AAAI Workshop on Intelligent Information Systems, July, 1999, Orlando, Florida
Co-Chaired by K. Hammond and L. Birnbaum

NED-IIS: An Intelligent Information System for Forest Ecosystem Management

W.D. Potter, S. Somasekar, R. Kommineni
Artificial Intelligence Center, University of Georgia, Athens, GA
and
H.M. Rauscher
USDA Forest Service, Bent Creek Experimental Forest, Asheville, NC
(contact: potter@cs.uga.edu)

Abstract

We view an Intelligent Information System (IIS) as composed of a unified knowledge base, database, and model base. The model base includes decision support models such as growth and yield simulation models, forecasting models, and visualization models for example. In addition, we feel that the model base should include domain specific problem solving modules as well as decision support models. This, then, allows an IIS to provide responses to user queries regardless of whether the query process involves a data retrieval, an inference, a computational method, a problem solving module (employing, for example, a non-production rule based heuristic search technique), or some combination of these. The unified integration of these components in a distributed environment for forest ecosystem management is the focus of our research.

Introduction

In the past decade, organizations have been moving mainframe-based systems toward open, distributed computing environments. The demand for interoperability has been driven by the accelerated construction of large-scale distributed systems for operational use and by increasing use of the Internet (Manola 1995). Distributed computing offers many advantages, including. location transparency to users, scalability, fault tolerance, load balancing and resource sharing. As such, much of the interoperability literature has been concerned with distributed computing; for example, the recent emergence of Java, the Object Management Group's CORBA (Common Object Request Broker Architecture) and Microsoft's DCOM (Distributed Component Object Model) are all for this purpose (Grimes 1997, Leppinen et al. 1997, OMG 1997). In addition, object orientation (00) is probably the most widely used approach in software development and the basis for the CORBA and DCOM interoperability architectures. 00 makes it easier to maintain software modules, and makes it possible to re-use existing software objects (Gamma et al. 1995). Consequently, platform independence, as well as language independence, has been a major focus in these interoperability architectures.

The interoperability architecture for our unified, distributed knowledge/data/model management approach is based on a combination of DCOM and Active KDL (Active Knowledge/Data Language). DCOM provides the middleware protocol necessary to handle distributed interaction among our forest ecosystem management applications. It is a built-in component of the Microsoft NT and Windows 98 operating systems. Active KDL follows the functional and object-oriented paradigms (Miller et al. 1991a, Miller et al. 1991b). Active KDL is based on the hyper-semantic data model, KDM (Knowledge/Data Model) developed in the mid-1980's (Potter and Kerschberg 1986). By hyper-semantic, we mean a data (information) mode1 capable of capturing even more of the meaning of an application area than captured via traditional, semantic, or object-oriented models (Potter and Trueblood 1988, Potter et al. 1989).

A typical forest ecosystem management decision support system (FEM-DSS) contains a user interface, database, geographical information system (GIS), possibly a knowledge base, simulation and optimization models, help/hypertext management, data visualization and decision methods (Rauscher 1999). Recent reviews have identified that at least 30 FEM-DSS have been developed for use in the United States (Mowrer, 1997; Rauscher 1999). Different FEM-DSS support different parts of the ecosystem management process. Functional service modules, also known as problem solving modules (PSM), provide specialized support for one or a few phases of the forest ecosystem management process. These PSM are further categorized by their function, for example, group negotiations, vegetation dynamics, disturbance simulation, and spatial visualization. Full service FEM-DSS on the other hand, attempt to be comprehensive by offering support for the complete forest ecosystem management process. These full-service systems may be further classified into the following by the scale of support: regional assessment, forest planning and project-level planning (Rauscher 1999).

Most FEM-DSS were developed independently of one another. As a result, they are typically large, monolithic, stand-alone systems. Although, collectively, the existing FEM-DSS are capable of addressing the full

range of support required for the management of a complex forest ecosystem no one system has been found to be completely satisfactory (Mowrer et al. 1997, Rauscher 1999). An ideal FEM system, therefore, requires many Of the available DSSs working together. This necessitates both functional service and full-service systems integration. Besides, it is often more cost-effective to re-use existing software than to develop custom software when an existing FEM-DSS is to be enhanced to provide additional services.

To achieve integrated operation of FEM-DSS, it is necessary to overcome the problems presented by the variety of platforms these legacy systems run on and the heterogeneity of their development environments. Existing FEM-DSS have been written in different software languages, they reside on different hardware platforms, they have different data access mechanisms, and different component/module interfaces. For example, non-geographical databases may be written in Oracle, geographical information system (GIS) databases in ARC/INFO, knowledge bases in Prolog, and a simulation model in Fortran. To date, efforts to achieve interoperability between FEM-DSS modules have used ad hoc techniques yielding unique, point-to-point custom solutions. While such unique solutions work, they are typically very difficult to maintain and extend by other developers due to their idiosyncratic nature. No comprehensive, theory-based interoperability standard currently exists for achieving integrated operations of FEM-DSS (Rauscher, 1999).

A cooperative research program between scientists of the Artificial Intelligence Center at the University of Georgia and the USDA Forest Service is currently underway to develop an Intelligent Information System (IIS) component for NED, a forest ecosystem management Decision Support System (Rauscher et al 1997, Twery et al. 1997). NED is a comprehensive full service FEM-DSS. The NED-IIS supports knowledge, data, and model management in a distributed environment. The immediate goal is the seamless integration of several existing, loosely coupled legacy USDA Forest Service systems within the NED architecture. Additional systems will be added at a later time using the interoperability standard designed and tested in this initial effort.

In this paper, we focus on three important facets of the NED-IIS. The first facet deals with how a forest manager would interact with our system. A manager interfaces with it via a standard client process that may provide knowledgeable assistance. The knowledge driving the interface and controlling the decision models is maintained within the interface controller. The interface controller is based on the notion of query driven processing. That is, a user specifies a query to the NED-IIS and the interface controller determines how best to respond to the query. The response may entail a variety of events taking place. Providing the hand-shaking architectural support for our distributed environment is the second facet. We use Microsoft's Distributed Component Object Model (DCOM) to facilitate the distributed integration of the decision model components (Microsoft 1995, 1996, 1997, 1998). The integrated decision model components of our intelligent information system prototype (the third facet) are FVS, SILVAH, and FIBER. FVS is a Forest Vegetation Simulator that projects the growth of forest stands under a variety of conditions (Teck et al. 1996, Teck et al. 1997). SILVAH (Marquis et al. 1992) is the SILViculture of Allegheny Hardwoods prescription system that aids forest managers in making treatment decisions. FIBER (Solomon et al. 1992) is a forest stand growth projection system that deals with interactions of a variety of tree species in the stand.

The organization of this paper follows the order of the three NED-IIS facets. We discuss the approach we take for providing distributed interoperable functionality. We briefly discuss the "middleware" layer of our approach (i.e., DCOM). Finally, we provide an overview of the integrated decision model components and our interface controller prototype.

Conceptual Architecture

Before discussing the details of Active KDL, we want to define our view of interoperability. One solution to many of the problems in systems integration is interoperability of software systems (Potter et al. 1992, Potter et al. 1994, Otte et al. 1996, Liu 1998). Interoperability is the ability for two or more software components to cooperate by exchanging services and data with one another, despite the possible heterogeneity in their language, interface and hardware platform (Heiler 1995, Wegner 1996, Sheth 1998). Interoperable systems provide a software standard that promotes communication between components, and provides for the integration of legacy and newly developed components.

The Active KDL Knowledge Data Base System (Miller et al. 1991a, Miller et al. 1991b) is capable of representing information in different forms: Stored Data, Rules, Constraints, Models, and Problem Solving Modules. These forms allow information to be retrieved, derived, checked, generated, and produced respectively. From an Active KDL user's point of view, queries may be answered by simple data retrieval, complex query processing, querying requiring heuristic/problem solving knowledge, model instantiation, or module instantiation.

Model instantiation may occur when Active KDL does not have sufficient data or knowledge to provide a satisfactory user response by other means. In such a case, Active KDL automatically creates model instances that are executed to generate enough data to give a satisfactory reply to the user. Depending on the complexity of the query, model instantiation may be a simple or quite complex process. The process centers on the creation of

sets of input parameter values that are obtained by schema and query analysis. Model instantiation has the potential to require an enormous amount of computation in response to a query. Therefore, control heuristics (explicit meta-knowledge) must be provided to control the amount of computation. Module instantiation is very similar to model instantiation except that it uses the query specific parameters to identify, instantiate, and start the execution of a problem solving module. In the event that some aspect of the problem is unavailable (or not derivable) from the NED-IIS, the user will be prompted for the information.

We use query driven simulation as an example of the model management facility of Active KDL. Query driven simulations' fundamental tenant is that simulationists or even naive users should see a system as a sophisticated (and intelligent) information system (Miller et al. 1990, Miller et al. 1991a, Miller et al. 1991b). Systems based on query driven simulation are able to store information about or to generate information about the behavior of systems that users wish to study. Active KDL can support query driven simulation by providing access to integrated knowledge, data, model, and problem solving module bases. The three sublanguages of Active KDL (schema specification, query, and database programming language) provide strong support for simulation modeling and analysis.

An example of a problem-solving module is a genetic algorithm-based diagnosis system (Potter et al. 1990). This type of PSM would be used to determine the most likely set of disorders that best explains a set of symptoms. The input needed is the set of symptoms that indicates that the forest under consideration is not achieving its specified goals. The output includes the diagnosis or set of forest components that are causing the problem(s). The domain knowledge used to guide the heuristic search for the solution would be acquired and placed within easy access of the diagnosis module in the IIS.

In our IIS approach, a problem-solving module is invoked in much the same way that a model is invoked. That is, whenever a user query is presented to the IIS where the other forms of response processing fail to produce results a PSM may provide the proper response. The IIS meta-knowledge that deals with preparing the plan of action to be taken by the query processor uses its available meta-knowledge to determine the appropriate response path. After determining that a PSM is appropriate, certain parameters are taken from the query specification (as in query driven simulation), as well as from the information content of the IIS (possibly via the application of meta-knowledge). These items are used to instantiate the selected PSM.

An important feature of the NED-IIS user interface is the recognition of unavailable (and necessary) parameter values. The interface would prompt the user for these values and save them for later use if necessary. For example, if a stands information has been specified and certain field test results have been given to the NED-IIS, the stand manager would be in a position to query for the diagnosis (the collection of disorders that were causing the stand to deteriorate). A diagnostic problem-solving module defined for the domain would be invoked to respond to the query. Critical information that was unavailable to the diagnostic module would need to be provided before the module could be executed. The user would be informed of the missing items and prompted for their values.

The decision making process on how best to answer a query is divided into two phases: Strategic Planning and Tactical Planning. The strategic planner is responsible for making such decisions as how many answers to give, and on what combination of bases (knowledge, data, model/module) to access, for example. The strategic planner deals with trade-offs involved in providing answers to queries and also concerning the nature of the base (of knowledge, data, and models/modules) itself. Example trade-offs include storage versus computation (should all or part of the generated and/or inferred data be stored or recreated when needed), degree of precision, and degree of completeness.

Tactical planning involves detailed decision making on how to achieve a specific, concrete goal. The techniques used are algorithmic, although in many cases the algorithms will need to be heuristic (e.g., query optimization is NP-Hard). For query optimization while accessing a database, a detailed plan is produced indicating what operations (e.g., join), access paths, and indices are to be used to retrieve the appropriate data. For rule selection, given a query that requires new data to be inferred, rules will be selected/indexed based upon their relevancy, and inference will be carried out by forward and/or backward chaining depending on the situation (a combined forward/backward approach is sometimes appropriate).

DCOM – The Middleware

Based on an evaluation of a number of systems, and because most existing forest decision support systems run on Microsoft Windows platforms, we selected a DCOM-based framework for the integration of forest decision support applications (Liu 1998). An earlier prototype using NED and FVS as example applications demonstrated the effectiveness and appropriateness of integrating legacy and newly developed applications using DCOM. The implementation also indicated that, based on our previous experience with CORBA (Maheshwari 1997), DCOM programming is much easier and more productive. This is because we only focus on the application-specific implementation while the framework does many routine tasks, for example, generating the templates necessary for creating DCOM objects and registering the applications.

An additional advantage with DCOM is that we can develop user friendly interfaces using Microsoft resources. The difficulty with DCOM is that, like CORBA, DCOM is a long and complicated specification that takes a good deal of time to master.

Conceptually, the general structure of our DCOM-based framework for integration has three major components: the caller, the controller that has DCOM as its middleware, and the applications. A caller is an entity that issues a request to an application via the controller, and usually acts as an interface between the entire integrated system and the user. This interface is visual and functions differently than those of application objects in the system. It does not have a corresponding implemented object. Rather, it only gives a "look and feel" of the integrated system. The caller can interact with one or more applications to accomplish its work.

The controller (also called the intelligent information server) is responsible for locating and activating applications. More importantly, it controls interactions between the caller and an application, and between applications. The controller uses DCOM as its backbone, since DCOM provides many system services that facilitate the registration and finding of application components, and the control of and communications between them. While running an application, the controller has the duty of managing the dialog with the user, for example, screen handling, data entry and validation, dialog box control, and menu interpretation. Sometimes, the controller may have the duty to display to the user the results passed to it by an application. An additional responsibility of the controller is to handle errors that may occur during an application's execution. Because of this, the entire process executes in a seamless way. Adding a new application to or replacing an existing one in the integrated system has minimal effect on how the framework looks to the user due to its "plug-and-play" nature. Depending on the complexity and need of the integrated system, the controller may also contain processing rules that help interpret the requests and instructions supplied by the user. Therefore, only through the controller can the applications participate in a coordinated fashion with the integrated information system.

An application is a component that provides services to the integrated system. Many of the forest decision support applications focus on single simulation, display, input/output, or analysis tasks. Each application is encapsulated within an interface that follows a standard format (we use MIDL the Microsoft Interface Definition Language). This approach makes it possible for the application to communicate with the rest of the framework, such that other applications can use the interface to access the services this application provides. Interfacing also provides an effective way to deal with legacy applications. Many legacy applications were developed with a stand-alone purpose. Their data and functionality may not be readily available to other applications; the APIs of those legacy applications may be proprietary, limited, or even lacking. Newly constructed interfaces to the legacy applications act like adapters so that these legacy applications and the rest of the framework can work together, hence enabling re-use of existing applications.

The architectural design should be general purpose, meaning that the framework should have distributed processing capability and provide platform independence so that it is ready to work in heterogeneous, cross-network environments if it is required to do so in the future. Overall, the design is general and makes no assumptions about the software applications to be integrated. its standardized interface scheme enables integration of a variety of applications. It is an open framework in the sense that application components can be added and/or removed easily without drastically affecting the htnctionality of the whole system. The adoption of DCOM as the middleware supports this design. Since most of the applications the USDA Forest Service is intending to integrate are running on Windows operating systems, DCOM is certainly a viable and logical choice.

The NED-IIS Prototype

Currently, we are integrating three legacy forest service applications, namely Forest Vegetation Simulator (FVS), FIBER, and SILVAH using the DCOM-based framework. The following paragraphs discuss the three applications in detail.

FVS is a system that uses common forest inventory information and a growth/yield model for projecting the growth of forest stands. FVS simulates growth and yield for major forest species, forest types, stand conditions and a wide range of silvicultural treatments. It is used in the U.S. forest management field and there are currently more than 20 variants for 20 different geographic regions covering much of the commercial forestland in the U.S. Numerous post processors have been developed for FVS. Post processors are independently developed computer programs that can be used along with FVS to further process the FVS simulation results for specific analysis needs. For example, the Average Summary Table post processor calculates an average summary table from the stand output of FVS. Typically, the simulation results from the FVS are used as input to the post processors that perform additional analysis. This feature makes the FVS a very useful and versatile tool because virtually limitless applications can be developed in the form of post processors without major modifications to the core FVS system. FVS is written in Fortran-77 and runs on PC and UNIX workstations (Teck et al. 1996, Teck et al. 1997).

FIBER is a forest growth model that predicts the growth interactions among species of the Northeastern

United States over a complete range of forest treatments (clear cutting to unmanaged stands), stand densities, harvest intervals, species composition, and different ecological land classifications. The user can use the software to predict the growth and yield over a specified time interval for individual forest stands or large forested areas. FIBER can be used for both even-age stand and multi-age stand management (Solomon et al. 1992).

SILVAH is a forest prescription model used for prescribing silvicultural treatments for Hardwood stands of the Alleghenies. It identifies important factors and determines how they function in regulating regeneration or stand growth. SILVAH also develops objective guidelines and prescribes optimal silvicultural treatments to achieve management goals. These guidelines have been integrated into a complete stand analysis and prescription procedure that provides a systematic way of measuring and evaluating critical stand conditions. This data is then used to arrive at a recommended treatment. The stand inventory and site factors are summarized and analyzed to evaluate the stand's potential growth and regeneration. Then, decision tables are used to determine proper prescription procedure based on the critical levels of the various factors in combination with landowner objectives (Marquis et al. 1992).

The communication between these three applications and the client interface is through the intelligent information server or the controller. The design makes no assumptions about the locations of the applications. The applications may reside on the local machine or on a remote machine. The controller is responsible for locating and activating the application whether it resides locally or remotely. The controller also contains processing rules that help interpret the user's needs. These processing rules determine the kind of information that must be collected from the user. Thus there is continuous interaction between the caller and the controller. The processing rules also determine which application to activate.

For example, ' if the user wants to have an assessment of bark beetle risk factors in a forest stand, the controller will decide that FVS is the right application to be used. It will also determine the right post processor to be used and guide the user through a visual interface to supply appropriate information that is needed to run FVS with this particular post processor. If the user wants to know the appropriate silvicultural treatments to achieve a certain management goal, the controller will decide to invoke SILVAH. If more than one application needs to be activated, the processing rules will determine the dependency between the applications and activate the applications in the right order. Under such a scenario, the output from one application may be needed as input to another application. The communication between applications is also routed through the controller. The controller activates the first application, and after receiving the output it may modify the output to achieve an acceptable input format for the second application. It then invokes the second application with this input. The controller also processes all outputs from the applications.

It is necessary to route all communications through the controller because the entire integrated system is composed of independent heterogeneous applications. We can make no assumptions about the language, data types and functionality of the individual applications. Each application participates in the integrated system through its wrapper. The wrapper around the application provides an interface to the application in a standard format. The controller accesses the application through the interface provided by its wrapper. New interfaces permit legacy applications and the rest of the framework to integrate seamlessly and work together. Addition of other applications can be done without affecting the existing integrated system.

Conclusions

We have designed a DCOM based framework for integration of legacy and future forest decision support applications. The design makes no assumptions about the individual software applications and is therefore a general model that will permit seamless integration of other legacy applications and future applications. The adoption of DCOM as the middleware provides the model with the capacity to run remote applications. From the client's point of view, the location of the application is not an issue. The application can be run locally or remotely as needed. We have built a visual interface common to our three sample applications that is capable of locating and running FVS, FIBER or SILVAH on remote or local machines. This has been achieved by building wrappers around each of these applications. The caller is able to access the functionality of the applications through these wrappers. We are currently working on the IIS component of the system, by adding rules to the controller.

References

Gamma, E., R. Helm, R. Johnson, and J. Valissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley. 395 p.

Grimes, R. 1997. Professional DCOM Programming. Birmingham, U.K: Wrox Press Ltd. 565 p.

Heiler, S. 1995. Semantic interoperability. ACM Computing Surveys 27(2): 27 I-273.

Leppinen, M., P. Pulkkinen, and A. Rautiainen. 1997. Java- and CORBA-based network management. Computer 30(6): 83-87.

Liu, S. 1998. Integration of Forest Decision Support Systems: A Search for Interoperability. Master's Thesis. Athens, GA: The University of Georgia. 122 p.

Manola, F. 1995. Interoperability issues in huge-scale distributed object systems. ACM Computing Surveys 27(2): 268-270.

Maheshwari, S.S. 1997. A CORBA and Java Based Object Framework for Integration of Heterogeneous Systems. Master's Thesis. Athens, GA: The University of Georgia. 92 p.

Marquis D.A., R.L. Ernst, S.L. Stout. 1992. Prescribing Silvicultural Treatments in Hardwood Stands of the Alleghenies (Revised). USDA, General Technical Report NE-96.

Microsoft. 1995. The Component Object Model Specification. Draft Version 0.9. http://premium.microsoft.com/msdn/library/specs/tech1/d1/s1d139.htm

Microsoft. 1996. The component object model: Technical overview. http://www.microsoft.com/oledev/olecom/

Microsoft. 1997. Distributed Component Object Model Protocol DCOM 1.0. http://premium.microsoft.com/msdn/library/techart/msdn_dcomprot.hhn

Microsoft. 1998. DCOM: A business overview. http://www.microsoft.com/oledev/olecom/

Miller, J.A., W.D. Potter, K.J. Kochut, and O.R. Weyrich. 1990. Model Instantiation for Query Driven Simulation in Active KDL. Proceedings of the 23rd Annual Simulation Symposium. Nashville, TN, pp. 15-32.

Miller, J.A., K.J. Kochut, W.D. Potter, E. Ucar, and A.A. Keskin. 1991a. Query Driven Simulation in Active KDL: A Functional object-Oriented Database System. International Journal in Computer Simulation. 1, 1, pp. I-30.

Miller, J.A., W.D. Potter, K.J. Kochut, A.A. Keskin, and E. Ucar. 1991b. The Active KDL Object-Oriented Database System and Its Application to Simulation Support. Journal of Object-Oriented Programming - Special Issue on Databases. 4, pp.30-45.

Mowrer, H. T., K. Barber, J. Campbell, N. Crookston, C. Dahms, J. Day, J. Laacke, J. Merzenich, S. Mighton, M. Rauscher, K. Reynolds, J. Thompson, P. Trenchi, and M. Twery. 1997. Decision Support Systems for Ecosystem Management: An Evaluation of Existing Systems. General Technical Report RM-GTR-296. Fort Collins, CO: USDA Forest Service, Rocky Mountain Forest and Range Experiment Station. 154 p.

OMG. 1997. The common object request broker: architecture and specification, Version 2.1. OMG Document, Object Management Group.

Otte, R., P. Patrick, and M. Roy. 1996. Understanding CORBA (Common Object Request Broker Architecture). Upper Saddle River, NJ: Prentice Hall.

Potter, W.D. and L. Kerschberg. 1986. A Unified Approach to Modeling Knowledge and Data. Proceedings of the IFIP TC2 Conference on Knowledge and Data (DS-2). Pp.v1-v27.

Potter, W.D. and R.P. Trueblood. 1988. Traditional, Semantic and Hyper-Semantic Approaches to Data Modeling. IEEE Computer. 21, 6, pp53-63.

Potter, W.D., R.P. Trueblood, and C.M. Eastman 1989. Hyper-Semantic Data Modeling. Data & Knowledge Engineering. 4, pp. 69-90.

Potter, W.D., B.E. Tom, M.R. Hilliard, G.E. Liepins, S.L. Purucker and R.T. Goeltz. 1990. Diagnosis, Parsimony, and Genetic Algorithms. Proceedings of the Third Int. Conf. on Industrial & Engineering Applications of AI and Expert Systems. pp. 1-8.

Potter, W. D., T. A. Byrd, J. A. Miller, and K. J. Kochut. 1992. Extending decision support systems: The integration of data, knowledge, and model management. Annals of Operations Research 38: 501-527.

Potter, W. D., J.A. Miller, and K.J. Kochut. 1994. A Hyper-Semantic Approach to Intelligent Information Systems. Integrated Computer-Aided Engineering. Vol. 1, No. 4, pp. 341-357.

Rauscher, H. M. 1999. Ecosystem management decision support for public forests: A review. Forest Ecology and Management 114: 173-197.

Rauscher, H. M., R. P. Kollasch, S. A. Thomasma, D. E. Nute, N. Chen, M. J. Twery, D. J. Bennett, and H. Cleveland. 1997. NED-1: A goal-driven ecosystem management decision support system: Technical description. Proc. of GIS World '97. pp. 324-332.

Sheth, A. 1998. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. Interoperating Geographic Information Systems. M.F. Goodchild, M.J. Egenhofer, R. Fegeas, and C.A. Kottman (eds). Kluwer Pub. Co.

Solomon, D.S., D.A. Herman, W.B. Leak. 1995. FIBER 3.0: An Ecological Growth Model for Northeastern Forest Types. USDA, General Technical Report NE-204.

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

Teck, R., M. Moeur, and B. Eav. 1997. The forest vegetation simulator: A decision-support tool for integrating resources science. http://www.fs.fed.us/ftproot/pub/fmsc/fvsdesc.htm

Twery, M.J., D.J. Bennett, R.P. Kollasch, S.A. Thomasma, S.L. Stout, J.F., Palmer, R.A. Hoffman, D.S. DeCalesta, J. Hombeck, H.M. Rauscher, J. Steinman, E. Gustafson, G. Miller, H. Cleveland, M. Grove, B. McGuinness, N. Chen, and D. E. Nute. 1997. NED-1: An integrated decision support system for ecosystem management. Proceedings of the Resource Technology '97 Meeting. pp. 331-343.

Wegner, P. 1996. Interoperability. ACM Computing Surveys 28(1): 285-287.