Machine Learning from Remote Sensing Analysis[1]

Daniel Charlebois, David G. Goodenough[2], and Stan Matwin
University of Ottawa, Ottawa, Canada

Abstract

An intelligent system (SEIDAM - System of Experts for Intelligent Data Management) is being developed for answering queries about the forests and the environment through the integration of remote sensing, geographic information, models and field measurements. SEIDAM consists of an hierarchical group of expert systems. In [1], it was shown how machine learning and planning can be used to create plans that execute image analysis software in order to recognize specific objects and perform a variety of different tasks. A query (task) could require, for example, that forest inventory stored in a GIS be updated to reflect past harvesting. As sensors become more numerous, the choices of data and options to recognize objects become more complex. These complexities can be reduced by making use of case-based reasoning. Only the data needed to answer the query will be used. The aim of case-based reasoning is to avoid having to build a solution to a problem from first principles, or by drawing on rare expertise, by adapting a known solution for an old problem to the new problem. There is a constantly growing variety of data sets, providing information at various degrees of accuracy and at often different cost. A non-expert user of this data could greatly benefit from reusing specific cases of queries, which convert the data into knowledge that other users were seeking before. A case, in this context, consists of a query and an example of the process (plan) that answers that query using a single or a multi-sensor data set, and geographic information, such as forest cover, topography, hydrology, etc. In our earlier work, we constructed a planner (LEAR), a planning system for creating expert systems by executing software for a case and interacting with a human expert. We now wish to raise the machine learning methods from creating expert systems for executing existing software to creating new rules (knowledge) derived from observing remote sensing analysis cases. We will also show how knowledge about objects acquired by the LEAR planner can be used to assist a case-based reasoner during both its retrieval step and its adaptation step.

I. Introduction

A. Background

The applications of forest management and environmental monitoring increasingly depend upon complex information systems. These information systems integrate forest cover descriptions, topographic maps, remote sensing, and application knowledge. The volume of remote sensing data is increasing. By the year 2000, each tracking station in North America could be receiving in excess of one terrabyte of remote sensing data per day. For most applications human screening of these data will not be practical. Another change in remote sensing is in the nature of the information which can be extracted. For example, it will be possible to map chlorophyll A and B distributions and timber volume using a mixture of imaging spectrometer and polarimetric SAR data. In the future, it will be possible to update these detailed attributes in a GIS. The need to reduce data costs for monitoring of large areas is leading to the data being available for the cost of reproduction. Therefore, it is hypothesized that the data management systems for resources will make intelligent selections of the incoming data in order to respond to users' goals, reduce complexity, and be more adaptable.

A project was begun in 1991 to develop a system of experts for intelligent data management (SEIDAM) for forest and environmental monitoring. The SEIDAM Project is a project conducted under the Applied Information Systems Research Program of NASA. The partners in the project include the Canada Centre for Remote Sensing (CCRS), the Pacific Forestry Centre (PFC), the B.C. Ministry of Forests (BCMOF), the B.C. Ministry of Environment, Lands and Parks (BCMELP), the Royal Institute of Technology (KTH) of Sweden, and the EEC's Joint Research Centre (JRC) at Ispra.

SEIDAM uses expert systems to control and integrate remote sensing image analysis software, geographic information systems, databases, and models. The user may specify his or her goals in two ways, by asking a question (query) or by specifying a list of desired products. The expert systems can control software developed by the partners, as well as commercial products, thus maximizing software reuse. The expert systems are instantiations of the shell RESHELL written in Quintus Prolog. The image analysis software includes CCRS' LDIAS software and the PCI image analysis software. The geographic information systems in SEIDAM are PAMAP GIS and GRASS. The

[1]This research was conducted under the SEIDAM (System of Experts for Intelligent Data Management) Project, part of NASA's Applied Information Systems Research Program. Research support was also received from Forestry Canada, NSERC, and FRDA.

[2]Also with Pacific Forestry Centre, Forestry Canada, Victoria, B.C.

Table 1. Test Data for Sites

Platform	Sensor	Description
LANDSAT	Thematic Mapper	7-channel optical.
SPOT	HRV	3-channel +1 optical.
NOAA	AVHRR	5-channel optical.
ERS-1	SAR	C-band.
J-ERS-1	SAR	L-band.
J-ERS-1	OPS	optical
Aircraft		
CCRS Convair 580	SAR	X, C -bands, HH and HV
Falcon Fanjet	MEIS	8-channel optical; 2 stereo.
Falcon Fanjet	MSS	11-channel optical.
NASA ER-2	AVIRIS	210-channel imaging spectrometer.
NASA ER-2	MAS	MODIS optical simulator
NASA DC-8	AIRSAR	C, L, P-band polarimetric SAR.
Falcon Fanjet	Aerial photography	
NASA ER-2	Aerial photography	
Ultralite	Spectron	optical profiles
Cessna 206	CASI	spectral profiles; 8-channel imagery.
GIS Sources		
BCMELP	TRIM topography	1:20,000 DTMs
EMR	topography	1:50,000 and 1:250,000 DTMs
BCMOF	forest cover	1:20,000 IGDS files.
M&B	forest cover	1:20,000
CFI	forest cover	1:20,000 ARC/Info

models support predictive queries and include forest growth and yield and pest spread over time. SEIDAM integrates software executing on SUN and SGI workstations, and DEC VAX computers. Motif is the windowing standard followed for the system. More than two million lines of code comprise SEIDAM.

B. Test Data and System Description

SEIDAM is a prototype of the kind of intelligent resource information system needed during the next decade. In order to develop and test SEIDAM, three test sites have been selected in British Columbia. These test sites are Greater Victoria Watershed (345 km^2) north of Victoria, B.C., Tofino Creek (270 km^2) in the Clayoquot Sound area of western Vancouver Island, and Parson (300 km^2) in the east Kootenay mountain range. The data to be collected over these sites is listed in Table 1.

Remote sensing data are being obtained from six satellite sensors and eight aircraft sensors. These sensors cover the visible, infrared and microwave spectral regions. The sensors also span a spatial resolution from one meter to one kilometre. The digital elevation models (DEMs) from BCMELP are at a scale of 1:20,000, on the NAD'83 datum, ungeneralized, and with ridge and break lines. These are the primary bases for the SEIDAM data. However, some of the GIS sources are based on older DEMs or on topographic maps at scales of 1:50,000 and NAD'27. SEIDAM has software to support the ingest of these data and their transformation to a common topographic base, to a correct edge-matched form, and to a common attribute description.

The overall architecture of SEIDAM is shown in Figure 1. The rounded boxes with numbers denote expert systems or collections of expert systems, such as SHERI. SHERI (System of Hierarchical Experts for Resource Inventories) is a collection of expert systems for using Thematic Mapper imagery to update and to perform quality control of forest cover GIS files. The major data stores for products, queries, cases, and answers are shown. SEIDAM is initially trained by examples which consist of a query, a single or a multi-sensor data set, and geographic information, such as forest cover and topography. The query or product list establish the goals to be satisfied. A case is a plan to solve a particular goal. These plans or cases are stored and manipulated as described later in this paper. Based on these cases, SEIDAM proceeds to solve the goals. In recognizing forest attributes, SEIDAM can dynamically select additional remote sensing data to achieve the goals. The data fusion expert system matches a query with the appropriate spatial resolution requirements based on knowledge of object scales and dynamically selects the best remote sensing data to recognize the forest object.

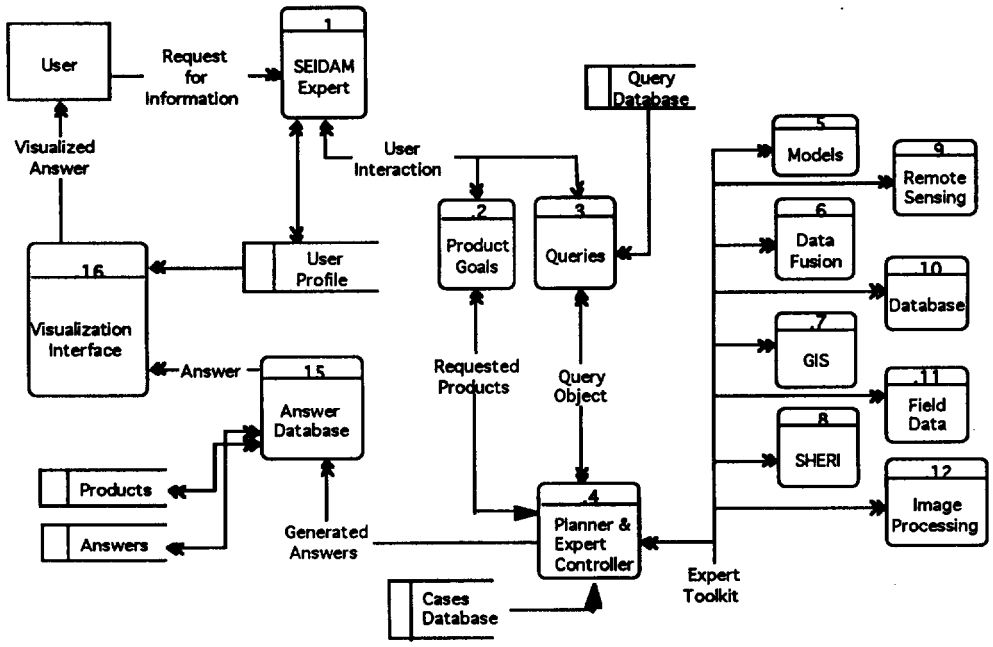

Figure 1 - SEIDAM Architecture

II. Queries

SEIDAM is driven by the user selecting products or queries for establishing data management and analysis goals. An experienced user may know that a particular product, such as a GIS file updated with remote sensing, can satisfy a number of queries. Products can include maps, updated GIS files, tabular summaries, visualizations, temporal change products, text, values, etc. Alteratively, the user can select a query. The most desirable interface for queries would be to have a natural language system. However, since that is beyond the present technical capabilities, SEIDAM uses a queries list and offers the user the ability to create new queries using a restricted vocabulary. The queries list was developed through interviews and documents provided by the BCMOF and BCMELP.

In this section our purpose is to show the issues related to answering queries and the need for planning. Examples of the queries supported by SEIDAM are:

How much forest do we have in the Tofino Creek test site?
How much old growth forest is in the test sites?
Who was the source of the data used to provide this answer?
Where has bud flush (new growth) been reduced?
What is the present timber volume of Douglas fir in this watershed?
What is the yearly rate of forest depletion in this test site over the past 20 years?
Where are the areas of highest root rot damage?
How much forest did or will we have in the Greater Victoria Watershed?
...

Let us consider the first query, *How much forest do we have in the Tofino Creek test site?* The types of possible answers are: (1) hectares of forest, (2) a GIS file showing forest distribution, (3) hectares of each species, (4) a GIS file showing species distribution, (5) timber volume table by species, and (6) a GIS file showing timber volume. Answers (1) and (2) respond to a class selection separating forest and non-forest. Answers (3) and (4) respond to a class selection separating forest by species. Answers (5) and (6) respond to a class selection separating species by attributes. How the query is answered depends upon the context, the date of the data on which the answer is built, the accuracy goals, and the knowledge in the system of past cases.

Let us suppose that the context for this query means that it is necessary to update a forest cover GIS file with Thematic Mapper imagery from 1993. What are the requirements to answer this query. A list of the data available is needed. For the Tofino Creek test site the following data are available:

(a) six, 1:20,000 scale, NAD'83 TRIM digital elevation files,
(b) six 1:20,000 scale, NAD'27 forest cover GIS files based on 1985 aerial photography;
(c) four LANDSAT images, three from Thematic Mapper (1989, 1990, 1993), and one MSS image (1973)
(d) several hundred ground samples of forest mensuration collected by forest industry.

For each data source SEIDAM carries meta-knowledge. The query could be answered by creating a mosaic of all data first. The query could also be answered by processing each map independently and combining the results. In either case, the digital elevation files would need to be processed to create a digital terrain model with elevation, slope and aspect. SEIDAM can call on the SHERI collection of expert systems which process one map at a time. The forest cover GIS files are on a different topographic base and datum and need to be transformed to the TRIM base. To create a mosaic over the test site, these forest cover files would also need to be cleaned, complexed, and edge-matched. The more recent LANDSAT TM images are geocoded to NAD'27 with 25 m pixels and no corrections for topographic relief. The MSS image is in a raw form. To answer this query we only need to use the 1993 TM image, and correct it to NAD'83 and for topographic relief. However, other queries may require combinations of imagery for which many processes would need to be executed to create these combinations.

The steps to answer a query can be ordered in a systematic way and our goal is to create plans that contain these steps. There are several methods to form plans, two of which are "planning" and "case-based reasoning", described hereafter. The presentation of the answer is an important part of the processing and therefore, must be included in the plan. This presentation will vary with the characteristics of the user (user profile), the context of the answer, the cost of the answer, and the information content of the answer. The plans we create have to exhibit flexibility in that the success of several processes are data dependent and may require that the system dynamically alter the plan.

III. Planning

One possible approach to create a system that can automatically answer a query or manufacture a product (eg. an updated forest cover map using remote sensing) is to use a planning system. A planning system can be described as follows:

Given:
- *initial world state,*
- *a set of rules or operators,*
- *a goal or a conjunction of goals,*

Then:
- *find a sequence of operators that can transform the initial world state into a state where all of the goals are true.*

Many planning paradigms exist, the simplest of which is linear planning. One widely recognized incarnation of a linear planning system is the STRIPS [2] planner. In STRIPS, the initial world and world states are described using a set of facts that are true at any given point during planning. Rules describe how to change the world state. In STRIPS, a rule has three components: a precondition, a delete list and an add list. The preconditions are the facts that must hold in the world before the rule can be applied. The delete list is the list of facts that no longer hold in the world after applying the rule. The add list is the list of facts that are true in the world after applying the rule.

The algorithm for STRIPS is:

```
procedure solve( Gs: list of goals )
repeat
    select a goal G from Gs
    if G is true in the current world then
            no other processing is neessary
    else
    repeat
            find a rule that has G in its add list
            solve the preconditions of the rule
    until successful or no rules can be found
    if not successful then exit( failure )
until Gs is empty.
```

What makes a planner linear is that the rule selection mechanism is only guided by the current goal. Although several rules may be applicable because their preconditions hold in the world, they are not selected because they do not add the current goal to the world. This approach can solve a wide range of problems, but is incomplete because it does not explore the whole space of possible world states to achieve the end goal.

For the SEIDAM system, a linear planner could help to create plans that can answer certain queries. For example, if a forester queried SEIDAM in search of the amount of forest in the Tofino Creek test site, it might use a rule such as the following to start building a plan to answer the query:

```
query_forest_cover_polygon( Site ) ::
    if      maps_available( Site ) and
```

```
        topography_available( Site ) and
        forest_cover_available( Site ) and
        tm_data_available( Site ) and
        execute_map_update( Site )
then    add forest_cover_polygon( Site )
```

For this example, in SEIDAM, the user goal is expressed as forest_cover_polygon_available(Site). Since this rule will add the user goal to the world, it would be selected by a linear planner. What is also expressed in this rule is that the information is not directly available and that to create it, maps, topography and TM data must all be available. If these data are available, the map update can be performed by the SHERI sub-system.

Although linear planning can help to solve a wide range of problems, there are two main reasons motivating the use of Case-Based Reasoning (CBR) for plan formation in SEIDAM. The first is that, to use a planning approach, a planner would need a complete set of operators and objects required to perform all tasks. The second is that, because linear planning uses goals and preconditions to select operators, the traversal of the space of possible world states is incomplete. Hence possible solutions may be overlooked. For the reasoning component of SEIDAM we propose to have some of the knowledge (i.e. operators and objects) given so that we may rely on linear planning to create plans for simple goals. For more complex goals, SEIDAM will use CBR and Analogical Plan Merging as described in section V.

IV. Case Based Reasoning

The goal of Case-Based Reasoning is to provide systems with the capability of solving problems based on past experience rather than trying to form solutions from first principal. In short, when a problem situation arises, a CBR system will search through a case base for a similar problem encountered in the past. If the case exists, the plan used to solve the previous problem is retrieved. The reasoning system must then modify the old plan such that it takes into consideration the differences that exist between the old problem and the new problem. There are two main types of CBR systems: transformational analogy [4] and derivational analogy [6].

In transformational analogy, the approach is to make minor changes to the old solution until it fits the new problem. As a simple example, consider the following problem whereby a forester must query a polygon cover map containing elevation and forest cover over a particular site for the main tree species at a given elevation. If a polygon cover map containing this information does not exist, the forester must overlay a forest cover polygon map over an elevation cover polygon map to create a new polygon level to answer the query. When using the PAMAP GIS, this can be accomplished by executing the following plan:

```
start_program( analyzer )
get_map( clayoquot )
overlay_polygon_cover_levels(
        forest_cover,
        elevation_cover,
        new_level )
stop_program( analyzer )
start_program( mapper )
get_map( clayoquot )
display_polygon_cover_level( new_level )
```

After executing this plan, the forester would be looking at the new polygon cover containing the information he requires. He can now simply query the cover. To continue the example, the forester must now find all forest species on north facing slopes at a different site, say Parsons. The CBR system should recognize that these two problems are similar and retrieve the solution that solved the old problem and adapt it to the new problem by substituting either the appropriate operators or operator parameters. The result would be:

```
start_program( analyzer )
get_map( parsons )
overlay_polygon_cover_levels(
        forest_cover,
        aspect_cover,
        new_level )
stop_program( analyzer )
start_program( mapper )
get_map( parsons )
display_polygon_cover_level( new_level )
```

Hence the result of transformational analogy is a plan that is similar in structure to the original plan. In this case, "clayoquot" was replaced by "parsons" and "elevation_cover" was replaced by "aspect_cover."

Derivational analogy uses knowledge about past operator selection during plan formation rather than modifying an existing plan. In the example described above, the forester had to go through the complete polygon cover creation exercise. If the forester needs only an approximate map of tree species on slopes facing north, he need not go through the whole process of creating a new polygon cover. He could, for example, display the aspect polygon cover over a remotely sensed surface cover map of forest species. Although the problems are similar, the solutions are significantly different. To find the new plan, a CBR system would have to review certain decisions that were made during plan formation. Consider the following (operators) rules:

```
get_cover_level( Level ) ::                          (1)
    if      current_map( Program, Map ) and
            Map has Level
    then    delete   all(current_cover_level(_,_ ))
            add current_cover_level( Map, Level ).
```

```
polygon_over_surface( Polygon, Surface ) ::   (2)
    if          Surface is_a surface_cover_level and
                Polygon is_a polygon_cover_level
                current_map( Program, Map ) and
                Map has Polygon and
                Map has Surface
    then  delete current_polygon_cover_level(_, _ ) and
          current_surface_cover_level( _, _ )
    add   current_polygon_cover_level( Map, Polygon ) and
          current_surface_cover_level( Map,Surface ).
```

Rule (1) will make the specified level, polygon or surface cover, the one currently displayed. Rule (2) will display a polygon cover over a surface cover map. During the planning process, the system would normally select rule (1) simply because it was encountered first. However, in derivational analogy, although the system can be successful by using rule (1), a previous case has recorded that, to get a simple view of tree species on north facing slopes, rule (2) should be used. In short, derivational analogy reuses past decision making experience.

V. System Design

The basic design of the Case-Based Planning system can be seen in figure 2. As is shown, there will be two types of users for the system. The first type of user is the domain expert. His main responsibility is to provide the system with all the necessary knowledge required to perform a variety of remote sensing, image analysis, and GIS tasks. The second type of user is the application user. This person will enter queries for which the system will attempt to construct a plan to answer the queries by retrieving solution examples supplied by domain experts. Plans are formed by adapting the old solutions to the new problem via analogical reasoning.

In the following subsections, we will describe in more detail the roles of each of the modules present in figure 2. In the figure, the rounded boxes show what type of information is travelling along the edges. The square boxes are modules that perform the information processing. The arrows indicate the direction of the flow of information. The figure does not show the interface between the users and the system. This interface will be a module based on a work station windowing environment.

At the outset, the case base is empty. In other words, the system has no experience in the expert's domain. The expert's responsibility is twofold. First, he or she must supply the system with a minimal set of rules that describe the basic methodology for performing a given set of domain specific tasks. He must then provide the system with examples (ie. cases) of tasks in which he must include: the goal of the example, the sequence of steps that must be executed to satisfy the goal, the information required for the successful execution of each step and a description of how the results should be presented to the application user. The system will also learn about the objects and processing involved in performing the different tasks and, as training progresses, the burden of providing extensive details for the examples will shift from the expert to the system. Part of this behaviour is assured by the LEAR planning system. As well as creating plans that use image analysis programs, LEAR assists in collecting information about the objects used in the domain (eg. image file formats, gradient images, etc.).

A design issue that must be addressed is case representation since it will determine how the system will behave during training as well as during use. Hammond [4], amongst others, has argued in favour of creating a case base of complete plans. When a new problem arises, the plan that matches the user's goals best is retrieved and adapted. We consider that the system to be much more effective and adaptable if we fragment the plans and store them in a database. These fragmented plans can be combined in a multitude of ways to achieve flexible goals. For example, when performing digital image analysis, one method of extracting information for the images is to apply classification algorithms. The results of classification can be used in different ways, however, there are several classification methods. One is to apply a maximum likelihood approach. Another is to use a segmentation algorithm. If plans that use classification as a means of extracting important image features are stored as complete plans, we may be restricting the size of the planning search space, but we may also be ignoring the possibilities of combining fragments from different plans. Hence, as a first draft prototype, plan fragments should be stored in such a way that they can belong to several plans. As is the case in LEAR, collecting plan fragments into a plan should guided by the objects that require processing to satisfy goals. This approach is analogous to the use of local cues in [5].

The next issue that must be addressed, is: what to remember about plans. Some approaches suggest remembering every plan created by the reasoning system. Others are proponents of remembering how old plans were adapted to satisfy new goals. We believe, that one approach should not prevent the other. Obviously, during training all the examples supplied by the domain expert should be remembered in their entirety. Since we propose to store plan fragment, complete plans can be considered as descriptions of how to assemble and adapt the plan fragments. This would allow the system to store new plans developed for the end users as well. The decision must still be made, however, on what new plans should be remembered.

Several approaches have been used to index case bases. Some of the more prominent approaches suggest that the first step is to notice a problem in order to characterize the problem situation. The characterization is the result of identifying relevant features, such as missing information, thus leading to plan failure. These approaches index the case bases according to the problems areas each solution addresses. Although we have not fully explored

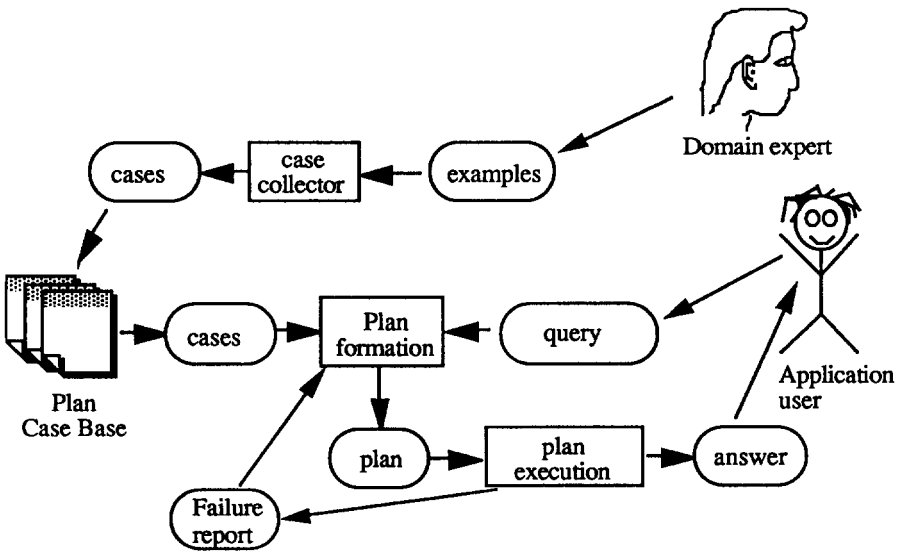

Figure 2 - Case-Based reasoning design for SEIDAM

the indexing problem, it would seem interesting to apply plan recognition techniques to highlight user goals to enable plan formation. The general idea is to match a user's high level problem specification with the "complete" program examples given by the domain expert. Whenever parts of the specifications do not match substitute them with other candidate plan fragments.

Obviously there must exist a measure to determine whether plan fragments are candidates. Most other approaches suggest methods akin to similarity matrices. Very interesting ideas in this respect have been presented in [3] whose CAESAR system receives program specifications, and builds a program from slices. This approach is based partly on data flow analysis. Briefly, a slice is a program code fragment that deals with a particular data object. Since using software libraries involves extensive object manipulation, indexing slices may be very similar to indexing plan fragments.

It is expected that, once it has a significant case base, the system will be used to create plans to solve unforeseen classes of problems. In an attempt to address these problems, the system must be able to learn. The type of learning has yet to be determined. However, some areas clearly present opportunities:

- the knowledge base is organized as a hierarchy of object classes; learning about these classes (i.e.: generalization, specialization of object descriptions) would seem to introduce the possibility of generalizing or specializing plans as well,
- the plan fragment hierarchy, should also lend itself well to learning techniques,
- the use of plan recognition as a tool to index into a case base should provide insight into learning and extending similarity matrices.

Plan formation is accomplished by retrieving from the case base one plan for every user goal that must be answered. Once these plans have been retrieved, they are merge to create a final complete plan that can satisfy all of the user goals. If for any reason a plan fails, the failure is reported to the plan formation module so that, when creating plans to solve similar goals, it will be aware of possible failure and will try to suggest alternative plans. One way to create an alternative plan is to remember cases that can satisfy the same goals by using different solutions.

Each case in the SEIDAM contains a goal, side effects and a plan. The goal should be satisfied if the plan stored in the same case is executed. Side effects are goals that can be satisfied by the plan as a result of executing it although they may not be part of the user goals. Hence, when a user enters a goal conjunction, a plan for each conjunct will be

retrieved from the case base. These plans must then be merged to form the final plan for the user goals. The merge may not always succeed. The problem can generally be linked to achieving goals in one plan inhibiting the achievement of goals in the other. Since it is possible to satisfy goals in different ways, side effects provide an easy mechanism to identify the best cases to retrieve and merge. The plan merging algorithm, Analogical Plan Merging (APM(k)), is:

```
1- for each user goal select a "base" plan.
   set a candidate list to empty for each plan
   set solution to empty
2- select a "current" operator from each plan.
3- while current operators from either plan do not clobber
     the preconditions of the other plan's operators
     place both operators in respective candidate lists
     set current operators to next operators
   if both plans do not interfere with each other
     append candidate lists to the solution
     exit
4- if not mutual clobbering then
     append to solution all operators up to and including
     clobbered operator
   goto 3
  else
   set X to 1
5-  in plan (X) while N < k and not safe
      set current operator to next operator
    if N = k and X = 1 then set X to 2 goto 5
    if N < k then
      if safe because both goals satisfied then
       append operators from plan X candidate list to
       solution
       set both candidate lists to empty
       set both current operators
      else
        if safe because clobbered preconditions no longer
         necessary
         append plan X operators to solution
         if preconditions for operator X' ok then goto 3
        else need extra operator(s)
         call linear planner
         if success goto 3
         else select substitute operator
           that has same effect as current as well as side
           effects
            that enable operator from other plan goto 3
            if no substitute exists then
              select another base plan for plan X
              if a plan exists goto 2
              else fail
```

where k is the number of operators to examine before declaring the merge a failure.

VI. Conclusions

SEIDAM is a system for intelligently answering queries for forest and environmental monitoring. Machine learning is used in SEIDAM to create new expert systems, to learn from training examples, and to learn from each subsequent query episode. In this paper we describe the role of planning and case-based reasoning in SEIDAM. It is shown that the plans created must be dynamically flexible to meet the varying characteristics of the users, the data-driven successes and failures, and the changing nature of the queries. The case-base is seeded with a minimum set of cases that represent typical problems. As SEIDAM is used, each new problem is treated as a new case which will be used to refine the case base, and thus learn from experience.

The plan merging algorithm, Analogical Plan Merging (APM(k)) was described briefly where "k" is the look-ahead depth in assessing which operators to merge. This algorithm addresses the clobbering of pre-conditions in operators in a plan by the operators in another plan.

VII. References

[1] Charlebois, D., Goodenough, D.G., Matwin, S., Robson, M., Fung, K., Reuse of Plans as a Tool for development of Remote Sensing Expert Systems, Proceedings IGARSS-92, Houston Texas, May 26-29 1992, pp. 1493-1496, 1992.

[2] Fikes, R.E., Nilsson, N.J., STIRPS: a New Approach to the Application of Theorem Proving to Problem Solving, Artificial Intelligence, vol. 2, pp 189-208, 1972.

[3] Fouqué, G. and Matwin, S., "CAESAR : a system for CAse basEd SoftwAre Reuse" Procs. of 7th Knowledge Based Software Engineering Conference, McLean, Va, 1992.

[4] Hammond, K.J., Case-Based Planning: Viewing Planning as a Memory Task. Perspectives in Artificial Intelligence, Academic Press, Boston, Ma, 1989.

[5] Konolige, K., Pollack, M.E., ASCRIBING PLANS TO AGENTS: Preliminary Report, IJCAI 89. Proceedings of eleventh international joint conference on artificial intelligence. 20-25 August 1989, pp924-930, 1989.

[6] Veloso, M., Learning by Analogical Reasoning in General Problem Solving, PhD Thesis, CMU, August 1992.