An integration management template (12) manages, monitors, and controls a complex physical system and includes a centralized database model (102) comprising a plurality of parameters (304) for modeling data and behaviors of the complex physical system. A graphical user interface (120) communicates between a user and the centralized database model (102). A first interface mechanism (14) receives data from external sources (18) and relates data from external sources (18) to the centralized database model (102). A second interface mechanism (24) interfaces a plurality of internal application programs, said internal application programs comprising application programs having varied data structures, operational characteristics, and fundamental assumptions.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Albania</td>
<td>ES</td>
<td>Spain</td>
<td>LS</td>
<td>Lesotho</td>
</tr>
<tr>
<td>AM</td>
<td>Armenia</td>
<td>FI</td>
<td>Finland</td>
<td>LT</td>
<td>Lithuania</td>
</tr>
<tr>
<td>AT</td>
<td>Austria</td>
<td>FR</td>
<td>France</td>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
<td>GA</td>
<td>Gabon</td>
<td>LV</td>
<td>Latvia</td>
</tr>
<tr>
<td>AZ</td>
<td>Azerbaijan</td>
<td>GB</td>
<td>United Kingdom</td>
<td>MC</td>
<td>Monaco</td>
</tr>
<tr>
<td>BA</td>
<td>Bosnia and Herzegovina</td>
<td>GE</td>
<td>Georgia</td>
<td>MD</td>
<td>Republic of Moldova</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
<td>GH</td>
<td>Ghana</td>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>GN</td>
<td>Guinea</td>
<td>MK</td>
<td>The former Yugoslav</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
<td>GR</td>
<td>Greece</td>
<td>MI</td>
<td>Republic of Macedonia</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
<td>HU</td>
<td>Hungary</td>
<td>ML</td>
<td>Mali</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
<td>IE</td>
<td>Ireland</td>
<td>MN</td>
<td>Mongolia</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>IL</td>
<td>Israel</td>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>BY</td>
<td>Belarus</td>
<td>IS</td>
<td>Iceland</td>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
<td>IT</td>
<td>Italy</td>
<td>MX</td>
<td>Mexico</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>JP</td>
<td>Japan</td>
<td>NE</td>
<td>Niger</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td>KE</td>
<td>Kenya</td>
<td>NL</td>
<td>Netherlands</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>KG</td>
<td>Kyrgyzstan</td>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>CI</td>
<td>Côte d'Ivoire</td>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
<td>NZ</td>
<td>New Zealand</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
<td>KR</td>
<td>Republic of Korea</td>
<td>PL</td>
<td>Poland</td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
<td>KZ</td>
<td>Kazakhstan</td>
<td>PT</td>
<td>Portugal</td>
</tr>
<tr>
<td>CU</td>
<td>Cuba</td>
<td>LC</td>
<td>Saint Lucia</td>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>CZ</td>
<td>Czech Republic</td>
<td>LI</td>
<td>Liechtenstein</td>
<td>RU</td>
<td>Russian Federation</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
<td>LK</td>
<td>Sri Lanka</td>
<td>SD</td>
<td>Sudan</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>LR</td>
<td>Liberia</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>EE</td>
<td>Estonia</td>
<td></td>
<td></td>
<td>SG</td>
<td>Singapore</td>
</tr>
<tr>
<td>SI</td>
<td>Slovenia</td>
<td>SK</td>
<td>Slovakia</td>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>SZ</td>
<td>Swaziland</td>
<td>TD</td>
<td>Chad</td>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>TJ</td>
<td>Tajikistan</td>
<td>TM</td>
<td>Turkmenistan</td>
<td>TR</td>
<td>Turkey</td>
</tr>
<tr>
<td>TT</td>
<td>Trinidad and Tobago</td>
<td>UA</td>
<td>Ukraine</td>
<td>UG</td>
<td>Uganda</td>
</tr>
<tr>
<td>US</td>
<td>United States of America</td>
<td>UK</td>
<td>United Kingdom</td>
<td>UZ</td>
<td>Uzbekistan</td>
</tr>
<tr>
<td>VN</td>
<td>Viet Nam</td>
<td>YU</td>
<td>Yugoslavia</td>
<td>ZW</td>
<td>Zimbabwe</td>
</tr>
</tbody>
</table>
INTEGRATION MANAGEMENT TEMPLATE METHOD AND SYSTEM

TECHNICAL FIELD OF THE INVENTION

The present invention relates to object-oriented computer languages and their use in modeling physical and other systems and, more particularly, to a method and system for use in a computer for modeling and monitoring physical systems, and even more particularly, to a method and system for providing an integrated infrastructure and tools for modeling and monitoring fundamental physical system data and behaviors that include an external data interface, interfaces to existing application programs, as well as integration of applications via object behaviors.
BACKGROUND OF THE INVENTION

In recent years, the art of computers and software programming has evolved from the detailed line-by-line writing of source code instructions to widespread use of a form of programming called "object-oriented programming." Object-oriented programming is preferred over traditional methods for developing software because it facilitates designs in a tangible or physical domain. With object-oriented programming, a program is written by designing each object separately and, once the individual objects are implemented, they are connected in a coherent fashion resulting in a modular system. The software objects are also extendable and portable. This means that they can be easily reused in other designs. Most object-oriented development is performed in the C++ programming language. Object-oriented programming objects are software entities comprising data structures and operations on the object's data. Together, these elements enable objects to model virtually any physical or real-world entity in terms of its characteristics as represented by its data elements, while its behavior may be represented by related data manipulation functions. Objects, therefore, can model concrete things such as people and computers. They can also model abstract concepts like numbers or geometrical concepts.

Planning, operations, and marketing functions of many companies or organizations have a need to integrate
planning, operations, and marketing centers as highly flexible and heterogeneous profit centers providing coordinated product and service delivery control. This need relates to information resources that assist in planning, operations and marketing.

Companies and organizations also need to address the common situation of having numerous, different systems using different business representations. These numerous, different systems often do not communicate with each other because they were built at different times using different software programming technologies and assumptions. These systems provide a variety, but not always a compatible selection, of ways to develop software, to develop additional programs and to meet additional business needs. For example, in a utility control center a great deal of data comes in from the field through other types of systems such as energy management systems. Also, within an organization there may be numerous smaller systems that support decision-making. All of these smaller systems, unfortunately, operate in isolation from other similar or possibly related smaller systems within the organization. These smaller systems employ their own application development approach for building software programs, thereby making applications development and support extremely difficult.

Another limitation of the numerous types of application programs that may exist in a company or
organization are the different functions and operations that must take place in order to make the various application programs operate and integrate with one another. The problems relate to there being different languages, different formats and different degrees of complexity within the various systems. On the other hand, eliminating all of these systems and replacing them with a new system is cost and time prohibitive.

Another limitation that existing systems experience relates to the time-consuming act of maintaining the many different databases for the many different individual smaller systems within a company or organization. Because the databases often reside in different locations in association with the different systems, there may be different business assumptions and data that adversely affect the output of the different systems. Because these different systems may use different business assumptions, the different application programs may not work well with one another.

Another limitation associated with existing smaller information systems is that these programs often only permit adjusting the information that is relevant to their model of the application. Although there may be other applications that changing simulation parameters affect, there is no existing system that permits making one change of parameters to an existing application program and then automatically making assessments
regarding the effects of this change on other applications.
SUMMARY OF THE INVENTION

As a result of the above, it is clear that there is a need for a method and system for communicating among existing different application programs that simulate or model certain physical parameters of a complex physical system to create a comprehensive and coordinated way of monitoring, modeling, and controlling the entire complex physical system.

There is a need to integrate smaller systems that form part of the modeling or management systems for a larger system to permit integrated and cohesive decision support.

There is a need for a method and system that avoids the problem of smaller application programs that form an organization's infrastructure by eliminating the complex and difficult application support and development tasks that exist for each of the application programs.

There is a further need for a method and system to overcome the problem of different languages and different formats which various smaller application programs may use to support a company's or organization's overall and complex business and operational infrastructure.

There is yet a further need for a method and system that permits maintaining the many databases that support individual application programs and systems, while eliminating needless expenditures of time and labor, and avoiding possibly erroneous inputs that result from
having different values for the same or similar parameters in the various application programs.

There is still a further need for a method and system that completely or substantially eliminates the problem of having to directly and independently modify parameters of many different smaller application programs based on the effect of changing one application program's parameters.

There is also a need for a method and system that avoids the fragmented views that numerous application programs provide in supporting management decisions relating to an information organization's infrastructure.

In light of these needs, the present invention, accordingly, provides an integrated management system and method for operations, planning and marketing in companies and organizations to yield an integration management template method and system that supports physically-based data and information.

One aspect of the present invention includes a computer-based integrated management template for monitoring and managing a physical system that includes a centralized database model having a plurality of parameters for modeling physical system data and behaviors. A graphical user interface permits communication between a user and the centralized database model. A first set of application program interface receives external data relating to the modeled physical
system. A second set of application program interfaces control and operate with a plurality of internal application programs for providing operations relating to the modeled physical system. In one further aspect of the present invention, the computer-based integration management system includes instructions for permitting case studies of physical system data and behavior. In another further aspect of the present invention, an object-oriented database model models complex physical systems data and behavior. As an illuminating example, one such complex physical system may be an electric utility or similar enterprise for generating and distributing electrical power.

According to another aspect of the present invention, there is provided a method for integrating the management, monitoring and control of a complex physical system. The method includes the steps of modeling data and behaviors of the complex physical system using a centralized database model having a plurality of parameters. The method further includes the step of communicating between a user and the centralized database model using a graphical user interface. Receiving data from external sources and relating the data from the external sources to the centralized database model is another step of the present invention. The method further includes interfacing a plurality of internal application programs, which may include varied data
structures, operational characteristics, and fundamental assumptions. One further important aspect of the invention includes the method of modeling the complex physical system data and behaviors using an object-oriented database model.

A technical advantage the present invention provides is a way to integrate numerous different systems that all use different representations of a given company or organization. The present invention provides a management space that includes a representation of a company, organization, or complex physical system in the form of an abstract representation that not only meets the needs of a specific application, but also provides information to manage and support an entire company or organization.

Another technical advantage of the present invention is that it provides an object model of a given company or organization that includes a domain object model together with a set of services that integrates the different application programs or decision and support services that the company or organization may use.

Still another technical advantage of the present invention is that it makes possible developing a centralized infrastructure for management of planning, operations, and marketing functions without discarding or having to reengineer or replace all of the various and different application programs already in use.
A further technical advantage of the present invention is its ability to communicate with multiple databases through a common interface to generate a single representation of all relevant data for an organization. The single data representation permits creating objects and asking for information, regardless of the separate or different database(s) that may be the origin of the data. The system, as a result, responds in a unified or consistent manner.

Yet another technical advantage of the present invention is the way in which the entire company or organization, or at least the operations, planning, and marketing portions of the company or organization is represented, so that a simulation or "what-if" change analysis propagates to all of the different applications within the system. This provides a more complete understanding of how changes in one sub-system may affect other sub-systems.

Another technical advantage of the present invention is the provision of a secure look at the information infrastructure of an organization while prohibiting arbitrary changes of values. Moreover, the security that the present invention provides may change according to the different access level that a system administrator may assign to the user.

A further technical advantage of the present invention is the provision of a set of integrating tools
or application program interfaces that make practical integration of all other application program interfaces. The present invention provides an open architecture where application program interfaces may interface an external program that supplies various processing services and data within the integrated management template. The present invention focuses on objects and permits modeling to a very low level of modeling degree in a fine-grained approach.

Yet another technical advantage of the present invention is the provision of historical archiving of data for maintaining accountability of changes to and within the system for tracking changes and monitoring changes, as well as for providing practical numerous data reversion operations. In the present invention, historical data, forecast data, and case management information may be used to develop a comprehensive and chronological view of a company's or organization's activities and operations.
BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantage thereof, reference is now made to the following description, which is to be taken in conjunction with the accompanying drawings in which like reference numerals indicate like features and wherein:

FIGURE 1 provides a conceptual view of the integration operations of the present invention;

FIGURE 2 illustrates one example of categories of an enterprise model formed in accordance with the present invention;

FIGURE 3 provides a conceptual illustration of intra-site object-oriented software applications for use with the present invention;

FIGURE 4 illustrates one aspect of the inter-site object-oriented software application features that the present invention provides;

FIGURE 5 depicts a case management hierarchy formed according to the present invention;

FIGURE 6 provides an example of a graphical user interface of integrated functions that the present invention makes possible;

FIGURE 7 illustrates examples of technology components for integration services that the present invention provides;
FIGURE 8 provides an exemplary object model diagram for the root object browser function of the present invention;

FIGUREs 9 and 10 provide an example of object model diagrams for creating function transfer metastatements according to the present invention;

FIGURE 11 provides a graphical user interface for events and exceptions according to the present invention;

FIGURE 12 provides an exemplary object model diagram for the data dictionary aspects of the present invention;

FIGURE 13 shows an exemplary object model diagram for the data dictionary entry aspects of the present invention;

FIGUREs 14 and 15 illustrate exemplary object model diagrams for the external application program interface operations that the present invention performs; and

FIGURE 16 illustrates an exemplary object model diagram for the system administrator and security functions of the present invention.
DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

The preferred embodiments of the present invention are illustrated in the FIGUREs wherein like numerals are used to refer to like and corresponding parts of the various drawings.

According to one aspect of the present invention, there is provided a sharable model of a company's or organization's operational and planning data that presents a consistent view of past, present, and future information for planning and analysis. The present invention establishes an environment for developing and integrating solutions for diverse business needs. In particular, the present invention provides a physically-based data and information management schema that employs a graphical user interface, decision and support tools, and application program interfaces for model, real time and historical data.

FIGURE 1 conceptually illustrates a preferred embodiment of the present invention of an integration management template 12 that includes fundamental electric utility data and behaviors, for example. Note that although the embodiment of FIGURE 1 relates to an electric utility, the present invention should in no way be limited to such an illustrative example. The present invention may be applied to numerous types of physical systems that include complex data and behaviors among complex parameters.
Within integration management template 12, application program interfaces (APIs) 14 provide interfaces to field and other external data. This may include real-time data, inter-site data. Electronic data interchange (EDI) block 18 indicates the various field or other external data interface functions that may relate to APIs 14. Also, integration management template 12 may include application programs via object behaviors 20 and 22 that relate specifically to the data and behaviors within integration management template 12.

Application program interfaces (APIs) 24 provide the necessary interface with existing application programs 26, office applications 30, new external applications 32 and existing systems 34, as well as other types of programs that may be part of the physical system (not explicitly shown in FIGURE 1).

Integration management template 12 provides a data management schema using an object model system that includes graphical user interfaces, application tool sets, and APIs such as APIs 14 and 24 for model, historical, and real-time information that a physical system, such as an electric utility system in the present example, may use in operations of planning and operating computer systems. Integration management template 12 uses or interoperates with a variety of object and relational database management systems to simultaneously
provide an open architecture and eliminate a customer's reliance on a single database vendor.

Integration management template 12 provides the key business processes that system operations and planning departments undertake by maintaining a common user interface and a central data repository for use by several applications. With the common user interface, users may easily access different applications that already exist within the company or organization. This makes it possible for users to run separate applications using the same data and to transfer results from one application to another, all without leaving integration management template 12.

Integration management template 12 contains a consistent set of tools for use by all applications that are integrated into or with integration management template 12, preferably including tools for editing and viewing data or results, tools for studying case maintenance, tools for system administration, and tools for database management. With these various sets of tools, users may quickly and easily jump from one application to another to achieve desired planning, operations, and marketing results. Because the user interface is consistent for all applications, users skilled with these tools for using one application may readily apply the tools to another application through the common interface of the present invention.
All applications within integration management template 12 use the same central database or data. All data and results from one application are immediately available for use by another application. Analysis requiring the user to run a series of applications and transfer the results from one to another is accomplished using integration management template 12.

Integration management template 12 includes a domain object model and a set of services that integrate various applications for using field and other external data. One such service is that integration management template 12 provides a centralized environment for application development that eliminates the problem of discarding work done in one application when making a change on another application. By establishing a centralized environment for application program development, integration management template 12 permits communication among multiple databases through a common interface to develop a single representation of the physical system, such as an electric power utility, in a unified way. Moreover, as yet another service, integration management template 12 provides a single, but flexible, way of asking for information or data regarding the model physical system, irrespective of the particular database of origin for the data.

FIGURE 2 provides object model diagram 40 that depicts the various categories of an object model.
residing within integration management template 12. In FIGURE 2, object model diagram 40 specifically relates to the present example of an electric enterprise such as an electric utility or electric system provider. For example, organization object 42 of object model diagram 40 may provide links with equipment object 44 and commodity object 46. Equipment object 46 may include connections with control process object 48, monitoring object 50 and energy production object 52. Monitoring object 50 may also interface with control process object 48. Commodity object 46 may communicate with energy usage object 54 and business rule object 56. End user object 58 preferably communicates with both energy usage object 54 and business rule object 56.

Integration management template 12 is based on a formal enterprise object model but represents key objects, attributes, relationships, and business activities of a complex physical system. As FIGURE 2 illustrates, integration management template 20 includes classes that represent, in this example, a power system and its attributes. Under each class category there is a hierarchy of classes. Many behaviors that are fundamental to an electric power system, for example, are also defined as part of the objects. This encourages the use of common assumptions and insures a high degree of data integrity among users of model data.
Because the different application programs and functions are influenced by modeling assumptions of one or a few functions, there is the difficulty of supporting multiple functions due to effects of the underlying database structure. Integration management template 12 overcomes this and other difficulties of integrating different suppliers' application programs and functions. The present invention carefully decomposes the models of a physical system, such as an electric utility, in a fine-grained, physically-based way that fits well with object-oriented software development.

The model definitions of integration management template 12 may be readily manipulated by the object-oriented software within integration management template 12. By virtue of this manipulation, the specific semantic and syntactical requirements of various models may be addressed. Within the categories of object model diagram 40, organization object 42 may represent business entities and the highest abstraction classes for node and link connectivity that represent a physical system. U.S. Patent Application Serial No. (08/611,412)

entitled "System And Method for Modeling Physical Systems," having common inventorship to the present application and commonly assigned to Electronic Data Systems Corporation is hereby expressly incorporated by reference. That U.S. Patent application describes one example of node and link connectivity applicable to the
present invention. Using the links and node connectivity, or a similar construct, the present invention effectively models complex physical systems while permitting effective interface with existing application programs. The following discussion lists exemplary objects and related classes for the preferred embodiment. In light of the present example – the modeling of a physical system, namely an electrical utility. Later the various classes and objects will be referred to as needed for describing the present invention.

Organization classes applicable to the present invention include, for example, an abstract link class, an aggregation class, a business entity class, a customer class, a hydrostation class, a link class, link and roll class, a nuclear station class, a physical link class, a power company class, a power pool class, a power system class, a station class, a substation class, a thermal station class, and a transmission corridor class. There may, however, be different, more, or fewer organization classes applicable to the present invention.

Equipment object 44 includes classes that represent various types of power system equipment. These classes may include, for example, a combustion turbine unit class, a conversion unit class, an equivalent hydroelectric unit class, an equivalent nuclear unit class, an equivalent thermal unit class, and a fossil
steam unit class. Moreover, equipment classes may include a hydroelectric unit class, a nuclear steam unit classes, pumped hydroelectric unit class, a qualifying facility class, and a steam unit class.

Commodity object 46 may include various commodity classes that represent anything of value that is bought, sold or exchanged between business entities. For example, commodity object 46 may include a capacity class, a commodity class, an energy class, a fossil fuel class, a fuel class, a nuclear fuel class, a steam class, and a transmission class, for example. Other numbers and types of classes may apply to commodity object 46.

Control process object 48 may include the system process classes to handle the distributed configuration of the present invention. Types of classes within control process object 48 include a client class, a configuration list class, a distributed callback class, a control panel proxy class, a general request broker class, a periodic function control class, a server class, a server copy class, a sign off class, a start up/shut down class, and a start up view class, for example. Other numbers and types of classes may apply to control process object 48.

Monitoring object 50 includes classes that provide an interface and repository for data obtained from an external supervisory control and data acquisition (SCADA) or energy management system (EMS). Classes within
monitoring object 50 may include an analog class, an analog archive class, an EMS transaction class, an energy management class, a measurements class, a monitoring external service class, and a status class, for example. Other numbers and types of classes may apply to commodity object 50.

Energy production object 52 preferably includes classes that supplement the conversion classes to model energy conversion. For example, energy production object 52 may include a second exponential transfer function class, a second transfer function class, a second transfer function view class, a third transfer function class, a curve class, a curve segment class, an exponential curve class, a segment collection class, and a transfer function class, for example. Other numbers and types of classes may apply to energy production object 52.

Energy usage object 54 may include various classes relating to energy usage. Business rule object 56 may include classes such as classes relating to operating rules. Operating rules classes provide a mechanism for users to define operating rules interactively. These operating rules can then be used to check application results or monitor other system conditions and use alarms through the events classes when operating rules are violated. Examples of operating rules classes may be, for example, alarm indicator classes, operating rule
formula classes, and the like. Other numbers and types of classes may apply to energy usage object 54.

End user object 58 may include numerous user interface classes, such as operations advisor interface classes. Such interface classes may be, for example, a comparison display value class, a display path class, a display transaction class, and a display value class. Also, user interface classes may include an increment/decrement set up view class, a multi-system result class, a net interchange class, and various other classes relating to user interfaces. Other numbers and types of classes may apply to end user object 58.

Integration management template 12 may also include other objects, as well as classes within objects not included within FIGURE 2. For example, integration management template 12 may include an activity register object that includes classes to provide a service for scheduling the execution of internal object processes on a timed, periodic or event-driven basis. These services may be available to both internal classes of integration management template 12 and external functions that are either loosely or tightly integrated with integration management template 12. Also, integration management template 12 may include an advisor engine that includes advisor engine classes to provide an object-to-C language interface for an operations advisor function. An area object within integration management template 12 includes
classes that model the physical and logical areas to which power systems, in the present example, are partitioned for organizational and electrical reasons. An audit object may include audit classes to provide mechanisms for logging and examining changes made to system data by users in external functions. Batch populate classes within a batch populate object may provide a convenient way to initialize data within integration management template 12.

An important set of classes for the present invention includes case management classes appearing within a case management object. The case management classes collaborate with the capabilities of persistency classes within a persistency object to support the generalized capabilities of integration management template 20 for "what-if" scenario cases. A containment object includes containment classes that represent physical entities that, in the present example, both conduct and store some physical quantities such as electricity, water, steam or fuel. A conversion object includes conversion classes that represent the conversion of energy from one form to another.

Integration management template 12 also includes a data dictionary object having data dictionary classes of instances that contain information about the persistent object representation and documentation of all classes within integration management template 12. Integration
management template 12 also includes an events exceptions object containing events exceptions classes that provide a generalized service through which classes of integration management template 12 or external integrated functions may route messages to users of integration management template 12. External application classes within an external application object of integration management template 12 provide a C+ computer language and FORTRAN language application program interface to the data and services of integration management template 12. Also, framework classes within integration management template 12 provide a unified way to build user interfaces for accessing information within integration management template 12, for integrating external applications, and for developing new decision support applications.

Integration management template 12 may also include a node topology object having node topology classes to represent connectivity for modeling electrical transmission and distribution, fuel transport, energy transfer, and other physical and business processes.

Within integration management template 12, there is a persistency brokers object having persistency brokers classes that allow integration management template 12 to use various relational and object database management systems for persistent storage of objects within integration management template 12. The persistency
brokers classes may include, for example, a gemstone broker class, an object database broker class, an object store broker class, an oracle broker class, a persistency broker class, a relational database broker class, and an object database broker class.

A persistency support object includes classes that support the ability of integration management template 12 to use various relational and object databases transparently from the point of view of functions within integration management template and integrated applications. Persistency support classes include, for example, query classes, query clause classes, reference classes, relational database reference classes, database access specification classes, database classes, and transaction classes.

The present invention further includes persistence collectors classes that work in collaboration with the various persistent objects classes to provide a uniform way to represent the future, current and historical states of all domain objects within integration management template 12.

Still other classes within integration management template 12 may be found in a security domain object and a security views object. In a security domain object, security domain classes include an access control class, a component group class, a component group reference class, a domain account group class, a password class, a
policy class, a resource control class, a resource limit class, a security access class, a security administration class, a temporal access control class, a user account class, a user account reference class, a user view class, and a user component reference class.

The security views object classes include a change password class, a component group tool class, a domain account group definition class, a log-in controller class, a log-in view class, a profile view class, a temporal access view class, a user maintenance view class, and a view account group definition class, for example. The security domain object classes and security views object classes allow the security administrator for integration management template 12 to control users' access to various user interface views and domain objects within integration management template 12.

FIGUREs 3 and 4 illustrate the aspect of the present invention for using distributive object-oriented software that goes well beyond traditional client/server applications. FIGURE 3 indicates the use of distributed objects for intrasite computing according to traditional client server applications. In particular, for client 62 having objects 64 and client 66 having objects 68 there is server 70 having internal objects 72. In a traditional client/server application, intrasite computing objects may be effectively distributed. FIGURE 4, on the other hand, shows the use of distributed
objects for inter-site computing. FIGURE 4 shows distributed management system 82 which has internal objects 84 and which establishes an inter-site link with bulk power control center 86 to communicate with distributed objects 88. Substation 90 includes objects 92 that make use of objects 84 within distribution management system 82 according to the control of bulk power control center 86. Using distributed object-oriented software, the present invention provides more flexible distribution of information to match the geographical nature of the business, while ensuring scalability according to the various sizes of supported enterprises.

FIGURE 5 illustrates the case management function of the present invention which handles multiple independent cases according to a process for which case tree 100 is but one example. Stemming from base case 102 may be, for example, Case A, at 104, and Case B, at 106. Case A may generate Case C, at 108, and Case D, at 110. Case D may generate Case E, at 112. Likewise, Case B, at 106 may generate Case F at 114, which itself generates Case G, at 116. The case management aspect of the present invention provides for storage of function results and permits case comparison features that are usable for future as well as initial functions.

The case management function provides a user with the ability to run studies with either modifications to
base case 102 or using actual system data. The present invention provides facilities to control data access to compare data or results from two or more cases and to keep track of data changes. The effect of data changes may be easily evaluated before being promoted into the base data, all without affecting other users of the base data.

U.S. Patent Application Serial No. 08/610,924 entitled "A Case Management System and Method for Managing Case Models of Physical Systems," having common inventorship hereto and commonly assigned to Electronic Data Systems Corporation, is here expressly incorporated by reference. The invention of that U.S. Patent application includes a software case management system and method for managing cases that represents models of an operation system; cases are created by a user having access to a base set of data containing variables. The invention of this referenced U.S. Patent application fully describes the operation of the case management of case tree 100 that may be used with integration management template 12 of the present invention.

FIGURE 6 shows graphical user interface 120 to illustrate operations advisor features, as well as to describe the generous graphical user interface features of the present invention. Graphical user interface 120 includes minimize and maximize buttons 122 that vary the size of the viewed portion of the graphical user
interface 120. Operation of button 124 controls operations associated with the operations advisor functions of the present invention. File pulldown menu selection 126, tools pulldown menu 128, application pulldown menu 130, window pulldown menu 132 and help pulldown menu 134 provide access to operations similar to those appearing in most graphical user interface application software programs in order to make the functions and operations of the present invention user friendly. The operations advisor features of the present invention, however, are unique and not in any way suggested by the prior art.

The operations advisor function of the present invention provides an effective and simple-to-use mechanism to acquire data from external inputs for using and sharing the data with various applications within integration management template 12. By working from a common object model, the present invention provides an economic dispatch which tells the user where the power system, for example, should generate power based on inputs such as existing generating capacity, existing system conditions, and projected system conditions. Button 136 indicates costs associated with electrical utility operation. Button 138 illustrates various transaction parameters. Button 140 describes peak demand and related information. Button 142 provides access for viewing and using other applications having a
relationship with the operations advisor function. Other buttons may be provided that provide information relating to the operation of electrical power utilities, for example. The output from operations advisor may appear in such forms as graph 144 that provides a timed load distribution as shaded portion 146. Capacity balance spread sheet 148 provides data points relating to load, such as that which graph 146 depicts.

The operations advisor function of the present invention may assist, for example, power system operators in making short-term operating decisions and in evaluating the economic, reliability, and environmental effects of those decisions. The operations advisor function performs economic system dispatches for predetermined time periods using the most current system operating data. For the example of a power system model, the operations advisor function considers generator constraints, transmission losses, noxious effects, data dispatches, fuel configuration, interchange transactions, transmission capacity, reserved and regulation requirements, as well as load forecasts in evaluating the power system production requirements.

The operations advisor function of the present invention permits evaluating the modeled physical system with a current forecast or in a "what-if" planning mode. Facilities of the operations advisor may allow, in the power system example, quick evaluation and "what-if"
scenarios such as unplanned unit outages, load forecast changes and fuel cost changes. The present invention preferably includes a detailed interchange transaction evaluation package to assist in the optimization of off-system purchases and sales. The operations advisor function may also automatically run a comparison of actual, planned, and optimized results for post-analysis.

The operations advisor, moreover, monitors specified system operating rules and provides notification of operating rule violations. Preferably, all violations are written to a log file. With the present invention, the operations advisor may also be used to notify operators when potential improvements could be made to the system.

The operations advisor of the present invention contains built-in security and data control features. Access to view and modify system data and to perform system studies is strictly controlled. All system based data changes are logged to allow complete reconstruction of system data at any point in time.

The operations advisor of the present invention, as FIGURE 6 shows, uses an advanced graphical user interface that provides simple, consistent access to data, results, and system options. Unique colors are used to visibly identify key components of the data and results. Data editing features include the ability to allow quick entry of large amounts of data, such as hourly interchange
transaction information. Also, standard error and warning boxes may be used and data changes may be permanently logged using the operations advisor function.

In the preferred embodiment, the operations advisor executes automatically four times per hour for the base case. This is called a real-time run. The operations advisor executes automatically, for example, once per hour for an incremental/decremental run. This is called a real-time incremental/decremental run. Once per hour, a post-analysis run may be automatically initiated.

The operations advisor function also allows for user-initiated study runs. These runs are initiated by entering data changes into a case study and selecting the run, increment/decrement, or post-analysis button on the operations advisor graphical user interface 120. The data view function in the operations advisor execution period may be specified by selecting the "time filter" pulldown menu within operations advisor graphical user interface 120. The operations advisor results are calculated and displayed for the execution period in the preferred embodiment.

Graphical user interface 120 includes title bar 121 which consists of the application title and icons to close, move, minimize, maximize or return to preferred window size, or to switch to another application. Menu bar 123 contains pulldown menus which open views or start applications. Clicking on a pulldown menu and then
selecting an item from that menu opens that view or
starts the chosen application. Button bar 125 contains a
sequence of buttons to execute applications. Clicking on
the appropriate button begins execution of the chosen
application.

Display region 127 may also contain display boxes
for context sensitive help, application status, case
name, and current date and time. The context sensitive
help box may display help relating to the region where a
mouse or other pointer may be positioned. This help
message could change as the mouse pointer location moves
to other regions of the screen or other application bars,
such as to title bar 12, menu bar 123, and button bar
125. A status box may display the text detailing the
status of the application execution. A name box may
display the current case name that a case manager
defines.

In the preferred embodiment, file pulldown menu 126
permits a launcher function to launch operation of
integration management template 12. File pulldown menu
126 also includes a setting selection for opening setting
windows relating to integration management template 12.
Also, file pulldown menu 126 includes a data and results
view for presenting tables, sharing several editing and
viewing features, a time filter for making accessible to
the user a menu of seven dates centered around a current
date, as well as for making accessible a time range setup
window. The time range setup window permits modifying the application view and execution period.

Tools menu 128 includes an alarm window selection for opening an alarm window that displays event from any log associated with the integration management template 12 application. Tools pulldown menu 128 also includes a case manager selection to activate the case manager functions of the present invention, thereby allowing users to run studies with modifications to the base and actual system data. Facilities of the case manager include the ability to control data access, to compare data results from two cases, and to keep track with data changes. The effective data changes may be easily evaluated before being promoted into the base data, all without affecting other users of the base data.

Tools pulldown menu 128 further includes an audit trial selection to access an audit trail system and further open a data entry selection window. U.S. Patent Application Serial No. 08/611,034, entitled "System and Method to Undo and Audit Changes Made in a Case Modeling System," having common inventorship herewith and commonly assigned to Electronic Data Systems Corporation is here incorporated by reference. That U.S. patent application describes and claims a method and system for use with the present invention for achieving the audit trial system functions of the present invention. Tool pulldown menu selection 128 accesses a data entry selection window that
displays names of case studies which have been changed and logged into the audit trail system. This window also displays all data change types, the identifiers of the users making changes, and objects whose changes have been logged into the audit trail system. Every data activity which has been logged to the audit trail system can therefore be located by selecting the appropriate combination of case, type, user, object and attribute. Study case creations may also be accessed through the audit trail system. The search can be refined by limiting the search time interval. Also, from the data entry selection window a system administrator may access information relating to a user's login or logout status and records.

Tools pulldown menu 128 includes a data query selection to access the object query editor. The object query editor provides access to all data base objects. Using this tool, a user may retrieve and display the date values for any item in the data base. Facility is provided to construct a logical expression to be used in the data query, only objects which satisfy the logical expression are returned from the query.

Tools pulldown menu 128 also preferably includes a file editor to open a small talk file editor. A table editor's menu selection within tools menu selection 128 opens a presentation description editor. The presentation description editor allows the user to create
user-defined views which are accessible from the tool pulldown menu 128. User-defined views are custom views that allow each user to create and save a customized set of screens to display application data or application results.

Tools pulldown menu 128 includes a client menu selection to permit viewing a list of users currently logged into integration management template 12. Selecting a user brings up a display on a user host name, port and logon time.

Tools pulldown menu 128 includes a domain editor selection that allows access to view and edit domain data. A domain editor is designed to allow browsing of the object model structure or to allow quick viewing of a specific data item. Most data changes by users occur in data editing and display tables, not in the domain editor. The domain editor allows for editing data that is not commonly changed by users or data which has a structure unsuited for standard data display screens.

Tools pulldown menu 128 also includes an object editor that permits viewing and editing objects which are represented in piecewise polynomial or exponential curves. Examples of these type of objects for a power system may include burner incremental heat rates, noxious emissions and SO, emission rates, as well as startup energy.
Application pulldown menu 130 permits opening application parameters. Application pulldown menu 130 includes a parameter selection for adding parameters, as well as a user defined view.

Window pulldown menu 132 provides selections to minimize or expand all "children" created from application toolbar 123. The Window pulldown menu 132 provides a submenu to select a "child" window to expand and make current.

Note that the above discussion describes only one of a wide variety of possible user interfaces that may permit access to and control of the innovation structures and functions of the present invention. Accordingly, in no way should the above discussion be used to limit the scope of the claims to the invention herein described.

FIGURE 7 shows the various technology components that may be used with the preferred embodiment of integration management template 12 of the present invention. Integration management template server 152 communicates over Ethernet or token ring network 154 using a TCP/IP protocol with application server/integration management template client 156, integration management template client 158 and integration management template client 160, for example.

Integration management template server 152 may be on a UNIX or similar workstation that includes VisualWorks® or Smalltalk 80® application programs 162 together with C
API program 164, which is a C+ language application program interface. Object base or relational database 166 and Distributed Smalltalk® 168 preferably interface with VisualWorks®/Smalltalk 80® application program, as does CORBA® application 170. Server 152 further includes UNIX sockets 172 for communicating with Ethernet or token ring network 154.

Application server/integration management template client 156 includes UNIX sockets 174 for interfacing Distributed Smalltalk® application 176, VisualWorks®/Smalltalk 80® application 178 and database client application 180. Integration management template client 158 includes Windows® sockets 182 for providing an interface between Ethernet or token ring network 154 and Distributed Smalltalk® application 184, VisualWorks®/Smalltalk 80® application 186 and database client application 188. Integration management template client 158 may use an Intel PC®, Windows 3.1, or NT platform. Integration management template client 160 provides an example of an Intel PC® or OS/2 platform that includes OS/2 sockets 190 for interfacing Ethernet or token ring network 154 with Distributed Smalltalk® application 192, VisualWorks®/Smalltalk 80® application 194, and database client application 196.

FIGURE 8 diagrams the root object browser function of the present invention. The root object browser function allows an administrator to view, edit, create or
delete root objects. To view or edit a root object, a user selects the object category, object class and root object from a category, class and object list. The present invention displays the root object and its attributes on a graphical user interface that may include multiple pages.

In the present invention, simple attributes may be edited by entering new values. Complex attributes may be edited by first selecting the attribute from the screen at which point the invention displays the complex attribute. Then, the complex attribute may be edited by individually editing the associate simple attributes or by editing in some other practical way. This process may be repeated to display any object at any level within the system. Commands that the root object browser provides include a remove command to remove the selected root object and a copy command to create an editable copy of a selected root object. Also, the root object browser provides an accept function that writes the selected root object along with any changes to the centralized database of integration management template 12. New root objects may be created by copying an existing root object, entering new attribute values and accepting the changes.

With reference to FIGURE 8, root object browser function 202 asks of each persistent object whether it is a root object. If the object is a root object, the present invention access the persistent object function
for maintaining and displaying the object. Root object browser object 202 sends a display-an-object message to the inspector object 206 so that the present invention will display the object. Inspector is used in a sub-canvas portion of the browser to display a list of instance variables, as well as a data entry form for the variables. Inspector object 206 builds a hierarchy collection with the object that it is inspecting. Hierarchy collection 206 is part of a selection-in list that the present invention uses to model the list of instance variables. Hierarchy collection 208 asks the current selection to perform an expansion selector function and answers or responds to a next-level list of instance variables. The answer is inserted into hierarchy collection 208, and the selection-in list holding of the collection is notified of the change.

Inspector object 206 is called upon to provide a suitable view object to display each instance variable on the selection-in list. An instance of a hierarchy-label-and-icon object is built for each list item as it is about to be displayed. Hierarchy-label-and-icon object 212 displays an appropriate icon on a graphical user interface as well as connecting lines between the icons. Hierarchy-label-and-icon object 212 also controls an indent level of an index signal to hierarchy collection 208. This permits determining how many levels of indentation to apply to the represented item.
Inspector object 206 also builds a custom data entry form for the currently selected object in a list of instance variables that it provides and maintains. Some variables are simple enough to be editable, others are too complex and are represented by their default display information. Those that can be displayed are linked to an edit field using aspect adapter object 214, which a selector model creates for an aspect message. The subject for the aspect adapter object 214 is an instance of inspector model object 216 which is one of a simple class of objects. Aspect adapter object 214 sends a perform-a-selector message to inspector model object 216. Inspector model object 216 implements the does-not-understand-a-message method to capture and fulfill all requests from aspect adapter object 214 for any object.

FIGURE 9 shows within a dash-line box formatted transfer metastatement system 222 of the present invention which provides a mechanism for transfer of data between integration management template 12 and the resident file system, which system maybe in some third party file format. A formatted transfer metastatement will extract and format the information to be transferred. Formatted transfer metastatement system 222 provides a user-friendly, menu-driven, point-and-click way to create formatted transfer metastatements for this purpose.
In FIGURE 9, domain services object 224 provides access to formatted transfer metastatement system 222. Populate facility 226 controls access to naming serving object 228. Naming service object 228 may use returns facility 230 to permit process flow to go to domain object 232. From domain object 232, formatted transfer metastatement system 222 uses stores facility 234 to access objects portion 236 of formatted transfer metastatement object 238. Naming service object 228 also uses request facility 240 to control input to targets portion 242 of formatted transfer metastatement 238. Formatted transfer metastatement object 238 includes, not only targets portion 242 and objects portion 236, but also metastatement request portion 244, metastatement errors portion 246 and formats portion 248. Formats portion 248 uses applies facility 250 to send formatted output to objects portion 236. Formats portion 248 also uses translate facility 252 to control process flow to external file object 254.

FIGURE 10 depicts in more detail format object portion 248 of formatted transfer metastatement object 238 to show how developers may develop optional methods to address unusually-structured inputs. Within format object portion 248, reader object 256 employs a print converter class and interfaces with fixed format object 258 and delimited form object 260. Metastatement interpreter object 262 also provides an input to reader
object 256. Reader object 256 employs converts facility 264 to convert the unusually-structured input from an external file 254 to a form that fixed format object 258, delimited form object 260, and metastatement interpreter 262 can use to generate a metastatement that is useable by domain object 232. Reader object 256 uses applies facility 266 to employ domain object accessor object 268 and metastatement request object 270. Metastatement request object 270 uses case name, effective time-stamp, and version time-stamp classes with regard to the unusually-structured input to generate an output to domain object 232.

In the present invention, a formatted transfer metastatement defines a mapping between domain object 232 of integration management template 12 and external file 254. The invention includes a formatted transfer metastatement editor to create or alter formatted transfer metastatements. The present invention makes it possible to define new formatted transfer metastatements for reading external file 254 and mapping this information to domain object 232.

A formatted transfer metastatement is an object which contains information containing a name, a comment, a formatted transfer metastatement type (e.g., create, update, validate) a format type (e.g. fixed or delimited), a delimiterator (if the format transfer metastatement is delimited), a target domain class, and a
collection of record file descriptions or formats. Each file record contains information relating to some root domain object. By convention, the name of a formatted transfer metastatement consists of the class of the root domain object in a mixed case. This is followed by a short descriptive mnemonic. For example, the records used to create a new instance of an equivalent thermal unit when applying the preferred embodiment of integration management template 12 to model a power system may be the name, "EquivalentThermalUnitDEF".

The comment is used to provide a short description of the mnemonic. The formatted transfer statement type denotes how the formatted transfer statement is to be used, e.g., to either create or update domain objects. A special formatted transfer metastatement type of "validate" establishes the information to populate validation objects. An external file is either of a delimited or fixed format type.

Each field on an external file record is described by an individual field descriptor or format. Each field may be thought of as containing either data or key information about the data. The dataset located at the bottom of the formatted transfer metastatement editor contains a list of currently defined formats for the active formatted transfer metastatement. Adding new formats is possible by accessing either the key accessors or value accessor window of the formatted transfer
metastatement editor. Once each new format is added to the dataset, information like data type, location, and unit of measure of conversion are the types of information that must be defined. Adding new formats is accomplished by selecting the appropriate field format and adding the use of a pull-down menu activated by a screen button or some other appropriate selection mechanism.

Creating a new format transfer metastatement

preferably includes the steps of (1) defining the name of the format transfer metastatement's name; (2) describing it via the comment; (3) denoting the type of formatted transfer metastatement; (4) describing the file format and, if applicable, the delimiter used; and (5) describing each field of the records. The following is a series of examples of the various types of formatted transfer metastatements that the present invention may provide.

One type of formatted transfer metastatement is the definition type formatted transfer metastatement that is used to create any named domain object in integration management template 12. These are the only objects which require the use of a definition type formatted transfer metastatement. Another type of formatted transfer metastatement is the simple keyed formatted transfer metastatement that allows the record to dynamically determine which objects are updated. Conceptually,
fields either define domain data or information about how to find an object to update. A related type of keyed formatted transfer metastatement, the temporal keyed formatted transferred metastatement, allows a record to dynamically define the time period at which temporal object becomes affected.

Second transfer function formatted transfer metastatements are exactly like temporal ones, except for the fact that they lack a terminate date. Second transfer function formatted transfer metastatements in general are considered to persist until a new effective instance is encountered. Indexed formatted transfer metastatements allow a record to dynamically define the order that data can be mapped into. Interval formatted transfer metastatements allow a record to dynamically define the series of data items which can be mapped into an interval type of domain object. Referential formatted transfer metastatements permit making associations between different objects. Description formatted transfer metastatements define the default value for a particular instance of an object when using that accessor. They also may be used to make associations between an object's accessor and some system measurement.

Validant formatted transfer metastatements are special objects that validate their associated domain objects. Validants may be defined at the class or instance level. There are two types of validants. A
first type is one that defines a numerated list. Another one is one which defines high/low ranges.

There may be other types of formatted transfer metastatements the user may design that are consistent with the purpose of transferring data between integration management template 12 and the resident file system for some third party file format. Within the constructs of FIGUREs 9 and 10, the formatted transfer metastatement system 222 makes these designer metastatements possible.

FIGURE 11 shows message box graphical user interface 280 that provides an events/exceptions overview to assist in the operation of integration management 12. Message box graphical user interface 280 includes control bar 282 for controlling operation of exceptions portions of integration management template 12. Within message box graphical user interface 280 are title portion 284 and text portion 286. Below text portion 286 are accept button 288, remove button 290, test button 292, and exit button 294. Menu portion 296 permits control of the illustrated selections for text portion 286 and includes in certain instances, submenus, such as submenu 298.

Exceptions portion of integration management template 12 provides a way to store and display system exception messages. A typical example of such a message in message box graphical user interface 28 is one which warns the user that there are changes to the method of exiting an application. Message box graphical user
interface (280) is responsible for displaying messages and for storing message titles in title portion 284 and text portion 286. The preferred embodiment also provides a message box editor which is a developer utility that includes message, titles and text and offers significant flexibility in applying fonts, font size, colors and emphasizes characteristics to message text in the preferred embodiment, the editor may be started from the VisualWorks® launcher tools menu.

With the present invention, there may be several different ways to evoke message box graphical user interface 280. In addition to reporting exceptions, the present invention permits viewing or browsing events that take place within a complex physical system as recorded and modeled by integration management template 12. Integration management template 12 maintains a log of important events that occur during planning as well as events that occur from actual operations activities. An event may be, for example, a notification that some aspect of the power of the physical system has gone wrong or that a previously recorded event has been corrected. An event could occur, for example, when an electrical generating unit operates above its maximum or below its minimum capacity, or when the physical system is below its spinning reserve requirements.

Events may be of high, medium or low severity, depending on the type of event. A user may define
additional logs to hold specific types of events. Maintenance, operation or planning events may be sent to their own log for use by maintenance, operations or planning functions, respectively. A user defines which logs and event is sent to, but an event is additionally always sent to the system log. The system log contains all events as they may be identified by their unique type, source, and reference values. User-defined logs contain all events sent to the specific logs. User-defined logs may also contain multiple events with the same type source and reference values.

An event type is name or type of events, such as "limit violation" or "advisor." The source is a name of a source of the event, which is typically the requesting application's name or the name of a resource management by the application (e.g. "Unit A" or "Rules."). The event referenced has a name for the subject of the event, such as "greater than maximum capacity" or "system not responding." The event parameters are a collection of other significant pieces for the event. The event collection facility also keeps track of when an event is created, when the event is acknowledged, who acknowledged the event, when the event is deleted and when the event is returned to normal.

The event log browser of the present invention provides a notebook page that allows a user to define new logs and browse existing logs. The event log browser
function of the present invention also allows a user to define and browse event message flags. Event message flags identify the type of event, the severity of the event, and the type of notification for the event. Notification may include the visual effect of flashing text or an audible notification when the event occurs.

To define a new log, the present invention permits selecting a "define logs" notebook page. The user may enter the log name in a name entry box of the notebook page and then select the applications to be notified when a new event is added to the notebook page. When an event is added to this log, the alarm window from the selected application is updated to display the event message. Then, the user may select an "accept" function to accept the new log definition.

To browse a log, the user may select a "browse log" notebook page from the associated graphical user interface. The user may then select a log from a pulldown menu of the graphical user interface. The event type, source, and reference values are displayed along with the time of the event. With the preferred embodiment, events may be sorted by type, time or reference by selecting specific buttons on an associated graphical user interface. The present invention also provides a report of the type, source, reference, parameters, priority, respondent and significant event types. The events reporting function of the present
invention provides a way to add new message flags that include severity level and audible or viable queues relating to the modeled physical system.

FIGURE 12 shows an object model diagram to describe the data dictionary aspect of the present invention. Within the dash-line box appears specified components of the data dictionary function 302 the present invention which components provide a central documentation repository containing information as to the content and underlying structure of integration management template 12. Data dictionary object 304 provides a way to document program and centralized database parameter definitions of integration management template 12. Programmers, developers or administrators may use data dictionary function 302 by accessing the model server associated with integration management template 12.

As FIGURE 12 indicates, numerous objects interface data dictionary object 304 for the various functions that data dictionary function 302 performs. For example, developer object 306 may use manages facility 308 to manage data dictionary object 304. User object 310 may employ requests facility 312 to retrieve data from 304. Object populator object 314 may use checks facility 316 to check the contents of data dictionary object 304. C API object 318 may use accesses facility 320 to access data dictionary object 304. Smalltalk® object 322 may
use compares facility 324 to compare various data
dictionary values as needed.

Within data dictionary function 302, table column
object 326, which is associated with or part of table
schema object 328, uses maps facility 330 for mapping
functions associated with data dictionary 304. Data
dictionary document object 332 accesses data dictionary
report object 334. Data dictionary word document object
336 employs formats facility 338 and determines necessary
formats relating to elements of audit result object 340.
Data dictionary object 304 uses produces facility 342 to
yield audit entries and identify audit problems relating
to elements or classes within data dictionary object 304.

Classes within data dictionary object 304 preferably
include an agent and event class, an assertion class, a
class definition class, an event initiation class, a
function class, an operation class, a predicate class, a
relationship class, a terminology class, a type
definition class, a variable class, an instance variable
class, a method class, an instance method class, and a
table schema class.

The graphical user interface associated with data
dictionary function 302 includes numerous tabs for
viewing and editing desired information. One such tab is
the class definition tab or class that accesses the class
definition notebook page for allowing entry of a class
definition or description for documentation purposes.
Class references are also entered on the class definition notebook page. The class reference is generally for documentation purposes only. In the data definition graphical user interface, the variable appearance of the tab accesses a variable appearance notebook page for allowing entry of class and instance variable display parameters. Also, a variable description tab for the data dictionary produces a variable description notebook page that allows entry of class and instance variable descriptions for documenting purposes.

A variable properties tab or class produces a variable property notebook page that allows entry of class and instance variable properties. A methods tab produces a methods notebook page that allows entry of class and instance method descriptions for documentation purposes. A domain reference notebook page allows definition of a relational database table for storing objects and instance variable values. In one implementation of the present invention, a documentation tab yields a documentation notebook page that allows the user to generate a Microsoft® Word® file that contains the documentation for selective classes. Moreover, a schema tab accesses a schema notebook page permitting the user to generate new relational database tables. Schema generation applies when defining a new table or when a structure of the date contained in a table is to be changed.
FIGURE 13 illustrates an object model diagram indicating the associated models for entry data into data dictionary object 304. In particular, data dictionary entry object 352 preferably includes various classes such as a name class, a description class, a reference class, and a sub-system class. Numerous objects associate with data dictionary entry object 352. These include agent and event object 354 for performing functions of sending events and receiving events relating to data elements. Assertion object 356 associates with data dictionary entry object 352 and performs conditioned function(s) relating to data dictionary elements. Class object 358 includes super class and table name functions associated with data dictionary elements.

Event definition object 360 includes direction and argument list functions that associate with elements of data dictionary entry object 352. Function object 362 preferably includes an argument list function, a return type function, and an expression function relating to elements of data dictionary entry object 352. Operation object 364 preferably includes operation kind function, operation agent function, and argument list functions for elements of data dictionary entry object 352. Predicate object 366 includes argument list functions and condition functions for data dictionary entry object 352, relationship object 368 and terminology object 370. Type definition object 372 includes type list collection and
type expression elements associated with data dictionary entry object 352.

Attribution object 374 and method object 376 associate with data dictionary entry object 352. Attribution object 374 include functions relating to various classes, including, for example, a variable type class, a variable size class, a variable organization class, a variable mutate class, a variable share class, a variable bind class, a variable class class, a variable name class, a variable units class, and a variable validity check class. Class variable object 378 and instance variable 380 associate directly with attribution object 374. Method object 376 includes functions relating to classes such as, for example, an argument list class, a return type class, and a method class. Class method object 382 and instance method class 384 associate directly with method class 376.

An important aspect of the present invention is external function execution that uses a remote procedure call, creation and submission of batch and shell files for re-execution or other ways to cause execution of a function that is non-inherent or non-native to integration management template 12, but to do so on the same or other platform. FIGURE 14, therefore, illustrates the object model for performing this function. External application object 392 includes operating system object 394 and external process object.
396. External application object 392 accesses through integration management template 12 boundary 398 as the dash-line box represents.

Within integration management template 12 boundary 398, access request facility 400 determines the need to enter the C API boundary 402 that the dash-line box describes. Within C API boundary 402 is external application registry object 404 that employs test access facility 406 to access security object 408 and notify facility 410 for accessing event log 412. Start facility 414 provides input from external application registry object 404 to external application object 416. External application object 416 includes process object 418 and channel object 420. Services object 422 associates with external application object 416. Application object 424, which is external to C API boundary 402, engages start facility 426 to interface external application object 416.

External application object 416 includes an application class and a state class. Within process object 418, an identifier class, a name class, and a priority class provide output for using controls facility 428 that access external application 430. Also, channel object 420 includes an inqueue class and an outqueue class for accessing post-request facility 432 for providing an input to external application object 430. Output from external application object 430 employs
"connect to" facility 434, "disconnect from" facility 436 and "send results" facility 438 to provide an appropriate input to channel object 420. Operating object 440 and external object 442 are within or controlled by external object 430.

FIGURE 15 shows in more detail further aspects of operations within integration management template boundary 398 in relation to operations occurring within C API boundary 402. As FIGURE 15 depicts, external application object 416 uses get time out facility 452 to use the external time out values class of time out library object 454. This operation occurs within C API boundary 402. External application object 416 also uses post request facility 432 within C API boundary 402 to engage, as appropriate, objects within or associated with services object 422 of integration management template 12. Such objects preferably include activity registry object 456, monitoring object 458, model data access object 460, data dictionary object 304, results object 462, and event log object 412. From activity registry object 456, monitoring object 458, model data access object 460 and data dictionary object 304, outputs may go to send results facility 438 to supply the output to external application object 416.

As FIGUREs 15 and 16 indicate, an external application may gain access to integration management template 12 by various methods. An external application
may be part of an application activated from within integration management template boundary 398. Alternatively, an external application may be accessed by an executable run of the operating system. In this case, the external application will request accesses to integration management template 12 via external application registry 404. In either case, external application object 416 acts as a liaison between operations within integration management template boundary 398 and external application 430.

External application object 416 includes process object 418 and channel object 420. Process object 418 controls the behavior of external application object 416, for example, to carry out, suspend, resume, or change priority operations. Channel object 420 acts as the communications line between operations of integration management template 12 and external application 430. Any requests or results from external application 430 will be passed to the requested services object 422. Any such requests include a time out event associated with it to ensure that no deadlock conditions may exist. Time out library object 454 includes values for these time out events.

To prepare for a request, external application object 430 may retrieve an object's data and size constraints via data dictionary object 304 within C API boundary 402. Prior to passing a request along to
services object 422, the request may be interrupted by a naming service which provides the ability to bind a string to an object instance within integration management template boundary 398.

Security for integration management template boundary 438 is assured by using two mechanisms within C API boundary 402. Access within integration management template boundary 398 via external registry object 404 requires external application 430 to provide information necessary to establish integration management template boundary 398 access rights for the process. This occurs by integration management template 12 communicating with a security object, as discussed in more detail below. If the security object permits access within integration management template boundary 398, the present invention uses these access rights to establish a user context for subsequent access within integration management template boundary 398. Services object 422 also uses this user context to maintain security at all times. Moreover, event log object 412 is notified of this security access, using notify facility 410 (shown in FIGURE 14) with each initial access to external application registry object 404.

FIGURE 16 shows an object model diagram describing the object model associations of the administrator's module of the present invention. An important aspect of the present invention is the security level used to
maintain control of integration management template 12. Each user possesses a profile that encloses user preferences and configuration information relating to the user. A user profile also provides the system with an ability to establish system features that the user may access. Using entry on an display that is accessible only by a system administrator for integration management template 12, users may be granted the privileges of viewing and/or entering data on each individual display associated with integration management template 12, as well as executing functions that integration management template 12 controls. Security control exists down to an instance level.

Entry to integration management template 12 and access to applications relating to integration management template 12 use name and type identifiers and passwords that are checked through a login view function. Users may change their passwords through a change password view function. The system administrator controls use of and restrictions on passwords through a user's security profile. A system administrator may grant or deny a user access to individual applications within integration management template 12. A user who does not possess access rights to a particular application of integration management template 12 does not see that particular icon for the application after successfully logging on to the system. Moreover, a system administrator may control
access to data and visual components of applications within integration management template 12 through the use of access levels and groups. For instance, the levels of access for the preferred embodiment are write, read-only, and no access. The system administrator may create or destroy access rights according these access levels. Also, access to root objects of data structures for integration management template 12 may be to domain access groups. View access groups may also access the visual components of windows or user interfaces for integration management template 12. Examples of visual components include action buttons, menu lists, or data entry fields. The system administrator may add or remove users from the domain and view access groups.

Referring in particular to FIGURE 16, system administrator object 472 uses modifies facility 474 for changing classes within policy object 476. Policy object 476 preferably includes classes such as an expiration interval class, a password individual class, a login attempt class, and a trap-door state class. System administrator object 472 also uses modifies facility 474 for modifying access control object 478 within secured domain object 480, as well as for modifying view object 482 and component group object 484. With enters facility 486, system administrator object 474 may enter information into user account object 488 and access control object 490, which is also within integration
management template boundary 398. View object 482 includes component group object 492. Component group object 484 includes access control object 494 that interfaces modifies facility 474, as well as view component object 496. User account object 488 includes password object 498 which calculates expiration facility 500 associates with policy object 476. Access control object 490 includes domain account group object 502, which contains a user accounts class. References facility 504 associates domain account group object 502 with access right object 506. Also, view account group object 508 uses application classes and interfaces with domain account group object 502. User account object 488 and secured domain object 480 use interacts facility 510 to reach user object 512. Interacts facility 510 also interfaces view object 482 and password object 514 which includes an expiration class. Changes facility 516 associates user object 512 with password object 498.

As FIGURE 16 indicates, the security system that system administrator object 472 controls governs restrictions on system login and passwords. To employ the relationship that FIGURE 16 describes, a system administrator may change a user profile using a system profile button or similar device on a toolbar of the graphical user interface for integration management template 12.
Using passwords with integration management template 12 is optional and is controlled through the system profile. If the password option is off, view object 482 removes the login view from the integration management template graphical user interface and only a valid login identifier is required to enter integration management template 12. If the password option is on, the password field appears on the integration management template 12 graphical user interface.

Users may change the passwords using changes facility 516 through accessing password object 498. A changed password view requires successful entry of the original password before enabling a new password field. The new password preferably should be entered twice correctly before the integration management template 12 accepts the password.

A system profile defines restrictions on the creation of passwords. A system administrator may impose a minimum or maximum length for passwords and an expiration for a given password using password object 514. The present invention employs user accounts to grant access to integration management template 12 into applications associated with or controlled by integration management template 12. The user account contains information such as login identifier, name, password, resource allowances, domain access groups, applications and view access groups.
User accounts are created and maintained by the system administrator through the use of view object 482. Also, a system administrator for integration management template 12 may control the amount of data created and kept by a user through the use of resource limits. The system administrator may add or remove an integration management template 12 application from the user's list. Also, a system administrator may add a user to one or more access groups.

Access to root objects are controlled through the use of domain access groups under domain access group object 502. Members of a domain access group are allowed to read or write root objects depending on the access level granted to the access group by the system administrator. Domain access groups are created and destroyed by the system administrator using domain access group object 502 of access control object 490.

Access to application windows for integration management template 12 and use of the components of these windows are controlled through view access groups of view object 482. Members of a view access group are allowed to enter windows and view or use components depending on the access level granted to the access group by the system administrator. The present invention permits the system administrator to create and destroy view access groups using a view access group definition window associated with view object 482. Component group object
restricts access to components through the creation of component groups. A component group may include a single component or several components that the system administrator desires to apply to common access and restrictions. If a component is not defined as a member of a component group, there is no restriction applied to the component. In the present invention, component groups may be created, modified and deleted from a component group definition window that relates to component group object 492.

The present invention also provides for historically archiving data to save information needed to allow recreation and study of past operating strategies or to review any other saved operations data. The historical archiving system also stores snapshots of selected real-time and calculated values at a user-designated periodicity. The historical portion of integration management template 12 provides a way to incorporate other types of historical data, such as logging of dispatcher actions and application-generated messages. Historically archived data may be used to recreate system conditions in a particular point in time or over a user-specified period. This data is retrieved into a study case that can be treated in the same manner as a "what-if" study case.

The present invention also includes an activity registry view to display all active processes that are
registered with the system. The display shows the request number, the time of registry, the request type (i.e., either simple or periodic), the server object performing the activity. The selector or method of activity. Activities are created or deleted by selecting the activity and the activity status may be automatically updated by selecting the automatic update function from one or more of the integration management template 12 graphical user interfaces.

Although the invention has been described in detail herein with reference to the embodiments, it is to be understood that this description is by way of example only and is not to be construed in a limiting sense. Moreover, while the illustrated embodiment applies to an electric utility, there may be many types of complex physical systems for which the present invention may provide valuable and novel benefits and results. It is to be further understood, therefore, that numerous changes in the details of the embodiments of the invention and additional embodiments of the invention will be apparent to, and may be made by, persons of ordinary skill in the art having reference to this description. It is contemplated that all such changes and additional embodiments are within the spirit and true scope of the invention as claimed below.
WHAT IS CLAIMED IS:

1. A integration template for managing, monitoring and controlling a complex physical system, comprising:
   a centralized database model comprising a plurality of parameters for modeling data and behaviors of the complex physical system;
   a graphical user interface for communicating between a user and said centralized database model;
   a relational database for maintaining and manipulating relationships within said database models;
   a first interface mechanism for receiving data from external sources and relating said data from external sources to said centralized database model; and
   a second interface mechanism for interfacing a plurality of internal application programs, said internal application programs comprising application programs having varied data structures, operational characteristics, and fundamental assumptions.

2. The integration template of Claim 1, further comprising case management instructions for performing case studies of said physical system data and behaviors according to relations of said plurality of parameters using said centralized database model.

3. The integration template of Claim 1, wherein said centralized database model comprises an object-
oriented database model for modeling the complex physical system data and behavior.

4. The integration template of Claim 1, wherein said centralized database model comprises a plurality of parameters for modeling a utility enterprise including a plurality of utility data and behaviors.

5. The integration template of Claim 1, further comprising a case management mechanism for managing a plurality of independent cases relating to said complex physical system, said case management mechanism comprising:
   - at least one case tree for establishing hierarchial relationships concerning said plurality of independent case and stemming from a base case;
   - instructions for storing results relating to operations arising from said independent cases; and
   - instructions for comparing ones of said plurality of independent cases to other ones of said plurality of independent cases.

6. The integration template of Claim 1, further comprising a formatted transfer metastatement function for extracting and formatting information relating to said centralized database model.
7. The integrated template of Claim 1, further comprising an events subsystem associated with said centralized database model for controlling and monitoring events of said physical system data and behavior.

8. The integration template of Claim 1, further comprising an audit trail mechanism for recording modifications to said plurality of parameters of said database model.

9. The integrated template of Claim 1, further comprising an activity registry for registering activity relating to said centralized database model.

10. The integration template of Claim 1, further comprising a security mechanism for maintaining security of said centralized database model.

11. The integrated template of Claim 1, further comprising a security mechanism for maintaining security of said physical system, said security system comprising a plurality of access levels for controlling access to said centralized database model according to a plurality of user privilege levels.
12. A method for integrating the management, monitoring and control of a complex physical system, comprising the steps of modeling data and behaviors of the complex physical system using a centralized database model comprising a plurality of parameters; communicating between a user and said centralized database model using a graphical user interface; receiving data from external sources and relating said data from external sources to said centralized database model; and interfacing a plurality of internal application programs, said internal application programs comprising varied application programs having varied data structures, operational characteristics, and fundamental assumptions.

13. The method of Claim 12, further comprising the steps of case studies of said physical system data and behaviors according to relations of said plurality of parameters within said centralized database model.

14. The method of Claim 12, further comprising the step of modeling the complex physical system data and behavior using an object-oriented database model.
15. The method template of Claim 12, wherein said modeling step comprises the step of modeling a utility enterprise including a plurality of utility data and behaviors.

16. The method of Claim 12, further comprising the steps of managing a plurality of independent cases relating to said complex physical system, said steps comprising:
   establishing hierarchial relationships concerning said plurality of independent case and stemming from a base case;
   storing results relating to operations arising from said independent cases; and
   comparing ones of said plurality of independent cases to other ones of said plurality of independent cases.

17. The method of Claim 12, further comprising the step of extracting and formatting information relating to said centralized database model using a formatted transfer metastatement function.

18. The method Claim 12, further comprising the step for controlling and monitoring events of said physical system data and behavior using events subsystem associated with said centralized database model.
19. The method of Claim 12, further comprising the step of recording modifications to said plurality of parameters of said database model using an audit trail mechanism.

20. The method of Claim 12, further comprising the steps of maintaining security of said physical system and controlling access to said centralized database model according to a plurality of user privilege levels.
FIG. 11

Message Box Titles and Texts

Title
Exit
Help

Text
Help
IMT App Running

One or more IMT applications are still running.
[Exit] anyway?

- copy
- cut
- paste
- hardcopy

- emphasis
- bold
- italic
- underline
- normal
- selectable
- position...
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

IPC 6 G06F 17/60

According to International Patent Classification (IPC) or to both national classification and IPC.

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SCHWARTZ J ET AL: &quot;INTEGRATED DATA MANAGEMENT IN A HETEROGENEOUS CIM ENVIRONMENT&quot; PROCEEDINGS OF THE ANNUAL COMPUTERS IN DESIGN, MANUFACTURING AND PRODUCTION CONFERENCE (COMPEURO), PARIS-EVRY, MAY 24 - 27, 1993, no. -, 24 May 1993, CROISIER A; ISRAEL M; CHAVAND F, pages 248-257, XP000463876 see the whole document --- -/- -</td>
<td></td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

Date of the actual completion of the international search

4 September 1997

Date of mailing of the international search report

16.09.97

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Wassenaar
Tgl. (+ 31-70) 340-2040, Tx. 31 651 epo nl,
Fax (+ 31-70) 340-3016

Authorized officer

Fournier, C
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>GRAEFE U ET AL: &quot;An enterprise model as a design tool for information infrastructure&quot; JSPE/IFIP TC5/WG5.3 WORKSHOP ON THE DESIGN OF INFORMATION INFRASTRUCTURE SYSTEMS FOR MANUFACTURING, DIISM '93, TOKYO, JAPAN, 8-10 NOV. 1993, vol. B-14, ISSN 0926-5481, IFIP TRANSACTIONS B (APPLICATIONS IN TECHNOLOGY), 1993, NETHERLANDS, pages 183-192, XP0002039651 see page 184, paragraph 2 - page 186, paragraph 4 ---</td>
<td>1-4, 12-15</td>
</tr>
<tr>
<td>A</td>
<td>RAFIUL AHAD ET AL: &quot;HP OPENODB: AN OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEM FOR COMMERCIAL APPLICATIONS&quot; Hewlett-Packard Journal, vol. 44, no. 3, 1 June 1993, pages 20-30, XP0000303911 see page 20, left-hand column, line 1 - page 22, right-hand column, line 53 see page 28, right-hand column, line 4 - page 30, left-hand column, line 1 ---</td>
<td>1,3,6, 10-12, 14,17,20</td>
</tr>
<tr>
<td>A</td>
<td>Dampney C N G ET AL: &quot;Semantic correspondence in integrating CASE tool repository schemas&quot; Information and Software Technology, 1994, UK, vol. 36, no. 2, ISSN 0950-5849, pages 87-96, XP0002039652 see page 88, left-hand column, line 19 - page 90, right-hand column, line 5 ---</td>
<td>1,3,12, 14</td>
</tr>
<tr>
<td>A</td>
<td>YU D C ET AL: &quot;A GRAPHICAL USER INTERFACE FOR DESIGN, SIMULATION AND ANALYSIS OF POWER PLANT ELECTRICAL AUXILIARY SYSTEMS&quot; IEEE TRANSACTIONS ON ENERGY CONVERSION, vol. 9, no. 2, 1 June 1994, pages 263-269, XP0000458865 ---</td>
<td></td>
</tr>
</tbody>
</table>