NetWeaver

Michael C. Saunders and Bruce J. Miller

Abstract NetWeaver is one of the foundational technologies of the Ecosystem Management Decision Support (EMDS) system. NetWeaver's graphical interface, real time evaluations, fuzzy-logic-based measures of uncertainty, and overall ease of use led to it being chosen as a major component of EMDS. By way of background on this stand-alone knowledge engineering tool, and in order to provide enhanced perspective on EMDS and its inner workings, we first present a description of the origins of NetWeaver, and how and why it was developed.

Keywords NetWeaver · Fuzzy logic · Decision support

1 Background

A computer-based model of human expertise is called a knowledgebase. NetWeaver is a knowledgebase development system that provides a graphical environment in which to construct and evaluate knowledgebases (Saunders and Miller 1997) built with dependency networks (see definition below). Before the advent of NetWeaver, we built knowledgebases by drawing dependency networks on whiteboards or flipcharts. These drawings were then handed off to a knowledge engineer to code. Before C++ was available, the dependency network drawings were hard coded as if-then statements in the C language.

M. C. Saunders (✉)
Department of Entomology, Pennsylvania State University, 501 Agricultural Science and Industries Building, University Park, PA 16802, USA
e-mail: mcs5@psu.edu

B. J. Miller
Rules of Thumb Inc., 11817 Cedar Mill Rd., North East, PA 16428, USA
e-mail: bjmiller@gmail.com

K. M. Reynolds et al. (eds.), *Making Transparent Environmental Management Decisions*, Environmental Science and Engineering,
DOI: 10.1007/978-3-642-32000-2_2,

The next stage of development was to separate the coding of knowledgebases from their code implementations. This meant creating a LISP-like scripting language to represent the knowledgebases and the creation of an inference engine to read and implement the new knowledgebases. In computer science, an inference engine is an application that acts to interpret a knowledgebase. The scripting language implementation had two main advantages: (1) a non-programmer could easily code a knowledgebase, and (2) a knowledgebase could be edited independently from the inference engine.

This early implementation led to large increases in our ability to produce knowledgebases. However, limitations of this approach became evident when a project came our way that involved 100s of dependency networks drawn by a subject matter expert (SME). Hand coding the scripts revealed two flaws: they were tedious to edit (but not nearly as tedious as hard-coding!), and it was difficult to verify that they accurately depicted the hand-drawn dependency networks.

Facing an overwhelming amount of work, a graphical dependency network editor was conceived. Enter NetWeaver. Conceived and written over winter break of 1991, it originally only displayed edited dependency networks, saving the knowledgebases in the same scripting system. But now graphical editing was possible: click a button to add a node, click a button to move a node. It was truly a what-you-see-is-what-you-get editor, so the flow of changes was easy to handle, and verification was as simple as comparing the hand-drawn dependency network to the one on the screen.

Once the dependency network was coded correctly, the next issue was ensuring that a given dependency network represented its intended logic correctly. The chosen solution was to integrate the inference engine into NetWeaver so that the user could test the graphical model with data. Now, knowledgebases could be designed and verified live. No more disjointed design, coding, and testing loops, and a SME could independently build knowledgebases and get real-time verification.

Over the years, NetWeaver has evolved and its capabilities have expanded. To deal with missing data and with qualitative concepts, we developed evaluation algorithms based upon fuzzy math (Saunders et al. 2005) that worked well (see below for more details).

In the early years of the 21st century, NetWeaver underwent a thorough rewrite to NetWeaver2. NetWeaver2's main improvements were:

- Internationalization—for both development and deployment.
- Runtime applications—deployable self-contained knowledgebase applications.
- Documentation—robust internal and exported documentation at all levels of the knowledgebase, and with all the capabilities of a modern word processor.
- Binary file format—to make the software smaller, faster, more extensible.
- Knowledgebase security—password protection for various aspects of the knowledgebase.
- Automated documentation—exporting the knowledgebase to HTML documents, complete with all linkages and embedded documentation.

(see http://rules-of-thumb.com/development_plans for a more complete review)

2 NetWeaver Concepts

Knowledgebase systems come in a variety of forms, but the dominant types currently in use are rule-based systems (Parsaye and Chignell 1988). Knowledge representation in NetWeaver, in contrast, is based on object-oriented fuzzy-logic networks (dependency networks). These types of dependency networks offer several significant advantages over the more traditional rule-based representation.

Compared to rule-based knowledgebases, NetWeaver knowledgebases are easier to build, test, and maintain because their underlying object-based representation makes them modular. The modularity of NetWeaver knowledgebases, in turn, allows the designer to incrementally evolve complex knowledgebases from simpler ones. Modularity also allows interactive knowledgebase debugging at any and all stages of development, which expedites the process.

Finally, fuzzy logic provides a formal and complete calculus for knowledge representation that is less arbitrary than the "confidence factor approach" (Negoita 1985), used in rule-based systems, and more parsimonious than bivalent rules.

Although the term "fuzzy logic" has a distinctly esoteric ring to it, the concept is actually quite simple. Fuzzy logic provides a metric for expressing the degree to which an observation on some variable belongs to a set that represents that variable. Alternatively, one might say that fuzzy logic is concerned with "aboutness." To make the concept clear, consider the following example.

Everyone has some concept of what it means to be an adult. For legal purposes, an adult, in western cultures is often defined to be a person who is 21 years old or older. A rule-based system dealing with legal issues can easily accommodate this bivalent definition: if a person is 20 years, 11 months, and 30 days old, they are not an adult, but if the person is at least one day older than 21 years, they are an adult. This characterization of adultness is sufficient if the concept of adult is limited to a simplistic legal one. However, if by adultness we instead are really interested in expressing something more complex such as an individual's emotional maturity, then the simple bivalent rule for determining adultness is no longer adequate. Most people would agree that a five-year-old has no, or at best minimal, adult qualities. In a 13-year-old, however, we might begin to see at least some early signs of adult characteristics. Some 18-year-olds demonstrate many adult qualities (they act very "grown up"). Conversely, most people can think of at least a few 25-year-olds they have met in their life that could not be called particularly emotionally mature. Thus, as a first step toward improving the characterization of adultness, one might construct a simple fuzzy membership curve that translates age into degree of membership in the set "adult."

We indicated that fuzzy logic allowed a more parsimonious knowledge representation than that which is possible with rule-based systems. The reason is simple. A single fuzzy membership function is sufficient to express the full spectrum of adultness. In contrast, rule-based systems are inherently bivalent, meaning that a rule is either true or false. To more precisely characterize adultness in a rule-based system, one would need to define, say, five age categories

Table 1 The NetWeaver logical node types and their function

OR	An OR node is true when any one of its antecedents is true. It is false when all of its antecedents are false. Functionally, it passes the value of its most true antecedent
AND	An AND node is true when all of its antecedents are true. It is false when any one of its antecedents is false. Functionally, it performs a weighted average of the values of its antecedents unless one of the antecedents is fully false. Compare this with the next definition of UNION
UNION	A UNION is true when all of its antecedents are true. It is false when all of its antecedents are false. As a practical distinction between AND and UNION nodes, antecedents to AND function like limiting factors, whereas antecedents to UNION function like compensating factors
NOT	A NOT node simply inverts the value of its antecedent
SOR	A SOR node (sequential OR) is a special class of node designed to select between alternative decision scenarios where there is a definite hierarchy of quality level associated with each possible data gathering method. In other words, the SOR node is a data route selector; it provides a method for selecting the best choice of paths within the scope of the currently given data. For example, the preferred path may involve decision making on the basis of acid neutralizing capacity (ANC), but if ANC is missing, then the decision can be based on an alternate parameter such as conductivity or pH. Connections to the antecedents of a SOR node are represented with dotted lines to indicate their relative position in the hierarchy
XOR	A XOR node (exclusive OR) is true when one and only one of its antecedents is true

corresponding to different levels of adultness, and each category would require a rule. Moreover, if our rule base also dealt with intelligence, and this attribute similarly had five categories (ranging from brilliant to ignorant, for example), then to jointly consider both adultness and intelligence in our rule base could require as many as 25 additional rules. In contrast, in a fuzzy-logic-based representation of this more complex situation, we only need one more fuzzy curve and perhaps a new network object to jointly evaluate the two fuzzy curves. So, in our example, two fuzzy curves have an expressive power that is equal to or better than 35 ($10 + 25$) rules. To summarize, the number of rules needed to adequately represent possible outcomes explodes approximately combinatorially, whereas the number of fuzzy curves and related objects needed to describe the same problem in an object-oriented fuzzy logic representation increases approximately linearly.

NetWeaver uses OR, AND, UNION, NOT, and XOR and SOR logic nodes to define the logical dependency of a network on antecedent networks, and on data links (Table 1). A data link is a type of dependency network object in NetWeaver. A data link is where data are interpreted by an argument (i.e., simple data links), or where data are used in a mathematical expression, the result of which is interpreted by an argument (i.e., calculated data links). If no data links antecedent to a network use fuzzy arguments, then the operation of these nodes conforms quite closely to their usage in conventional logic. The only real difference in this context between these nodes, as used in NetWeaver, and in standard logic is that true $= 1$, false $= -1$ in NetWeaver.

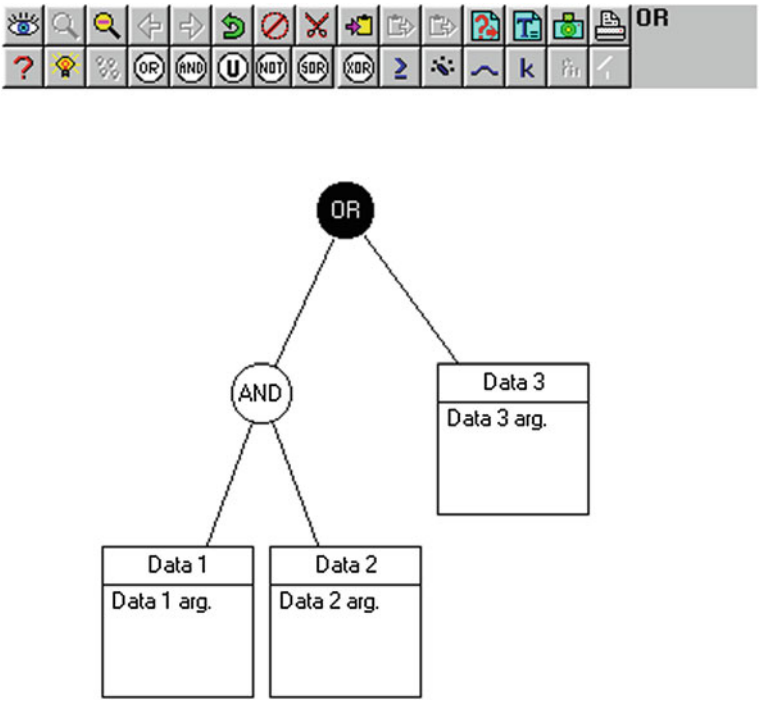

Fig. 1 A dependency network as represented in NetWeaver. In this dependency network, there are three data links represented by the squares at the bottom of the figure. Each of the data items is evaluated relative to the degree to which it satisfies its arguments. The network can be read as a rule as follows: "IF Data 1 satisfies the argument Data 1 arg. AND Data 2 satisfies the argument Data 2 arg. OR Data 3 satisfies the argument Data 3 arg. THEN the assertion is true." The degree to which the assertion is true is a function of the degree(s) to which the individual data satisfy their arguments and the types and arrangements of the logical nodes used within the network

NetWeaver allows simple or calculated data links to take fuzzy arguments to determine a data value's membership in a fuzzy set. In order for fuzzy set membership to be propagated through a knowledgebase, the definitions of the conventional logical operators OR, AND, NOT, and XOR have been extended to handle measures of fuzzy-set membership (Table 1). The SOR node object is unique to Net-Weaver, and we describe its operation in a later example.

As previously mentioned, models in NetWeaver are based on dependency networks which are graphical depictions of rules (Fig. 1). At the bottom of a dependency network are data links (e.g., Data 1, Data 2), which are used to hold, fetch, or modify raw data. There are two types of data links; simple and calculated. Simple data links fetch and hold data from various sources (databases, GIS map layers, direct data input, environmental variables, and other sources). Calculated data links modify data (e.g., calculate an ecological index or a mathematical relation from raw data) through networks of calculation nodes chosen from a toolbox of arithmetic,

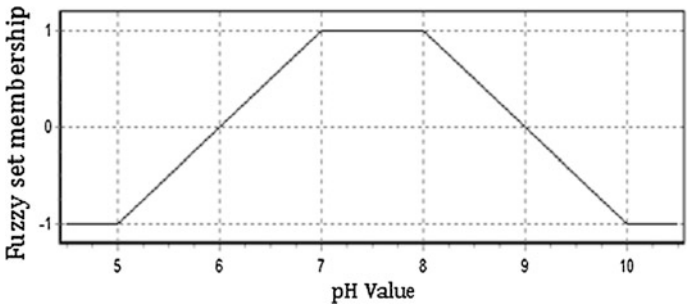

Fig. 2 A fuzzy argument used for interpreting the pH value of a stream. The fuzzy membership is shown on the Y axis with −1 indicating no fuzzy set membership (i.e., False) and 1 indicating complete membership in the fuzzy set (i.e., True). For this example, pH values between 7 and 8 fully satisfy the argument and indicate that the pH is indicative of a healthy stream. pH values less than 5 and greater than 10 are unacceptable pH values for a healthy stream

trigonometric, selection, summation, and other tools. Both types of data link are visually represented as a square object in a dependency network.

To provide a "trueness" level that can be used in a dependency network, the data within a data link are compared to an "argument." Arguments can be reference conditions, ecological thresholds, ecological index set points, or other types of indicator measures (e.g., single values or ranges of pH values, ranges of water-temperature values). NetWeaver provides two types of arguments, the standard argument and the fuzzy argument. The standard argument compares data values against an argument to return a TRUE or FALSE value (or undetermined when data are absent). An example of a standard argument is presence (TRUE) or absence (FALSE) of a particular species. The fuzzy argument compares the data values against a fuzzy set membership function that returns a level of trueness based on the degree of membership in the fuzzy set. In NetWeaver, fuzzy set membership is measured on a scale of −1 (no membership in the fuzzy set TRUE, which is equivalent to 100 % FALSE), to 0 (UNDETERMINED in the case of no data, or if there are data provided, it represents 50 % membership in the fuzzy set TRUE), to 1 (complete membership in the fuzzy set, which is equivalent to 100 % or completely TRUE). There are four break points provided to define a fuzzy argument within a data link, each of which can be defined to be TRUE, UNDETERMINED, or FALSE. An example of a fuzzy argument is the range of pH that is ideal to support aquatic organisms (Fig. 2).

3 Why Use NetWeaver Knowledgebases

Knowledge-based reasoning is a general modeling methodology in which phenomena are described in terms of abstract entities and their logical relations to one another (Holsapple and Whinston 1996). There are two basic reasons for using knowledge-based reasoning:

- The entities or relations involved in the problem to be solved are inherently abstract, so that mathematical models of the problem are difficult or even impossible to formulate.
- A mathematical solution is possible in principle, but current knowledge is too imprecise to formulate an accurate mathematical model.

Both cases are common. The first case naturally arises when the nature of the problem involves relatively abstract entities. These problems may simply be easier to solve with logic. The second case arises very frequently, particularly when dealing with ecosystems, because there are an almost unlimited number of relations of potential interest. Agencies, academia, and others have developed numerous mathematical models to describe some of the important relations of interest to ecosystem management, but many relations have not been studied in sufficient detail to provide generally applicable mathematical models. However, there is often a wealth of human experience in these same institutions that can be drawn upon to develop useful, more qualitative, knowledge-based models to guide decision making.

Another valuable aspect of knowledge-based reasoning that makes it ideal for use in environmental assessment is that such systems can provide clear reasoning with incomplete information. The NetWeaver engine provides partial evaluations of system states and processes based on available information, and provides useful information about the influence of missing data, which can be used to improve the logical completeness of an assessment.

In its most basic form, a NetWeaver knowledgebase is a collection of dependency networks. It can also include such things as supporting documentation and hyperlinks. A NetWeaver knowledgebase represents relations among concerns, system states and processes, and data requirements. Uses of dependency networks include:

- Evaluation of the truth value of assertions about system states and processes, given existing data.
- Identification of data requirements for an analysis.
- Ranking of missing data in order of relative importance to the analysis.

One of the virtues of a dependency-network representation is that a single knowledgebase may incorporate a very wide variety of topics. This is particularly valuable in the context of ecological assessments in which topics of interest might include, for example, many different topics and subtopics related to terrestrial vegetation and wildlife habitat conditions, native fish population status, available recreation opportunities, water and air quality conditions, visual and aesthetic concerns, and commercial concerns or opportunities. The number of topics and the interrelations that can be represented in a knowledgebase is only limited by the state of knowledge held by SMEs, and by a computer's dynamic memory.

4 Evolving Knowledgebases

NetWeaver is a rigorous, object-oriented, knowledgebase development system. One of the more practical implications of object-oriented knowledgebases is that it is very easy to start with simple knowledge representations and gradually to evolve them into large, complex systems because they are extremely modular. A basic modeling principle that has motivated development and application of object-oriented technology, in general, is well captured by Gall (1978):

> A complex system that works is invariably found to have evolved from a simple system that worked… A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.

NetWeaver is designed to build a knowledgebase in an incremental and evolutionary fashion. The best possible advice we can give to the novice user is to avoid designing large, complex systems at the outset, and instead, start with a simple knowledgebase, built from a small number of dependency networks, and gradually evolve this simple representation into a more complex representation of the problem.

As another practical matter, we have found that it is almost always best to start at the top with the highest-level (primary) dependency networks that apply to the problem domain, and develop the structure downward. To get started, create and document at least a few of the primary dependency networks. It is not necessary to identify an exhaustive list of primary networks at the outset. Because of the modular structure of NetWeaver knowledgebases, new dependency networks can easily be added later in development without upsetting overall knowledgebase structure.

For each primary network in the knowledgebase, create antecedents. Unless it is an unusually simple knowledgebase, there will usually be at least one or two levels of antecedents before a chain of dependencies terminates in a data link.

In a completed knowledgebase, you will normally want to be sure that each chain of dependencies terminates in a data link. However, while a knowledgebase is under development, it is always possible to evaluate a network object, regardless of how complete the network structure is.

As a knowledgebase structure evolves, you will probably find occasion to use existing antecedents (both dependency networks and data links). Multiple occurrences of dependency networks and data links in a single knowledgebase are not a problem because NetWeaver objects are reusable. In fact, the presence of a dependency network in two or more other networks within the same knowledgebase is an important mechanism by which networks are interrelated through shared antecedents.

5 Applications

NetWeaver has been used to develop a broad array of knowledge-based models, many of which are detailed in this book and in Wikipedia (http://en.wikipedia.org/wiki/Ecosystem_Management_Decision_Support). For purposes of illustration, we refer to a recent effort that sought to characterize watershed conditions within the Delaware Water Gap National Recreation Area (DEWA) and Upper Delaware Scenic and Recreational River (UPDE) (see Mahan et al. 2011 for a complete description of this model).

The focus of this modeling effort was on natural resources at these two park units; however, our assessment was conducted at the watershed scale in each park. The assessment was developed to assist superintendents and natural resource managers with: (1) strategic planning, (2) general management planning, (3) park reporting on land health goals, and (4) overall natural resource management and conservation.

As a positive consequence of employing a NetWeaver modeling approach for this systematic natural condition assessment, we delivered not only the required final report complete with tables and maps, but also an operational system that park managers can revisit as new data become available. Periodic re-running of the delivered NetWeaver model will allow managers to generate new reports to assess the new conditions and to identify any trends.

All data used in this assessment were compiled from relevant reports, scientific literature, and data files, and were initially managed using an online Wiki tool. We used these data and information available in the scientific literature to develop thresholds for our overall natural resource assessment model. We also used a variety of GIS-based analytical models to synthesize landscape data and develop indices of landscape condition across the two parks. These landscape condition indices included input data on impervious surfaces, forest fragmentation, and land use within watersheds at both parks.

The overall assessment examined a variety of aquatic and terrestrial ecosystem components and their interactions. Components included: chemical and physical elements of water quality, biologic elements of water quality, and forest landscape condition elements. For the water quality chemical and physical elements, we used a water quality index to assess water quality in each watershed. For the water quality biologic elements we used the Ephemeroptera, Plecoptera, Tricoptera (EPT) and the Hilsenhoff indices. Finally, for the landscape forest condition elements of our model, we used percentage area in forest, and percentage area with impervious surfaces measures. In addition, the DEWA/UPDE model contains natural resource elements that were not included in the overall assessment per se, but may be useful for management of park resources. For instance, our model included information on the number of stream crossings, dams, road miles, and rare species per watershed, and in each park.

In most cases, data did not come from single sources. For many elements, the data sources were varied. For example, pH values for any given watershed could

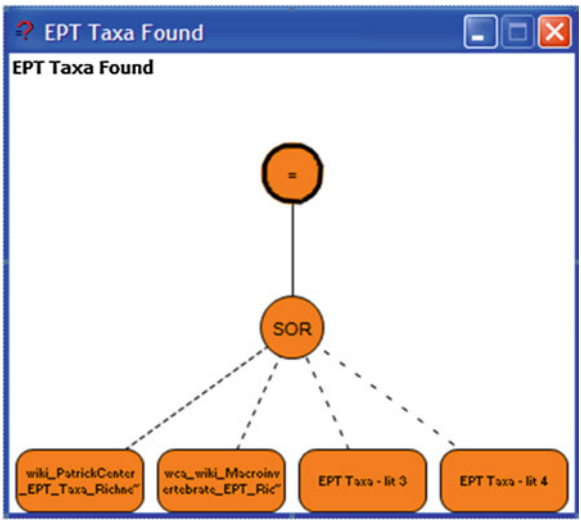

Fig. 3 Use of a sequential OR (SOR) node for the preferential selection of data from multiple data sources

have come from a handful of published reports, National Park Service (NPS) curated databases, or from gauge-station data of the US Geological Survey. NetWeaver facilitated the prioritization of these disparate data sources for a watershed based on the available data for a given watershed. The ability to aggregate data sources greatly enhanced the coverage of the analyses.

In some cases, multiple analyses were performed using competing analytical measures to observe differences in results, such as when using EPT or Hilsenhoff index when evaluating water quality, with respect to available aquatic insect taxa (US EPA 2002). Where data were sufficient, the results of the competing methods could be compared. Where data were lacking for one or the other method, the results from the method with sufficient data were used to represent conditions, according to NPS preference.

The logical nodes that are available for use within NetWeaver are shown in Table 1. In addition to purely logical operators such as AND, OR, and NOT, there are other operators that can be useful when dealing with multiple sources of data that vary in terms of desirability, accuracy, quality, and relevance. In the DEWA/UPDE model, the sequential OR (SOR) node was used frequently. This node (Fig. 3, EPT taxa found) provides the modeler with the ability to specify the order in which the model will use data, and in so doing, consistently apply the best available data to the model. In this example, the data for extant EPT taxa within a given watershed could be found in at least four reports. Some of these reports (and the data within) were preferred over others, often based on the originating agency of the report, report recency, and other factors. In Fig. 3, we show how a SOR node can be used to ensure that the most preferred data source is used. The SOR node always ensures that the left-most source of extant data is used. If there are no data present, the SOR node seeks its value from the next data source to the right. In the case of the DEWA/UPDE watershed model, any EPT value from any of the

four sources was evaluated the same. However, cases arise whereby a modeler may wish to interpret less reliable data in a more conservative manner than would be applied to more reliable data. This can be easily implemented by attaching a data link or dependency network to the SOR node that is designed to evaluate these less reliable data sources.

6 Conclusions

NetWeaver software provides graphic tools for constructing executable dependency networks that permit both forward- and backward-chained reasoning. Because the inference engine is integrated, networks can be evaluated in real-time with nodes changing color to indicate their changing "trueness" levels. This ability to peer into the logical workings of a knowledge network greatly optimizes the knowledge engineering process by:

- Providing the ability to run and evaluate freshly elicited knowledge in the presence of the domain expert(s).
- Enabling the knowledge engineer to trace the logic structure from data to conclusions.
- Allowing the knowledge engineer to quickly identify and edit errors and inconsistencies in the logic.
- Providing consistent analyses across a landscape.
- Providing intelligent prioritization of data and results.
- Allowing competing analytical methods to be employed simultaneously.

References

EPA US (2002) Methods for measuring the acute toxicityo of effluents and receiving waters to freshwater and marine organisms. Document no. EPA-821-R-02-012, Washington

Gall J (1978) Systemantics: how systems work and especially how they fail. Pocket books, New York, 158 pp

Holsapple C , Whinston A (1996) Decision support systems: a knowledgebased approach. West Publishing Co., Eagan, 713 pp

Mahan CG, Miller BJ, Saunders MC, Young JA (2011) Assessment of natural resources and watershed conditions for Delaware Water Gap National Recreation Area and Upper Delaware Scenic and Recreational River. U.S.D.I National Park Service, Natural Resource Report NPS/NER/NRR 2011/429–NPS 620/108557; 647/108557, July 2011, 90 pp

Negoita CV (1985) Expert systems and fuzzy systems. Benjamin Cummings Pub. Co., San francisco, 190 pp

Parsaye K, Chignell M (1988) Expert systems for experts. Wiley, New York, 462 pp

Saunders M C, Miller BJ (1997) A graphical tool for knowledge engineers designing natural resource management software: NetWeaver™. Proc ACSM/ASPRS Ann Conv Res Technol 4:380–389

Saunders MC, Sullivan TJ, Nash BL, Tonnessen KA, Miller BJ (2005) A knowledgebased approach for classifying lake water chemistry. Knowl Based Syst 18(1): 47–54

http://www.springer.com/978-3-642-31999-0

Making Transparent Environmental Management Decisions
Applications of the Ecosystem Management Decision Support System
Reynolds, K.M.; Hessburg, P.F.; Bourgeron, P.S. (Eds.)
2014, XIX, 337 p. 69 illus., 49 illus. in color., Hardcover
ISBN: 978-3-642-31999-0