Guiding Object Oriented Design Via the Knowledge Level Architecture: The Irrigated Wheat Testbed[1]

Kris Schroeder, Ahmed Kamel & Jon Sticklen
Intelligent Systems Laboratory
Computer Science Department
Michigan State University
East Lansing, MI 48824, USA
sticklen@cps.msu.edu

Rick Ward, Joe Ritchie, & Urs Schulthess
Crop and Soil Sciences Department
Michigan State University
East Lansing, MI 48824, USA
22857MGR@msu.edu
Rhines%Staff%CSSDept@banyan.cl.msu.edu
urs@psssun.pss.msu.edu

A. Rafea & A. Salah
Central Lab for Ag Expert Systems
Agricultural Research Center
Ministry of Agriculture and Land Reclamation
Cairo, EGYPT
ESIC%EGFRCUVX.BitNet@pucc.princeton.edu

Abstract: In this paper, we detail our current work on a crop management system for irrigated wheat in Egypt. The goal of our work is to develop a system that will address all aspects of crop management including varietal selection, planting/harvest date selection, sowing parameters decisions, insect/disease/weed identification and remediation, and irrigation/fertilization management. The approach we take to solve this problem is the Generic Task Approach to expert systems development pioneered by Chandrasekaran et al. By using the Generic Task (GT) approach, we set out to model the behavior of an expert in wheat crop management. To accomplish this goal, the GT approach builds on top of the object-oriented methodology and acts as a guiding overlay for analyzing knowledge intensive problems, such as wheat crop management. As a multi-task problem, wheat crop management provides a testbed for the ideas of a Knowledge Level Architecture introduced by Sticklen. The Knowledge Level Architecture (KLA) provides a means of understanding large systems in terms of cooperating sub-agents. This paper describes the GT and KLA methodologies, focusing on the support they afford to the description and understanding of knowledge-based systems from an object-oriented perspective.

1.0 Introduction

Object-oriented technology has become a powerful means of handling the complexity inherent in many systems. Object-oriented technology has influenced and benefited from research in the arena of Artificial Intelligence (AI) and Knowledge-based Systems (KBS). Thinking in terms of objects provides considerable leverage when dealing with many complex problems. However, it is often

1. Accepted for presentation at the Object-Oriented Modeling Of Nature And Problem Solving in Ecosystem And Natural Resource Management Workshop, Ontario, Canada, September 12-16, 1994 and publication in a special issue of Mathematical and Computer Modeling by Pergamon Press.

insufficient when dealing with knowledge intensive problems. In such problems, the key component is knowledge. However, understanding how knowledge should be organized and controlled within a KBS has been a difficult issue facing AI since its onset. Determining the objects needed during problem solving and how they should be organized takes considerable experience and expertise. For the knowledge engineer faced with a problem solving scenario for the first time, this experience is not available. Therefore, we feel a knowledge-based overlay to the object-oriented approach is needed. The overlay should assists in the identification of objects as well as how they should interact. The “Generic Task” methodology pioneered by Chandrasekaran et al. (Chandrasekaran, 1986) provides us this overlay.

The Generic Task (GT) Methodology is one of several Task Specific Architectures (TSA’s) developed to address problems associated with early expert systems. These early systems were often rule or frame-based systems. Two central problems were encountered by the builders of these early systems. First, control issues were implicit in the representation, not allowing the explicit representation of control issues relevant to the specific type of problem being addressed. Second, knowledge engineers often found it necessary to create organizational structures on top of the already existing formalisms. Pioneers in TSA work recognized that these problems were calls for control and organization structures beyond what the simple rules and frames could provide (Chandrasekaran, 1983). They also recognized that certain problems often had the same control and organization structures even though they were operating in different domains. For example, diagnosis performs the same *type* of problem solving regardless of whether the domain is medicine or mechanics. It is in this atmosphere that the GT Methodology developed. GTs provide the knowledge organizations and control structures specific to certain kinds of problem solving across numerous domains as will be discussed in section 4.

The current problem we set out to address with the Generic Task approach is the development of a district-level wheat management consultation system for Egypt. This system will address all aspects of crop management including varietal selection, planting/harvest date selection, sowing parameters decisions, insect/disease/weed identification and remediation, and irrigation/fertilization management. As the length of this list implies, crop management is a complex, multi-task problem. Previous research in GTs typically focused on applying one GT to one problem. However, we will utilize numerous GTs, as well as the CERES crop simulation methodology pioneered by Ritchie et al. (Ritchie,Godwin, & Otter-Nacke, 1985). The GT approach alone does not provide a template for the integration of these problems solvers. We will use the Knowledge Level Archi-

tecture proposed by Sticklen in (Sticklen, 1989) as the template by which we will organize our system.

This paper describes Task Specific Architectures, Generic Tasks and the Knowledge Level Architecture methodologies, focusing on the support they afford to object-oriented design and implementation of knowledge-based systems. As a starting point, in section 2 we discuss Object-Oriented Modeling. Although taking an object-oriented approach to design and development of a system does offer significant leverage, the use of an object-oriented technique alone does not guarantee a successful system. In section 3, we will discuss how Task Specific Architectures have built upon what object-oriented paradigms have already established. In section 4, we give an overview of one Task Specific Approach known as Generic Tasks that is a principle methodology used in our system. We discuss GTs in light of the object-oriented approach and how the Generic Task approach provides a guiding hand during the design and implementation of expert systems. In section 5, we discuss our proposed Knowledge Level Architecture and how it guides the organization of Generic Tasks. Next, we discuss of our approach to building a regional wheat management system. We highlight the architecture of our overall system in sections 6 and 7. In section 8, we discuss the decomposition of the management system into the modules that address the various facets of wheat crop management. Finally, in section 9, we close the paper with a discussion of our plans for future research.

2.0 Object-Oriented Modeling

In (Rumbaugh,Blaha,Premerlani,Eddy, & Lorensen, 1991) the authors describe object-oriented modeling and design as “a new way of thinking about problems using models organized around real world concepts.” In the area of knowledge-based systems, the problems we deal with are knowledge intensive problems. Thus, the real world concepts discussed in the definition are not tangible items. Instead, they are often abstract knowledge organizations or problem solving methodologies which experts find useful when solving a problem. The major objective in building a knowledge-based system is to effectively select these knowledge structures and problem solving methodologies needed to solve the problem. We recognize the value of an approach that emphasizes knowledge organization and control.

There are a number of central tenets that run through object-oriented approaches (Rumbaugh et al., 1991). As will be seen, these ideas are also central to the Generic Task approach as a realization of the Knowledge Level Architecture. These tenets are:

1. **Abstraction:** During the design of a system, the goal is to uncover the domain "objects" to reach an understanding of the domain. However, in knowledge-based systems, what constitutes an object may be unclear since a knowledge intensive problem often deals with intangible objects. Thus, we emphasize the need to provide a means for the knowledge engineer to identify potential objects. We do this by guiding the knowledge engineer toward the selection of commonly used knowledge organizations and problem solving methodologies.
2. **Encapsulation:** The goal of encapsulation is to hide the implementation details of an object from its external users. In object-oriented programming, an object is defined in terms of its data structures and its behavior. Other objects should not have access to these data structures and behaviors. The advantage of encapsulation is that objects which communicate with other objects, must only be concerned with the interface to these other objects rather than their implementation details.
3. **Reusability:** One of the benefits arising from object-oriented approach is that the analysis of objects in the problem domain often leads to reusable code. After analyzing the domain, the object may be reused in other problem solving situations. However, taking an object-oriented approach to a problem does not insure reusability. The reason for this may be in the improper analysis of what constitutes an object or improper encapsulation of an object by not attributing the proper behaviors to the proper objects.
4. **Inheritance:** Often objects within a domain can be broken into classes/subclasses where relationships among classes follow a hierarchical organization. Depending on the granularity chosen in the class/subclass hierarchy, these relationships may vary. It is the responsibility of the knowledge engineer to determine the proper granularity for the domain.

A central argument for using an object-oriented approach is that by looking at a domain in terms of its objects, we can reach a better understanding of that domain. However, we believe that the use of object-oriented analysis is not sufficient in the area of knowledge-based systems. Problems arise when one is faced with a knowledge intensive problem for the first time, such as diagnosis. How does one determine the objects in this case? If we apply a pure object oriented approach to this problem we might say that each hypothesized diagnosis is an object. However, once this is decided, we must have a means of using this knowledge efficiently during problem solving. Therefore, we must determine the organization of the knowledge and control strategies for its use. Unless one has encountered a similar problem in the past and has experience in deciding what are the objects and how to organize these objects, a pure object-oriented approach does not necessarily provide the necessary means for organizing such knowledge and its associated control structures.

As the above analysis suggests, taking an object-oriented approach to a problem does not guarantee success. Our approach to knowledge-based system guides the object-oriented approach in terms of higher level concepts. The Generic Task approach as well as other Task Specific Architecture (TSA) approaches provide high level problem solving modules that combine both knowledge organization and control, assisting in both the design and development of knowledge-based systems (KBSs).

Before discussing the Generic Task approach it is necessary to understand the landscape of the TSA area.

3.0 Task Specific Architectures

Task Specific Architectures (TSAs) are a reaction to the problems encountered by "first generation expert systems." These systems were often rule or frame-based and considered "universal approaches" appropriate for all knowledge-based problem solving situations. Proponents of these architectures argued in favor of the modularity, uniform representation and single control strategies these architectures provided. However, the use of these lead to disadvantages as well (Chandrasekaran, 1987). In particular there were two difficulties:

- important control issues were hidden behind clever programming artifices at the implementation language level, and
- system builders encountered the need to organize knowledge in the system using constructs outside the formalism provided by the architecture, such as MYCIN's *context hierarchy* (Buchanan & Shortliffe, 1984) and PROSPECTOR's *models* (Duda & Gaschnig, 1979).

While a KBS could be built at the level of these architectures, these implementations do not provide guidance in terms of the conceptual problem of the analysis and design of the system. Problems at higher level of abstraction needed to be addressed.

Two intuitions grew from common experience with first generation approaches: (1) Certain knowledge and control structures may be common to a particular task (say, design or diagnosis) across different domains, and (2) the structures for different task types will likely differ. Both retrospective analysis of existing systems and prospective design of new systems indicated that an effective KBS will contain — either explicitly or implicitly — a model of the problem solving process it realizes. The evolving task-specific approach recognized the advantages of representing

explicitly the conceptual organization of domain knowledge assembled to solve a particular type of problem following a given method. This approach denoted a paradigm shift away from use-independent, uniform knowledge bases toward a view of KBSs as collections of diverse conceptual structures organized for use in targeted ways. Universal methods are appropriate when no further knowledge of a domain is available, but typically expertise in a domain affords a more structured understanding of how knowledge is used to solve problems efficiently. This new outlook signified one of the central lessons of the first generation.

Many problem solving approaches at the task level arose during the 1980's. A classical example of defining problem solving activity in terms of a higher level task vocabulary is due to Clancey (Clancey, 1981). After recognizing that the control strategies implicit in the MYCIN/GUIDON knowledge base could be expressed independent of domain terminology (Clancey, 1981), Clancey isolated heuristic classification as a method for performing diagnosis and other selection tasks (Clancey, 1983; Clancey, 1984; Clancey, 1985). This method decomposes selection tasks into a set of high-level subtasks that characterize the type of problem solving performed by many existing KBSs. By moving to this more abstract level of description, Clancey and his colleagues were able to reformulate MYCIN into NEOMYCIN (Clancey, 1981), a system whose control knowledge made no reference to the application domain and constituted an abstract model of inference independent of implementation.

McDermott and his colleagues have formulated a view of expert problem solving, based on the notion of role-limiting methods (RLMs), that is strongly driven by experiences in knowledge acquisition. The RLM approach (McDermott, 1988) posits that a large knowledge base can be constructed, maintained, and understood more fruitfully by organizing it according to the various roles that different kinds of knowledge play. On this view, "each role-limiting method defines the roles that the task-specific knowledge it requires must play and the forms in which that knowledge can be represented" (McDermott, 1988). Like Chandrasekaran and Clancey, McDermott holds that families of tasks exist for which the problem solving method and its control knowledge can be abstracted away from the peculiarities of a task instance. This approach, however, focuses its concern with these methods on how they circumscribe the roles and the representation of the task-specific domain knowledge on which they operate. The goal of that research is to identify task families having these characteristics, to abstract their methods, and then to construct an architecture that assists knowledge acquisition for the corresponding tasks. For the purpose of knowledge acquisi-

tion, the RLMs represent an important class of methods because they direct the acquisition process at a more abstract level while still providing a broad coverage of tasks in a variety of domains.

Steels (Steels, 1990) has advocated a framework for system analysis and development with some similarities to Chandrasekaran's task-oriented approach. Following Steels, one first conducts a thorough task analysis in which the task is decomposed into subtasks based on the nature of their inputs and outputs and on the nature of the mappings among them. Second, one constructs a model of the domain knowledge available to perform the task and subtasks. Finally, one applies problem solving methods geared to solving individual subtasks and to structuring subtasks in the pursuance of higher-level tasks. The method selected for each task depends on the kind of knowledge available to solve the task, as captured in the domain model. This methodology differs from that espoused by Chandrasekaran and McDermott, however, in that it allows for a representation of domain knowledge — in the domain model — independent of the method to be selected.

Founded on similar intuitions, KADS (Breuker & Wielinga, 1989) is a methodology for the construction of knowledge-based systems that offers an explicit software life cycle and a set of languages for describing and creating KBS structures. This methodology rests on the assumption that task methods share "ways of using knowledge" at a level of abstraction higher than that of concepts in particular domains (Breuker & Wielinga, 1989). The languages in KADS support the development of a conceptual model of the problem solving process and a design model of the target KBS at a level of abstraction corresponding to the types of knowledge employed. KADS proposes a four-layer representation of knowledge: (1) a domain layer of domain-dependent concepts, relations, and structures; (2) an inference layer that describes what inferences can be made in terms of the roles that domain-level entities play; (3) a task layer that controls when inferences are made in terms of goal structures; and (4) a strategy layer for goal generation and task monitoring. Like Steels' approach, KADS allows "task-neutral" representation of domain knowledge but then stresses the importance of having high-level task structures through which to view problem-solving knowledge. These structures include primitive "knowledge sources" at the inference level for solving particular subtasks and goal structures at the task level for representing task decompositions.

4.0 Generic Task Approach

The *Generic Task* (GT) approach of Chandrasekaran and his colleagues is one of the earliest and one of the most fully developed of the task specific approaches to knowledge-based systems. The

assumption of the GT approach is that knowledge takes different forms depending on its intended function (Chandrasekaran, 1986; Chandrasekaran, 1987). Following the Generic Task view, a problem is analyzed according to methods associated with solving it where each method can be specified by the forms of knowledge and inferences necessary to apply the method, and by the subtasks that must be solved to carry it out. These subtasks can then be recursively decomposed in a similar fashion. When analyzed in light of an object-oriented approach, we parallel tasks to objects. Thus, complex tasks are decomposed into smaller tasks as objects are decomposed into smaller objects. Furthermore, the tasks are encapsulated in terms of knowledge organization and control strategies which parallels the encapsulation of data structures and behaviors in the object oriented terminology.

The Generic Task approach sets out to identify *generic tasks* - basic combinations of methods, knowledge structures and inference strategies that serve as subproblems for complex problem solving across numerous domains. A number of Generic Tasks are currently available. However, for purposes here, the three most significant are:

- **Hierarchical Classification and Structured Matching** (Chandrasekaran & al, 1979; Gomez & Chandrasekaran, 1981; Mittal, 1980). Hierarchical classification is intuitively a knowledge organization and control technique for selecting among a number of hierarchically organized options. The abstract engine used for hierarchical classification, known as CSRL, was the first TSA shell and is described in (Bylander & Mittal, 1986).
- **Routine Design** (Brown & Chandrasekaran, 1986; Chandrasekaran,Josephson,Keuneke, & Herman, 1989). Routine Design was proposed by Brown as an architecture for performing design and planning tasks in which substantial experience is available (not for design or planning in totally novel situations).
- **Abductive Assembly** (Josephson, 1987). Abductive Assembly is based upon the abductive reasoning work of Josephson et al. Given a list of findings, the goal of Abductive Assembly is to form a composite hypothesis that will collectively explain the set of findings.

The Generic Task approach guides the object-oriented approach in terms of higher level concepts. When a knowledge engineer is face with a new problem, he/she can perform a task decomposition of the problem. If a task matches one of the generic tasks, the knowledge and control structures are specified. The knowledge engineer must only obtain the domain knowledge to fill in the knowledge structure. The data structures and the behavior associated with the task are already encapsulated within the generic tasks. Having a pre-enumerated set of generic tasks from which to choose gives the knowledge engineer significant direction during the analysis phase of system

development. The identification of objects, which is often difficult in object-oriented programming is directed by what has been useful in the past.

When building an expert system, knowledge acquisition (elucidating knowledge from the expert) is often the limiting bottleneck. Of central importance to the knowledge engineer is the availability of a vocabulary that is appropriate for the problem at hand. For example, in diagnosis we speak of malfunction hierarchies, ruleout strategies, and setting up differentials. On the other hand, design uses an entirely different vocabulary. According to the Generic Task approach the vocabulary used by the same task is the same across domains.

The vocabulary provided by object oriented approaches is in terms of the objects of the domain. The Generic Task methodology builds on top of this by providing the correct vocabulary for describing the problem, as well as a vocabulary for use during the design and development of the knowledge-based system. This vocabulary defines the knowledge structures and inference strategies specific to the task. The power of this approach is that if the problem matches the function of a generic task, the vocabulary is determined. Therefore, the knowledge engineer has a means by which to talk with the experts in terms that are familiar to them thus aiding in knowledge acquisition.

5.0 Knowledge Level Architecture

The problem of irrigated wheat management is a multi-task problem with several differing task types needed to solve the problem. Therefore, there is a need to integrate multiple Generic Tasks into one problem solver. The Knowledge Level Architecture (KLA) proposed by Sticklen (Sticklen, 1989) provides an organizational overlay to the basic Generic Task Approach to facilitate integration.

The KLA is based upon the Knowledge Level Architecture Hypothesis (KLAH). This hypothesis builds on of what Newell proposed in his AAAI presidential address of 1980 (Newell, 1982). Newell's proposal was the existence of a distinct level of analysis for systems, the “Knowledge Level” which existed above the symbol level. What the Knowledge Level provides is a way of understanding a problem solving agent apart from the implementation of the agent. Although this allows deeper understanding of problem solving, Newell recognizes in his address that the behavior of an agent cannot always be predicted at the knowledge level. The reason for this deficiency is the lack

of any discussion of problem solving control. The KLAH on the other hand, allows discussion of the control issue, but only in terms of *knowledge organization and control.*

Knowledge organization and control are captured in the Knowledge Level Architecture according to the Knowledge Level Architecture Hypothesis as stated in (Sticklen, 1989, p.343):

> **Knowledge Level Architecture Hypothesis**: A problem solving agent may be decomposed into the cooperative efforts of a number of sub-agents, the larger agent can be understood at the Knowledge Level by giving a Knowledge Level description of the sub-agents and specifying the architecture the composition follows.

This hypothesis leads to the specification of a system by explicitly representing the interactions between its agents. There are two defining aspects of KLA:

- First, there is a distinct message protocol that exists between problem solvers. The message protocol between two cooperating agents is defined in terms of the functionality of the agents. In other words, the protocol provides a means for the agents to request work and respond in a vocabulary that the other can understand.
- Second, to allow communication between cooperating problem solvers, communication channels are established. By decomposing the agent into subagents and fixing the communication paths, the KLA provides a way of organizing the knowledge of the agents.

These aspects provide a means of organizing the knowledge of differing agents. Furthermore, since control is passed to an agent only when another agent sends a request, the KLA provides a means of understanding the problem solving activity taking place among the cooperating agents of an integrated system.

Another researcher in the TSA community has explored similarly motivated extensions to the original Knowledge Level concepts of Newell. Walter Van de Velde developed an extension to the Knowledge Level orthogonal to that proposed in the KLA. Although orthogonal, Van de Velde's extension at the bottom line also deals with integrating control issues at the Knowledge Level. This work, although relevant to our research, is beyond the scope of this paper. Interested readers are referred to (Van de Velde, 1991).

6.0 Knowledge Level Architecture for Wheat Crop Management

To address the problem of irrigated wheat management, we integrate the Generic Task Approach with CERES Wheat crop simulation model (Ritchie et al., 1985) into a Knowledge Level Architecture. Within our integrated system, we leverage compiled[1] level expertise to “propose” a manage-

ment scheme, CERES Wheat to test that scheme, and compiled level expertise to (possibly) modify the suggested management in light of CERES results.

Figure 1 shows a high-level overview of the system. To understand the processing performed by the system, we describe the architecture according to the object-oriented paradigm upon which it is based. At the top level, our system is composed of three cooperating agents, the Strategic Planner, CERES Wheat and the Plan Critic. These agents can be viewed as specialists whose tasks are plan generation, plan testing and plan critiquing respectively. The strategic planning module uses compiled knowledge of wheat crop management to propose a plan. Next, the CERES Wheat Module simulates the growth of the wheat under the circumstances set forth by the plan. Finally, the outcome of testing is handed off to the Plan Critic, where both experience-based knowledge and the farmer's opinion can be used to critique the plan. If the plan is found unsatisfactory, a redesign request is sent to the strategic planner. The farmer may also be asked to change any preferences that have a negative impact on the plan's performance. Finally, if the plan is approved by the Plan Critic specialist, it is handed off to the farmer.

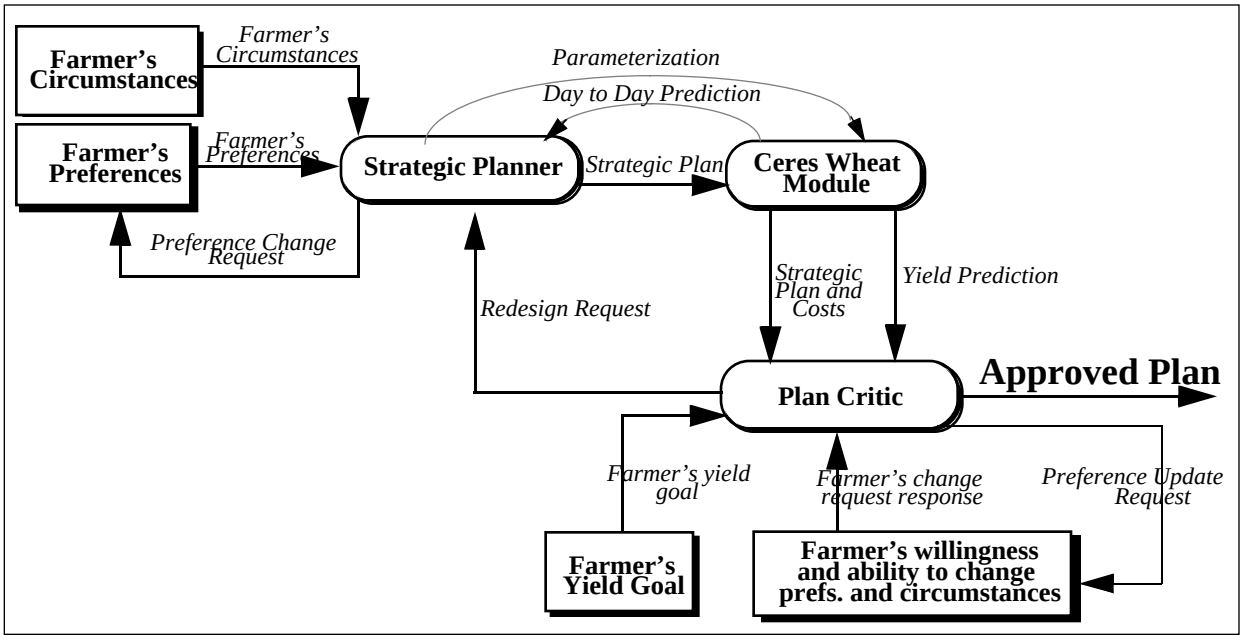

Figure 1: Irrigated Wheat Crop Management System

Our plan tester, CERES Wheat (Ritchie et al., 1985) is one of a family of dynamic process-orientated models which simulate the growth, development, and yield of major cereals. Our objective is

1. Compiled level expertise refers to knowledge based on previous experiences.

to exploit the model's simulation capabilities as a dynamic knowledge base for prediction of input demand and yield as influenced by farm and district-level management decisions. The model is sensitive to crop management decisions including choice of variety, date of planting, fertility levels, and irrigation amount and timing. With recently incorporated modifications, it can also simulate the impact of long-term climatic change on yield, crop duration, and nutrient losses. Modeling can thus be used to evaluate long-term agricultural productivity and sustainability in light of management decisions.

As shown in Figure 1, we differentiate between two sets of inputs to the strategic planner. The first is "Farmer's Circumstances." This input provides the system with information about set conditions under which the farmer is proposing to plant wheat. We contrast circumstances with preferences. The input of "Farmer's Preferences" provides the system with information about what the farmer would like to do or has done in the past. Often there are circumstances in which the farmer must plant that he/she has no control over, but there are also farmers who impose limits on them selves by sticking with traditions.

We feel it is important to make to the above distinction between circumstances and preferences. A major thrust of our work is the education of both inexperienced and experienced farmers. Therefore, we would like to allow a farmer to input their preferences for management decisions and if necessary critique these decisions. For example, if a farmer has always grown durum wheat, he/she may have a strong preference toward growing this type of wheat again. If we view this input as a preference, we can analyze the remaining conditions the farmer faces. Then we can determine if this decision is ideal, or if it would be better to plant a bread wheat in his region. If we find a farmer's preference runs counter to an optimal decision, we can offer the optimal decision with an explanation as to why the preference is suboptimal. The farmer is then allowed to choose between the preference and what the system determined as optimal.

7.0 Strategic Planning Module

From an expert systems viewpoint, the heart of our system is the Strategic Planning Module. The role of the strategic planning module is to generate a plan for the management of a wheat crop during an entire cropping season. To design this planning module, we follow the Knowledge Level Architecture approach. At the highest level, the task of creating a strategic plan is seen as a design problem - designing a plan for the management of wheat. Thus, the Routine Design Specialist is

incorporated as the top level problem solving agent. Although Routine Design was first developed for application in the design of engineering artifacts, it has proven applicable to plan generation (Chandrasekaran, Josephson, & Keuneke, 1986).

Routine Design makes use of hierarchical structures of design specialists to perform design, each responsible for a particular part of the overall plan. Hierarchies are used not because the plan is intrinsically hierarchical, but because hierarchical decomposition is a typical means utilized to manage complexity. The input to the Routine Design agent is a set of planning constraints, and the output should be a full set of specifications for the required plan.

To understand the problem solving conducted by a Routine Design agent, it helps to view the design problem solver as consisting of a collection of design specialists. Each specialist is responsible for accomplishing a small part of the overall design. Associated with each specialist is a list of plans that can be carried out to achieve its part of the design. S1 in Figure 2 has two such plans from which to choose. Generally each specialist chooses just one of its plans. Typically, the actions that constitute a plan include doing a calculation for a local value, satisfying a local constraint, and requesting another specialist to refine the current plan. For example, the left plan in S1 invokes the S2 specialist. If a plan fails, then alternate plans are tried. If part of a plan fails then an attempt is made to *redesign* the part of the plan that caused the failure. Potential causes of failure (i.e., where to try to fix a plan) are precompiled into the specialist.

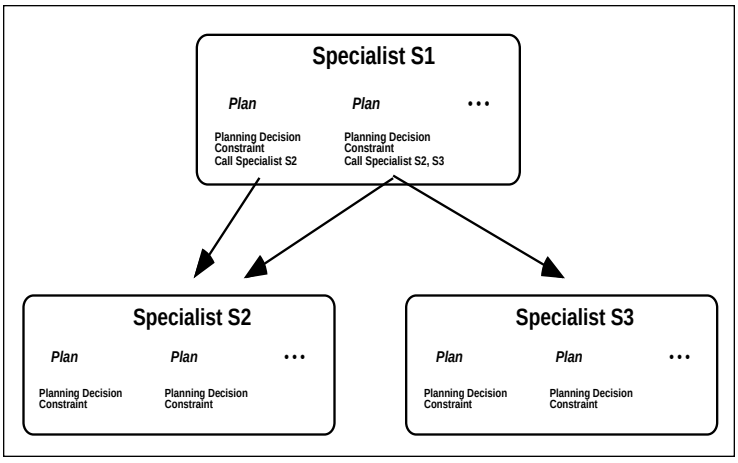

Figure 2: General structure of a
Routine Design problem solver.

Considering the Knowledge Level Architecture discussed above, the ability of another "specialist" to perform part of a plan is of particular importance. Normally, these subspecialists are other

routine design specialists. However, these specialists may be other problem solvers from the Generic Task tool kit or from another source, such as quantitative simulation models (CERES Wheat in our case). The KLA forces us to identify the communication channels which exist between two cooperating specialists and to identify the message protocol which they will use to communicate. In other words, how the Routine Designer requests work from other problem solving agents and what it expects in return must be stated explicitly.

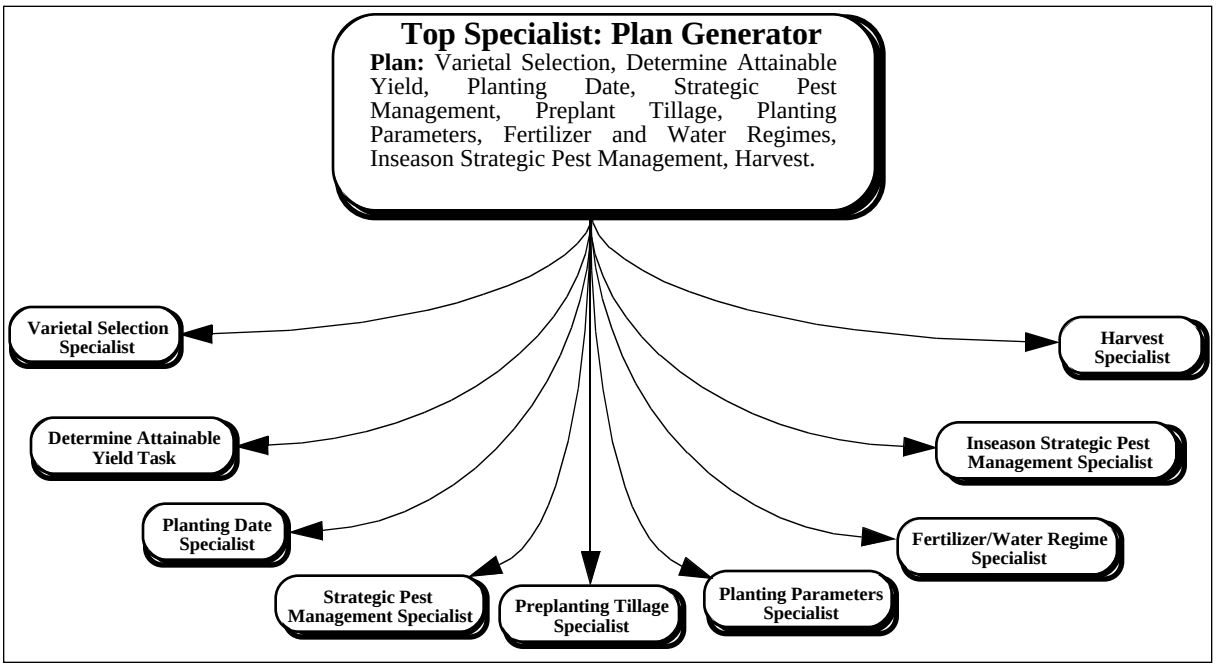

Figure 3: Top level Routine Design agent to perform wheat management plan generation.

Figure 3 depicts the top level architecture of the strategic planning module. The top specialist uses the subspecialists to perform the task of plan generation. We follow a linear ordering of the specialists to generate the plan. If any of the specialists fail due to constraints set forth by a previous specialist, redesign knowledge which is precompiled into the problem solver is used to choose an alternative design.

8.0 Module Descriptions

In the discussion that follows, we will give brief highlights of selected modules that demonstrate the applicability of our approach.

8.1 Varietal Selection Module

During the analysis of the wheat crop management problem, we identified varietal selection as a classification problem. We have identified a list of factors which experts consider when selecting a variety for planting. The varieties can be classified in terms of the factors we have identified. When the user presents his/her situation to the system, the system will classify this situation in an attempt to find a variety which is appropriate for the field in question.

We have chosen a Generic Task known as Hierarchical Classification (HC) to perform the varietal selection part of the plan. A classification tree has been built which classifies possible varieties in terms of the following factors:

- **Heat Resistance:** susceptible, medium susceptibility, tolerant or resistant.
- **Height:** dwarf, semidwarf, or tall.
- **Loose Smut susceptibility:** susceptible, medium susceptibility, tolerant or resistant.
- **Market:** bread or durum wheat.
- **Maturity:** number of days to maturity.
- **Region:** Upper Egypt, Middle Egypt, South Delta, North Delta, Fayoum, Southwest Coast.
- **Rusts susceptibility:** susceptible, medium susceptibility, tolerant or resistant to leaf, stem, and/or yellow rusts.
- **Planting Date:** optimal planting date.
- **Salinity:** tolerant or susceptible.

When the user runs the system, he/she enters the circumstance on their farm. For example, the system will ask the user if there are rust problems or salinity problems on the farm. Furthermore, the farmer can enter preference about various factors (such as planting date, seed color, time for maturity, etc.) as well as their willingness to change these preferences. The system will then classify the users circumstances/preferences to determine which varieties strongly match, match or weakly match their requirements.

8.2 Irrigation/Fertilization Module

The high-level picture of our system in Figure 1 shows CERES Wheat as a plan tester. However, this will not be the only function of CERES. We will also use CERES in plan construction. The irrigation and fertilization specialists will interact with CERES Wheat as shown in Figure 4. CERES Wheat will provide these module with estimates on irrigation/fertilization timings and amount by indicating water and Nitrogen stress to the plant.

The Knowledge Level Architecture (KLA) forces us to explicitly identify the points of interaction and the communication that can take place between cooperating agents. The interaction

between the quantitative simulation model CERES and our generic task Routine Design, shows that the agents which compose a complex problem solver need not come from the Generic Task perspective. Provided the communication and message protocol can be determined, any type of agent can be integrated. This offers considerable flexibility with our KLA approach since we do not limit our choice of agents to those provided by the Generic Task tool set.

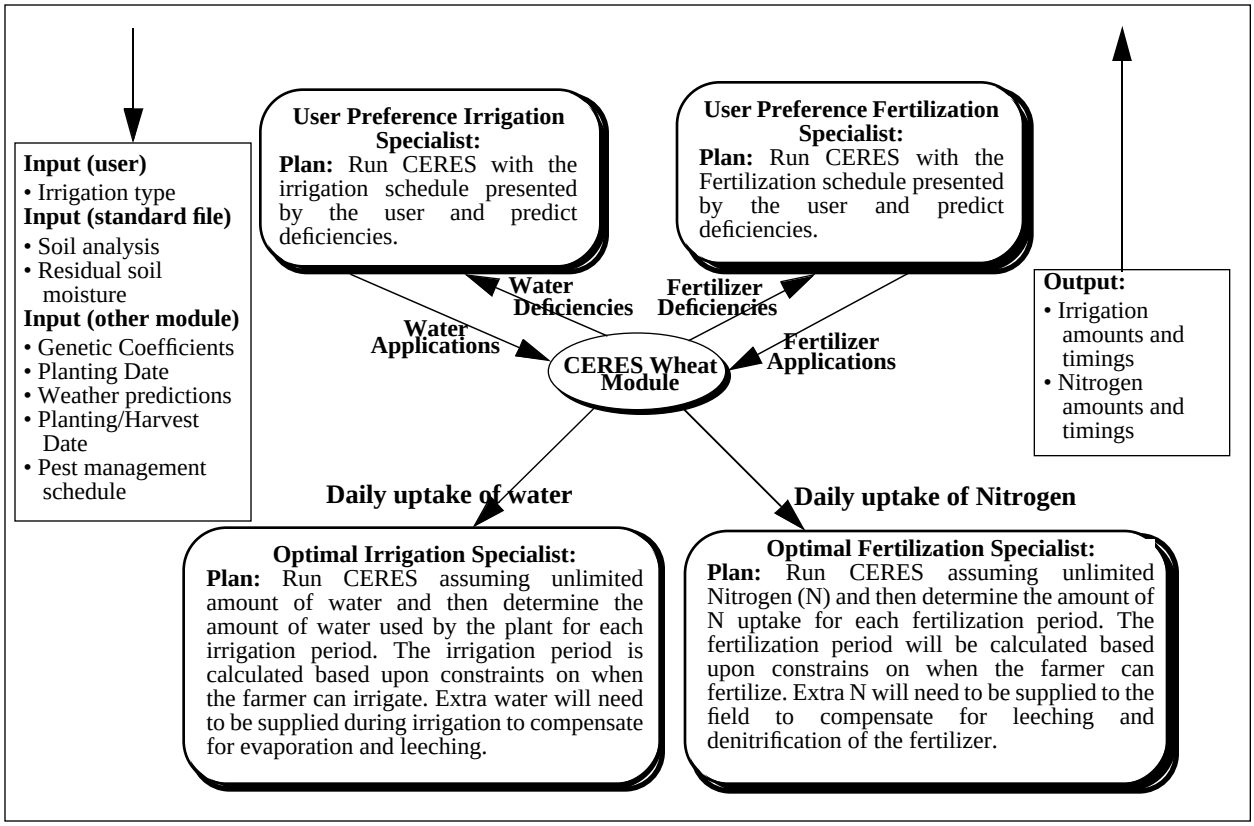

Figure 4: Interaction between CERES Wheat and Fertilization/Irrigation

As Figure 4 shows, if the user has a predetermined irrigation or fertilization plan, it will be tested by CERES Wheat. CERES Wheat will perform a simulation to show daily plant intakes of water and Nitrogen and show when any stresses occur. The Optimal Irrigation/Fertilization specialists will request CERES Wheat to simulate assuming unlimited water and Nitrogen supply according to an upper bound for the region. These specialists will then use the simulation data to plan amounts and timings of water and fertilization applications within constraints set by the user.

The weather data upon which the simulation is based is predicted from past weather data. Since this prediction can never be completely accurate, the schedule for irrigation/fertilization may need modification during the season to compensate for irregularities in the weather. However, by bring-

ing as much knowledge as possible to the task of Irrigation/Fertilization planning, we can make the best prediction possible.

8.3 Weed, Disease and Insect Management Specialist

Crop protection decisions at the strategic planning level are considerably more complex than those that occur with the treatment of a single pest[1]. The difference in complexity is due to the nature of both problems. The treatment of a single pest is often done by seeking the advice of an extension agent or company which specialize in crop protection. However, strategic planning must consider the entire set of pests which require treatment at a given time during the cropping season. The treatment of these problems cannot be considered in isolation due to the interaction which might take place between treatments. Examples of these interactions are:

- Treatment of one pest may cover multiple pests. Thus, it would be advantageous to consider treatments that would cover the largest number of pests in one application.
- If multiple treatments are needed, it may be more efficient to perform all treatments on the crop at the same time (as in the case of tank-mixes). However, due to interactions between treatments, this may not be possible.
- Treatments in the past may constrain future pest treatments, as well as future cropping plans. For example, irrigation scheduling can be affected by when a farmer wishes to spray his/her crop. Furthermore, chemical treatments which are safe for the current crop, can often have adverse effects on other crops during the next cropping season.

Due to the number of interactions which can exist between crop planning decisions, we must use a problem solving organization/strategy which can handle these complexities. The Generic Task which matches this description is Abductive Assembly. The goal of Abductive Assembly is to generate a composite hypothesis which will explain a set of findings. In the case of pest management, the goal of Abductive Assembly is to create a composite treatment which will treat all pests occurring in the field. The generation of such a composite treatment follows the same abductive algorithm described in (Punch, Tanner, Josephson, & Smith, 1990) as follows:

1. Given the list of possible pests (insects, weeds, or diseases) which need treatment, generate a list of possible treatments for each pest (consider chemical, mechanical and biological treatments).
2. Select one pest which needs to be treated.
3. For the pest chosen in step 2, select those treatments which are not yet ruled out.

1. We use the term “pest” to mean any weed, insect or disease which is damaging to the crop.

4. If only one treatment is available, select that treatment, otherwise select the most plausible treatment.
5. Add the treatment from step 4 to the compound treatment. Test for incompatibilities with existing treatments. If they exist we have two choices, either select another treatment for the current pest, or remove the incompatible treatments and mark the pests that they treat as untreated.
6. Mark as treated any other pests which can be treated by the treatment added in step 5. If some pests are still left untreated, then return to step 2.

By using this form of abductive problem solving, we hope to handle the complexity inherent in treating combinations of insects, diseases and weeds which occur in the field at a given time. The composite treatment plan we seek is the most efficient, economical and safe treatment available.

9.0 Conclusion

This paper emphasizes the connection between Generic Tasks, Task Specific Architectures, the Knowledge Level Architecture and object-oriented design and development. Object-oriented design provides a vocabulary for organization and control within an object by speaking in terms of the encapsulation of the data structures and behavior associated with objects in the domain. However, for knowledge intensive problems, a vocabulary which assist the knowledge engineer in identify these objects and their interaction is required. Generic Tasks (and other TSAs) provide this vocabulary by explicating organization and control structures specific to problem solving tasks which are common across domains. For multi-task problems, such as wheat crop management, the Generic Task approach alone is not enough since GTs do not provide a language to describe the integration of problem solvers. We use the Knowledge Level Architecture (KLA) vocabulary to describe the organization and control of several problem solvers integrated to perform multi-task problem solving.

Investigations into the KLA leads to a distinction between three levels of description of knowledge-based systems:

1. **The individual problem solvers:** A functioning system is an example at this level. The vocabulary used here is in terms of objects specific to the domain. For example, for the functioning MYCIN system, the terms would include *strep infection, patient fever,* etc.
2. **The problem solving types:** Since problem solving types span numerous domains, the vocabulary at this level uses problem solving specific, but domain independent terms. For example, classification is used in a number of different domains. Terms such as *classification hierarchy, established nodes*, etc. are used to describe the problem solver at this level.

3. **The Knowledge Level Architecture:** This is the highest level of description. The vocabulary used at this level is in terms of *agents, communication channels and message protocols.*

By describing a KBS in terms of these levels, a better understanding of the actual problem solving being performed is possible.

It is important to emphasize that the problem solving types from level 2 which are integrated into a KLA need not be from the Generic Task tool set. As the wheat crop management system shows, other problems solvers such as simulation models can also be incorporated easily into our system. Other Task Specific Architectures (TSAs) can be integrated as well. The other TSAs provide primitives which are of smaller grain sizes than the GT primitives. Therefore, the construction of a single problem solving agent often requires the combination of numerous TSA primitives. Once constructed, however, these problem solving agents can interact with the GTs along the communication channels to perform cooperative problem solving.

The irrigated wheat test bed provides an excellent trial for our ideas on the Knowledge Level Architecture. We have completed the design of the architecture for the irrigated wheat crop management system. Current work is focused on the implementation of the individual modules which will be integrated into our system. The module for varietal selection has been completed, as well as a picture based identification for pests. In the future, we wish to extend the ground work laid here to the management of other crops. The systems for various crops can interact and consider the possibilities of crop rotations over multiple years.

10.0 Acknowledgments

The Egyptian/American team which is performing this work consists of the two computer science groups led by A. Rafea in Cairo and Jon Sticklen at Michigan State University, and three agricultural groups: a simulation group at Michigan State led by Joe Ritchie, a wheat group at Michigan State led by R. Ward, and a wheat group at the Agricultural Research Center in Cairo led by A. Shafi. Without smooth integration between the agricultural groups and the computer science groups, this project would not be possible.

This research is supported by a joint USDA (OICD) / NARP funding. The Intelligent Systems Laboratory, MSU, receives substantial equipment support from Apple Computer.

11.0 Bibliography

Breuker, J., & Wielinga, B. (1989). Models of Expertise in Knowledge Acquisition. In G. Guida & C. Tasso (Eds.), Topics in Expert Systems Design: Methodologies and Tools Amsterdam: North Holland Publishing Company.

Brown, D. C., & Chandrasekaran, B. (1986). Knowledge and Control for a Mechanical Design Expert System. IEEE Expert, 3(July), 92-100.

Buchanan, B., & Shortliffe, E. H. (1984). Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley.

Bylander, T., & Mittal, S. (1986). CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling. AI Magazine, 7(2), 66-77.

Chandrasekaran, B. (1983). Towards a Taxonomy of Problem Solving Types. The AI Magazine, 4(winter/spring), 9-17.

Chandrasekaran, B. (1986). Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design. IEEE Expert, 1(3), 23-30.

Chandrasekaran, B. (1987). Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks. In IJCAI-87, . Milan:

Chandrasekaran, B., & al, e. (1979). An Approach to Medical Diagnosis Based on Conceptual Structures. In Proceedings of IJCAI 6, .

Chandrasekaran, B.,Josephson, J.,Keuneke, A., & Herman, D. (1989). An Approach to Routine Planning. International Journal of Man Machine Studies, 377-398.

Chandrasekaran, B.,Josephson, J. R., & Keuneke, A. (1986). Functional Representations as a Basis for Generating Explanations. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, .

Clancey, W. J. (1981). NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application To Teaching. In Proceedings of IJCAI 7, .

Clancey, W. J. (1983). Advantages of Abstract Control Knowledge in Expert System Design. In Proceedings of AAAI, .

Clancey, W. J. (1984). Classification Problem Solving. In Proceedings of AAAI, .

Clancey, W. J. (1985) Representing Control Knowledge as Abstract Tasks and Metarules. Stanford University, Palo Alto.

Duda, R. O., & Gaschnig, J. (1979). Model Design in the Prospector Consultant System for Mineral Exploration. In D. Michie (Eds.), Expert Systems in the Micro-Electronic Age Edinburgh University Press.

Gomez, F., & Chandrasekaran, B. (1981). Knowledge Organization and Distribution for Medical Diagnosis. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1), 34-42.

Josephson, J. R. (1987). A Mechanism for Forming Composite Explanation Hypotheses. IEEE Transactions on Systems, Man and Cybernetics, 3, 445-454.

McDermott, J. (1988). Preliminary Steps Toward a Taxonomy of Problem-Solving Methods. In S. Marcus (Eds.), Automating Knowledge Acquisition for Expert Systems (pp. 225-258). Boston: kluwer Academic Publishers.

Mittal, S. (1980) Design of a Distributed Medical Diagnosis and Data Base System. Computer and Information Science Department, Ohio State University.

Newell, A. (1982). The Knowledge Level. AI Magazine, Summer, 1-19.

Punch, W. F. I.,Tanner, M. C.,Josephson, J. R., & Smith, J. W. (1990). Using the Tool Peirce to Represent the Goal Structure of Abductive Reasoning. IEEE Expert, 5(5), 34-44.

Ritchie, J. T.,Godwin, D. C., & Otter-Nacke, S. (Ed.). (1985). CERES Wheat. A Simulation Model of Wheat Growth and Development. College Station, Texas: Texas A&M University Press.

Rumbaugh, J.,Blaha, M.,Premerlani, W.,Eddy, F., & Lorensen, W. (1991). Object-Oriented Modeling and Design (1 ed.). Englewood Cliffs, NJ 07632: Prentice Hall.

Steels, L. (1990). Components of Expertise. AI Magazine, 11(2), 28-49.

Sticklen, J. (1989). Problem Solving Architectures at the Knowledge Level. Journal of Experimental and Theoretical Artificial Intelligence, 1, 1-52.

Van de Velde, W. (1991). Tractable Rationality at the Knowledge Level. In L. Steels & B. Smith (Eds.), Artificial Intelligence and Simulation of Behaviour (AISB) Leeds, GB: Springer-Verlag.