Object-Oriented Methods Drive Protective Relay System

Qing Liu,* Shijie Cheng*

China is a developing country, and shortages in electrical power supply exist in most provincial power systems. Stability margins are always very limited, which requires operators to run the system carefully. With each change in either the power system configuration or operating conditions, a series of actions must be taken to guarantee the safety of the power system. The actions adopted for the protective relays include arranging protective devices appropriately and setting relays at their appropriate values for all elements involved. In addition, as power systems are developing very fast in China, the number of skilled operators available is far less than required. Expert systems are urgently needed.

This article describes the use of object-oriented (OO) methods to develop expert systems and, in particular, features the object-oriented expert system for the power system protective relay operation and management (OOPRES).

Expert System Development

Expert systems have become a major field of interest in power system applications, and the first successful attempt was to apply an expert system to power system restoration after an emergency. Over the years, expert systems have been developed for fault diagnosis, planning, alarm processing, etc. Most of these expert systems were developed using conventional structured programming techniques. This method has proven effective for the development of simple or single functional expert systems.

Due to the large size and complexity of power systems, however, with more functions being embedded in expert systems, expert systems have developed into large and complex software programs. A particular problem in developing and maintaining such large and complex software is that extremely long times may be needed. Since the software codes developed using conventional methods are inflexible, maintaining and updating them also presented difficulties. An additional problem is that it is difficult for knowledge engineers to consistently acquire enough knowledge from the domain engineers in different aspects of the application field.

To overcome these difficulties, object-oriented (OO) methods have been used to develop a power system protective relay operation and management expert system. It has been developed in a PC/Windows environment using the Borland C++ for Windows programming language. It can help power system operators operate and manage protective relays correctly. This expert system is actually a subset of a multifunctional expert system. It has been successfully used in a provincial power system dispatch center in China to help system operators operate and manage the power system.

Object-Oriented Methods/Programming

The OO methods are relatively new and have attracted more and more interest in recent years. OO methods recognize and analyze the real world in a way that is very close to that of human beings. The real world is composed of different objects and associated interconnections. OO methods enable the development of software in a flexible and natural way. These features enable the software to be developed in a highly modularized and easy-to-maintain manner.

Application of these methods in power systems is a relatively new discipline. Research reported so far is mainly concentrated on the areas of databases, graphi-
Class Hierarchy Definition of OOPRES

In object-oriented programming, every type of individual in the real world is represented by an object. Representing the real world in this way enables knowledge engineers to directly map objects from the problem domain to the expert system. Consequently, it is easy for the expert system to pick the objects out of a simulated world if OO methods are used. This is quite difficult for conventional expert systems, because conventional expert systems are developed entirely on data structure and procedures.

Using OO terminology, objects are instances of a class declaration that consists of both data and operations. A power system is usually an extremely large system that contains a large number of elements, and its main feature is the hierarchy in its structure. Consequently, it is better to describe power systems by class hierarchies. Different class hierarchies for power systems have been proposed:

- **Prototype** class hierarchy is used for an object-oriented graphical user interface for the power system.
- **Component** class hierarchy is used for object-oriented power system modeling software.

Protective relays are mainly used to protect different types of generation and transmission elements of the power system from being damaged during abnormal situations. The elements involved in the power system include generators, transformers, buses, and lines. They are situated in different power plants and/or substations. Each of these elements has its own protective devices. Operation of the protective devices depends mainly on system operating conditions and system configurations.

Any change, either in the operating condition or in the system configuration, must be accompanied by a series of changes in arranging the protective devices and their setting values. This will be performed by OOPRES and is its problem domain.

Based on the study of problems that exist in the problem domain, a class hierarchy structure, which is similar to the structure of a physical power system, is proposed. The graphical representation of the class hierarchy is shown in Figure 1. Elements situated in the bottom level of the class hierarchy are the breakers on which the protective relays act. Information about the protective devices and the coordinating methods for each type of breaker is specified in the classes situated at this level.

A table can be developed to represent the class hierarchy in C++ syntax. It should be pointed out that two abstract classes are defined:

- **Calculation** class defines the calculation of the setting values of the protective relays.
- **Special** class is defined if the knowledge of operation for an object is quite different from those used for other objects of the same type.

The database that is developed shows the plant as the root class. The transformer is a subclass of plant, the transformer-breaker is a subclass of transformer and a sub-subclass of plant. The inheritance mechanism, which the OO method provides, enables a subclass to inherit the attributes of a class if the relationship of inheritance is specified by the subclass. Some attributes of the class, such as the plant's name, the name code of the plant, etc., are automatically inherited by all the subclasses and sub-subclasses. Data and methods defined in a class are available in all of its subclasses. This is a very useful feature for the development of expert systems.

Knowledge Acquisition and Representation

Knowledge acquisition plays a very important role in developing expert systems. However, it is always a bottleneck in the design of a conventional expert system, because the knowledge engineers and domain engineers usually deal with the same objects from different points of view and by different ways. Domain engineers are interested in the property of every individual object, and they are used to considering objects individually. Knowledge engineers mainly deal with the database, knowledge base, and inference engine, and they try to find a general way to
treat all kinds of objects. The difference in the approaches causes difficulties, but the OO method provides a way to overcome this difficulty. It specifies the objects with classes. Different types of objects are defined with different classes. The class itself defines all information about the objects, such as the data, knowledge, and method of inference. As this type of knowledge representation enables the knowledge engineer to recognize the objects in a way that is similar to the domain engineer, it helps in overcoming the difficulty of knowledge acquisition that exists in designing conventional expert systems.

An example of the line class and line-breaker class is represented in C++ syntax as follows:

```cpp
Class Line : Public Plant {
    Protected :
    Attributes BI, ... Bk;
    Public:
    Line(); // constructor
    -Line(); // destructor
    Virtual void getFrom(); // get an object's attributes from database
    Virtual void Update(); // update an object's attributes in database
    Virtual void Delete(); // delete an object's attributes in database
}
Class Line-Breaker : Public Line { // inherits class Line
    Attributes Al, ... Ai; // static attributes
    Facts F1, ... Fj; // the facts about an object
    Public:
    Line-Breaker();
    ~Line-Breaker();
    D_read(); // read data from database
    K_Display(); // display rules
    K_Edit(); // edit rules
    K_Insert(); // insert a rule into knowledge base
    K_Delete(); // delete a rule in knowledge base
    Inference_forward(); // inference forward
}
```

It can be seen from this representation that the line-breaker class is a subclass of the line class. The attributes of the line class include the code of the line's name, the voltage level of the line, and the maximum current of the line. Three types of data are defined for the line-breaker class: attributes, facts, and rules. The attributes of the line-breaker class specify the static information, such as name code of the line-breaker, which is used to specify the location of the breaker in the power system, and the mode of the connection, which specifies the way that this object connects with the other objects. The facts of the line-breaker class contain the information needed for reasoning, such as the type of protective device or the status of protective relay. The rules of this class include plenty of knowledge acquired from the protective engineers, such as the expertise of the coordination of the protective devices and the rules used for the operation of the protective devices. In the present application, this kind of knowledge is represented by a set of if-then production rules.

The following is an example of the production rules used:

If line-breaker named xxx is put into repair and a bypass breaker named yyy is used to replace it, then the phase comparison carrier relay on the other side of the line should be switched off, and the carrier relay used for line-breaker xxx should be switched to the bypass breaker yyy, and the setting values for the protective relay calculated.

For those objects in the real world, if the knowledge of operation for the object is quite different from those used for the other objects of the same type, a special class is defined. In a special class, knowledge is also represented by a set of production rules. The rules are put in the set one by one. An example of the special class is represented in C++ syntax as follows:

```cpp
Class Special : Public Plant {
    Protected :
    Rules BI, ... Bk;
    Public:
    Special(); // constructor
    ~Special(); // destructor
    K_Display(); // display rules
    K_Edit(); // edit rules
    K_Insert(); // insert a rule into knowledge base
    K_Delete(); // delete a rule in knowledge base
    Match(); // matching
}
```

Following are two examples of production rules used in the special class:

If transformer #1 in the power plant Linzhushang needs to be repaired, then ground the neutral of at least one transformer connected with 110 kV bus.

If breaker named K610 in substation Baonanshan needed to be repaired, then ground the neutrals of both sides of transformer #2 and unground the neutral of transformer #1 of this substation.

Knowledge Management
With the growth of power systems, some new elements
may be put into operation in the system and some old elements may be replaced by new ones. Corresponding changes must be made for the protective relay devices. An expert system used for protective relay operation and management should be able to cope with these changes.

In conventional expert systems (Figure 2), a centralized database (DB) and a centralized knowledge base (KB) are used to store all information, and a single change in a physical system will cause a number of modifications in both centralized bases. Sometimes modifications may be needed at different places in the centralized bases. In this situation, avoiding the conflict and keeping the consistency of the stored knowledge presents difficulties.

OO methods provide an effective way to overcome these difficulties. In an OO-based expert system, the KB and knowledge-base management system (KBMS) are arranged in a decentralized manner. For each type of object, there will be one KB and one corresponding KBMS. Also, both knowledge and inference engine are encapsulated in one object. As a result, changes in the physical system only result in changes within the corresponding object. Obviously, conflicts of the knowledge stored in the KB can be easily avoided, and consistency of the knowledge in the KB can be easily guaranteed.

![Image](image.png)

**Figure 4. OOPRES reasoning results**

**Interfaces**

Three kinds of interfaces are provided to link the expert system with the outside world. They are the interfaces for power system operators, the SCADA system, and other expert systems.

In order to appropriately use the expert system, windows and menus should be provided at each processing stage of the expert system. In OO programming, as the windows and the menus are all defined as objects, their execution can be requested by message passing. This provides a flexible way to activate appropriate windows or menus by different kinds of objects whenever the windows or menus are required. Also, a starting object is developed to accept the original inputs of the operators to start the expert system. By doing this, a very flexible and friendly human-machine interface is developed.

As the other functional expert systems are all programmed with the OO method, the interface between OOPRES and the other functional expert systems is realized easily by message passing. Every object has its own database and associated management system. The interface between OOPRES and the SCADA system is located in the individual objects.

**Inference Engine**

The inference engine is the core of the expert system. Three kinds of inference mechanisms can be used: forward, backward, and hybrid.

*Forward* inference can be characterized as an event-driven inference. *Backward* inference is sometimes referred to as the goal-driven inference. In OOPRES development, in general, a forward inference is used. However, the OO method provides an ability of distributed reasoning. It allows different kinds of objects to use different kinds of inference methods. The inference engine used by each individual object can be specified by the object itself.

The reasoning between the objects is implemented by message passing. Message passing is one of the most important features of the OO method. It can be used to describe all kinds of operations requested between the objects. A message to be passed has two or more arguments. The first one is usually the name of the object to which the message is sent. The second one specifies the operation requested. Arguments may be required by the operation described by the message passing. They are additional parameters of the operation. For example, sending a message to an object plant to inquire its information with the parameter Aug1 can be represented as follows:

```
Object_plant.Read( Aug1 );
```

The overall reasoning of OOPRES is summarized as follows and represented in Figure 3. Execution begins with a starting object. This object receives the request from the operators. The request is usually the code of the element. After receiving the request, the object passes a message to the special object trying to find out if this element is contained in the knowledge set of the special object. If it is, corresponding actions will be taken, and the results will be obtained. Otherwise, the special object will pass a message to the corresponding object. If, in this case, calculation of the setting values of the protective relays is needed during the reasoning, the object
calculation will be requested by message passing. This procedure continues until the final results are obtained. It can be seen that inference in this way can considerably reduce the size of the search space. The efficiency of inference is, therefore, greatly increased.

Application Examples

OOPRES has been successfully used in a dispatch center of a provincial power system in China. The following example shows how OOPRES works in practice. A breaker of a transmission line (coded as K602) in a substation called Balinbin is going to be placed under repair. Changes to the system configuration and load rearrangement will be done by the other subordinate expert systems of the multifunctional expert system. Now, OOPRES is used to make the necessary arrangement of the protective relay devices used.

Upon starting the expert system, the object start is activated. A window such as the one seen in Figure 4 comes up on the monitor. The power system operator inputs the information by selecting the items of the menu that the window provides. For this particular example, the information contains:

- Substation's name, Balinbin
- Name code of the breaker to be repaired, K602
- Operation mode, breaker K602 switched off, bypass breaker K618 is used to replace breaker K602.

Actually, object start can receive this information by message passing from the objects of the other subordinate expert systems if the multifunctional expert system is used. Then the name code of B03 for the substation Balinbin is obtained from the database of the object substation by message passing. The object start of the OOPRES passes a message B03_K602 to the object special, trying to match this message in the object special. As B03_K602 is not a special object, matching fails. This message is then sent to the object line-breaker by the object special. The attributes and facts associated with B03_K602 are obtained from the database of the object line-breaker. It is from the database that:

- The line in which breaker B03_K602 is used is connected with a power plant.
- The name code of the plant is F03.
- The name code of breaker used at the other side of the line is K606.
- The ratio of the current transformer in B03_K602 is 1200/5.
- The ratio of the current transformer ratio in the bypass breaker is 600/5.

Using all the information, the inference engine of the object line-breaker executes reasoning. As the bypass breaker is going to be used, the setting values of its protective relay must be recalculated.

Some of the setting values calculated are shown in Table 1. The results of the protective relay arrangement are obtained:

- Breaker K618 is used to replace breaker K602. The setting values of the protective relays will be given in a separate form. The phase comparison carrier relay for breaker F03_K606 should be switched off.
- The carrier relay for breaker K602 should be switched to breaker K618.
- If the carrier relay is switched off, the reclosing time should be changed according to the form.
- The system configuration will be changed to a single bus operation scheme of a provincial power system in China.

For Further Reading


Biographies

Qing Liu received her BS and MS degrees in computer science from Wuhan University of Technology, Wuhan, China, and PhD in electrical engineering from Huazhong University of Science and Technology (HUST), Wuhan. Her research interest is the application of artificial intelligence in power systems.

ShiJie Cheng graduated from the Xi'an Jiaotong University, Xi'an, China, received an MS degree from HUST, and PhD from the University of Calgary, Calgary, Canada. Both degrees are in the electrical engineering. He is a full professor at HUST. His research interests are power system control, power system stability analysis, and application of artificial intelligence in power systems. He is an IEEE senior member.