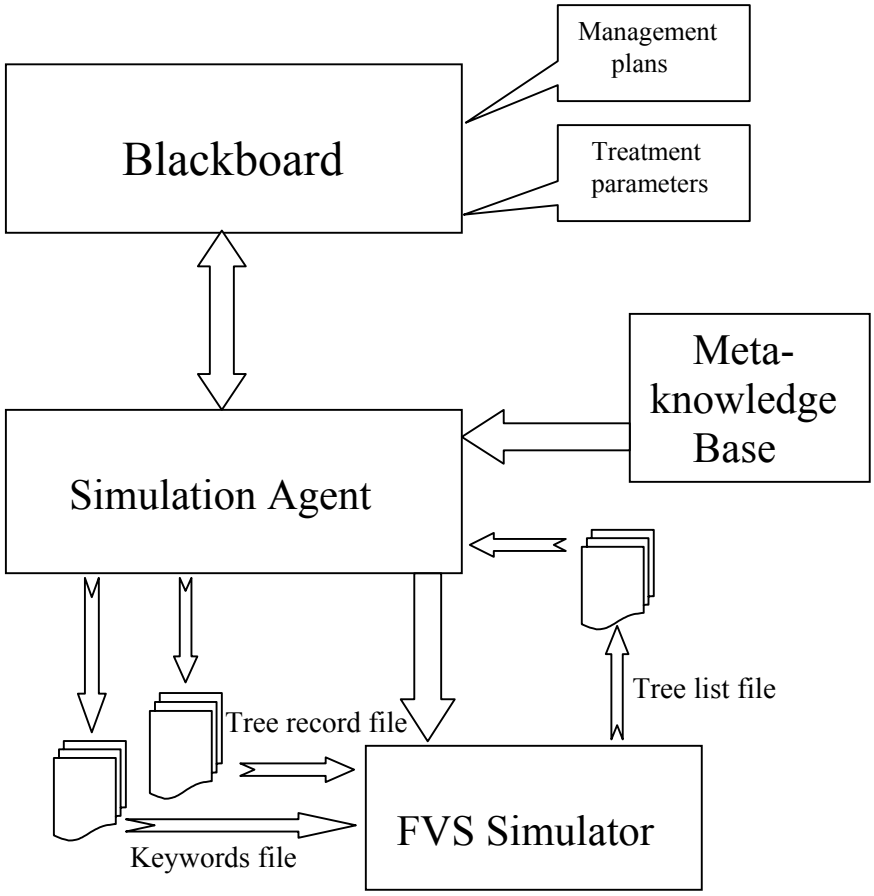

Figure 3: NED/FVS integration

A meta-knowledge base related with FVS was developed. It provides the simulation agent with the knowledge needed to translate data between the NED-2 format and the FVS format. This meta-knowledge base also tells the simulation agent what control codes the FVS simulator understands.

4.3 Implementation Details

4.3.1 The Simulation Agent

The simulation agent plays two roles: baseline generation and plan simulation. Figure 4 shows the procedure for the baseline generation. Figure 5 shows the procedure for the plan simulation.

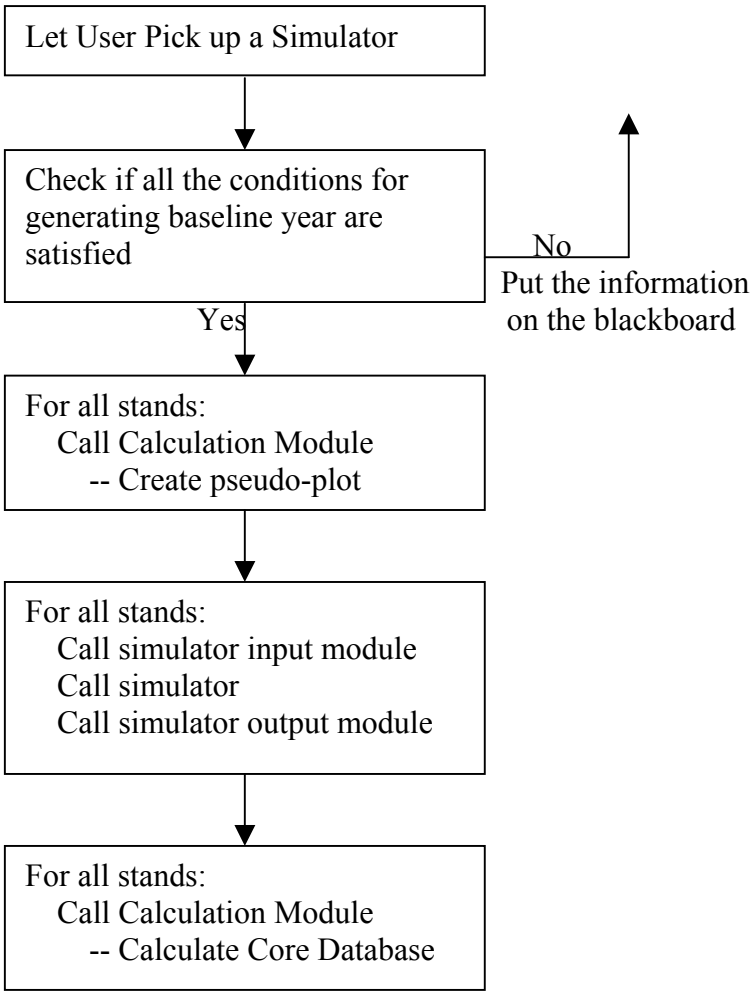

Figure 4: The procedure for baseline generation

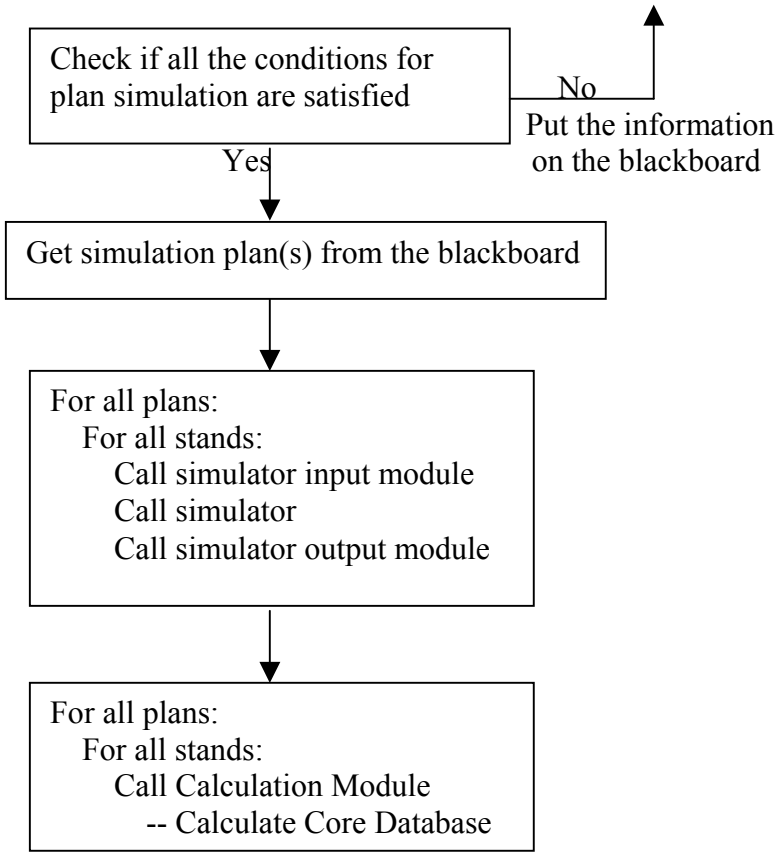

Figure 5: The procedure for plan simulation

NED-2 needs to create baseline data for all stands because all treatment plans must start in the same year for every stand. Users may take inventory for each stand in different years, and the last inventory information for every stand may not be in the same year. To create data for the same common baseline year for all stands, NED-2 will simulate data for those stands where the baseline inventory information is not available. When the user selects a baseline year and asks NED-2 to generate the baseline, the simulation agent will process the simulation. It will get the baseline year value from the

blackboard, pop-up a window to let the user pick a simulator, and determine all the conditions that must be satisfied for the simulator to function. For example, the baseline year must be selected, and there should be at least one stand in the core database. Also, FVS needs stand information such as stand year of origin, site index species, and site index. If a condition is not satisfied, the simulation agent will post a request on the blackboard. Once all conditions are satisfied, it will check every stand in the specific management unit under consideration. If the last inventory year for a stand is the baseline year, the simulation agent does not need to take any action. However, if the inventory year is different from the baseline year, it will run a simulator to grow the stand from the data for the last inventory up to the baseline year.

To simulate a user's plan, the simulation agent will first retrieve the user's treatment plan from the blackboard, and then run a simulator. A treatment plan includes which stands will be simulated, how long the simulation will run, how to implement any treatments, and when to treat the stand. Currently, several kinds of treatment, such as light thinning, medium thinning, and clear cut can be simulated by the FVS simulator. Also, before running FVS for simulating a treatment plan, the agent needs to check whether or not a plan is selected, and whether or not the baseline is created.

The simulation agent includes three main modules: the NED-2 calculation module, the simulator input module, and the simulator output module.

The function of the NED-2 calculation module is to run a dynamic linked library called NEDcalc.dll. In NED-2, trees are identified by the stand, cluster, and plot that they belong to. According to the tree's diameter at breast height (DBH), tree data are stored in the overstory observation table and understory observation table of the core database. But

in FVS, trees are identified by the stand and plot. To facilitate the retrieval from the NED-2 database, the simulation agent runs the NED calculation module before calling the simulator input module. The NED-2 calculation module then runs several routines that will create a "pseudo_stand" combining all trees in a cluster into one "pseudo_plot". After running the simulator output module, the simulation agent calls the NED-2 calculation module again. This time the NEDcalc.dll shuffles the simulation results into the overstory observation table and understory observation table, and then computes the other tree data at both the tree level and stand level.

The simulator input module sets up the simulator input files. For simulator FVS, the module takes the stand data, plot data, and tree data from the blackboard and creates a keyword file and a set of FVS tree record files. The simulation meta-knowledge base tells the simulator input module what format these files have and how to create them.

The simulator output module takes the simulation results as input and inserts appropriate data back to the blackboard. For simulator FVS, the module selects each line in the tree list file. It checks if the data record for this tree is duplicated. If it is, the record is ignored and processing continues with the next tree record. If it is not, the module uses the simulation meta-knowledge base to convert the record to the NED-2 database format and puts it into a new "snapshot" in the NED-2 database. A snapshot in the NED-2 database is a set of records representing simulated data for a single stand in a given year for a selected treatment plan.

To integrate a new simulator, we only need to tell the agent for what situation this simulator is suitable, how the corresponding input module and output module work, and

how to call the simulator. All of this knowledge is stored in the simulation meta-knowledge file.

4.3.2 The Simulation Meta-knowledge Base

Currently, the simulation meta-knowledge base includes the meta-knowledge for the variants SN and NE of FVS.

4.3.2.1 Creating A Tree Record Input File

Each stand has its own tree record file. Each tree has a record in the tree record file. Each record contains cluster identification (column 1 - 4), tree identification (column 5 - 7), tree stems per acre (column 8 - 13), tree history code (column 14), tree species code (column 15 - 17), and tree DBH (column 18 - 21). Each record has a fixed width. Here is a tree record file.

```
0001001    10.1SM  2.1
0001002    10.1AB  2.2
0001003    10.1SM  7.3
0001004    10.1SM  2.8
0001005    10.1AB  2.1
0001006    10.1AB  3.1
0001007    10.1SM  7.0
0001008    10.1SM  5.7
0001009    10.1SM  8.3
0001010    10.1OH  5.4
0001011    10.1AB  2.6
0001012    10.1AB  2.6
0001013    10.1SM  7.2
0001014    10.1SM  3.3
0001015    10.1SM  8.9
0001016    10.1SM  7.8
0001017    10.1SM  3.6
```

The meta-knowledge tells the simulation agent to get inventory data from the overstory observation table and understory observation table in the core database. The format of each tree record is:

```
tree_records_format ((fvs,_),columns,[
        (cluster,[increment(1),pad(front,4,'0')]),
        (tree_id,[increment(1),pad(front,3,'0')]),
        (stems_per,[places(0),pad(front,6,' ')]),
        (tree_alive,[translate([(0,8),(1,1)])]),
        (species,[species_code(ned2,alpha),pad(back,3,' ')]),
        (dbh,[places(1),pad(front,4,' ')])
        ]).
```

Here, the cluster identification number and tree identification number are increased by 1. The tree species code is converted from NED-2 code to corresponding alpha code. The function *pad/3* and the function *places/1* are used to get correct data width and decimal place.

4.3.2.2 Creating A Keyword Input File

The keyword file tells FVS what to do with the data. It consists of a series of FVS keywords, some of which have parameters and some of which do not. Here is a keyword file for stand growth.

```
!!Suppose
!!Top
Comment
Starting year for simulation is 1999
Ending year for simulation is   2002
Min and Max inventory years are 1999 1999
Common cycle length is          3
End
!!End
!!Stand
StdIdent
001      Stand 001 at Bent Creek
!!SK
```

```
Screen
InvYear         1999
ModType            1
StdInfo                  M231Aa         84         80         20       1810
Design          -1.0        1.0        0.1          5
SiteCode          SM         60
!!End
!!TK
TimeInt                     3
NumCycle           1
!!End
!!C
!!SW
!!P
!!K
NoTriple
!!End
!!C
!!SW
!!P
!! max
!!K
TreeList           0         3.           1          1          0          0          0
!!End
!!TR
Open               2
0000105.fvs
TreeData           2          1
Close              2
!!End
SPLabel
   All, &
   !StandsInNoDefinedGroup
Process
!!EndStand
STOP
!!G
!!S
!!Subset
!!G
!!End
```

The format of the keyword file for baseline generation is:

key_file_format((fvs,Variant),baseline,Scenario,Snapshot,Stand,
Ending_year,Starting_year,Cycle_len,Time_int,[
'!!Suppose',nl,

```
'!!Top',nl,
'Comment',nl,
'Starting year for simulation is ',Starting_year,nl,
'Ending year for simulation is   ',Ending_year,nl,
'Min and Max inventory years are ',Starting_year,' ',Starting_year,nl,
'Common cycle length is ',[Cycle_len,[pad(front,10,' ')]],nl,
'End',nl,
'!!End',nl,
'!!Stand',nl,
'StdIdent',nl,
[Stand,[increment(1),pad(front,3,'0')]],
        '     Stand ',[Stand,[increment(1),pad(front,3,'0')]],
        ' at ',MU_Name,nl,
'!!SK',nl,
'Screen',nl,
'InvYear        ',Starting_year,nl,
'ModType          1',nl,
'StdInfo   ',[LocationCode,[pad(front,10,' ')]],
        [Habitat,[pad(front,10,' ')]],
        [YearsSinceOriginal,[pad(front,10,' ')]],
        [Aspect,[pad(front,10,' ')]],
        [Slope,[pad(front,10,' ')]],
        [Elevation,[pad(front,10,' ')]],nl,
'Design    ',[Baf,[pad(front,10,' ')]],
        [Size,[pad(front,10,' ')]],
        [BreakPointDBH,[pad(front,10,' ')]],
        [TotalCluster,[pad(front,10,' ')]],nl,
'SiteCode  ',[Spp_ned1,[pad(front,10,' ')]],[Index,[pad(front,10,' ')]],nl,
'!!End',nl,
'!!TK',nl,
'TimeInt           ',[Time_int,[pad(front,10,' ')]],nl,
'NumCycle          ',Cycles,nl,
'!!End',nl,
'!!C',nl,
'!!SW',nl,
'!!P',nl,
'!!K',nl,
'NoTriple',nl,
'!!End',nl,
'!!C',nl,
'!!SW',nl,
'!!P',nl,
'!! max',nl,
'!!K',nl,
'TreeList        0       3.       1       1       0       0       0',nl,
```

```
'!!End',nl,
'!!TR',nl,
'Open            2',nl,
[Stand,[increment(1),pad(front,5,'0')]],
        [TotalCluster,[pad(front,2,'0')]],',fvs',nl,
'TreeData        2       1',nl,
'Close           2',nl,
'!!End',nl,
'SPLabel',nl,
'  All, &',nl,
'  !StandsInNoDefinedGroup',nl,
'Process',nl,
'!!EndStand',nl,
'',nl,
'STOP',nl,
'!!G',nl,
'!!S',nl,
'!!Subset',nl,
'!!G',nl,
'!!End',nl]).
```

Here, the keyword *StdInfo* has six parameters, including stand location code, stand habitat type code, stand age, stand aspect, stand slope, and stand elevation. The keyword *Design* has four parameters, including a basal area factor, inverse of the small-tree fixed area plot, breakpoint DBH between large-tree and small-tree sample design, and number of plots in the stand. The width of each parameter is 10.

If the keyword file is used for plan simulation, except all the keywords for baseline generation, the corresponding treatment keyword(s), such as THINBBA, will be added in the keyword file. Here is an example of format information for an FVS keyword.

```
[treatments(Scenario,
        Stand,
        [[keyword,[pad(back,10,' ')]],
        [treatment_year,[pad(front,10,' ')]],
        [residual_ba,[pad(front,10,' ')]],
        [efficiency,[pad(front,10,' ')]],
```

[min_dbh,[pad(front,10,' ')]],
[max_dbh,[pad(front,10,' ')]],
[min_ht,[pad(front,10,' ')]],
[max_ht,[pad(front,10,' ')]])].

Here, the keyword has seven parameters, including treatment year that treatment is scheduled, residual basal area, treatment efficiency, smallest DBH to be considered for removal, largest DBH to be considered for removal, shortest tree to be considered for removal, tallest tree to be considered for removal. The width of keyword and its parameters is 10.

A user must provide the treatment keyword parameters required by the preferred simulator for all treatments that would be included in the treatment plans. All the treatment keywords and their parameters are stored in the treatment parameter database, which is on the blackboard (Figure 6). If a user's treatment plan is "for stand 10, do light thinning in 2010 and clear cut in 2015", the meta-knowledge will tell the simulation agent to go to the treatment parameter database to get all the information relating "light thinning and clear cut", and then convert them to the format of the treatment keyword:

```
THINBBA      2010       120        1.0          1          7          0        999
THINDBH      2015       1.0      999.0        1.0        ALL          0          0
```

4.3.2.3 Converting A Tree List Output File

NED needs to get tree DBH, tree stems per acre, and tree species code from the results of a simulation. For simulator FVS, this information is in the FVS output tree list file.

The FVS tree list file includes a header and a set of tree records for each simulation cycle (Figure 7). In the meta-knowledge base, the tree record is described as:

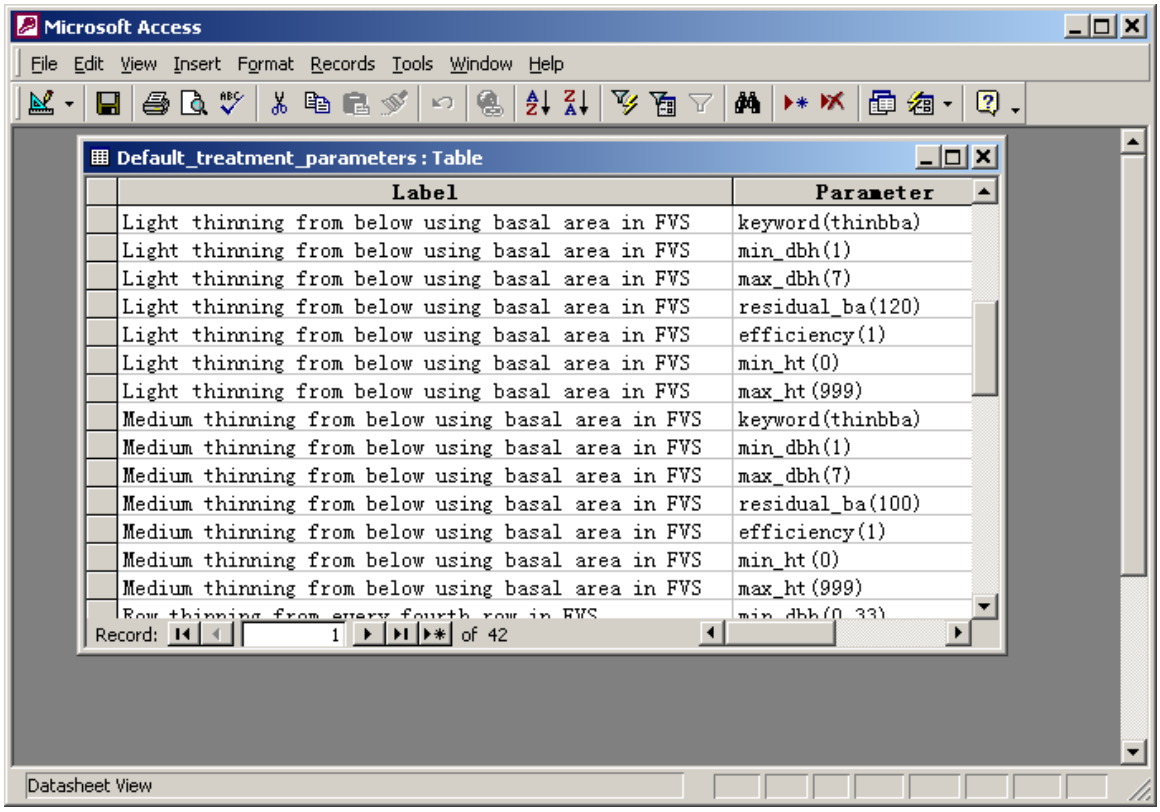

Figure 6: The treatment parameter database

trl_file_format(normalTree,Plot,Obs,TreeId,Spp_ned2,Dbh,Tree_alive1,Tpa):-
fread(n,3,0,PlotNumOld),
fread(a,3,0,TreeNum),
fread(a,6,0,_),
fread(a,2,0,Spp_ned1),
fread(a,9,0,_),
fread(n,4,0,PlotNum),
fread(a,3,0,_),
fread(s,6,0,Tpa_s),
fread(a,11,0,_),
fread(s,4,0,Dbh_s),
fread(s,0,-1,_),
name(Tpa_s,Tpa_l),
removeall(32,Tpa_l,Tpa_list),
name(Tpa,Tpa_list),
name(Dbh_s,Dbh_l),
removeall(32,Dbh_l,Dbh_list),
name(Dbh,Dbh_list),
Plot is PlotNum - 1,

```
number_atom(Obs1,TreeNum),
Obs is Obs1 - 1,
number_atom(PlotNum,PlotNuma),
number_atom(Obs1,Obs3),
cat([PlotNuma, ':',Obs3],TreeId,_),
(
 Tpa == '    0',
 Tree_alive = '0'
;
 Tpa \== '    0',
 Tree_alive = '-1'
),
number_atom(Tree_alive1,Tree_alive),
vari(Variant),
ned_alpha_species_code( Variant,Spp_ned2, Spp_ned1 ).
```

This clause shows that in tree list file, column 3-5 is the plot number, column 6-8 is the tree number, column 14-15 is species code, column 49-53 is tree DBH, and column 32-38 is tree stems per acre. For each tree, the simulation agent will use this information to convert the simulation results to a record in the NED-2 database, and then insert it back to the database.

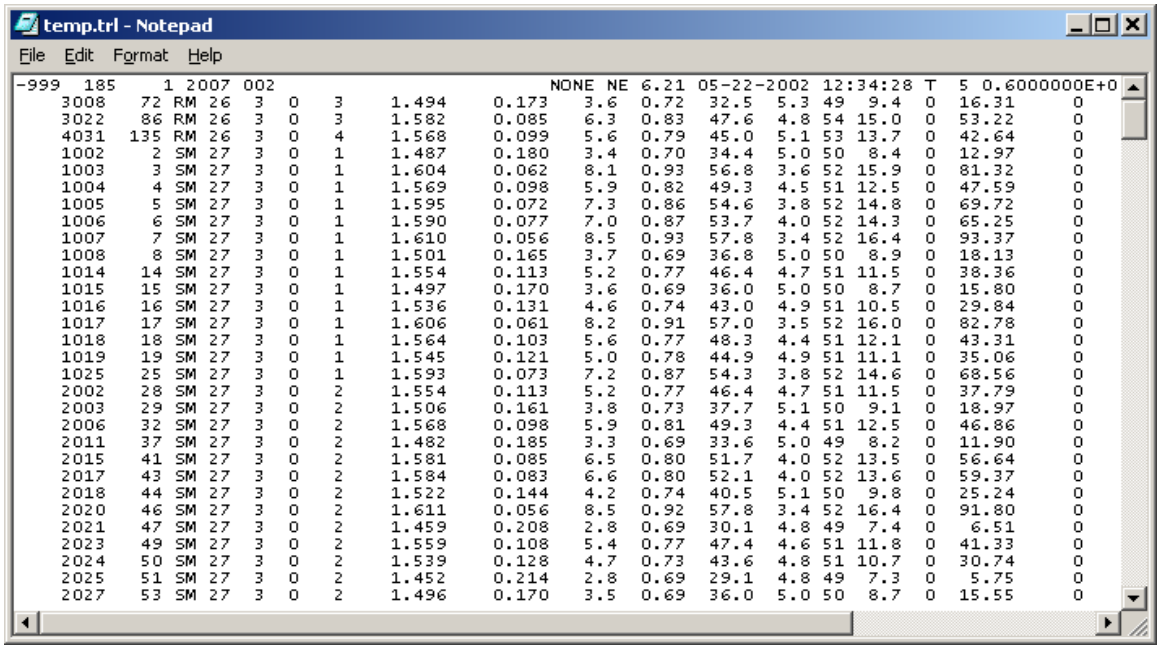

Figure 7: A tree list file

4.3.2.4 Calling A Simulator

PROLOG provides a predicate to call an executable file:

exec(Program, Command, Code).

To run the FVS simulator, a batch file for calling this simulator is invoked. Meta-knowledge will tell the simulation agent how to control the batch files.

A set of batch files is built to satisfy a user's different requests. For example, if a user wants to run the FVS southern variant simulator and get the copies of all FVS files, including tree record input files, keyword input files, and tree list output files, the corresponding batch file is

```
rem stdFVS run on DOS.
copy data.key %1
echo data.key > data.rsp
echo data.tre >> data.rsp
echo data.out >> data.rsp
echo temp.trl >> data.rsp
echo data.sum >> data.rsp
echo data.chp >> data.rsp
SN1.exe < data.rsp
del data.rsp
copy temp.trl %2
```

So, the meta-knowledge base will give the format for calling this batch file as:

```
run_command_format(sn,fvs,Stand,Command,copy_fvs_files):-
        number_atom(Stand,Standa),
        cat(['fvs_sn_copy.bat stand',Standa,'.key stand',Standa,'.trl'],
                Command,_).
```

If a user just wants to run the FVS southern variant and does not need the input and output files, the batch file will be

```
rem stdFVS run on DOS.
echo data.key > data.rsp
```

```
echo data.tre >> data.rsp
echo data.out >> data.rsp
echo temp.trl >> data.rsp
echo data.sum >> data.rsp
echo data.chp >> data.rsp
SN1.exe < data.rsp
del data.rsp
del *.fvs
```

Then, the meta-knowledge base will give the format for calling this batch file as:

```
run_command_format(sn,fvs,Stand,Command,no_copy_fvs_files):-
        number_atom(Stand,Standa),
        cat(['fvs_sn.bat'],Command,_).
```

4.4 The Process for Simulating A User's Plan in NED-2

As mentioned, to simulate a management plan, a user needs to generate the baseline year first. Then he will create the plan by using a plan generation screen (Figure 8). Once the plan is saved in the core database, the user can invoke FVS to simulate it. The user can review the results in another screen (Figure 9).

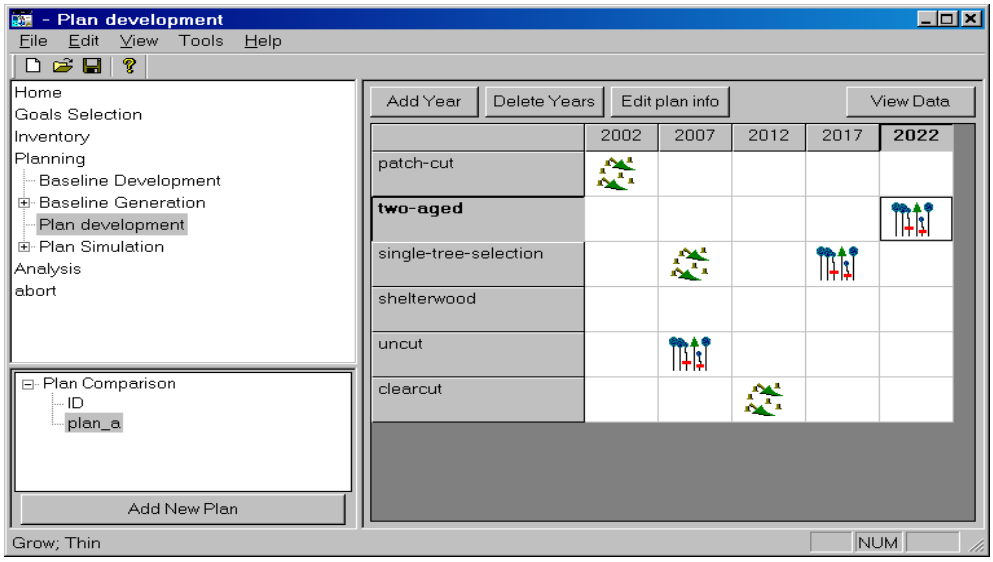

Figure 8: A graphic interface for the plan generation

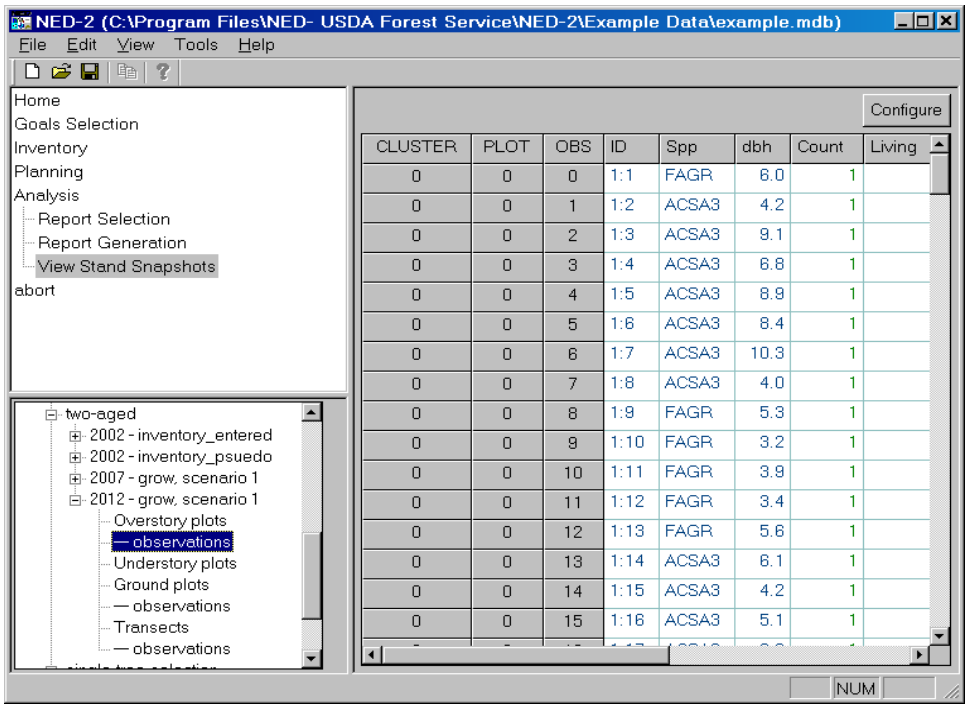

Figure 9: The simulation result

CHAPTER 5

SUMMARY AND CONCLUSIONS

NED-2 is a robust, full-featured Intelligent Information System designed to provide decision support for forest ecosystem management. NED-2 uses a blackboard architecture with semi-autonomous intelligent agents to make decisions and respond to user queries. NED-2 uses external heterogeneous information source management agents to integrate various resources. An EHISM agent is a multi-source manager. Each EHISM agent has a meta-knowledge base that stores the declarative knowledge about when and how to access related information sources. The blackboard architecture with EHISM agents makes integration of external information sources easier, faster, and more powerful. Since the declarative knowledge and procedures are separated, the NED system can evolve easily.

The first simulation model, FVS, has been integrated into NED-2. The structure of the meta-knowledge will be extended to make the simulation agent more intelligent. Other simulation models, such as SILVAH, will be integrated into NED-2. Currently, by consulting the meta-knowledge base, the simulation agent can only simulate user management plans that involve one treatment keyword. That is not enough. Multiple treatment keywords can appear in the keyword file for FVS, telling the simulator how to simulate more complex treatments. NED-2 will be able to respond to high level queries, like “Show me how Stand 10 will look in 25 years if I remove all hardwoods under six

inches in diameter today." The simulation agent will reformat the original user query and run an appropriate growth and yield model according to the meta-knowledge base. It will also need to place appropriate requests on the blackboard so a visualization agent can then display the results of the simulation. This allows seamless data base, knowledge base, and model base interaction that is transparent to the user. The point is to increase the intelligence of the software to reduce the complexity that the user must deal with directly.

REFERENCES

Adali, S., & Emery, R. (1995). A uniform framework for integrating knowledge in heterogeneous knowledge systems. *Proc. of 11th International Conference on Data Engineering* (pp.513-520).

Arens, Y., Knoblock, C.A., & Shen, W.M. (1996). Query reformulation for dynamic information integration, *Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration*, *6*(2/3), 99-130.

Bayardo, R., Bohrer, W., Brice, Cichocki, R., Fowler, A., G., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., & Woelk, D. (1997). InfoSleuth:Semantic integration of information in open and dynamic environments. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 195-206).

Bird, S. D. (1993). Toward a taxonomy of multi-agent systems. *International Journal of Man-Machine Studies*, vol. 39, 689-704.

Bressan, S., & Goh, C. (1998). Answering queries in context. *Proceeding of the International Conference on Flexible Query Answering Systems, FQAS'98,* (68-98). Roskilde, Denmark.

Bayardo, R., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., & Woelk, D. (1997). InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. *Proceeding of the ACM SIGMOD International Conference on Management of Data* (pp.195-206).

Cheung, W., & Cheng, H. (1996). The model-assisted global query system for multiple databases in distributed enterprises. *ACM Transactions on Information systems*, *14*(4): 421-470.

Craig, I. D. (1995). *Blackboard Systems*. Ablex Publishing Corp., Norwood, NJ.

Hass, L. M., Kossman, D., Wimmers, E. L. & Yang, J. (1997). Optimizing queries across diverse data sources. *23rd Conference on Very Large Databases Systems* (pp. 276-285). Athens, Greece.

Heimbbigner, D., & McLeod, D. (1985). A federated architecture for information management. *ACM Transactions on Office Information Systems*, *3*(3): 253-278.

Levy, A., Rajaraman, A., & Ordille, J. (1996). Querying heterogeneous information sources using source descriptions. *Proceedings of the 22nd VLDB Conference* (pp.251-262). Bombay, India.

Maier, F. W. (2002). *Notes on a Blackboard: Recent work on NED-2.* M. S. thesis, University of Georgia.

Marquis, D. A., & Ernst, R. L. (1992). *User's guide to SILVAH: Stand analysis, prescription, and management simulator program for hardwood stands of the Alleghenies.* Gen. Tech. Rep. NE-162. Radnor, PA: U.S. Department of Agriculture, Forest Service, Northeastern Forest Experiment Station.

Mena, E., & Illarramendi, I. (2001). *Ontology-based query processing for global information systems.* Kluwer Academic Publishers, ISBN 0-7923-7375-8

Nute, D., Kim, G., Potter, W. D., Twery, M. J., Rauscher, H. M., Thomasma, Bennett, S., D., & Kollasch, P. (1999). A multi-criteria decision support system for forest management. *Environmental Decision Support Systems and Artificial Intelligence*, AAAI-99, Technical Report WS-99-07 (pp. 68-73). AAAI Press, Menlo Park, California.

Nute, D., Potter, W.D., Maier, F., Wang, J., Twery, M., Rauscher, H.M., Knopp, P., Thomasma, S., Dass, M., & Uchiyama, H. (2002). Intelligent model management in a forest ecosystem management decision support system. *iEMSs 2002*. Lugano, Switzerland. (to appear)

Ozsu, M. T., & Valduriez, P. (1999). *Principles of distributed database systems* (2nd ed.). Prentice Hall.

Papakonstatniou, Y., Garcia-Molina, H., & Ullman, J. (1996). MedMaker: A mediation system based on declarative specifications. *IEEE 12th Int. Conference on Data Engineering* (pp. 132-141). New Orleans.

Potter, W. D., Deng, X., Somasekar, S., Liu, S., Rauscher, H. M., & Thomasma, S. (2000). Forest ecosystem management via the NED Intelligent Information System. *Proceedings of the 13th Int. Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE'2000* (pp. 629-638). New Orleans.

Roth, M. T., & Schwarz, P. (1997). Don't scrap it, wrap it! A wrapper architecture for legacy sources, *Proceeding of the 23th VLDB Conference* (pp. 266-275). Athens, Greece.

Sheth, A., & Larson, J. (1990). Federated database systems for managing distributed, heterogeneous and autonomous database. *ACM Computing Surveys*, *22*(3), 183-236.

Teck, R., Moer, M., & Eav, B. (1997). The forest vegetation simulator: a decision-support tool for integrating resources science. Retrieved on May 20, 2002 from website: *http://www.fs.fed.us/ftproot/pub/fmsc/fvsdesc.htm*

Teck, R., Moeur, M., & Eav, B. (1996). Forecasting ecosystems with the forest vegetation simulator. Journal of Forestry, *94*(12), 7-10.

Twery, Mark J., Rauscher, H. M., Bennett, D. J., Thomasma, S., Stout, S., Palmer, J., Hoffman, R., DeCalesta, D., Gustafson, E., Cleveland, H., Grove, J. M., Nute, D., Kim, G., & Kollasch, R. P. (2000). NED-1: Integrated analysis for forest stewardship decisions. *Computers and Electronics in Agriculture*, 27, 167-193.

Wang, J., Potter, W.D., Nute, D., Maier, F., Rauscher, H.M., Twery, M. J., Thomasma, S., & Knopp, P. (2002). An Intelligent Information System for forest management: NED/FVS integration, *Proceeding of the 2nd FVS Conference.* Fort Collins. (to appear)

Wiederhold, G. (1996). Foreword: Intelligent integration of information. *Intl. Journal of Intelligent Information Systems*, *6*(2/3), 93-97