Modern power systems have grown both in size and complexity. Various constraints, such as security, economy, and environmental regulations, are forcing power systems to operate closer to their design capabilities and security margins. Utilities are required to operate as optimally as is feasible. Energy control centers play a vital role in optimal and secure operation of a power system. Applications in an energy control center require monitoring and control of the large interconnected power system. Energy management system (EMS) decision-support tools, such as optimal power flow (OPF), base their decisions on the real-time model of the power system network along with the real-time data. Usually, real-time modeling of a power system follows a four-step procedure that involves the following:

- Network configuration analysis
- Observability analysis
- State estimation and bad data processing
- Network application functions.

A network topology processor (NTP) processes real-time circuit-breaker (CB) status to obtain current electrical network topology. This article builds on information presented by the authors in the October 2000 issue of IEEE Computer Applications in Power and focuses on an object-oriented design for an NTP. The design and application is expected to be part of an overall object-oriented EMS.

**Network Topology**

Network topology processors perform a time-critical EMS operation but also add to the complexity of the EMS software system, which can be handled elegantly in the object-oriented paradigm. Object-oriented systems stress reuse of software components. Typically, with changes in technology, applications evolve (Figure 1); however, physical laws governing them remain the same. Thus, an object-oriented approach can be used to develop a toolkit for power system applications.

Figure 2 shows a typical power system network (PSN) in which substations are connected through circuits (lines, transformers, cables, etc.). In each of the substations, buses are interconnected through CBs (Figure 3). Various substation configurations exist, such as ring main, breaker and a half; with varying costs, flexibility of operation, and reliability. In real-time, breakers may operate at any time, changing the system configuration dynamically. The substation status must be analyzed to update connectivity; as with the operation of a CB, any of

**Figure 1. Evolution of applications, with dates reflecting when applications were widely accepted as technologically viable**

<table>
<thead>
<tr>
<th>Application</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Flow Analysis</td>
<td>(1950 onwards)</td>
</tr>
<tr>
<td>Short Circuit Analysis</td>
<td></td>
</tr>
<tr>
<td>Transient Stability</td>
<td>(1960 onwards)</td>
</tr>
<tr>
<td>EMTP</td>
<td></td>
</tr>
<tr>
<td>EMS Applications</td>
<td>(1970 onwards)</td>
</tr>
<tr>
<td>Optimal Power Flow</td>
<td>(1980 onwards)</td>
</tr>
<tr>
<td>Voltage Stability</td>
<td>(1985 onwards)</td>
</tr>
<tr>
<td>Deregulation</td>
<td>(1995 onwards)</td>
</tr>
</tbody>
</table>
the following effects can be observed:
- Circuits may or may not be disconnected
- Substations may split, possibly forming new nodes
- An interconnected system may split into several separated areas.

Let us define a physical node as an interconnection point of two CBs or end points of a CB. Figure 3 shows A, B, C, D, E, and F as the physical nodes, with CB 2 connected between nodes A and B. A closed CB between two physical nodes establishes zero impedance path between them, creating one electrical node. For example, in Figure 3, if all the CBs are closed, we get only one electrical node. Opening of CB 2 creates an open circuit between nodes A and B. However, with other CBs remaining closed, an alternate path of closed CBs, A-F-E-D-C-B still exists between them, establishing a condition of zero impedance path. On the other hand, opening of CBs 2 and 5 leads to two distinct electrical nodes. Thus, if a graph of the substation is created, with edges depicting the closed CBs, then each component of the graph forms an electrical node. Electrical nodes are, thus, a function of CB status and substation topology. Given the real-time status of the CB, an NTP performs the following tasks:
- Based on breaker status within a substation, groups physical buses at the substation into electrical buses
- Builds connectivity from electrical buses to the network branches (e.g., transmission lines, transformers, etc.)
- Analyzes electrical network connectivity for islands.

Hierarchical Approach
Software complexity can be elegantly tackled in an object-oriented paradigm. Some of the important attributes of a complex system are:
- Frequently, complexity takes the form of a hierarchy, whereby a complex system is composed of interrelated subsystems that have, in turn, their own subsystems, and so on, until the lowest level of elementary components is reached.
- Hierarchic systems are usually composed of only a few different kinds of subsystems in various combinations and arrangements. In other words, a complex system has common patterns.

The network shown in Figure 2 can be viewed as a two-level hierarchical structure. At the top-most level, the substations are connected through power apparatus. If we zoom into a substation, we discover yet another network consisting of buses interconnected through CBs (Figure 3). The hierarchical structure, in turn, suggests a corresponding solution approach. The first task of an NTP is to analyze various substation configurations, i.e., create a mapping of physical nodes of the substation onto unique electrical nodes. For this, an electrical graph for each substation is created. This analysis is referred to as substation-level processing (SLP).

Once SLP is complete, network-level processing (NLP) can be initiated. A network is composed of substations connected through circuit elements. A circuit (e.g., line, transformer, etc.) connects physical nodes of one or more substations. Knowing the mapping to electrical nodes in each substation, a graph of the electrical network can be constructed. In this solution methodology, one notices a recurring theme or pattern, emphasizing reuse of the graph-related software component. This graph is further analyzed during NLP to find the number of islands in the PSN. Figure 4 illustrates these two important scenarios (SLP and NLP), which can be used to describe NTP.

Identification and Design of Classes
The vocabulary of problem domain contains nouns like CB and substation. This gives us a cue to model CB and substation. A CB associates two physical nodes. A Boolean variable is required to represent its on/off sta-
tus. Structure *spmat* has been designed to model static linked list data structure, used in storage of sparse matrices. When instantiated by integer data type, *spmat* can be reused to store CB information, including status.

A substation is primarily a collection of CBs and is modeled by an array of *spmat*. A natural convention to label a substation and its physical nodes is by character strings. Since graph-theoretic operations are facilitated by integer-node representation, a mapping from character strings onto integers is required. A separate support class that implements associative array caters this functionality. While class *sub_stn* models an individual substation, a class *All_Substn* is designed as a collection of all substations. An instance of class *asso_array* in the instance of class *All_Substn* is delegated the work of assigning new numbers to the electrical nodes created in the SLP of all substations. Moreover, so as to process the topology of a network, we also need to model the system network. A system network is composed of power apparatus such as lines, transformers, etc. As such, class network is shown as a composition of these apparatus classes (Figure 5). NTP functionality requires creation of a graph by interconnecting substations through circuits. The hierarchical structure discussed is best captured by inheritance. Hence, class *NTP* is logically derived from class *All_Substn*.

The other set of classes, arising from the description of solution, involves terminology like nodes, path, and graph, making NTP a graph-theoretic application. A class *graph* is defined to handle graph-theoretic applications. Structure *spmat* comes handy again to store the information about the nodes of a graph. In other words, a graph of a system can be composed of an array of *spmat*. The storage scheme of symmetric
sparse matrix is reused in class \textit{graph}. As a matter of implementation detail, we have derived class \textit{graph} from class \textit{symmetric-sparse-matrix}. As such, class \textit{graph} can efficiently handle large sparse problems. Knowing the structure of a graph, adjacency of a node can be derived and stored as a set. Methods of class \textit{graph} can be efficiently implemented using a set-theoretic approach, making class \textit{set} an indispensable partner of class \textit{graph}. With the help of the unified modeling language (UML), all these classes with their internal relations have been shown in the class diagram (Figure 5).

\section*{Substation-Level Processing}

Class \textit{sub_stn} has been defined to model the substation configuration. Every substation has a unique name. A substation in itself is an electrical network consisting of buses interconnected through CBs. Specific topologies (like one-and-half breaker, ring bus) can be easily derived (as subclasses) from the class \textit{sub_stn}. Circuit elements (like lines, transformers) terminate at the physical buses in a substation. They are mapped onto integers. The CB data is then stored in structure \textit{spmat}, with field \textit{row} corresponding to the \textit{from_node} and field \textit{col} corresponding to \textit{to_node} of the CB. Field \textit{val} can be used to store the status of CB, with 1 representing a closed CB and 0 as an open CB. Integer -1 is reserved for a phantom breaker. The electrical connectivity within a substation, corresponding to closed CBs, is replicated in an array of \textit{spmat}. An instance of class \textit{graph} is now created. Method \textit{find_component()} of class \textit{graph} is used by method \textit{SLP()} of class \textit{sub_stn} to compute components of the graph. In this case, a component represents a set of connected physical nodes through zero impedance branches and hence alias an electrical node. Thus, class \textit{graph} and its method \textit{find_component()} play a crucial role in \textit{SLP()}. The SLP can be illustrated on a part of the IEEE 14-bus network shown in Figure 6. Table 1 provides the CB connectivity for substations \textit{sub_7} and \textit{sub_9}. Let the status of these CBs be as shown in the table. Consider the substation \textit{sub_9} that consists of five buses. Being the first character string of this substation, \textit{bus_29} is mapped onto integer 0; with \textit{bus_16} as the next mapped onto integer 1 and so on, by the associative array of the substation. A similar exercise is also performed on substation \textit{sub_7}. It can be seen that, as the CBs between buses \textit{bus_29} and \textit{bus_14} and between \textit{bus_29} and \textit{bus_9} are open, substation \textit{sub_9} gets decomposed into three electrical nodes.

Once all substations are processed, the electrical nodes of each substation aliasing the physical nodes are known. Each electrical node of a substation is assigned a new number (name), by a unique character string, using associative array. Concatenating the component number of the substation to which that particular physical node belongs with the substation name creates this new string. As shown in Table 2, a lookup table can be created to store many-to-one mapping.

\section*{Network-Level Processing}

NLP is carried out by methods of class \textit{NTP} (refer Figure 5). The role of class \textit{NTP} is to connect the various electrical nodes of all substations by circuit elements, thereby developing a graph of the system (single-line diagram). First, an offline database containing circuit details by the substation name and the physical node name at the substation is read. Then, terminals of these circuit elements are redefined in terms of electrical node names, generat-

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Substation & From Node & To Node & CB Status* \\
\hline
\textit{sub_9} & \textit{bus_29} & \textit{bus_16} & 1 \\
& \textit{bus_29} & \textit{bus_17} & 1 \\
& \textit{bus_29} & \textit{bus_14} & 0 \\
& \textit{bus_29} & \textit{bus_9} & 0 \\
\hline
\textit{sub_7} & \textit{bus_27} & \textit{bus_15} & 1 \\
& \textit{bus_27} & \textit{bus_14} & 0 \\
& \textit{bus_27} & \textit{bus_8} & 1 \\
\hline
\end{tabular}
\caption{CB connectivity for Figure 6}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{IEEE_14-bus_system}
\caption{IEEE 14-bus system}
\end{figure}
ed during SLP. Inheritance of class \textit{NTP} from class \textit{All\_Substn} permits visibility of this information to class \textit{NTP}. The database of a transmission line \textit{TL} originating from \texttt{bus\_14} of substation \texttt{sub\_9} and terminating at \texttt{bus\_14} of substation \texttt{sub\_7} reads as follows:

\texttt{TL sub\_9 bus\_14 sub\_7 bus\_14}.

The new names are searched in the lookup table, and the transmission line now gets connectivity as follows:

\texttt{TL sub\_9 1 sub\_7 bus\_14}.

All network apparatus connectivity is translated in the same manner. In NLP, we are interested in identifying islands in the system. The process is similar to SLP. An instance of class \textit{graph} corresponding to the new apparatus connectivity is created. Now, a component of the graph \textit{alias} islands in the electrical network. It can be seen that the design pattern of SLP, i.e., creation of graph and finding components, is reused in NLP.

\textbf{Alternative Scheme}

So far, we have outlined one possible object-oriented design for an NTP. Alternative designs and implementations within the same framework with varying trade-offs are possible. For example, class \textit{sub\_stn} can be equipped with a method that analyzes all possible CB configurations. Though the procedure is enumerative in nature, it will not pose computational hazards, as the number of CBs in a substation are finite. The result of the analysis can be stored in a lookup table (database). This work has to be done only once during the initialization process. Later, during substation analysis, the effort required is just to search the lookup table for the required CB status. The tradeoff in such an implementation would be additional memory requirements, albeit negligible computations during the substation analysis phase.

Both the schemes (the one that is implemented in this work as well as the alternative scheme) are suitable for distributed processing. For example, SLP can be completed locally at substation host computers. It will transmit the mapping of physical nodes to the energy control center (Figure 4). The burden at the energy control center would then reduce to only NLP. Considering that utilities now have access to fiberoptic communication, such a scheme may be feasible.

Thus, from the above design, it can be seen that class \textit{sub\_stn} can be developed to model any arbitrary substation layout. The proposed design enables development of simple and elegant NTP schemes that can be implemented efficiently. The scheme is amenable to distributed computing as well. This approach has no inherent drawback in design to limit its computational speed. Class \textit{graph} plays a strategic role in the proposed design. It has been reused in other power system applications, such as relay coordination and observability analysis.

\section*{For Further Reading}


\section*{Biographies}

Shubha Pandit is an assistant professor at the S.P. College of Engineering, Mumbai, India. She is presently working toward a Ph.D. at Indian Institute of Technology at Bombay (IIT Bombay), Mumbai, India. Her research interests include object-oriented programming, software engineering, sparse matrix computations, and power system analysis.

S. A. Soman is an assistant professor at IIT Bombay, India. He received his Ph.D. in 1996. His research interests include power system analysis, object-oriented programming, and computer-aided protection and automation of power system. He may be reached by e-mail, soman@ee.iitb.ernet.in.

S. A. Khaparde is a professor at IIT Bombay, India. He received his Ph.D. in 1978. He is a senior member of IEEE. His research interests include power system analysis, neural networks, and deregulation applications to power system.