Design of integrated forest resource information systems

Douglas K. Loh & Hannu Saarenmaa

TIIVISTELMÄ: INTEGROITUJEN METSÄTIETOJÄRJESTELMIEN SUUNNITTELU

Loh, D.K. & Saarenmaa, H. 1992. Design of integrated forest resource information systems. Tiivistelmä: Integroitujen metsätietojärjestelmien suunnittelu. Silva Fennica 26(2): 111–122.

Managing forests and other natural resources requires merging of data and knowledge from many fields. Research efforts in many countries have simultaneously aimed at computer applications to help managing the large amounts of data involved and the complexities of decision making. This has invariably led to large integrated systems. An integrated system is software that consists of modules for various tasks in natural resource management such as data base management, spatial analysis, simulation and optimization, diagnostic reasoning, and communicating with the user. The paper presents an overview of the need, levels, and historical development of integrated systems. Newly emerged technologies, especially object-oriented programming and the X Window System with its associated environment have given new flexibility and transparency to the designs. The client-server architecture is found out as an ideal model for integrated systems. The paper describes an implementation of these ideas, the INFORMS system that supports the information needs of district level forest management planning.

Metsien ja muiden luonnonvarojen hoito edellyttää monista eri lähteistä saatavan tiedon yhdistelyä. Tällä hetkellä työskennellään lukuisissa tutkimus- ja kehittämishankkeissa eri puolilla maailmaa tarkoituksena rakentaa tietojärjestelmiä niiden suurten tietomäärien käsittelyyn, joita päätöksenteko edellyttää. Säännöllisesti tuloksena on ollut ns. integroituja järjestelmiä. Integroitu järjestelmä koostuu useista moduleista, jotka on tehty eri tarkoituksiin, kuten luonnonvaroja koskevan tiedon hallintaan, paikkaan sidotun tiedon analysointiin, simulointiin ja optimointiin, diagnostiseen päättelyyn ja käyttäjän kanssa kommunikointiin. Julkaisussa esitetään katsaus integroitujen järjestelmien tarpeeseen, toteutustapoihin ja historialliseen kehitykseen. Viimeaikainen kehitys, erityisesti olio-ohjelmointi ja X-ikkunointi ohjelmointiympäristöineen ovat tuoneet uutta joustavuutta ja selkeyttä järjestelmien toteutukseen. Asiakas-palvelinarkkitehtuuri on monella tapaa ideaalinen malli integroiduille järjestelmille. Tässä tutkimuksessa kuvataan yksi näiden ajatusten toteutus, INFORMS-järjestelmä, joka tukee alueellisen metsätalouden suunnittelun tarpeita.

Keywords: forest management, information systems, X Window System, databases.
FDC 62

Authors' addresses: *Loh:* Texas A&M University, Department of Rangeland Ecology and Management, College Station, Texas 77843, USA, loh@starr-gw.tamu.edu; *Saarenmaa*: Finnish Forest Research Institute, Information Systems Group, Unioninkatu 40A, SF-00170 Helsinki, Finland, hannu.saarenmaa@metla.fi.

Accepted October 7, 1992

Introduction

In recent years, computer applications have had an increasing role in natural resource management. Examples include database management systems for tracking resource inventories, geographic information systems for analyzing spatially referenced data, simulation models for projections of growth and yield, expert systems for assisting diagnostic reasoning and planning, and spreadsheets and statistical packages for summarizing data.

As the public concerns over resource conservation and environmental protection have grown, so do the number and sophistication of relevant computer tools, and the quantity and complexity of data and information. To facilitate a more efficient use of such growing information resources, today's firms and agencies are facing the important task of restructuring their information architectures to meet the emerging requirements on organizational information processing (Integrated forest... 1989, Forest service... 1990, Saarenmaa et al. 1991).

Most information system projects in the forestry sector seem to be directed toward integrated resource management systems, or integrated systems for short. For example, one year after the national GIS procurement plan was drafted, the USDA Forest Service (Forest service... 1990) decided that it is more appropriate to expand the scope of the plan toward integrated corporate information resource management. Other examples include numerous research projects in various countries that aim at integrated systems (see Buhyoff 1988, 1991, Saarenmaa 1991).

This pursuit for the integrated systems reflects the fact that computer software should be a model of the activity it is made to support (Mallach 1990). Forestry is an area that requires assimilation of knowledge from many fields of expertise before decision making. These include ecological, technological, economical and social aspects. Each of these aspects has to be considered in turn, and the consideration must be properly scheduled with the other steps of the decision making. Planning also takes time to complete and involves many parties. Consequently, the operational practices of forestry organizations are rigorously defined. Their computer applications are only reaching towards similar quality and integration.

What does this requirement of sofware as a model mean? For instance, the various windowing user environments in workstation computers adhere to a desktop metaphor. An integrated resource management system should be a model of how the various aspects of natural resources management are properly taken into account. A proper metaphor is intelligent agents acting on a landscape (Saarenmaa 1991). These models have been approached in various strategic plans of information systems (Integrated forest... 1989, Forest service... 1990, Mikkola and Saarenmaa 1992, Saarenmaa et al. 1991), but they are far from complete.

Once the models of the activies have been laid out, the following steps in systems building are analysis, design, and construction (e.g. Martin 1989). In the analysis phase, the tasks and data stores are identified. The design phase is concerned with the mechanisms of module interaction and layout of the user interface.

The purpose of this paper is to make an overview of the evolution of integrated systems in natural resource management. In the past few years important new technologies have emerged which make many old designs obsolete. In this paper we present a design that is in line with the recent advancements of information technology. It is based on a flexible client-server communication.

Levels of integration

For a system to be integrated, it must support a mechanism for data sharing and exchange between modules. If this requirement can be met, then programs such as GIS are able to reference to a firm's inventory database for spatial analysis. Similarly, queries made to the tabular database could result in graphic display of the whereabouts of the entities of interest. An information system framework that meets this minimum requirement of allowing data sharing and exchange represents the *rudimentary form* of an integrated system.

There are flaws in this rudimentary data exchange. Data consistency is not guaranteed, and data redundancy may be beyond the tolerance level. The query capability and the programming interface of the system may be rigid and limited. These flaws can be corrected, however, when a relational database management system (RDBMS) is added to the system framework.

RDBMS is important for three primary reasons. The first relates to the types of information that can be extracted from the database. A relational database has the most flexibility for finding relationships among data features that have not been physically linked, particularly those that involve combining spatially referenced and tabular attributes. The second is that the relational data model is adaptive; it does not require database elements to be predefined. As information processing needs change, new relations and data elements can simply be added to the existing database. The third is the emerging trend of using a standard language for database manipulation. Since the mid-1980s, Structured Query Language (SQL) has become the standard for relational database management systems (Date 1986, Tucker 1990). An information system framework that incorporates a RDBMS as its database foundation starts to possess the *operational form* of an integrated system.

For most resource management tasks, integrated systems of either rudimentary or operational form are sufficient, e.g. drawing the buffer zone of an endangered species protection area, or generating a timber sale report. Such tasks are called structured ones, and they require mainly independent access to individual tools such as a GIS or a spreadsheet program.

On the other hand, interpreting a firm's abstract strategy into a tangible implementation plan requires a more sophisticated data access. For that purpose, information from various sources needs to be integrated. Indeed, most forestry tasks require a number of actions to be executed before they are completed. Actions are often interconnected and may be constrained so that action A be at a certain state while action B is executed. Examples of this are planning of harvest scheduling and timber transportation while taking into account forest protection.

Identifying managerial opportunities is another example where structured access to individual tools is not adequate. A forest manager may want to query the inventory database for a graphic display of the whereabouts of the qualified entities. To him, whether a GIS is executed and how spatial data is cross referenced is irrelevant. As another example, he may want to use the expert system to evaluate a forest pest infestation problem. Again, it is not his concern as to how data pertaining to intersection of spatial layers of historical defoliation boundaries and that of the specified stands is produced, and how such data pertaining pest/host is organized and fed into the reasoning process. He is only interested in browsing the comprehensible results that relate to the problems at hand. To assist these unstructured tasks, integrated use of an array of tools becomes a necessity.

To achieve this kind of transparency, the system ought to add two additional components to its framework: a set of inter-program communication protocols and a user interface shell. Inter-program communication protocols are needed for a standardized access to the database and a common mechanism for implementing data exchange among application programs. Ideally, if all the tools possess the same layout and interface style, it would greatly reduce the amount of time needed to learn the use of those programs (Durant et al. 1987). However, it may be unrealistic to expect that all applications will deliver the same "look and feel." One pragmatic measure is to impose a user interface shell on top of the information architecture. The shell's purpose is to facilitate the use of the application programs in an integrated environment. When designed correctly, the interface shell does not hamper the direct use of the underlying application for the structured tasks. A system framework that has these two added features starts to possess the *functional form* of an integrated system.

Historical perspectives

The concept and design of integrated systems has evolved as information technology has advanced. The advancement and evolution in turn dictate and reflect the form and capabilities of the systems. Historically, there have been four known approaches toward the design of integrated resource management systems. More or less in chronicled sequence, they are (1) integration to a high level language (HLL) program, (2) integration to a shell program, (3) integration with the aid of an operating system, and (4) integration to an operating environment. While the first three will be described here, the fourth one will be deferred till the next section.

Integration to a high level program

This method of integration is perhaps the most conventional one. Its main thrust was seen during the mainframe era of computer age in the seventies and early eighties. The paradigm of integration to a HLL based program is simple: it uses a master/slave style of program structure. In this structure, the main program, written in a HLL such as FORTRAN, serves as the master

control module. The application tools such as spatial analysis functions and simulation models are arranged as a hierarchy of callable slave subroutines. Examples of this type of integration can be found in Southern Pine Beetle Decision Support System SPBDSS (Rykiel et al. 1984), Integrated Pest Impact Assessment System IPIAS (Hunter et al. 1988), and in the METIK programming standard (Mäkeläinen 1988).

An advantage of this integration method is that it does not involve programming complexity. It is basically straightforward assembling of the slave programs into a predefined data flow diagram. Because the program logic is well controlled by a sole main routine, such systems usually render reasonably high performance.

On the other hand, the rigid data flow scheme and calling sequence make the information processing rather sequential. This is not desirable when dealing with more unstructured problems. In addition, the predefined data structure hinders system upgrades. When the information processing needs of the firm change, or when any component modules are modified, a major overhaul of the system diagram and subsequent tedious modification and re-compilation become inevitable. This drawback explains at least partially why programs like SPBDSS faded away.

Integration to a shell program

In natural resource management communities, a popular approach is to use an expert system shell or a GIS package to integrate other applications. One typical method is to provide macro language support for the user to develop additional procedures, e.g. the AML of ARC/INFO (AML users... 1989). Others would furnish hooks for temporary exit from the shell to run an outside application such as a simulation model (basically a system call) and then return the control to the shell, e.g. the CLIPS Expert System Shell (CLIPS reference... 1988). For more sophisticated needs, the source or object codes can be available. The most advanced integration can be found in Smalltalk environments and in LISP-machines, where the operating systems themselves are huge object bases and extendable by users.

Examples of this type of integration are abundant. Thieme et al. (1987) used an expert system tool to combine heuristic knowledge and a network analysis algorithm in road planning for forest resource management. Folse et al. (1990) incorporated a simulation model into a GIS to make a simulation system that allows modeling of spatially heterogeneous processes in the landscape.

An advantage of this type of integration is that the hooks or programming interfaces to a shell are usually well developed and documented. And if a firm's information processing need leans toward the type of operations that require mainly the built in capabilities of a shell package (e.g. GIS for spatial analysis) and less toward other tools, then the resulting performance should be reasonably satisfactory.

The disadvantage, however, is that the integration is constrained by what the shell can provide and what platforms it runs on. Some local needs, e.g. bridge to a DBMS package or a particular user interface style, if not supported by the shell vendor or developer, will not be conveniently fulfilled. In some instances, the shell does provide some desirable hooks, but they may not be general purpose enough. For example, the database bridge of ARC/INFO to the relational database of ORACLE allows no more than five active links, which may fail user's expectation in certain circumstances. These and other constraints limit the portability and extensibility of the target system. Furthermore, the thing a firm needs to be concerned most is whether to put all eggs in one basket. As the information technology markets are becoming increasingly competitive, the vendor of a proprietary system may disappear.

Integration to an operating system

Next level of integration may occur at the operating system level. At this level, each program is incorporated as it exists into an integrated system framework, i.e. they retain all their original features. Like the first approach, usually there is also a main control program. The difference is that the needed programs are usually pre-loaded and stay resident and suspended in memory. Their activations are done by the operating system interrupt signals and key sequences. The function of the main control program is basically intercepting and parsing these key sequences. TEAMS, a relational database application for terrestrial ecosystem analysis and modeling is an example of integration at the operating system level (Covington et al. 1988).

The advantage of this type of integration is the familiarity of the user with the underlying tools. Because all original features of the tools are retained, not much learning is required. Obviously, a disadvantage of this type of integration is that it lacks the control of consistent user interface. Also, programming at this low level, there is no safety for the frequent use of system interrupts. One accidental mis-mapping of the memory may lead to system crash. Due to the rapid advancement on both operating systems and windowing environments, this approach is becoming obsolete.

Design of a modern integrated system

The ideal architecture

The pros and cons of different approaches discussed above have provided some perspective on the design of integrated systems. Next we try to shed light on the shape of the new designs to come. Although there is more than one way to depict the desirable features of a system design, a generalized one may have the construct as is shown in Fig. 1. In this construct, there is a relational corporate database that solely manages and serves data to the application client programs (including an interactive human client). The database in general can be distributed across the networks and on different computers, but it is transparent to the clients. The client programs are those tools that are essential to resource management, e.g. resource inventory system, GIS, and simulation models. They all access this corporate database. The block in the center of the diagram depicts the user interface component which is supposed to contribute to a smooth interface surface. The implementation of the model construct can be described in three parts: (1) the operating environment, (2) the system traffic, and (3) the system structure.

The operating environment

A sound operating environment is a necessary platform for the development of integrated systems. An operating environment is a higher level add-on interface to the underlying operating system. An operating system may support one or more operating environments concurrently. For example, MS-Windows is an operating environment of DOS. On the other hand, an operating environment may be supported by more than one operating system. For example, the X Window System runs on virtually every major systems including UNIX, VMS and DOS. By the virtue of being high level, an operating environment provides a higher degree of program and data abstraction. This feature makes applications more machine- and operating system-indepedent, hence greatly increasing their maintainability and portability.

Loh et al. (1988) pointed out that a sound operating environment for integrated resource management systems is the one with multi-tasking and graphic windowing capabilities. Multitasking is essential in that it allows running more than one application concurrently (e.g. GIS, DBMS, and expert systems). It also enables the spawning of executables as deemed necessary. Graphic windowing is important because it allows and manages multiple virtual screens (windows) for both text and graphic data display and exchange. The best candidate which possesses the multi-tasking and windowing capabilities is the UNIX / X Windows duo.

UNIX, since its conception in the sixties, is probably the most powerful operating system for scientific computing. It runs on virtually any hardware platform. Along its migration path, UNIX added its strength by incorporating many enhancements for networking and graphical user interface as its standard features. The direct relevance of these standard features to the design of an integrated resource management system are (1) its network transparent client-server architecture, (2) programming interface, and (3) inter-client communication.

Client-server architecture

The X Window System specifies that executable programs fall into two categories: clients and servers. A client is any application program that runs in an X Window environment. A server is the program that controls the physical resources of display. The server distributes user input to and accepts output requests from various client programs located either on the same machine or elsewhere in the network (Scheifler et al. 1988). For instance, client programs such as GIS and simulation models do not need to physically reside on a same machine, yet can be executed and displayed on a same X server. Called client-server architecture, this opens the door for a convenient distributed processing for better system performance and computer resource utilization.

In X windows, all client programs communicate with an X server through a fundamental layer of X window system called X Protocol (Lee 1988). By means of X Protocol, a client gives instructions to a server to carry out its screen display activities such as drawing a line or a box.

Programming interface

As the X Protocol is quite rudimentary and cumbersome for direct programming interface, it is grouped and organized into a higher layer library of functions called Xlib. All implementations of user interfaces for X provide a window manager whose responsibility is to keep track of the status and to control the appearance of any X client programs running on a server. It should be noted here that the X server merely interprets X Protocol into screen display as it is told. It is not responsible for the "look and feel" of a client, nor it cares how do multiple windows stack up or when to refresh the images. These added flavors are the responsibility of a window manager.

The X Window System itself does not define the user interface layer of X. The window manager is regarded as individual preference. Since all the toolkits are built on top of Xlib, and all the requests to a server are in the forms of X Protocol, the functionalities of a client program usually will not be hindered, regardless of the toolkit used.

X-windows is an event-driven system and almost all client requests detected by the X server generate events. Examples of events include keyboard events, such as key press or key release; mouse events, such as button press or button release; or window state notification events (Scheifler et al. 1988). As there are ten major categories of events with over thirty types of them, the efficient handling of events is a main concern for any system design under X-windows. Unless a designer wants to start from scratch, he may adopt the event handling mechanism already built into some toolkits (Heller 1990).

Inter-client communication

One message type that is unique in the X-windows environment is the inter-client communication (ICC). It is a set of conventions that X clients use to pass message to each other to request services, and to inform the requester the status of the services. ICC provides programs an effective way to communicate and exchange data. Information processing with ICC is freed from the more rigid format of conventional subroutine structure. A whole system can be broken down into a number of stand-alone client programs. Each client program requests and responds to messages in a standardized "conversational" style. This allows a modular approach in designing an integrated system.

Structure of an integrated system

A functionally integrated resource management system needs an integrated corporate database and five groups of clients (Fig. 1). There is only one database client and user interface shell client, but there may be many of each of the other types. It reflects the integrated control of data and a unified, consistent user interface. The plural form of models, spatial analyzers, and inference procedures means that there can be more than one tool for a domain.

Corporate database

The database of an integrated resource management system should be treated as a sacred cow of the firm. As mentioned before, the access method should be uniformly SQL. On top of the database lies the relational database management system (RDBMS) engine, interface programs, and application programs. The interface programs can be interactive SQL parsers, forms managers, or report generators. The RDBMS shall provide a high level programming interface for client applications to embed SQL statements for data access.

All data should be stored in one (logical) database, and ideally, all data should be relationally accessible across the network. This includes tabular inventory attributes, spatial data, and systems resources. Most RDBMS packages these days do support distributed databases. Combining with the UNIX/X duo, the distributed setup is even more simple.

In reality there usually are extraneous data sources that have to be reconciled with. For example, GIS like ARC/INFO has its own database, INFO, which manages spatial data files such as the ARC (arc coordinates and topology)

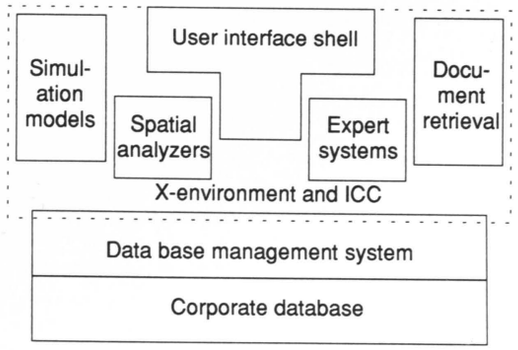

Fig. 1. All components of an integrated resource management systems should handle their mutual communication via the X-substrate. Outside communication is through the corporate database and a user interface shell.

file, and attribute tables such as the PAT (Polygon/Point Attribute Table). When ARC/INFO is used independently, the content of those spatial tables is referenced and accessed by key attributes in the INFO base. However, when it is used in conjunction with others such as the corporate inventory database, a problem occurs. The INFO attributes may only be a subset of the corporate inventory database. If that is the case, then the spatial data can not be cross-referenced with those attributes missing from the INFO base. If data is updated, unless it is done in both places then discrepancy will occur. One remedy is to keep this anomaly to a minimum. That is, use only the primary keys in INFO, e.g. keys of compartment and stand numbers. These INFO keys in INFO can be mapped with the corresponding ones in the corporate inventory database and maintained as key mapping tables.

The database client

This client provides interface to the system's relational database, e.g. an ORACLE database. Ideally, the database client should be the sole interface to the underlying database. It would be desirable that other clients simply use ICC to the database client. It receives ICC database service requests from other clients and parses them into SQL format. It then log into the underlying database management system for data access. This approach would ensure higher degree of abstraction. If the underlying RDBMS is converted to other package, or if the SQL language evolves, the only place requiring modification is the database client.

Model clients

The model clients are simulation programs used for growth and yield projections, pest infestation predictions, soil erosion and sedimentation computation, etc. They are usually developed in conventional languages in conventional main-subroutine format, and obviously will continue to be this way for a long time. Even so, they can be worked into the framework of an integrated system by wrapping. That is, an X client can be written to wrap around and spawn a conventional model so that its data input /output needs can be managed by this wrapper. The wrapper itself may use ICC to communicate with the database client to request for database services for the wrapped model. Or, it may simply use the RDBMS programming interface to embed SQL for data access.

Spatial analysis

Spatial analyzers are programs that perform analysis on spatially referenced data. The spatial functions that are most relevant to integrated use include: intersection, union, difference, buffer, adjacency and distance. These functions may be served by one or more of vector- or cell-based GISs, image and map processing packages, etc (Loh and Rykiel 1992). Like models, this group of programs are usually written in conventional format and need to be "X-wrapped."

Knowledge-based components

Inference procedure clients are rule- and other knowledge-based systems. In dealing of diagnostic tasks, the ones with backward chaining inference procedures are useful. For planning problems, forward chaining systems can be used. Again, they need to be X-wrapped.

Document management

Document management clients handle the textual and pictorial information that is used to support decisons. Essentially these clients contain multimedia databases from which the user

can browse facts of rules and regulations, pest information, etc. The user may create documents and enclose the information from these clients into his/her own reports. Knowledge-based systems may utilize the services of these document management clients.

User interface

The integrated user interface client is the "control center" of an integrated system. The control panel will be constructed as a graphical interface with standard windowing characteristics. In the layout of the control panel, there will be a number of top level menu buttons each representing a major type of service. For all practical purposes. The user interface client is the only client that a user will be interacting with. The user specifies the service needed through a control panel. The service request is then translated by the user interface client into internal message events (e.g. X Protocol or ICC message) and sent to the X server or other relevant clients to fulfill the service. When results are sent back, the user interface client will manage and organize them for their screen display. Standard characteristics of a windowing environment can be seen in Fig. 2. They are quite similar across the various user interfaces, MS-Windows, Open Look, and MOTIF.

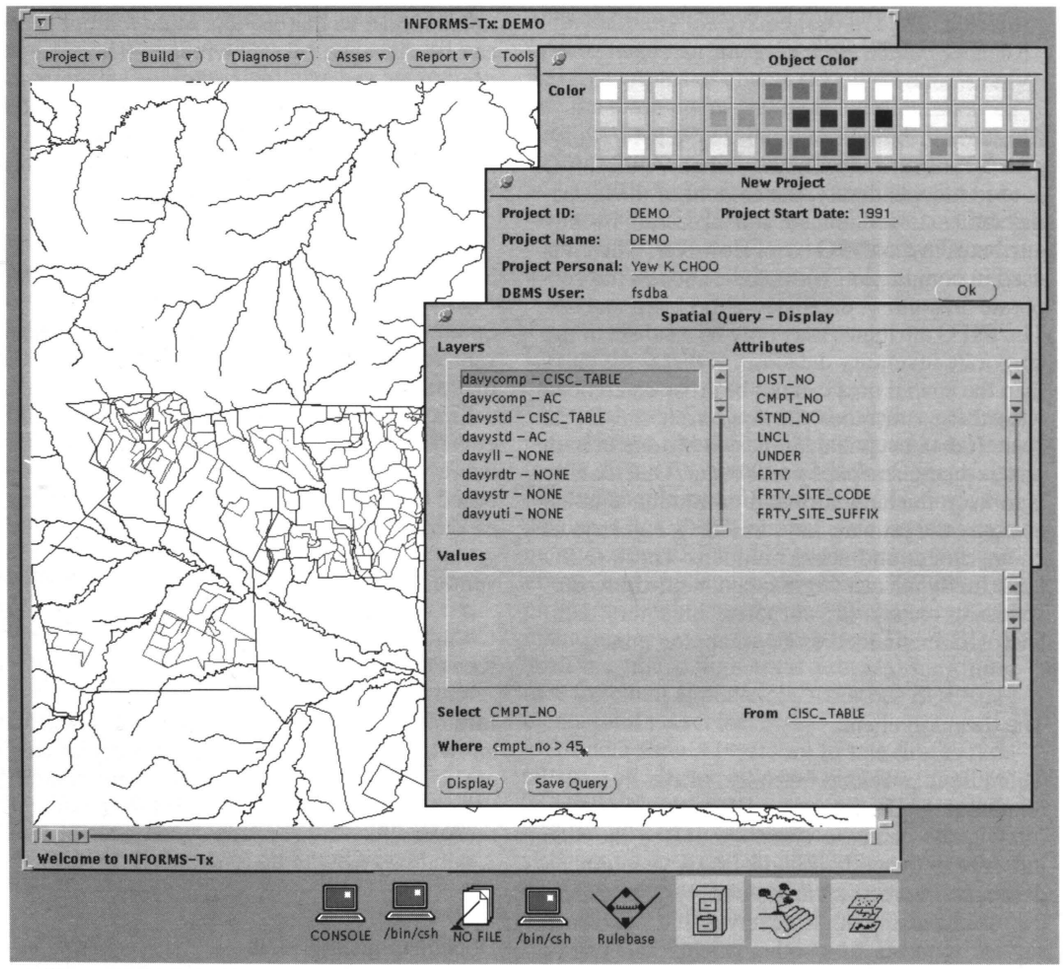

Fig. 2. The characteristics of a windowing graphical user interface that conforms to the OPEN LOOK user interface definition. The application shown is the INFORMS DEMO-TX system, with windows for project definitions. At the bottom of the screen, icons for rulebases, GIS, and the relational database management system are shown.

Functions of an integrated system

In the following, the technology described above is used to build a system for supporting the decision making of district level forest management planning. Its functions are depicted through the six top level menu buttons, each representing a group of related functions. They are: Project, Build, Diagnose, Assess, Report and Tools (Fig. 2). These selections are in the order of logical use following a decision process. The design is built to permit flexible progress through each phase of project planning and evaluation such as describing the project location, building the set of alternative project activities, diagnosing existing and potential site conditions, assessing of potential impacts of proposed actions and reporting of analyses for inclusion in project documentation.

This implementation is based on the working model designed by the USDA Forest Service's Forest Pest Management for the national forests in Texas. Its main features are the same throughout large-scale natural resource management in the coniferous forest region, but of course, for each country and firm, adjustments are needed. The implementation described here is called INFORMS for Integrated Forest Management System. There are varieties called IRMA (Loh et al. 1988) and INFORMS DEMO (Loh et al. 1991).

Project

This menu enables the user to initiate, select, fetch, use, and save a project plan. To start a new plan, the user first queries and selects a mapset from the database. If the mapset name entered by the user is a known district or in the mapset list, then the district map is shown on the screen. Otherwise, the user will be prompted to define the mapset. Once the map is displayed the user can use a pointing device, such as a mouse to identify a general project vicinity from an overview map, generally a district-wide map. The project vicinity map is, by default, a scanned image of the quadrangle maps associated with the vicinity. It is intended for referencing purposes. Additional feature maps such as stand boundaries, roads and lakes can be loaded. These feature maps are normally prepared through a digitizing process.

By pointing to natural boundaries, such as roads, streams and ridges, the project boundary is finalized. The data encompassed by this project area will drive models used later in the impact assessment for proposed activities. After the project boundary has been defined, the resource manager will need to become more familiar with the site conditions stored in the corporate database. By using a tool called Theme Display, the user can have themes displayed for the project area.

Build

After setting the boundaries of the project area, the INFORMS user can start to build alternative action sets. A fully developed alternative is a mix of management actions proposed for the area. Examples of action sets include enhancing a dispersed recreation site, rehabilitating an abandoned or eroding roadway, designating a fuelwood sale area and/or thinning stands susceptible to fire or pest management. The user can add or delete actions from a selection of common activities. INFORMS allows an iterative and flexible *Build* procedure to refine a project.

To support such prescription tasks, the INFORMS user can access rulebases (expert systems) developed by resource specialists. The rulebases can be used to score sites for certain emphasis or suitability for an action under consideration. The scores or ranking are displayed in various color intensities. Many scoring systems have been developed on National Forest pilot projects. Examples include: hazard rating, wildlife habitat suitability, fiber production, prescribed fire suitability, mast production, timber stand improvement and uneven-aged management. Many other applications and rulebases are feasible.

Diagnose

The selection or proposal of activities within a project area should consider potential adverse conditions in addition to the existing site conditions. For example, rulebases can be used for risk rating of pest, fire or other hazards over the entire project area. They can also be combined with other tools to further enhance the diagnos-

tic capability. For example, the combination of imagery and expert systems can assist in effective diagnosis of symptoms recognized during field investigations of a project site. This strategy can make more diagnostic expertise available to other resource managers and can free the limited number of pest specialists to tackle more challenging opportunities.

Assess

As alternatives are being developed for a project area, their consequences and impacts have to be evaluated. For this purpose, mathematical models are usually used. In our vision, models to assess alternative will be executed in an integrated and automated manner. All automated models have certain input and output parameters which the user can change to control the processing of a model. The system will provide the user with the opportunity to modify process control parameters such as simulation interval, time increment and output format.

Additionally, designated resource specialists will be able to modify the behavior of a model through appropriate means such as changing the coefficients of core equations in the model. Similarly, the calibration, verification and validation of models should be controlled by experienced resource specialists with the knowledge and authority to modify models for a certain geographic area. As an option, a special tool set can be made available for use by resource specialists serving as consultants on interdisciplinary teams or working with smaller project plans or compartment prescriptions.

The system would allow the user to select formats for output of results from the assessment process. They will include tables, histograms, pie charts and 2D or 3D visualizations. Among them, visualization is becoming an increasingly vital part of an assessment. By integrating complex visualization methods with site condition data, sophisticated graphics can mimic a multi-dimensional scene. These scenes convey results more meaningful than reams of numeric listings and tabular results.

Report

Report functions allow the user to produce reports from information generated during the decision process. For example, the original display of maps, tables or visualizations on the screen can be retained for printing on a variety of devices. In addition, the results generated by running impact assessment models over multiple alternatives can be saved and printed for later comparisons.

The system can support the comparison of alternatives by placing control of assessment/analysis in the hands of the resource managers. One way of organizing the comparison of assessment results is to use a spreadsheet-type format for recording and calculating weighted preferences or subjective values for each decision criterion. If the user desires, the system can automate the format and means of storing and calculating the weighted values. Using this method the user must first define criteria and assign weighted values to each. A summary of the recorded results can then be used to subjectively rate each alternative. Recommendations based on these "quantified" ratings can then be evaluated against intangible and qualitative judgements applied by the interdisciplinary teams, resource specialists, and decision makers.

Tools

At any stage of a session, the user should be allowed to perform structured tasks such as standard spatial analyses like depicting buffers, intersections, unions, difference calculations, etc. These functions should be available in the submenus under *Tools*. This is similar to using a library of mathematical functions such as average, mean, square root, division, or sum calculations which are provided through reporting and query functions of spreadsheets and database management systems.

Conclusions

Integrated resource management systems would provide better decision support services to resource specialists by allowing integrated use of the tools and data, e.g. combining spatial and tabular information concerning management directions for a particular geographic area. It facilitates the identification of project areas for implementation of management practices through a combination of database accesses and the use of GIS functions. It enables a user to gather all relevant information in one place so that he can conveniently examine the consequences of alternative management activities in a project area and evaluate the pros and cons of each alternative project set.

An integrated resource management system offers several advantages to resource managers: (1) all needed tools and relevant planning and management information will be in one place, (2) a large number of alternatives can be considered with the same intensity while realizing labor and monetary savings over manual techniques, and (3) the decision making process can be documentable, and decisions can be repeatable and consistent.

The client-server architecture and communication mechanisms provided by the X environment represent important new technological advancements that makes the building and operation of integrated systems flexible and more open than in the past. Currently, programming in X introduces extra complexity because X itself is object-oriented, but the rest of the software usually is not. Embedding the X/ICC mechanisms into higher level objects via pure object-oriented programming will probably be the next step in the pursuit for integrated resource management systems.

References

AML users guide: ARC macro language and user interface tools. 1989. Environmental Systems Research Institute, Inc. Redlands, California.

Arnold, J. 1989. The X window system. MIPS (1)12: 82–87.

Buhyoff, G.J. (ed.). 1988. Resource Technology 88, International Symposium on Advanced Technology in Natural Resource Management, Fort Collins, Colorado, June 19–23, 1988. American Society for Photogrammetry and Remote Sensing Publishing. Falls Church, Virginia. 277 p.

— (ed.). 1991. Resource Technology 90, International Symposium on Advanced Technology in Natural Resource Management, Washington D.C., November 12–15, 1990. American Society for Photogrammetry and Remote Sensing Publishing. Falls Church, Virginia. 830 p

CLIPS reference manual. Version 4.2. 1988. Artificial Intelligence Section. Lyndon B. Johnson Space Center.

Covington, W.W., Wood, D.B., Young, D.L., Dykstra, D.P. & Garret, L.D. 1988. TEAMS: a decision support system for multiresource management. Journal of Forestry 86(8): 25–33.

Date, C.J. 1986. An introduction to database systems, vol. I. Fourth edition. Addison-Wesley, Reading, Massachusetts.

Durant, D., Carlson, G. & Yao, P. 1987. Programmer's guide to WINDOWS second edition. Sybex Inc., San Francisco, California.

Folse, L. J., Mueller, H.E. & Whittaker, A.D. 1990. Object-oriented simulation and geographic information systems. AI Applications in Natural Resource Management 4: 41–47.

Forest service integrated information management program (Project 615) specifications (draft). 1990. USDA Forest Service.

Heller, D. 1990. XView programming manual: an OPEN LOOK toolkit for X11. O'Reilly & Associates, Inc., Sebastopol, California.

Hunter, D.O., White, W.B., Daniel, T.C. & Buhyoff, G.J. 1988. Integrated information technology for natural resource management. In: Buhyoff 1988 (above). p. 109–120.

Integrated forest resource management system: system design workshop report. 1989. RFQ RM 89–64. USDA Forest Service. 58 p.

Lee, E. 1988. Windows of opportunity. Unix World (6): 46–61.

Loh, D.K., English, K.J, Choo, Y.K., Chu, Y.T. & Sun, M. 1988. Integrated natural resource management automation – the knowledge shop. In: Buhyoff 1988 (above). p. 158–167.

— , Janiga, P.E., Tsai, C.M., Holtfrerich, D.R. & Chu, Y.T. 1991. INFORMS DEMO: The design concept of integrated resource management systems. In: Stuth, J. (ed.). Proceedings of the international conference on decision support systems for resource management. Ranching Systems Group Publications, College Station, Texas. p. 137–140.

— & Rykiel, E.J. Jr. 1992. Integrated resource management systems: Coupling expert systems with database management and geographic informations. Journal of Environmental Management 35(1): 15–25.

Mäkeläinen, J. 1988. Ylläpitojärjestelmä (METIK programming standards). Silva Carelica 10: 78–88.

Mallach, E. 1990. Without a data model everything falls down. Computerworld, August 20. p. 25.

Martin, J. 1989. Information engineering, vol. I. Introduction. Prentice Hall, Englewood Cliffs, New Jersey.

Mikkola, E. & Saarenmaa, H. 1992. Strategic planning of information system in Finnish forestry. 66–75. In: Minowa, M. & Tsuyuki, S. (eds.). Integrated Forest Management Information Systems. IUFRO S4.02, S4.04 Symposium, Tsukuba, Japan, October 13–18, 1991. Japan Society of Forest Planning Press.

Rykiel, E.J. Jr., Saunders, M.C., Wagner, T.L., Loh, D.K., Turnbow, R.H., Hu, L.C., Pulley, P.E. & Coulson, R.N. 1984. Computer-aided decision making and information accessing in pest management systems, with emphasis on the southern pine beetle (Coleoptera: Scolytidae). Journal of Economic Entomology 77: 1073–1082.

Rosenthal, D.S. 1989. X window system version 11, inter-client communication conventions manual, version 1.0. Sun Microsystems, Inc.

Saarenmaa, H. 1991. Artificial intelligence, decision support and information systems strategies in for-

estry. Proceedings of the IUFRO XIX World Congress, Montreal, Canada, August 5–11, 1990, vol. B. p. 379–390.
— , Kaila, E., Lehto, K., Piippo, H., Salminen, H. & Pöntinen, J. 1991. The new information system architectures and strategies of the Finnish Forest Research Institute. In: Buhyoff 1991 (above). p. 569–578.
Scheifler, R.W., Gettys, J. & Newman, R. 1988. X WINDOW SYSTEM: C Library and Protocol Reference. Digital Press, Bedford, Massachusetts.
Thieme, R.H., Jones, D.D., Gibson, H.G., Fricker, J.D., & Reisinger, T.W. 1987. Knowledge-based forest road planning. AI Applications in Natural Resource Management 1: 25–33.
Tucker, M.J. 1990. The inevitable merging of SQL. Unix World 7(2): 68–74.

Total of 29 references