ABSTRACT

A Simulation of a Monitoring and Alarm System in an Energy Management System

by
Hui-Jen Liu

This is a simulation for an energy system's monitoring and alarm and working under KEE's environment. Because of KEE's powerful capabilities, we can monitor and get warning signals from several active windows that we attaching them from KEE's system knowledge base to our source code.

It is very convenient for us to set every member of the energy system on an image panel. We can monitor the entire conditions of that member and get highlighted warning signals from its own image panel. Also we can initialize them by a kind of method-actuators windows if we want to restore the system.

A SIMULATION OF A MONITORING AND ALARM SYSTEM IN AN ENERGY MANAGEMENT SYSTEM

by
Hui-Jen Liu

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

Department of Mechanical and Industrial Engineering
May 1993

APPROVAL PAGE

A Simulation of a Monitoring and Alarm System in an Energy Management System

Hui-Jen Liu

Dr. Nouri Levy, Thesis Advisor (Date)
Associate Professor of Mechanical and Industrial Engineering, NJIT

Dr. Rajpal Sodhi, Committee Member (Date)
Director of Manufacturing Engineering
Associate Professor of Mechanical and Industrial Engineering, NJIT

Dr. Rong-Yaw Chen, Committee Member (Date)
Professor of Mechanical and Industrial Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Hui-Jen Liu

Degree: Master of Science in Mechanical Engineering

Date: May 1993

Undergraduate and Graduate Education:

- Master of Science in Mechanical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1993

- Bachelor of Engineering in Mechanical Engineering,
  Tamkang University, Tamsui, Taiwan, R.O.C., 1991

Major: Mechanical Engineering

This thesis is dedicated to
my parents

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his supervisor, Professor Nouri Levy, for his guidance, friendship, and moral support throughout this research.

Special thanks to Professor Rong-Yaw Chen and Rajpal Sodhi for serving as members of the committee.

And finally, a thank you to E-Pin Lin for his help.

TABLE OF CONTENTS

Chapter **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1
INTRODUCTION

1.1 Chapter Synopsis

This chapter describes a knowledge-based expert system, an Energy Management System [6] and the KEE (Knowledge Engineering Environment) [11] which is an expert-system-building-tool. This chapter also introduces the reasons and values of an expert system to be used as an alarm system to monitor an energy management system. This work is dedicated to those who want their energy transmission system to be under control easily.

1.2. Expert System

An expert system is a set of programs that manipulates encoded knowledge to solve problems in a specialized domain that needs human expertise. Energy systems are a recent product of artificial intelligence. An expert system is a knowledge-based system, that gets its power from the expert knowledge that has been coded into facts, rules, heuristics, and procedures. This knowledge is stored in a knowledge base separate from the control components. This makes it possible to add new knowledge or refine existing knowledge without recompiling the control programs. This greatly simplifies the construction and maintenance of knowledge-based systems [4].

1.3 Energy Management System (EMS)

Energy Management System is viewed by many electrical utilities as new and emerging technology. Operators can operate, control, monitor and restore a whole power system by EMS. The major components of an energy management system include: hardware, application programs, communications, man-machine interface and software support [4].

Many energy management system functions have been devised to answer three questions:

1. What might be done to improve system efficiency and reliability, and reduce stress on equipment?
2. Can the system withstand any single or a sequence failures?

What will be the state of the power in the near future after planned outage and generation changes occur?

All of the above questions are addressed by energy management experts with the development of computer and Artificial Intelligence theories. Now, expert systems can manage the energy systems very well, no matter what the energy systems are, simple or complex.

1.4 KEE (Knowledge Engineering Environment)

KEE is a kind of knowledge engineering language for frame-based representation. It also supports rule-based, procedure-oriented, and object-oriented representation methods. Its main features include multiple knowledge bases to facilitate modular system design, forward and backward chaining for its rule interpreter KEE's support environment includes a graphics-oriented debugging package, flexible end-user interfaces using windows, menus, and an explanation capability with graphic displays which can show inference chains.

KEE was developed by Intellicorp, is LISP based and is operated on systems such as Symbolic machines, Xerox 1100's, or TI Explorers. KEE has been used for the development of intelligent user interfaces, diagnosis and monitoring of complicated systems, planning, design, process control, scheduling and simulation. [2,3,4].

1.5 Monitoring System

An expert system used as a monitoring system has some basic characteristics:

1. Distinguish if alarms are necessary or unnecessary.
2. A short reaction time.

3. Keep the operators aware of most emergency situations.

4. Remind the operators of the problems as they occur. and of emergency actions to be taken.

To be useful in a monitoring system, an expert system will have to have the following characteristics:

1. Energy management system being a real time system is time critical and high speed processors are an absolute requirement for proper operation.

2. Many of the development and maintenance tools now used in stand-alone workstations can be made available to the Artificial Intelligence software environment [1,5,7,8,9].

1.6 Energy System

The Energy System, also called a Power system, is a huge and complex combination. Generally speaking, it includes several layers

1.6.1 The Distribution Level

The distribution level, which represents the lowest structural level of the power system, usually consists of two voltage levels

1. The so-called primary or feeder voltage which is moderately high (for instance, 240KV or above).

2. The so-called secondary or consumer voltage which is low (for instance, 120V).

1.6.2 The Transmission Level

The transmission system interconnects all the generating stations of the system; consequently, in addition to transmitting power to the bulk power substations and extremely large consumers, the transmission system also governs the power exchange or mutual assistance which results from the interconnection of power plants.

CHAPTER 2

THE PURPOSE OF THE MONITORING SYSTEM

In our modern world, people can not live without electricity. So, how to maintain electricity supply normally is one of the most important issues to the power companies. Power companies spent lots of money to build power plants, power distribution systems, and power transmission lines, just to provide enough electricity to people for using in their daily life. But with the growing population power systems had to grow up simultaneously. Now the modern power system has become huge and complex, and also hard to maintain and manage. As a result maintaining and monitoring power systems become a new engineering development topic.

We simulated a real-time monitoring system to monitor a power system. The so-called real-time means to respond immediately to inputs from sensors, conventional programs, or other computer-based devices. So, a real-time monitoring system means a system that can monitor and respond is inputs from sensors or other computer-based devices and that is what the system we developed is supposed to do. The monitoring system monitor all the power systems through sensors and collects all the data output from sensors or other computer-based devices (or we can just call them monitor points in general). The monitoring system will compare the data we collected from monitor points and expect normal value. If the data fluctuates between a acceptable range, then we treat them as normal conditions and record them. But if some of the data go out of the normal range, then our monitoring screens instantly to warn the operator to under take some proper procedures to maintain or restore the abnormal parts of the power system.

We developed this monitoring system focused on setting the normal value, display the actual value we collect from the sensors and to display warning signals.

By using the KEE environment to develop our displaying warning signals, we could get a humanized and easy-operating screen; a particular advantage of using the KEE environment is that it executes all the data comparing and displaying, automatically, Even some changes can be made using an ActiveImage, LISP-Listener or a conclusion of rules (all of them are functions of KEE system). No matter where the change originates, the comparing and displaying will take place automatically.

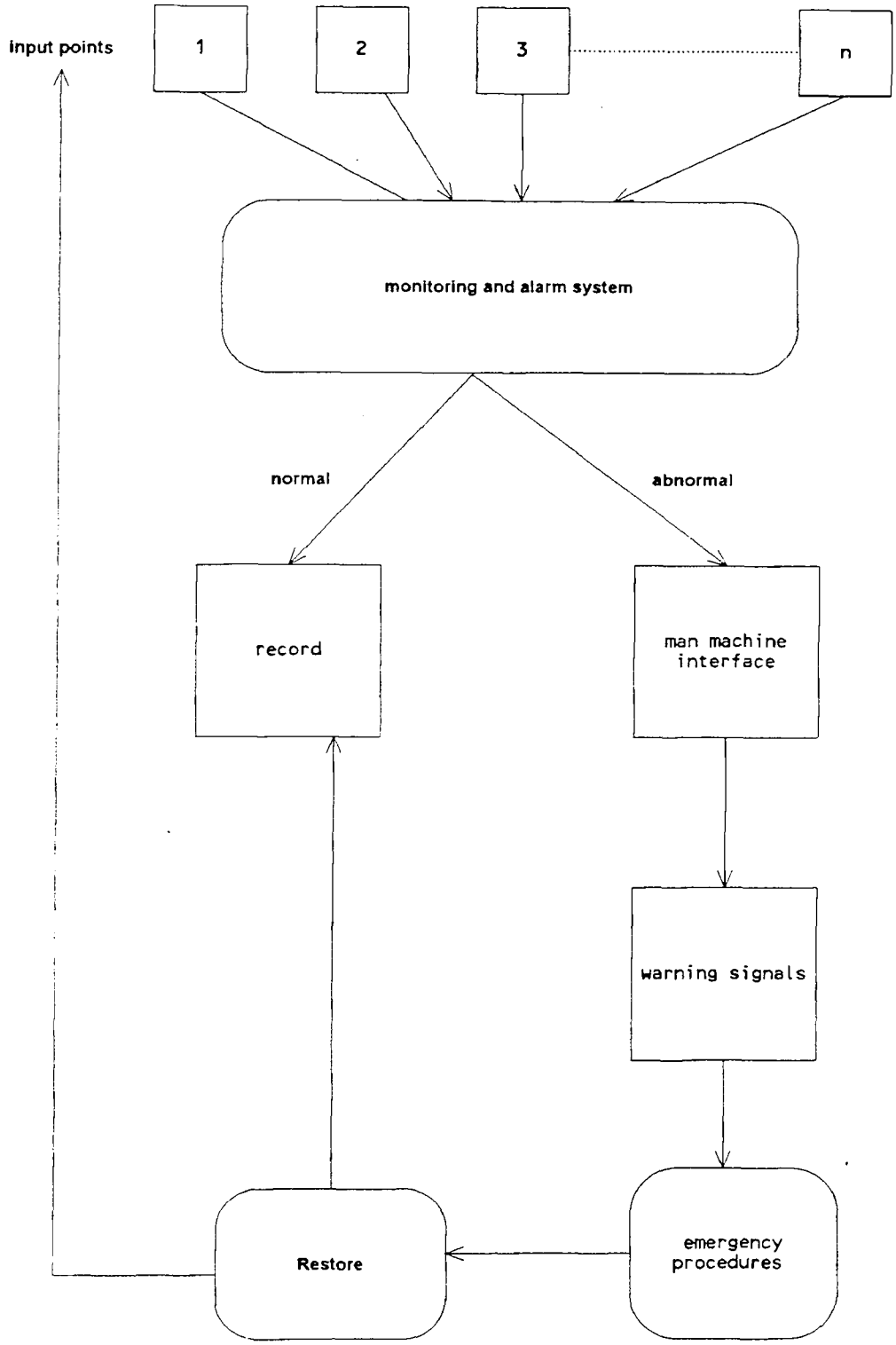

Fig 1 System Operation Flowchart

CHAPTER 3

KNOWLEDGE AND INFERENCE ORGANIZATION

3.1 The KEE System Structure

When we work with the KEE system, a collection of objects (while solving the problems, we probably think in terms of "objects", which can either be tangible objects: like cars, airplanes, etc., or intangible objects: like some procedures.) will be called a knowledge base. For tangible as well as intangible objects, it usually makes sense to differentiate between class objects and non-class objects. A class object represents a collection of objects sharing similar characteristics or attributes.

3.2 Objects

Objects in KEE system are usually called units, and from now on the term object and unit will be used interchangeably (some other system call the same thing a "frame" or an "entity"). A distinction is made in the KEE system between two types of objects (or we can call them units).

1. Class units
2. Non-class units

Non-class units are frequently linked to class units as members or instances of the more general class.

Attributes of objects are represented as slots in the KEE system. Slot can be used to represent two kind of information:

1. Descriptive or factual information such as the height of building.
2. Procedural information in the form of a code.

One benefit of having class units, and units that are member of classes is conceptual clarity. More important, however, we can control how slots (attributes) of units are

inherited by objects lower in the hierarchy. Whether a slot in a unit is inherited is in part determined by the type of link between two objects. In the KEE system, we can work with two kind of links:

1. Class-subclass relationship (represented by a solid line)
2. Class-member relationship (represented by a dashed line)

shown as Fig 2.

3.3 Attributes of Objects

Descriptive or sometimes procedural information about units is stored in slots. To make the distinction between local and general information possible, the KEE system offers two kind of slots: member slots and own slots. To make the distinction between existence and full specification of a slot possible the KEE system differentiates between slots and slot values.

3.3.1 Member Slots

Member slots are used to store information relevant for a class of objects. They will be inherited by each descendant of the class unit. Member slots are called member slots because all members of a class share them. We can give a member slot a value to be passed to its descendants, or we can leave the value unspecified.

Member slots stay member slots if they are inherited down a class-subclass link (solid line). Member slots become own slots if they are inherited down a class-member link (dashed line)

3.3.2 Own Slots

Own slots hold information specific to one object. They should therefore not be inherited by other objects lower in the hierarchy. The information is specified only locally and is not passed down to any of the members of the class shown as Fig 3.

3.4 Summary

We can get some concepts in our mind after reading the description about KEE system above.

1. Units are often organized into hierarchies.
2. Objects hierarchies are composed of class units and their subclass and member
3. Units have characteristics or attributes that are stored inside the unit as slots.
4. Slots can be entered at any level and specified at any level in the hierarchy.

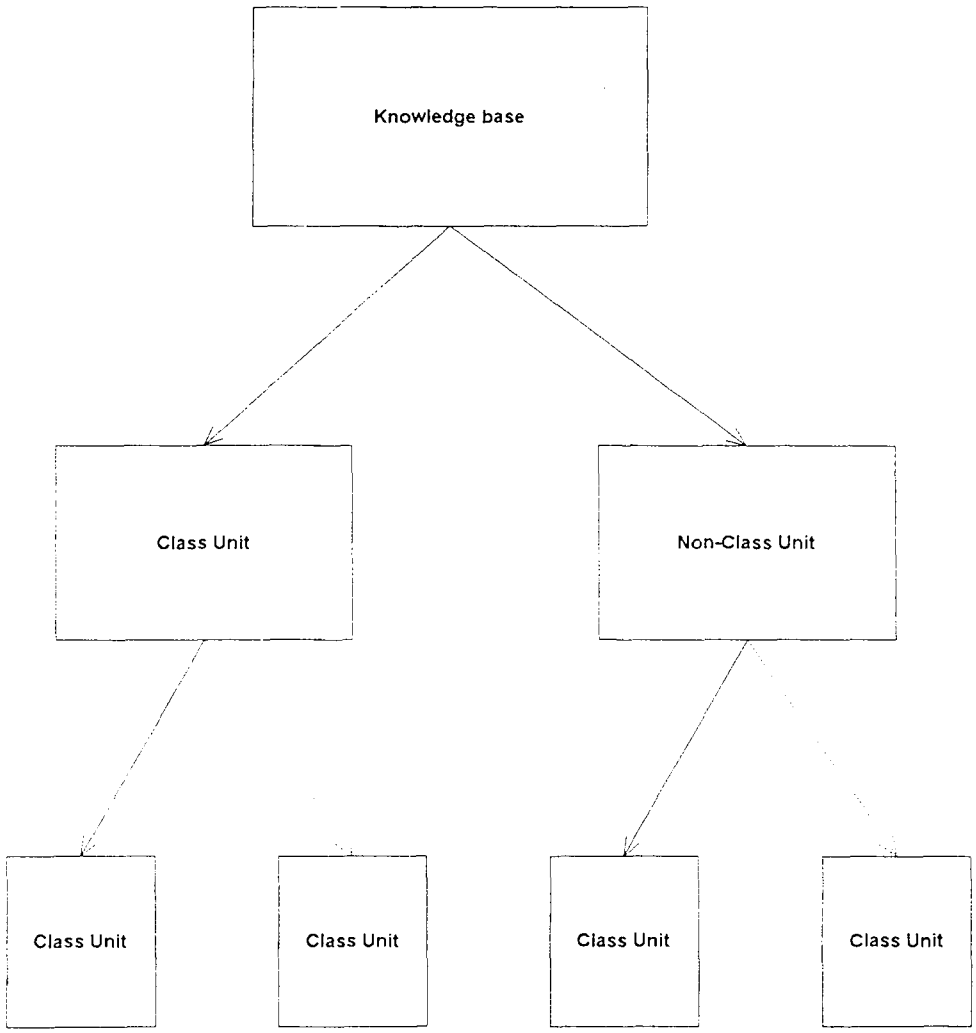

Fig 2 KEE System Structure

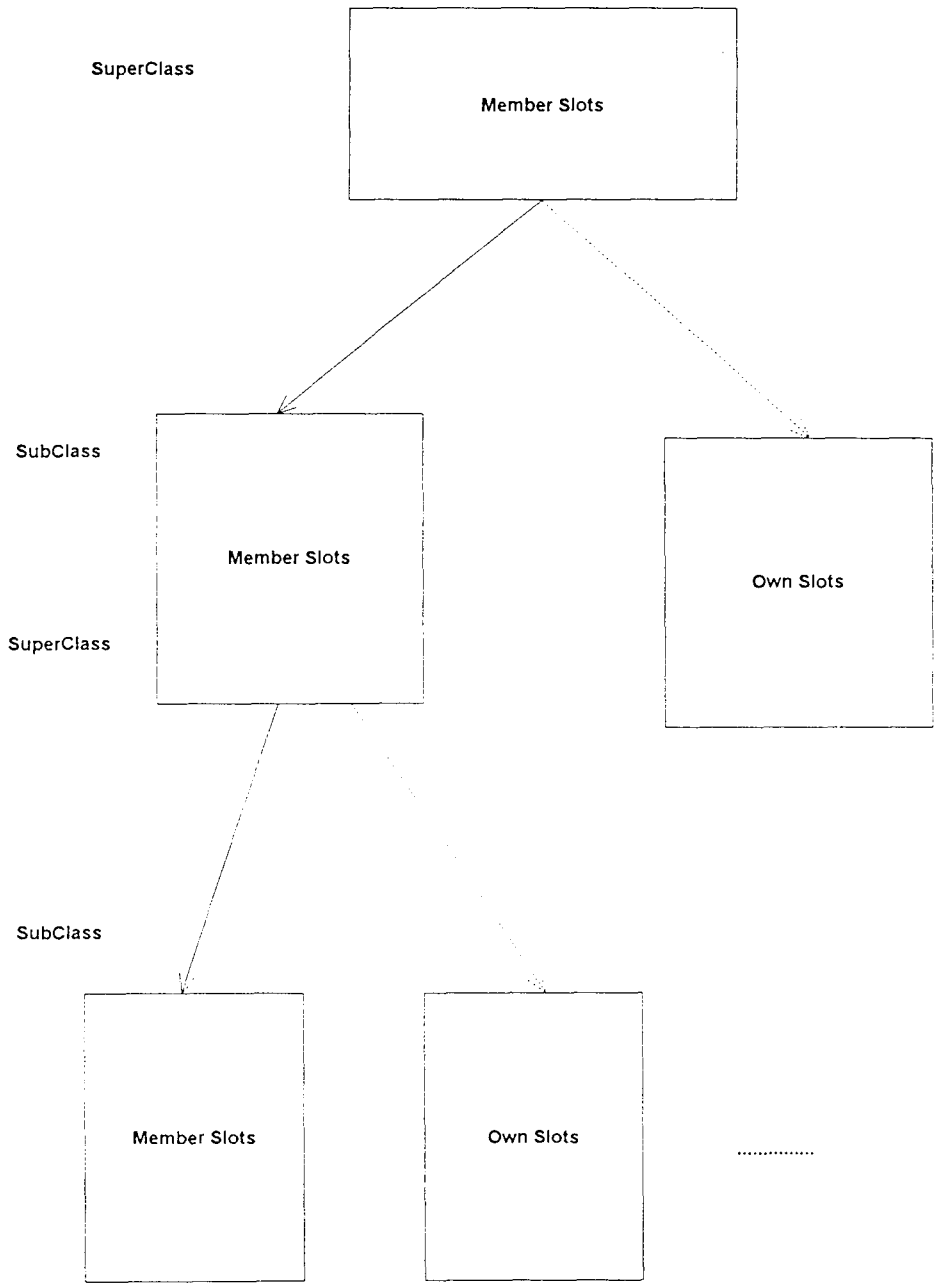

Fig 3 KEE System Hierarchical Organization

CHAPTER 4

THE ENERGY MANAGEMENT MONITORING AND ALARM SYSTEM DESIGN DETAIL

4.1 Chapter Synopsis

This chapter introduces the Monitoring and Alarm System of an Energy Management System. This Monitoring and Alarm System is created under KEE's environment and written in COMMON-LISP language. The function of this Monitoring and Alarm System includes monitoring circuits, fuses, buses (loads). If there is any state abnormal, it also will have a warning signal to warn the operator that what or where the wrong state is happening. This system also focuses on the restoration of the Energy System. Operators can initialize the system from the METHOD-ACTUATOR which says "MANAGER::INITIALIZE!METHOD".

4.2 The Structure of the System

The Monitoring and Alarm System is written in COMMON-LISP, and used under KEE's environment. The whole system contains two parts:

4.2.1 Knowledge Base

It includes a header part and 14 knowledge bases. It exists as a filename "MANAGEMENT" with a extension "U".

4.2.2 Function File

It contains 9 function definition files which exists as a filename "MANAGEMENT" with a extension "LISP". After compiling, the KEE's system creates a new file with name "MANAGEMENT" and an extension "sbin".

We explain the two parts in detail follows

4.2.3 Knowledge Base: MANAGEMENT.U

The header part: This program is a collection of several small knowledge bases with a header at the beginning of this program. This part is something like a declaration. It tells us the name of this program, when and who created and modified this program, the corresponding function file's name, the size of this knowledgebase, and several version numbers we used.

The knowledgebase part: This part contains 14 knowledgebases. We call these knowledgebases as "unit" or "slot".

4.2.3.1 ELECTRICAL.COMPONENTS

We see this knowledgebase as a class unit. From the program, we can see that the contents of this unit includes the name of this unit and the name of who created it. This unit has a super class "ENTITIES", which is inherited from KEE's System's KNOWLEDGEBASE "GENERICUNITS", and is a member of "CLASSES" in "GENERICUNITS". It also has two subclasses named "FUSES" and "CIRCUITS" respectively, which we can see later. This unit has three members slot called "INITIALIZE", "PLACES.AFFECTED", and "STATUS" respectively. The "INITIALIZE" slot is a kind of Method as we can see in the program. The second slot is called "PLACES.AFFECTED". It has at least one value from unit "PLACES". The third slot is called "STATUS". It is much like "PLACES" but it only has one value from its value class which contains: OK, NOW.OK, OVERLOAD, BLOWN, REPLACED, NEEDS.REPLACEMENT.

4.2.3.2 CIRCUITS

It is a class unit of "ELECTRICAL.COMPONENTS" and has two members named "CIRCUIT1" and "CIRCUIT2". It has seven member slots. The first slot is called "APPLIANCES". Its value is coming from unit "ELECTRICAL.APPLIANCES". The second slot is called "CURRENT.FLOW". It has only one integer value. The third slot is called "VOLTAGE". It also has only one integer value. The fourth slot is called "INITIALIZE", which is a method as we can see in the program. The fifth slot is called "REVELANT.FUSE". It has only one value in the unit "FUSES". The sixth slot is called "STATUS". It has only one choice from "OK, NOW.OK, OVERLOADED". The last slot is called "TOTAL.CURRENT". It has a choice from "TOO.HIGH, NOW.NONE, OK".

4.2.3.3 ELECTRICAL.APPLIANCES

It is a class unit of the KB "Management". It is the same as class unit "ELECTRICAL.COMPONENTS" which has a super class "ENTITIES" and is a member of "CLASSES". There are six member slots in this unit. The first slot is called "CURRENT.DRAWN". It has only one integer value. The second slot is called "INITIALIZE", which is a kind of method as we can see in the program. The third slot is called "LOCATION". It has only one value. The fourth slot is called "RELEVANT.CIRCUIT". It has only one value from class "CIRCUITS". The fifth slot is called "STATUS". It has only one choice from "OK, OUT". The last slot is called "SWITCH". The choice of its value is one of "ON" and "OFF".

4.2.3.4 FUSES

It is a class unit of "ELECTRICAL.COMPONENTS" and a member of "CLASSES". It has two members named "PLACE1.FUSE" and "PLACE2.FUSE". There are four member slots in this unit. The first slot is called "CHECK.FUSE". It is a method as we can see in

the program. The other slots "INITIALIZE", "PLACES.AFFECTED", and "STATUS" are as mentioned before.

4.2.3.5 PLACES

It is a class unit of KB "Management". It is a easier unit. It has two member slots in this unit called "INITIALIZE" and "STATUS" respectively as we mentioned before.

4.2.3.6 MANAGER

It is a class unit of KB "Management" and has one own slot named "INITIALIZE.ELECTRICAL.SYSTEM". This slot is a method as we can see in the program.

4.2.3.7 PLACE1 and PLACE2

They are two members of "PLACES".

4.2.3.8 PLACE1.FUSE and PLACE2.FUSE

They are two members of "FUSES" and are connected with unit "PLACES" by slot "PLACES.AFFECTED".

4.2.3.9 BUS1 and BUS2

They are two members of "ELECTRICAL.APPLIANCES". Both of them have a "CURRENT.DRAWN" integer value. They are connected with units "PLACES" and "FUSES" by slots "LOCATION" and "REVELANT.FUSE" respectively.

4.2.3.10 CIRCUIT1 and CIRCUIT2

They are two members of "CIRCUITS". They have two integer values for "CURRENT.FLOW" and "VOLTAGE" respectively. They also connected with units

"ELECTRICAL.APPLIANCES", "FUSES", and "PLACES" by slots "APPLIANCES", "REVELANT.FUSE" and "PLACES.AFFECTED" respectively.

4.3 The Operating Procedures

These procedures are getting start with going into the KEE's environment then:

4.3.1 Open an "IMAGE PANEL" for "CIRCUIT1"

4.3.2 Attach Proper Image Windows to Every Slots in the Unit "CIRCUIT1"

1. APPLIANCES: Simple-Value-Displays.
2. CURRENT.FLOW: Numeric-Dials;
   Digimeters;
   Numeric-Alarms (2);
   Value-History-Monitors.
3. INITIALIZE: Method-Actuators.
4. PLACES.AFFECTED: Simple-Value-Displays.
5. REVELANT.FUSE: Simple-Value-Displays.
6. STATUS: Vertical-Push-Bottoms;
   State-Alarms;
   Value-History-Monitors.
7. TOTAL.CURRENT: Vertical-Push-Buttons;
   State-Alarms;
   Numeric-Alarms;
   Value-History-Monitors.
8. VOLTAGE: Numeric-Dials;
   Digimeters;
   Numeric-Alarms (2);

Value-History-Monitors.

When every slot has been set up, the system is at normal state. But if you change the value of the slot "CURRENT.FLOW" and let it go out of the limits which are set by two "Numeric-Alarms", then the relative alarms will highlight till you initialize the system.

4.4 The Chosen Windows and Their Functions

To choose a proper image window is an important step for operating this system. The right choices can show you the situations that the system is situated now, easily and clearly. The choices we made for those slots are as follow:

4.4.1 BUS1 and BUS2

1. SWITCH: State-Dials
2. CURRENT.DRAWN: Numeric-Dials;
3. Digimeters;
4. Numeric-Alarms (2);
5. Value-History-Monitors.
6. INITIALIZE: Method-Actuators.
7. LOCATION: Simple-Value-Displays.
8. REVELANT.CIRCUIT: Simple-Value-Displays.
9. STATUS: Horizontal-Push-buttons;
   State-Alarms;
10. Value-History-Monitors.

4.4.2 CIRCUIT1 and CIRCUIT2

as we mentioned above.

4.4.3 PLACE1.FUSE and PLACE2.FUSE

1. CHECK.FUSE: Method-Actuators.
2. STATUS: Vertical-Push-Buttons;
   State-Alarms
3. Value-History-Monitors.
4. INITIALIZE: Method-Actuators.
5. PLACE.AFFECTED: Simple-value-displays.

4.4.4 PLACE1 and PLACE2

1. STATUS: Vertical-Push-Buttons;
   State-Alarm;
   Value-History-Monitors.
2. INITIALIZE: Method-Actuators.
3. MANAGER: INITIALIZE: Method-Actuators.

4.4.5 The Functions of These Image Windows

1. Simple-Value-Displays: These kind of image windows can display the value of the attaching slot. You can not update their values from those kind of windows. So it is only good for the slots with fixed values.

2. State-Alarms: If the value of this slot is equal to the value of the State-Alarms, this alarm will highlight.

3. Vertical-Push-Buttons or State-Dials: They are good for that the slots which have a valueclass of "ONE.OF".

4. Numeric-Dials: These kind of windows are good for integer valueclass slots. They can be updated easily but it is not easy to see the actual value. So we combined it with a Digimeter for a clear digital display.

5. Digimeter: As mentioned above, it displays a value in digits but can not be updated from window directly.

6. Numeric-Alarms: They can set up digital limits for integer valueclasses.

7. Value-History-Monitors: They can record many values that happened before sequentially.

8. Method-Actuators: These kind of windows are good for a method valueclass slot.

You can activate this method by left clicking the mouse when the cursor is in its window.

4.5 Using ActiveImages

ActiveImages is a graphics package that is part of the KEE system. We can use ActiveImages to graphically represent slots, objects, and knowledge base. The ActiveImages package is built on KEE pictures which is developed on top of Common Windows.

4.5.1 Types of Images

When we create an image, the KEE system creates a unit and its graphic representation on the screen. The image unit is a member of the image class we choose in the ACTIVEIMAGES3 knowledge base and it inherits functionality from this image class.

4.5.1.1 Panel Image

Panel Images are a type of window that allows the user to collect all related images into one central location. There can be many panels in one application that would monitor different parts of the system. Panels can be attached to units, knowledge bases, or even other image panels.

There are two main advantages to using panels

1. We can visually arrange related images on the screen. That means we can have a predetermined layout that is then fixed for that knowledge base until we decide to change it.

2. More importantly, we can use image panels to manipulate related images as a group. For instance, it is possible to bury, close, delete, move, or shrink all images on a panel with a few simple mouse clicks. That means that we can shrink a screen size panel with two mouse clicks, and when it is expanded later, all images on it will appear together the appropriate place.

4.5.1.2 Slot Images

The greatest variety of images is given in the SLOT-IMAGES class. What type of image we can attach to a slot depends on the valueclass of the slot.

4.5.1.2.1 Numeric Images

A numeric value class will result in a selection of numeric images. For example, if we attach an image to a VOLTAGE slot and the valueclass of the slot is numeric (that is, NUMBER, INTEGER, etc.) the system gives us a choice of images suited for numbers, such as DIGIMETERS, THERMOMETERS, and so on.

4.5.1.2.2 State Images

If we have a valueclass (ONE OF A B C), we will get, among others, the choice of STATE IMAGE. A state image can show the possible values of a slot, the current values, or no value if the value is unknown. It can also change the value if an actuator image is chosen. VERTICAL-TRAFFIC-LIGHT is such a image.

4.5.2 Attaching Images

We can attach images through the mouse and menu interface or programmatically. When we attach an image to a slot through the mouse and menu interface, the system offers us a wide selection of possible images from which to choose. The images to choose from come from the subclass of SLOT-IMAGES in the ACTIVEIMAGES3 knowledge base shown above. We will not have to choose among all classes each time, however, since the KEE system is designed to pass down the choices so that only the appropriate ones are revealed to the user.

In addition to representing slot values, SLOT-IMAGES can also be used to change slot values. For many SLOT IMAGES, the system offers us an interactive image or a simple display image. The interactive image is often called an actuator and is usually distinguished by having "actuator" as part of the image name.

There are more than twenty SLOT-IMAGES that could be attached to a slot potentially. The KEE system helps us to choose the appropriate image by narrowing down the selection and offering any images good for a slot if attachment takes place through the mouse-and-menu interface. The system makes the selection by looking at the valueclass of the slot to which we are attaching the image. The system only displays images that make sense for the given valueclass.

4.6 The Structure of the Program

We introduce the structure of the program in Fig 4. From the diagram, we can see that there is a knowledge base "Management" and that it has four subclasses: "electrical component"; "electrical appliance"; "places" and "manager".

4 6.1 Electrical Components

This subclass means that the component we use in power system is a monitor point. It includes two subclasses: Fuses and Circuits and three member slots: Initialize , Place

Affected, and Status. Both of these two subclasses will inherit the characteristic of these three members from their superclass: Electrical Component.

4.6.2 Electrical Appliances

This subclass means the products we use to consume electricity. All these kind of products we called them buses. These two buses mean two products we connect on the circuits to consume power and also we use them as two monitor input points. They have three slots: Current Drawn; Location and Revelant Fuse. Current Drawn has a integer value and means the value of current flow through the bus. Location means where the bus is located and this slot also attaches to the Places subclass. Revelant Fuse means which fuse is taking care of this bus and it also attach to the Fuses subclass. This subclass represents the relation between the current fuse and its location and is connected by buses. If the current flow through the bus is over its normal value (Current Drawn value) then the fuse status is blown and the Location status becomes dark, and hence we can easily know where and which component is out of order.

4.6.3 Places

This subclass means where we put our monitor input points and which fuses are going to take care of them. There are four slots in this subclass: Check Fuse Status Initialize Place Affected. Check Fuse can inform an operator to check a fuse if some abnormality just happened and the system was restored already. Status can display the place now dark or light, and PlaceAffected shows us where the location has all these components, and their conditions.

4.6.4 Manager

Manager is a subclass different from the other three subclasses The other three subclass can do some judgments and display any one of abnormal conditions to warn the operators

to do some emergency procedures. After the abnormal conditions are recovered, operators must use this Manager to restore the system, and let all the values set to their normal conditions indicating what emergency is now over.

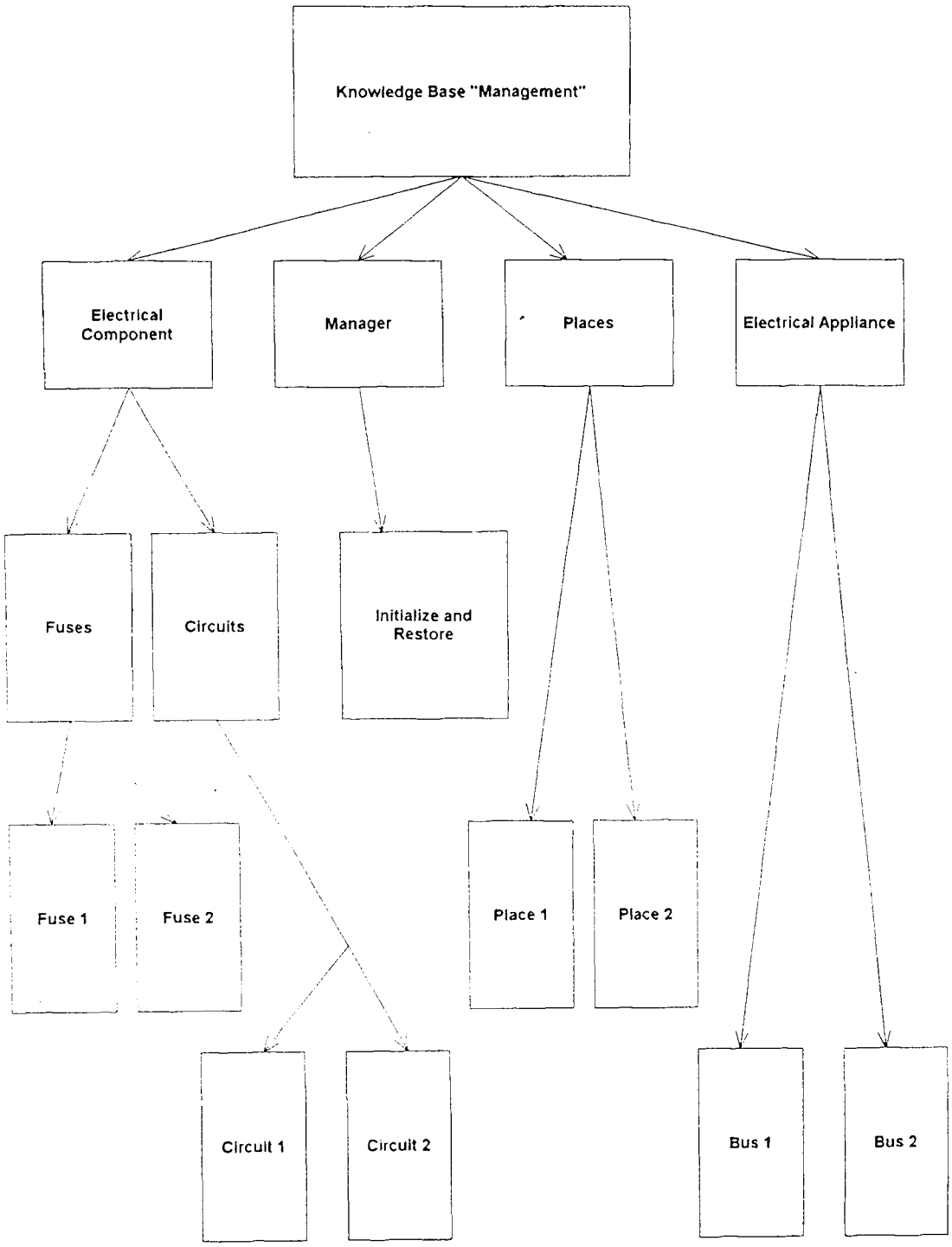

Fig 4 Program Structure

CHAPTER 5
RUNNING THE SYSTEM

The Monitoring and Alarm System of an Energy Management System that we developed before has been operating on a SUN WORKSTATION under KEE's environment successfully. We can monitor all the conditions on the transmission lines and receive alarms for abnormal states. We also restore the energy system after the abnormal state(s) was under control.

We monitor circuits by several kinds of active image windows. Simple-Value-Displays windows can show the slot value of the three slots "APPLIANCES", "PLACES.AFFECTED" and "REVELANT.FUSE". These values in these kind of windows can not be changed by activating these windows. For example, if there is any abnormal state that happened in the "CIRCUIT1 IMAGE PANEL", then we can know what load is connected to this circuit, what places are affected by this abnormal state and which is the relative fuse right away. On the other hand, we monitor the "CURRENT.FLOW", "VOLTAGE", "STATUS", "TOTAL.CURRENT" by more complex alarms. We attach 5 image windows to monitor and alarm "CURRENT.FLOW" and "VOLTAGE" slots. We use "Numerical-Dials" to adjust the current flow and "Digimeters" to see the actual number of current flow (or we can replace these two image windows by a "Digiactuators" and a "typescript" window). Two "Numeric-Alarms" can set up the upper and lower limit for ranges of current flow and voltage. If the values of these two slots ("CURRENT.FLOW" and "VOLTAGE") are going out of their ranges, then "Numerical-Alarms" will highlight till the values goes back to the fixed ranges. The last one is a "Value-History-Monitors". This window can record every change of the current flow and voltage as a reference for operators.

The other units are attaching image windows similar to the "CIRCUITS" unit. So we do not want repeat them again.

Because of KEE's powerful capabilities, can easily attach any kind of monitoring or alarm windows to our source codes. There is also a Method-Actuator that can activate the "INITIALIZE" slot to restore the units respectively no matter what circuits, fuses or buses. We can even initialize the whole system by activating the "MANAGER" unit.

KEE system has two way of giving objects behavior. The first way is to create a method slot and enter the behavior as Lisp code in the slot. The behavior is activated by sending an object a message. The other way is using active values as we used in this project.

Active values are a special kind of unit in the KEE system that have the pre-defined task of watching a slot. When using an active value, we make a behavior an inherent part of the slot (almost like its cardinality or valueclass). The behavior executes automatically under pre-defined conditions as illustrated.

A particular advantage of using an active value is that it will execute the behavior automatically. Every change can be made using an Active Images, using a Lisp method, or as a conclusion of a rule. No matter where the change originates, the checking and message printing will take place automatically.

ActiveImages can be used as a tool in two major areas:

1. During prototyping, we will find them helpful to monitor or change slots that are accessed frequently. During that stage, they function as a debugging facility or just as an easy switch for setting values. ActiveImages can be attached quickly to slots that require monitoring or frequent changes, and detaching them is just as easy.

2. When our prototype becomes more complete, ActiveImages can be used to develop extensive user interface. Various types of images offer a wide range of uses. ActiveImages can act as a window to the knowledge bases; allow user input; provide instructions; or permit the user to alter the knowledge bases.

The KEE system takes advantages of active values in a variety of ways. One very prominent use is ActiveImages. ActiveImages are pre-defined graphics tools that can be

used to monitor slots. ActiveImages are a subclass of active values and their behavior is pre-defined in the system. Instead of a user-defined message, we have a system-defined change in a graphical representation on the screen.

The implementation of active values in the KEE system lends itself well to working in an object-oriented fashion since code reuse is encouraged. We can have the same active value attached to a variety of objects in our knowledge base. The code also resides directly in the object and is actually part of the behavior of that object, which also has certain implications:

1. Active values are inherited with member slots they are attached to. We can attach an active value to a slot at one level in our object inheritance; it will then monitor that slot at that level and any lower level at which the slot exists.

2. Just like methods that use the obligatory argument SELF to get information from the unit in which the method is messaged, active values can pick up appropriate information from the unit in which they are activated using an argument in the argument list. This menu, in the project above, the PLACE.AFFECTED slot will have a different value for different individual fuses and the active values will work with that value.

The most valuable achievement of this system is that it very easy to understand and operate. Operators do not need to know much about KEE or Common-Lisp. They can control it easier then ever. All the system becomes is many active graphs as you can see. You can operate or control the system through all the active image windows. By the way, all we can describe about this system is that it is "humanized", and that is just what we are trying to do.

CHAPTER 6
FUTURE WORK

A modern energy system is combined with many different and complex subsystems. For example, the energy system includes power plant, power transmission system and power distribution systems. They all are huge and complex. So, a good energy management system must be also big and complex, to manage the whole energy system properly.

In the previous work, we simulated a simple monitoring and alarm system for a small range on the energy system. This is a beginning step for us to create a whole energy management system piece by piece. In this step, we are going to set up a real-time emergency states advisory system. That is because when a abnormal state alarm is highlighted to warn an operator to take some procedures, operators can do it with sufficient time and make a proper decision. But if there comes a good deal of highlighted alarms in a short period of time, then all the operators must have a advisory system to assist them doing the right and proper procedures to keep the energy system working normally.

In this project, we simulated a real-time monitoring and alarm system. These system are used to respond immediately to inputs from sensors, conventional programs, or other computer-based devices. Real-time system makes instantaneous knowledge-based decisions that are needed to respond to different environmental situations. These expert systems are seen in combat support roles in fighter and bomber airplane and in commercial airplane as pilot advisors. Control systems for chemical plants, power plants and manufacturing machining process control are other examples of real-time expert systems

The working principle of the monitoring and alarm system we developed is that the system gathers data from multiple monitored points and compares them to expected normal values. If there is any difference then the system generates a warning signal. We can show the procedures as figure 1.

A good energy management system not only tell you where and what happened, but also show you what you should do. At that time you can say that you have a real expert system.

CHAPTER 7
CONCLUSION

All power companies operate under more or less formal contracts with their customers. In general, these contracts specify the maximum quantity and the quality of the power to be supplied. The quality of the power is described by

1. The frequency and the limits between which it is constant.
2. The voltage and the limits between which it is constant.
3. The continuity of service.

In the normal, steady operating conditions of the power system, the objective is to produce the power where it can most economically be done at the time, and to distribute it to the points where it is needed while the voltage and frequency are maintained within suitable tolerance.

When faults occurs they must be eliminated by selectively de-energizing the faulted line section or equipment, by opening the circuits breakers which are adjoined to the faulted apparatus. It is the duty of the protective relays of the network protection system to give the tripping commands to the right breakers.

The network protection system gets the wrong state messages from the monitoring and alarm system. The monitoring and alarm system shows that what and where faults happened. When a warning signal is highlighted, it means a kind of fault happened. Operators can get all of the information about the wrong state, relative parts, locations and even restore the system from the image panel that we set up already.

Because of the development of Expert System, we have many convenient Expert-System-Building-Tools now. Such as KEE we used in this project. KEE's powerful window-frame capabilities provided us an excellent tool to simulate such a humanized and easy-operating system. It is very simple for operators to operate and do not need other special language skill. That is why expert system's application developed so fast. It is also

extensible. We do not have many units in this project. But if it is necessary, we can add as many units as we need very quickly and easily. So its scale is very flexible.

An expert system used as a monitoring and alarm system is convenient and effective as we described above. After this project, we can say that an Expert System is not only a computer program but also a real applied expertise.

APPENDIX
SOURCE CODE

```
;;;   -*- Mode:Common-Lisp; Syntax:Common-lisp; Package:KEE;
Base:10. -*-
(MANAGEMENT
("hxl7866" "11-26-92 11:22:33" "hxl7866" "11-27-92
15:23:12")
NIL
(KNOWLEDGEBASES)
************************************************************
*  Above is a general description about this program.      *
*  It includes the language and package we used            *
*  and program name.                                       *
*  It also has the log in name and time.                   *
************************************************************
NIL
()
 (
(DELETE.KB ((BEFORE (BEFORE-AI3-KB-DELETE SELF))
(BEFORE (BEFORE-KP-KB-DELETE SELF))))
(KBMETHODFILE ("MANAGEMENT"))
(KBSIZE (14))
(KEE.DEVELOPMENT.VERSION.NUMBER (0))
(KEE.MAJOR.VERSION.NUMBER (3))
(KEE.MINOR.VERSION.NUMBER (1))
(KEE.PATCH.VERSION.NUMBER (0))
```

```
************************************************************
*  This part includes program name and how many units in   *
*  this knowledge.base                                     *
*  We also can see its version.                            *
************************************************************

(ELECTRICAL.COMPONENTS
("hxl7866" "11-26-92 11:24:23" "hxl7866" "11-27-92
11:25:14")
((ENTITIES GENERICUNITS))
((CLASSES GENERICUNITS))
NIL
((INITIALIZE ((LAMBDA (SELF)  (REMOVE.ALL.VALUES SELF
'STATUS)))
METHOD (#[Unit: METHOD KEEDATATYPES]))
(PLACES.AFFECTED NIL NIL
(#[Unit: PLACES MANAGEMENT]) NIL ((CARDINALITY.MIN (1))))
(STATUS NIL NIL
((ONE.OF OK NOW.OK OVERLOADED BLOWN REPLACED
NEEDS.REPLACEMENT)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
)
())
************************************************************
*  Electrical.Components is a class-unit.                  *
*  There are 3 member slots inherited to its               *
*  subclasses:Fuses, and                                   *
```

```
*  Circuits.                                                    *
*  The 3 member slots are: Initialize                           *
*  ( a method defined in Manager                                *
*  Place.Affected (relate to Places ), and                      *
*  Status                                                       *
*****************************************************************

(CIRCUITS
("hxl7866" "11-26-92 11:27:56" "hxl7866" "11-27-92
11:34:43")
(ELECTRICAL.COMPONENTS)
((CLASSES GENERICUNITS))
NIL
((APPLIANCES NIL NIL (#[Unit: ELECTRICAL.APPLIANCES
MANAGEMENT]))
(CURRENT.FLOW NIL NIL (#[Unit: INTEGER KEEDATATYPES]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
(VOLTAGE NIL NIL (#[Unit: INTEGER KEEDATATYPES]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
(INITIALIZE ((BEFORE (PUT.VALUE SELF 'CURRENT.FLOW 0)
(PUT.VALUE SELF 'VOLTAGE 110))))
(RELEVANT.FUSE NIL NIL (#[Unit: FUSES MANAGEMENT]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
(STATUS NIL NIL ((ONE.OF OK NOW.OK OVERLOADED)))
(TOTAL.CURRENT NIL NIL ((ONE.OF TOO.HIGH NOW.NONE OK)) NIL
((CARDINALITY.MAX (1))))
)
```

```
())
************************************************************
*  Circuits is a class-unit.                               *
*  There are 7 member solts inherited to its subclasses:   *
*  Circuit1 and Circuit2.                                  *
*  The 7 member slots are Appliances (related to           *
*  Electrical.Appliances )                                 *
*  Current.Flow ( has only one integer value ),            *
*  Voltage ( has only one integer value )                  *
*  Initialize ( will give Current.Flow and Voltage         *
*  initialize values                                       *
*  0 and 110, respectively.                                *
*  Revelant.Fuses (relaed to Fuses ).                      *
*  Status, and Total.Current                               *
************************************************************

(ELECTRICAL.APPLIANCES
 ("hxl7866" "11-26-92 11:34:12" "hxl7866 ""11-27-92
11:32:12")
((ENTITIES GENERICUNITS))
((CLASSES GENERICUNITS))
NIL
((CURRENT.DRAWN NIL NIL (#[Unit: INTEGER KEEDATATYPES]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
(INITIALIZE ((LAMBDA (SELF) (REMOVE.ALL.VALUES SELF 'STATUS)
(REMOVE.ALL.VALUES SELF 'SWITCH)))
METHOD (#[Unit: METHOD KEEDATATYPES]))
```

```
(LOCATION NIL UNION NIL NIL ((CARDINALITY.MIN (1))
(CARDINALITY.MAX (1))))
(RELEVANT.CIRCUIT NIL NIL (#[Unit: CIRCUITS MANAGEMENT]) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
(STATUS NIL NIL ((ONE.OF OK OUT)) NIL ((CARDINALITY.MIN (1))
(CARDINALITY.MAX (1)))) (SWITCH NIL NIL ((ONE.OF OFF ON)))
)
())
************************************************************
*  Electrical.Appliances is a class-unit.                  *
*  There are 5 member slots inherited to its subclasses:   *
*  Bus1, and Bus2.                                         *
*  The 5 member slots are: Current.Drawn                   *
*  ( has an integer value ),                               *
*  Initialize ( a method ),Location ( has only one value ) *
*  Relevant.Circuit ( related to Circuits ), and Status    *
************************************************************

(FUSES
("hxl7866" "11-26-92 11:37:43" "hxl7866" "11-27-92
11:45:41")
(ELECTRICAL.COMPONENTS)
((CLASSES GENERICUNITS))
NIL
((CHECK.FUSE
((LAMBDA (SELF)
  (IF (EQUAL (GET.VALUE SELF 'STATUS) 'BLOWN)
```

```
      (PUT.VALUE (GET.VALUE SELF 'PLACES.AFFECTED) 'STATUS
'IN.THE.DARK)
      (REMOVE.ALL.VALUES (GET.VALUE SELF 'PLACES.AFFECTED)
'STATUS))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
(PLACES.AFFECTED NIL NIL NIL NIL ((CARDINALITY.MAX NIL)))
(STATUS NIL NIL ((ONE.OF OK BLOWN REPLACED
NEEDS.REPLACEMENT)))
)
())
************************************************************
*  Fuses is a class-unit.                                  *
*  There are 3 member solts inherited to its subclasses:   *
*   Place1.Fuse                                            *
*  and Place2.Fuse.                                        *
*  The 3 Member solts are Check.Fuse                       *
*  ( a method checks which place is                        *
*  in the dark and alert operator ),Place.Affected         *
*  ( no limit values),                                     *
*  and Status                                              *
************************************************************

(PLACES
 ("hxl7866" "11-26-92 11:48:21" "hxl7866" "11-27-92
11:49:16")
((ENTITIES GENERICUNITS))
((CLASSES GENERICUNITS))
```

```
NIL
((INITIALIZE ((LAMBDA (SELF) (REMOVE.ALL.VALUES SELF
'STATUS)))
 METHOD (#[Unit: METHOD KEEDATATYPES]))
(STATUS NIL NIL ((ONE.OF LIT IN.THE.DARK)) NIL
((CARDINALITY.MIN (1)) (CARDINALITY.MAX (1))))
)
())
************************************************************
*  Places is a class-unit.                                 *
*  There are 2 member slots inherited to Place1,           *
*  and Place2                                              *
*  They are Initialize ( a method ), and                   *
*  Status ( can has only one value ).                      *
************************************************************

(MANAGER
("hxl7866" "11-26-92 11:55:25" "hxl7866 ""11-27-92
11:54:42")
NIL
((ENTITIES GENERICUNITS))
NIL
()
((INITIALIZE.ELECTRICAL.SYSTEM
((LAMBDA (SELF)
  (DECLARE (IGNORE SELF))
```

```
(LOOP FOR OBJECT IN (UNIT.DESCENDANTS 'ELECTRICAL.COMPONENTS
'MEMBER)
  DO (UNITMSG OBJECT 'INITIALIZE))
(LOOP FOR OBJECT IN (UNIT.DESCENDANTS 'ELECTRICAL.APPLIANCES
'MEMBER)
  DO (UNITMSG OBJECT 'INITIALIZE))
(LOOP FOR OBJECT IN (UNIT.DESCENDANTS 'PLACES 'MEMBER)
  DO (UNITMSG OBJECT 'INITIALIZE))))
METHOD (#[Unit: METHOD KEEDATATYPES]))
))
************************************************************
*  Manager is a method class                               *
*  It can initialize all the member slots and              *
*  own slots in this 3 class-units.                        *
*  The three class-units are Electrical.Components,        *
*  Electrical.Appliances, and Places                       *
************************************************************

(PLACE1
  ("hxl7866" "11-27-92 11:59:15" "hxl7866" "11-27-92
12:02:12")
 NIL
 (PLACES)
NIL
()
())
************************************************************
```

```
*  Place1 is a subclass of Places                                *
*  It has no own slot                                            *
****************************************************************

(PLACE2
  ("hxl7866" "11-27-92 11:29:36" "hxl7866" "11-27-92
12:21:43")
 NIL
 (PLACES)
 NIL
()
())
****************************************************************
*  Place2 is a subclass of Places                                *
*  It has no own slot                                            *
****************************************************************

(PLACE1.FUSE
 ("hxl7866" "11-26-92 12:23:21""hxl7866""11-27-92 12:27:16")
 NIL
 (FUSES)
 NIL
 ()
 ((PLACES.AFFECTED (#[Unit: PLACE1 MANAGEMENT]))
  ))
****************************************************************
*  Place1.Fuse is a subclass of Fuses                            *
```

```
*  It has 1 own slot: Place1.                                    *
*****************************************************************

(PLACE2.FUSE
 ("hxl7866" "11-26-92 12:34:22""hxl7866""11-27-92 12:36:12")
 NIL
 (FUSES)
 NIL
 ()
 ((PLACES.AFFECTED (#[Unit: PLACE2 MANAGEMENT]))
))
*****************************************************************
*  Place2.Fuse is a subclass of Fuses                            *
*  It has 1 own slot: Place2.                                    *
*****************************************************************

(BUS1
("hxl7866""11-26-92 12:45:12""hxl7866""11-27-92 12:52:34")
NIL
(ELECTRICAL.APPLIANCES)
NIL
()
((CURRENT.DRAWN (5))
(LOCATION (#[Unit: PLACE1 MANAGEMENT]))
(RELEVANT.CIRCUIT (#[Unit: CIRCUIT1 MANAGEMENT]))
))
*****************************************************************
```

```
*  Bus1 is a subclass of Electrical.Appliances              *
*  It has 3 own slots                                       *
*  They are Current.Drawn ( defalut value is 5 ), Place1,   *
*  and Circuit1.                                            *
************************************************************

(BUS2
("hxl7866""11-26-92 13:23:11" "hxl7866""11-27-92 13:23:12")
NIL
(ELECTRICAL.APPLIANCES)
NIL
()
((CURRENT.DRAWN (7))
(LOCATION (#[Unit: PLACE2 MANAGEMENT]))
(RELEVANT.CIRCUIT (#[Unit: CIRCUIT2 MANAGEMENT]))
))
************************************************************
*  Bus2 is a subclass of Electrical.Appliances              *
*  It has 3 own slots                                       *
*  They are Current.Drawn ( defalut value is 7 ), Place2,   *
*  and Circuit2.                                            *
************************************************************

(CIRCUIT1
("hxl7866" "11-26-92 13:23:51" "hxl7866" "11-27-92
13:32:15")
NIL
```

```
(CIRCUITS)
NIL
()
((APPLIANCES (#[Unit: BUS1 MANAGEMENT]))
(CURRENT.FLOW (0))
(VOLTAGE (110))
(RELEVANT.FUSE (#[Unit: PLACE1.FUSE MANAGEMENT]))
(PLACES.AFFECTED (#[Unit: PLACE1 MANAGEMENT]))
))
*****************************************************************
*  Circuit1 is a subclass of circuits.                          *
*  Circuitd1 includes 5 own slots.                              *
*..The 5 own slots is Bus1, CURRENT.FLOW                         *
*  ( default value is 0),                                       *
*  Voltage ( default value is 110 ), Place1.Fuse, and           *
*  Place1.                                                      *
*****************************************************************
(CIRCUIT2
("hxl7866""11-27-92 13:45:23" "hxl7866" "11-27-92 13:48:18")
NIL
(CIRCUITS)
NIL
()
((APPLIANCES (#[Unit: BUS2 MANAGEMENT]))
(CURRENT.FLOW (0))
(VOLTAGE (110))
(RELEVANT.FUSE (#[Unit: PLACE2.FUSE MANAGEMENT]))
```

```
(PLACES.AFFECTED (#[Unit: PLACE2 MANAGEMENT]))
))
************************************************************
*  Circuit2 is a subclass of circuits.                     *
*  Circuitd2 includes 5 own slots.                         *
*..The 5 own slots is Bus2, CURRENT.FLOW                    *
*  ( default value is 0).                                  *
*  Voltage ( default value is 110 ), Place2.Fuse           *
*  and Place2                                              *
************************************************************

KBEND
************************************************************
*  Program end.
************************************************************
```

REFERENCES

1. Amelink, H., A. M. FORTE, and R. P. Guberman. August, 1986. "Dispatcher Alarm and Message Processing."*IEEE Transacions on Power Systems.* Vol.PWRS-1, No.3. 188-194.

2. Berk, A. A. 1985. *Lisp, The Language of Artificial Intelligence.* New York: Van Nostrand.

3. Hunt, V. D. 1986. *Artificial Intelligence and Expert Systems Sourcebook.* New York: Chapman and Hall.

4. Patterson, D. W. 1990. *Introduction to Artificial Intelligence and Expert System.* New Jersey: Princeton-Hall.

5. Sakguch,T., and K. Matsumoto. Feb. 1983. "Development of a Knowledge Based System for Power System Restoration." *IEEE Transactions on Power Apparatus and System.* Vol.PAS-102, No.2. 320-329.

6. Silverman, S., and A. Shoop. 1991. "A Real-Time Expert System in the Area of Energy Management." *HANDBOOK OF EXPERT SYSTEM IN MANUFACTURING.* Maus,R.,and J. Keyes. N.Y. McGraw-Hill.

7. Talukdar, S.N., E. Cardozo, and T. Perry. August 1986. "The Operator's Assistant-an intelligent, Expandable Program for Power System Trouble Analysis." *IEEE Transactions on Power Systems.* Vol.PWRS-1, No.3. 182-187.

8. Tesch, D. B.,D. C. Yu, L. M. Fu, and K. Vairvan. February 1990. "A Knowledge-Based Alarm Processor for an Energy Management System." *IEEE Transactions on Power Systems.* Vol. 5, No.1. 268-275.

9. Wollenberg, B. F. May 1986. "Feasibility Study for an Energy Management System Intelligent Alarm Processor." *IEEE Transactions on Power Systems.* Vol. PWRS-1, No.2. 241-247.

10. Zaborszky, J.,and J. W. Rittenhouse. 1969. *Electric Power Transmission, The Power System in the Steady State.* Troy, New York: The Rensselaer Bookstore.

11. 1989. "KEEtutor." IntelliCorp.