Decision Support for Multiple Purpose Forestry, April 23-25, 2003, Vienna - Austria

AN AGENT ARCHITECTURE FOR AN INTEGRATED FOREST ECOSYSTEM MANAGEMENT DECISION SUPPORT SYSTEM

Donald Nute[1]*, Walter D. Potter[1], Mayukh Dass[1], Astrid Glende[1], Frederick Maier[1], Hajime Uchiyama[1], Jin Wang[1], Mark Twery[2], Peter Knopp[2] Scott Thomasma[2], H. Michael Rauscher[3]

[1]Artificial Intelligence Center
The University of Georgia
Athens, GA 30605, U.S.A.
dnute@uga.edu
potter@cs.uga.edu

[2]Northeastern Research Station
USDA Forest Service
P.O. Box 968, Burlington, VT 05402-0968, USA
mtwery@fs.fed.us

[3]Creek Experimental Forest, Southern Research Station
USDA Forest Service
Asheville, NC, USA
mrauscher@fs.fed.us

Keywords: agent architecture, blackboard architecture, ecosystem management, model management, decision support system, knowledge based system, Prolog.

Summary: *A wide variety of software tools are available to support decision making in the management of forest ecosystems. These tools include databases, growth and yield models, wildlife models, silvicultural expert systems, financial models, geographical information systems, and visualization tools. Typically, each of these tools has its own complex interface and data format. To use these tools in combination, a manager must learn each interface and manually convert data from one format to another. NED-2 is a robust, intelligent, goal-driven decision support system that integrates tools in each of these categories. NED-2 uses a blackboard architecture and a set of semi-autonomous agents to manage these tools for the user. Each agent has the procedural knowledge needed to operate a class of decisions support tools used in forest ecosystem management. The simulation agent can set up input for growth and yield models and knows how to interpret the output from these models. Meta-knowledge bases provide the simulation agent with information about the data formats and control codes used by different growth simulators. The GIS agent can merge information with a shape file and knows how to invoke a geographical information system to display the information. The visualization agent can generate input for stand and landscape visualization tools. The blackboard system itself includes a powerful agent that integrates a database and a set of Prolog clauses into a single blackboard. The interface agent provides access to all the tools in the system through a single user interface. Other agents allow development of alternative treatment plans; provide analysis of timber, wildlife, water, ecology, and visual goals; and generate a wide variety of reports useful in forest management. The agent architecture is designed to facilitate integration of new third-party decision support tools as they become available.*

1. Introduction

Information technology provides a broad range of tools useful in the management of forested lands. These tools include databases, growth and yield models, wildlife models, silvicultural expert systems, financial models, geographical information systems, and visualization tools. To use these tools, the forest manager must invest considerable time and effort to learning how each tool functions. After this initial investment, more time and effort must be spent on integrating the tools into the decision process. Each tool will require that data be entered in a particular manner and each tool will output its results in its own format. To use several tools together, the forest manager must take the output from one tool and convert it into the format required by the next tool to be used. In these circumstances, it would be surprising if many forest managers took advantage of all the decision support tools that are available to them. What is needed is a tool that helps the manager integrate decision support tools such as growth simulation models, wildlife models, financial models, and models for selecting silvicultural treatments. What is lacking is a model of the decision process that integrates these other models, a tool that models how to use models in forest management. What is missing is a forest management metamodel.

NED-2 is an attempt to model a decision process that integrates a broad range of decision tools and models. The NED-2 architecture has been developed with a few key design principles in mind. First, NED-2 users should be able to enter data and view results using a single, simple interface regardless of how those results are produced. Second, NED-2 should perform all conversions between the data formats used by different tools and models without user intervention. Third, NED-2 should have an open architecture that allows additional third-party components to be added to the system without extensive revision of existing NED-2 code. Fourth, developers of new NED-2 components should not need to know very much about how other components work. These design criteria resulted in three essential features of the NED-2 architecture. The first is a robust ontology that supports communication between NED-2 and a wide variety of forest management decision support tools. This ontology is provided by a complex internal data model implemented using both database and logic programming technologies. Second, NED-2 is an agent-based system. Individual tasks are performed by semi-autonomous agents. Third, NED-2 agents coordinate their behaviors and communicate information using a blackboard architecture (Ni 1989). Agents do not need to know what tasks other agents perform or what information other agents need to perform those tasks. All information is placed on the blackboard where all agents can access it. Whenever an agent needs help in performing a task, it can place a request on the blackboard and does not need to know who will respond to its request.

A companion to this paper (Twery *et al.* 2003) describes NED-2 functionally from the point of view of the user. Here we will focus on the implementation of NED-2. We will describe its ontology, its architecture, and several of its agents in detail. Finally, we will look at some of the agents we plan to build and tools we plan to integrate into subsequent versions of NED.

2. Overview of the NED-2 Architecture

Wherever possible, NED-2 integrates already existing tools. So far, these include a Microsoft Access database, the Forest Vegetation Simulator (FVS) (Crookston 1997), and ArcMap. For some tasks, including goal analysis, no appropriate tools were available and the NED developers had to construct the needed component. When integrating a third-party component, a NED-2 agent was developed that knew how to use that component. Where there was no available third-party component, the necessary functionality was built directly into the NED-2 agent.

A complex software system that integrates many components can have any number of architectures. One fundamental design question is whether control of the system will be centralized or distributed. The NED architecture is distributed. This was seen as the best choice for a system that allowed additional components of unanticipated kinds to be added later. NED-2 is a community of agents, each performing some specialized task. Another question is how the components will communicate with each other. Hierarchical models and other architectures where one agent sends requests or information directly to

other agents require that agents have considerable knowledge of what other agents do and of what information they need to perform their tasks. NED-2 uses a blackboard architecture. With this approach, all information and requests for tasks to be performed are placed on a blackboard where all agents can view them. This approach minimizes the amount of knowledge each agent must have about other agents in the system. The diagram below illustrates the NED-2 architecture. Components included in the diagram will be explained in later sections of the paper.

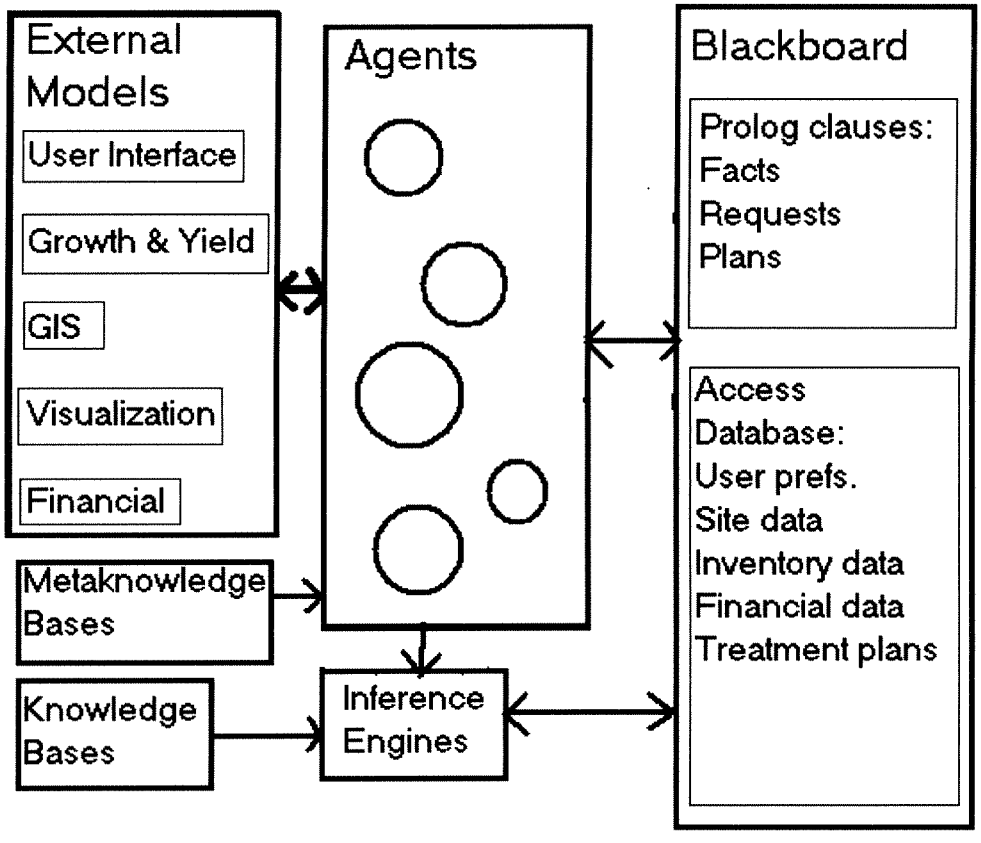

Figure1. NED-2 Architecture

3. The NED-2 Ontology and Internal Data Model

There are specialized tools such as DAML (Hendler and McGuinness 2000) for representing ontologies. However, the ultimate purpose of an ontology is to provide a system with the ability to represent domain knowledge and to utilize information and knowledge about the domain. We have adopted a less abstract and more practically motivated method for representing the NED-2 ontology. The ontology for NED-2 is incorporated into the design of the NED-2 database and the design of a set of Prolog clauses that store temporary information during a NED-2 session. To communicate with a variety of external components, to represent a forest over time, to evaluate a potentially large set of management goals, and to generate all the reports that a manager might require, NED-2 requires a robust ontology and internal data model.

A management unit is divided into stands. Database tables were designed to hold permanent information about the management unit and the stands such as location, size, and physical characteristics. Management goals must be coordinated with knowledge bases for evaluating whether a particular goal has been satisfied. So the set of available goals is limited to those included in the database in a goals table.

New goals can be added to the system by entering the goal in the goals table and providing additional rules in the database for evaluating the goal. There is also a table that includes necessary information about all the different reports NED-2 can generate. There will be a goal report for each goal in the system. Currently the goals table includes timber, wildlife, water, ecology, and visual goals. Fire management goals are under development.

Neither the management unit nor the stand is static. So most information about stands is stored as snapshots representing the stand at a point in time under a particular treatment regime. The initial snapshot for each stand represents a year when inventory was taken for at least one plot in the stand. Each snapshot includes a set of observations for the overstory, understory, and groundcover component of the stand. These include individual tree data for the overstory and understory, and the presence of herbaceous plants of various categories, of coarse woody debris, etc., for the understory and the ground layers.

All plans are developed from a common baseline year. If inventory has not been taken on every stand during the baseline year, then simulated data is generated for that stand using a growth simulator. To generate the baseline and to simulate treatment plans later, a simulation model must be chosen for each stand. The user selects treatments from a pre-established list and either accepts default parameters or enters his own parameters for each treatment. The user's treatment definitions are stored in a knowledge base and in a treatment table in the database.

The user develops one or more treatment scenarios for the management unit, specifying both silvicultural and non-silvicultural treatments and the years they are to be applied. This information is stored in a scenarios and a secenarios_design table in the database. When a plan is simulated, NED-2 will generate pre- and post-treatment snapshots for each year that a treatment has been specified. Each snapshot is represented in the database by a snapshot number. This coordinates the snapshot observations with the treatment entry in the plan or scenario.

Facts are stored temporarily on the blackboard as Prolog clauses. Persisting information is stored in the database or in Prolog knowledge bases. Facts on the blackboard are represented as Attribute-Object-Value triples. The Attribute in an AOV triple is always represented as a simple "atom", but the Object and the Value can be complex. For example, the triple

```
tpa([snapshot(17),species(oak),size(6)],24)
```

indicates that there are 24 six-inch oaks per acre in snapshot 17, while the triple

```
goal([balance_size_classes],[[21,0.0],[22,0.4],[23,1.0])
```

indicates that snapshot 21 satisfies the balanced size class goal with confidence 0, snapshot 22 with confidence 0.4, and snapshot 23 with confidence 1. Objects are always indicated by a list of identifiers. In the first example, the attribute is trees per acre; here the object is complex (six-inch oaks in snapshot 17) but the Value is simple (24). In the second example, the attribute is balanced size classes; now the object is simple (a timber goal) but the value is complex (a list of snapshots together with the confidence that each snapshot satisfies the goal.)

The NED-2 ontology and internal data model must be rich enough to support exchange of information with all external components integrated into the NED-2 system. The structure of the database and of the Prolog facts on the blackboard provide the concepts of the ontology. We also need to establish critical relations between these concepts. These are provided partly by knowledge bases of rules relating the concepts and partly by utilities that calculate the values of certain attributes from the values of other attributes. These knowledge bases and utilities may be accessed by any NED-2 agent that needs them.

Finally, part of the ontology of the NED-2 system is stored in metaknowledge bases that describe the structure of the NED-2 database including relations between information in different tables. The function of these metaknowledge bases is described in the next section.

4. The NED-2 Blackboard/Database Integration

When a user opens a NED-2 working file containing his inventory, goals, treatment definitions, and plans, this database becomes an integral part of the NED-2 blackboard, not physically but conceptually. Any NED-2 agent may access a fact from the blackboard using the Prolog query

```
known(Attribute(Object,Value))
```

whether that fact is stored as a Prolog clause or as a record or set of records in the database. To achieve this integration, the blackboard must be an active set of procedures rather than just a static set of facts. These procedures analyze the request for information to determine how it may be satisfied. First, it looks to see if there is a corresponding Prolog fact. If not, then it determines whether or not the information is stored in the database. To do this, the blackboard must consult a metaknowledge base that tells it the structure of the NED-2 database. This may require performing joins across several tables. For example, the query

```
?- known(ba([stand(17),plan(`Maximize Timber`),year(2023)],BA).
```

asks what the basal area of stand 17 will be in 2023 if we implement the plan called `Maximize Timber`. The blackboard will look in the [Scenarios] table to find the scenario number for the `Maximize Timber` plan. Then it will look in the [Scenarios_design] table to find the snapshot number corresponding to that plan for stand 17 and year 2023. Then it will look in the [Treatment_measurements] table to find the basal area for that snapshot, which will have been calculated from values in the [Overstory_obs] table when the plan was simulated. All of this will be automatic. The developer building a new NED-2 agent does not need to know the underlying structure of the database that supports the NED-2 ontology in order to formulate his query because the blackboard knows this. The knowledge is provided by the metaknowledge base.

Updating knowledge on the blackboard is more complex. Any agent can put temporary data on the blackboard as a Prolog clause, but only certain agents may update persistent information in the database. These include the interface agent, the treatment development agent, and the simulation agent all described below. These agents construct specific SQL queries for database update. So these agents must have specific knowledge of the database and its relationship to the NED-2 ontology. For more details on integrating the database with the Prolog blackboard, see (Maier 2002).

Requests are also stored on the blackboard as Prolog clauses. Requests can be as simple as

```
request(interface).
```

or as complex as

```
request([

analysis('Wildlife',[ american_goldfinch, cedar_waxwing,
northern_flying_squirrel], [inventory, baseline]),

analysis('Timber', [cubic_foot],  [inventory, baseline])

arcview(,[ american_goldfinch, cedar_waxwing, northern_flying_squirrel,
cubic_foot], [inventory, baseline], 'mystandmap.shp')

])
```

The first of these simply requests that the interface dialogs be enabled. This is the request that initiates interaction with the user when NED-2 starts. The second request is actually a plan of action including several component requests: analyze the inventory and baseline data to see whether habitat is available for the American goldfinch, the cedar waxwing, and the northern flying squirrel; analyze the inventory and baseline data to see whether the goal of focusing timber production on cubic foot production is satisfied; and display the results of this analysis on a map defined in the shape file `mystandmap.shp`. Different agents will satisfy these requests as explained below. Actually, this is a very simple request to find on the NED-2 blackboard.

5. NED-2 Agents

NED-2 Interface Agent

NED-1 (Twery *et al.* 2000) is a program for analyzing inventory, evaluating how well goals are satisfied by current management unit conditions, and generating reports. It was a C++ program with a Prolog inference engine to handle the goal evaluation. The user interface in NED-1 is well-designed and forms the basis for the NED-2 interface. However, to facilitate integration of third-party simulation programs, geographical information systems, and visualization packages NED-2 is primarily a Prolog program. The NED-1 user interface and inventory analysis codes were redesigned as a set of Dynamic Link Libraries (DLLs). The NED-2 interface agent starts and suspends these modules as appropriate. When the user requests some service that the DLLs cannot provide, the interface agent places a request for that service on the blackboard.

The NED-2 interface DLLs handle inventory entry and analysis, updating the user's Microsoft Access database directly. These DLLs also provide dialogs for goal selection plan development and simulation, and report selection. The user's goal sets, treatment plans, and report sets are also stored in the working database by the DLLs. Other NED-2 agents written in Prolog also access and update the working database.

NED-2 Treatment Development Agent

Before the user can develop a treatment plan, he must create a knowledge base of available treatments including all required parameters. The NED-2 treatment development agent assists in this process. This agent consults a default knowledge base that contains a set of available treatments for each simulation program integrated with NED-2. It displays these to the user in lists. The user can then select a simulation program he plans to use for one or more of his stands, review the default parameters for any treatments he wishes to use, and either accept or modify the default parameters. Users can also use the treatment development dialog to define custom treatments. When the user is finished, his treatment definitions are stored in his personal treatments knowledge base.

Of course, a user responsible for managing more than one management unit can import the treatments knowledge base developed for one unit into the working knowledge base for another unit. Users can also exchange treatments knowledge bases. These knowledge bases are stored outside the management unit database as a set of Prolog clauses. A table within the database records a pointer to the appropriate knowledge base and basic information that can be displayed in the plan development user interface.

NED-2 Simulation Agent

Growth and yield simulators typically require input data files in very specific formats and complex instructions for simulating treatments. They also output cut lists and tree lists in specialized formats. For this reason, simulators often have complex interfaces to assist the user in setting up the simulation run and to display the results of the run. The NED-2 simulation agent is designed to do all this for the user. It takes the user's inventory and treatment plan, sets up all data and control files to run the simulator, then interprets the results so they can be displayed by the standard NED-2 interface. It is not surprising that this agent is one of the most complex in the entire NED-2 system.

In the NED-2 design philosophy, we think of simulators as knowledge sources. To use one of these knowledge sources, the simulation agent needs to know the semantics of the input and output files and of the control structures for the simulator. That is, it requires knowledge about the knowledge source, what we might call *metaknowledge*. There are several kinds of metaknowledge files used by the simulation agent. Some of these provide the semantics for the data files used and produced by the agent. The semantics indicates how items in these files correspond to fields in the NED-2 database and to each other. Some of this metaknowledge is just tables that allow translation of species codes between NED-2 and a simulator. Other metaknowledge provide templates for data input files together with instructions for inserting data into the templates at the proper places. To set up the controls for a simulation run, the simulation agent must understand how a particular silvicultural treatment is to be represented. This information is stored in the treatments knowledge base constructed using the treatment development agent described in the last section. Different metaknowledge of each of these types must be developed for every simulator incorporated into NED-2. We currently have integrated the Northeastern and Southern variants of FVS. While the formats for the data files and the control codes for these two simulators are essentially the same, the two variants do handle different sets of species and even have different codes for the same species in some cases.

Before a plan can be simulated, a single baseline year must be established for the management unit. When there is no single year for which inventory was taken for all stands, this requires that data for some stands will have to be simulated. Stand inventory data are organized in NED-2 in clusters where each cluster includes exactly one overstory plot. Associated with each overstory plot will be one or more understory and ground plots. By default, all trees with a dbh of 1" or greater go into the overstory and smaller stems go into the understory, but the user may set another value for this purpose. Typically, all trees with dbh of 1" or larger should be included in data sent to a simulator. So it may be necessary to take trees from an understory plot, send them to a simulator along with tree data for the corresponding overstory plot, and then place them in a new snapshot for the understory plot after the simulation is finished. A simulator may also create new trees that, because of the user's preference, belong in the understory. It is impossible to determine which understory plot in a cluster these trees should be assigned to. So before any simulation is performed, NED-2 converts all the understory plots in a cluster into one representative understory plot and similarly for ground plots. The baseline data are then created by growing all these clusters with their "pseudo-plots" for understory and ground level data up to a single year. All plans are then simulated from this baseline year forward.

NED-2 Goal Analysis Agents

Goals can be evaluated for inventory data, at the baseline year, or for a point in the future. Goal evaluation in the future must be tied to a specific treatment plan. The NED-2 goal analysis agents analyze data at both the stand and the management unit levels to determine whether they satisfy a given goal. So goal analysis is often a two step process. First, determine which stands satisfy the goal. Then apply a higher-order rule to determine whether the goal is satisfied at the management unit level. For example, a stand might pass a balanced size class goal if the trees in the stand divide into size classes in a certain way, but the management unit would pass the goal only if a certain percentage of stands, or a certain percentage of the management unit area, passes the lower-level rule.

Knowledge bases contain rules that relate a high-level goal to a set of measurable or at least observable variables for which values are available in the inventory data or in simulated data for the baseline or future years. These rules are applied to a snapshot of a stand representing a year and a treatment plan (or the inventory or baseline data) by an inference engine. The inference engine determines whether a stand fails, nearly passes, minimally passes, or passes a goal. It performs a variety of fuzzy reasoning to arrive at these conclusions. The inference engine puts the results of this analysis on the blackboard. The analysis agent then uses these facts to test the higher level rule for the management unit and puts these results on the blackboard.

There is a different goal analysis agent for each category of goals: timber, ecology, water, visual, and wildlife. These agents do not display the results of their analysis to the user directly. The request for

analysis that one of these agents responds to also indicates whether the analysis results will be presented to the user by a GIS system or in a written report. If the analysis is to be displayed on a map, the agent places facts on the blackboard that the GIS agent can use. If a written report is to be constructed, the agent generates an HTML file that can later be incorporated into a hyper-document by the report writing agent. To do this, the agent retrieves a knowledge base that includes the template and information about how goal analysis results are to be included in the template.

NED-2 GIS Agent

NED-2 uses a commercial geographical information system, ESRI's ArcMap, to display inventory data, simulated data, the results of data analysis, and the results of goal analysis. NED-2 does not use GIS to analyze data. The NED-2 geographical information system agent consists of two distinct parts: a Prolog portion that prepares a temporary database to be combined by ArcMap with a shape file for the management unit, and Microsoft Visual Basic for Applications (VBA) code incorporated into an ArcMap project file that tells ArcMap which files to merge and then provides the user with a custom interface to view the NED-2 data.

The Prolog portion of the GIS agent responds to a complex request placed on the blackboard by a planning agent described below. This request tells the agent which views, which data, and which goals to display on a map of the management unit. A view can be either inventory for the management unit, the baseline, or a combination of a treatment plan and a year. Any data can be displayed that represents some property of an individual stand. This can be physical characteristics such as the presence of ponds or the percentage of rocky areas, or it can be silvicultural data such as basal area or the percentage of canopy closure. The GIS agent can also display the results of goal analysis. On the map of the management unit, it will color any stand that fails a goal red, any stand that nearly passes yellow, any stand that minimally passes light green, and any stand that clearly passes a darker green.

The Prolog part of the GIS agent creates a temporary Microsoft Access database containing the data to be displayed by ArcMap. This database also contains information about which views are represented in the data. Then the Prolog part of the GIS agent starts the ArcMap project containing the VBA code that makes up the second part of the GIS agent. The dialog generated by the VBA code allows the user to choose which variable or goal to display, and the ArcMap interface then allows the user to move between views of the management unit as the map is colored to show the values of the data variable or goal selected.

By embedding the VBA code into an ArcMap project, NED-2 also allows the user to access the results of NED-2 analysis outside NED-2. If the user starts up the ArcMap/NED-2 project directly, the VBA code will prompt the user for a data file. By directing it to the temporary data file created by the Prolog part of the simulation agent, the user can merge the data with his shape file as though he were running NED-2. The user can also send the ArcMap/NED-2 project file, the data file generated by the Prolog part of the GIS agent, and his shape file to someone else. If ArcMap is available to the recipient of these files, he can open the project, merge the data and shape files, and examine the NED-2 analysis without even having NED-2 available on his computer.

NED-2 Report Writer Agent

The NED-2 report writer agent prepares a hypertext document incorporating a potentially large set of HTML reports. It then starts up the user's default Web browser to display the top level of the document. This will include a summary of management unit information and an organized list of clickable links to the other reports contained in the document.

The report writer responds to a complex request indicating which reports are to be included in the hypertext document. Also on the blackboard the report writer will find facts telling it the full names for goal analysis reports generated by the goal analysis agent. All other reports, such as vegetation reports for each stand, will be created by the report writer agent. Corresponding to each of these reports there is a knowledge base that includes a template for the report and information about what data values are to be placed in the report and where they are to be placed. In some cases, the values to be included in a report

are not stored in the NED-2 database. The knowledge base will also include rules for calculating these values using other values that are in the database.

As the report writer generates each report to be included in the hypertext document, it places a fact on the blackboard that holds information about the report. Finally, the agent refers to all the facts placed on the blackboard by it or by the goal analysis agents and uses these facts to generate the top level of the hyper-document. It then removes all the temporary facts used to generate the complete report from the blackboard and sends the name of the top level HTML file to the default Web browser.

NED-2 Planning Agent

Some tasks performed by NED-2 require that a number of different NED-2 agents perform a set of subtasks in a particular order. For example, the user might ask for a GIS display or for reports to be generated. Before either of these tasks can be completed, NED-2 must determine which plans, years, data variables, and goals are to be included in the GIS display or reports. And if any goal analysis is to be included, the goal analysis agents must first perform the analysis. The agents in NED-2 have a great deal of autonomy, but they also know little about what other agents in the system do. So how can they coordinate their efforts to accomplish such a complex task?

The answer to this problem is planning agents. A NED-2 planning agent knows what steps must be performed to produce a GIS display or a set of reports. We will describe what this agent does for a GIS display; the process is similar for reports. First, the agent confers with the user to determine which views to include in the display. The user can always include inventory and baseline views. He can also select a treatment plan and all years included in that plan, or he can include a year and all plans for which data will be available in that year. The user then selects groups of data variables and goals to include in the display.

After conferring with the user to determine what will go into the GIS display, the planning agent determines the snapshots that correspond to the views selected by the user. It then formulates a set of requests for the chosen goals to be analyzed for each of these snapshots. These go into a list together with a request for a GIS display to be presented. This list is placed in a request on the blackboard. The steps on the list are in the order that they are to be completed, and the list therefore constitutes a plan for achieving the complex objective. Facts are also placed on the blackboard indicating which views and data variables are to be included in the GIS display. The other agents in the system see the plan on the blackboard. Any agent that can perform the first sub-task in the plan does so and removes that task from the plan. Then any agent that can perform the second sub-task, which is now the first item in the list, performs that task. And so on until the plan is completed. After all the goal analysis is finished, the last step is to actually generate the GIS display. The GIS agent responds to this request, gathering the information about what goes into the display from the blackboard. This information will have been placed on the blackboard by the NED-2 planning agent and by the various goal analysis agents as they performed earlier steps in the plan. Finally, the GIS agent will remove all the facts it used to create the GIS display from the blackboard.

6. Future Plans for NED

Integrating so many different decision support tools into a single DSS has been challenging. The two foundations for this integration are the rich NED-2 ontology and internal data model, and the distributed control using autonomous agents and a blackboard architecture. The ontology and data model make it possible to develop semantics for the data formats and control structures of external models. Programmers developing different NED-2 agents can also depend on a stable model for the information those agents will use in performing their tasks.

An alpha version of NED-2 that has all the functionality described in this paper has been completed. This system will be used to conduct a set of case studies to test the NED-2 decision process and to determine how well NED-2 supports this process. There are also plans to perform usability studies on NED-2 to determine how easy it is for users to learn to use the system effectively and how easy it is for developers

to integrate new components into the NED-2 architecture. Several expansions of NED-2 are already on the drawing board.

(1) We plan to improve and extend the goal analysis rules in NED-2. For example, the wildlife rules were developed for the Northeastern United States. We plan to develop similar rule sets for the Southeast. We also plan to develop new rule sets for additional categories of goals. For example, we have begun work on a rule set for analyzing how well a management unit satisfies goals to maintain an acceptable fire risk level.

(2) NED-2 currently has no model for simulating changes in the understory and ground levels of a stand. This data only changes if the growth and yield simulator used for the overstory level generates trees through some regeneration model that are small enough to fall below the user-set boundary between the overstory and understory. Many NED-2 wildlife rules use understory and ground level data to test for species habitat. Understory and ground level models are in development.

(3) The only regeneration models available for NED-2 are those in the Northeastern and Southern variants of FVS. Other models are under development, and the NED-2 ontology and simulation agent have been designed to allow the user to choose not only the overstory growth model, but also the regeneration and understory/ground level models for each stand. We also plan to allow users to build their own regeneration tables as separate files so NED-2 can use those upon request.

(4) Additional overstory growth and yield models will be incorporated into NED-2. We expect SILVAH to be incorporated next.

(5) Visualization tools will be integrated into NED-2. Current plans are to incorporate ENVISION (McGoughey 2000).

(6) Although the NED-2 ontology and internal data model includes such financial information as projected costs of treatments and prices of timber, the system does not currently include a model for performing financial analysis on this data. We plan to integrate such a model in a future version of NED-2.

7. References

Crookston, Nicholas L. 1997. Suppose: an interface to the forest vegetation simulator. In: Teck, Richard; Moeur, Melinda; Adams, Judy. 1997. *Proceedings: Forest Vegetation Simulator Conference. 3-7 February 1997, Fort Collins, CO.* Gen. Tech. Rep. INT-GTR-373. Ogden, UT: U.S.D.A., Forest Service, Intermountain Research Station.

J. Hendler and D. L. McGuinness. 2000. The DARPA Agent Markup Language. *IEEE Intelligent Systems.* [Online]. *15 (6)* , pp. 67-73. Available: http://www.ksl.stanford.edu/people/dlm/papers/ieee-daml01-abstract.html

McGaughey, R. 2000. ENVISION – environmental visualization system. ENVISION home page, http://forsys.cfr.washington.edu/svs.html.

Maier, Frederick. *Notes on a Blackboard: Recent Work on NED-2.* Masters Thesis, Artificial Intelligence Center, The University of Georgia, Athens, GA, 2002. http://graduate.gradsch.uga.edu/edtarchive/summer2002/maier_frederick_w_200205_ms.pdf.

Ni, H. Penny. 1989. Blackboard systems. In Avron Barr, Paul R. Cohen, and Edward A. Feigenbaum (eds.), *Handbook of Artificial Intelligence, Vol. IV.* Reading, MA: Addison-Wesley.

Twery, Mark J., Rauscher, H. Michael, Bennett, Deborah J., Thomasma, Scott A., Stout, Susan L., Palmer, James F., Hoffman, Robin E., DeCalesta, David S., Gustafson, Eric, Cleveland, Helene, Grove, J. Morgan, Nute, Donald, Kim, Geneho, and Kollasch, R. Peter. 2000. NED-1: integrated analyses of forest stewardship decisions. *Computers and Electronics in Agriculture* **27**: 167-193.

Twery, Mark J.,_H. Michael Rauscher, Peter D. Knopp, Scott A. Thomasma, Donald E. Nute , Walter D. Potter, Frederick Maie[c], Jin Wang, Mayukh Dass, Hajime Uchiyama, and Astrid Glende. 2003. NED-2: an integrated forest ecosystem management decision support system.