Integration of Design Modeling Techniques: Operations Automation Systems Scenario

William W. Owen
CEGELEC ESCA
Tucson, Arizona 85727, USA
wwo@esca.com

Jerzy W. Rozenblit
Dept. of Computer and Electrical Engineering
The University of Arizona
Tucson, Arizona 85721, USA
jr@ece.arizona.edu

Abstract

This paper presents an application scenario in which knowledge-based and object oriented modeling techniques are applied and used in the analysis and design of complex, computer-based systems. The methodology is presented in the context of an operations automation and information management system in the electric utility operations problem domain. This system is referred to as OAS, for Operations Automation System in the remainder of this paper.

The analysis and design methodology utilizes several techniques to analyze different aspects of the system in relative isolation. System Entity Specifications (SES) are used to decompose the system and to classify its components. The SES also serves as an anchor for linking the other models for traceability purposes.

Use Case scenarios are used to model the system requirements. Object Modeling methods provide a rich mechanism for specifying the attributes and behavior of the system components. Dynamic modeling techniques, presented through the use of interaction diagrams, are used to model the dynamic behavior of system components and assist in the detailed system specification.

The system presented in this paper is treated at a high level. The methodology supports iterative design and development so that the high-level design can be repeatedly refined, prototyped and tested until a suitably detailed design is produced.

Introduction and Background

The electric operations control room is an office, typically manned 24 hours a day by at least one electric system operator. The operator's fundamental job is to ensure that customers are supplied with electric power 24 hours a day, 365 days a year with minimal interruptions.

Customers expect to be able to consume power at any time they wish, in any quantity, limited only by the capacity of the equipment and protective devices serving their residence or commercial location. The power quality (voltage and frequency characteristic) is expected to be high at all times.

A major component of the operator's job is the management and direction of field crews in a variety of system activities. These activities include the installation of field equipment, routine maintenance of feeder equipment, and emergency restoration of power.

The safety of field crews is the chief concern in this type of activity. For this reason, elaborate procedures, (typically paper and pencil based), have been developed to ensure equipment is in the proper state before construction, maintenance or re-energization activities are executed.

The physical electrical system for most utilities is quite large. Major utilities organize their service territory into divisions. Divisions are often further subdivided into districts. The typical district contains between 20 and 100 distribution substations. Each substation provides power to between three and eight feeder circuits. The each feeder circuit delivers power to as many as 1500 of the utility's customers.

While there is usually one control room per district for daytime operation, it is common for a utility to consolidate operations and control into a single office responsible for providing "after hours" service for several districts. Control authority can be re-distributed at any time if there is a storm or other event that requires more operators.

The following list summarizes the typical tasks required of the electric system operators [1]:

- Know the current state of the system and have a good understanding of what states may occur based on system load, weather, date and time.
- Keep system voltage and loading within limits.
- Monitor total system load, and in some cases shed load if it exceeds limits (for economic or system

0-8186-7355-9/96 $05.00 © 1996 IEEE

security reasons).

- Plan, document, and supervise pre-planned switching operations.
- Analyze and respond to emergency trouble calls.
- Document all changes to the electrical system.

Over the past decade, a variety of computer-based information systems have been added to the electric operations control room to augment the existing paper-based systems. These systems typically operate independently, as "islands of automation." Examples of these systems and tools are summarized below:

- Paper maps provide a detailed geographic and schematic view of the electrical system. These maps are often tiled together on the walls of the control room to provide a contiguous view of the entire service area.
- Telephone and radio communication equipment are used for communicating with field crews as well as with operators in other locations.
- One or more Supervisory Control and Data Acquisition (SCADA) systems provide real-time information regarding the state of the electrical system. Devices that are equipped with sensors report their current conditions back to the SCADA systems, while devices with automatic control capability can be remotely actuated by the SCADA systems. SCADA systems include a graphical user interface that typically presents schematic and form-based drawings.
- A mainframe-based customer information system provides information regarding individual customers, including their power consumption history, their payment history, and in some cases their physical and electrical location in the system.
- Geographic information systems provide a wealth of data, including very detailed information about each physical component in the system as well as detailed geographic maps for the utility's service territory. A graphical user interface is used to present the maps.
- Other systems such as trouble call reporting and analysis, field crew tracking, and weather information are additional tools that aid the operator monitoring and maintaining the system. Each typically has its own user interface with geographic or schematic displays of the utility's service territory.

It is the operator's job to acquire and assimilate information from all sources in order to carry out his job. As stated above, similar information is presented in several locations and in similar formats. What is needed is a mechanism to aid the operator in acquiring, integrating, and utilizing information from these disparate systems. Furthermore, automation of manual and time critical tasks is a high priority function.

The OAS is a real-time system that interfaces with other information sources, acquiring data of interest, incorporating that data into a model of the physical system, and presenting the information to the user in a consistent, intuitive manner. Applications that control and analyze the system and automate user activities are layered on the model infrastructure.

Requirements - Use Case Descriptions

Requirements modeling is arguably the most important step of system design. It is critical that requirements be understandable, clearly documented, and easily traceable through the steps of the design and implementation process. Jacobson [2] promotes Use Case scenarios as the vehicle for documenting all system requirements.

Fundamentally, use cases are a description of each function the system provides to its users. Like the other components of the design process, use cases can be made more detailed and refined in an iterative process.

High level use cases for the OAS include activities like device control, area of responsibility management, alarm management, and diagram navigation. Note that in later sections, these sample use cases are expanded and their relationships to other system entities are discussed.

Use cases are linked to System Entity Structures, described below, so that the requirements can be traced through the design and implementation process. This helps to ensure that all system requirements are considered in the system design and implementation processes.

System Model

In this section a knowledge representation called System Entity Structure (SES) [3, 4] is presented for the Operations Automation System. The three primary aspects of the OAS SES are described. These are the hardware architecture, communications architecture, and the software architecture. The software architecture is expanded to show decomposition and specialization of the system models, views and controls. Additionally, an object model [5, 8] for a portion of the system's software architecture is presented. This is used to better show the relationships between software entities in the system.

Both the System Entity and Object Modeling descriptions reflect the hierarchical, modular manner in which the system is constructed. Typically, low-level components are designed, implemented, and tested.

Higher level components are then designed, implemented and tested using low-level components. This process continues until the design and construction are complete. Booch refers to this method as the "round trip gestalt" approach [8]: the design is essentially top down, while construction is bottom up, but the overall process is iterative between phases of analysis, design, construction, and test.

System entity structure is a mechanism for modeling the entities that are used to construct a complex system [3, 4, 6]. The two fundamental relations in SES are decomposition and specialization. In SES diagrams, decompositions are typically called aspects and denoted with a single vertical line. Specializations are denoted with dual vertical lines.

Figure 1, below, shows the top level SES for the Operations Automation System. The system is decomposed into three fundamental aspects: the hardware architecture, software architecture, and communications architecture. Each of these architectures is further refined in the actual design process. In the next section, further analysis of the software architecture is presented.

Note that the next level of analysis for the hardware architecture is a specialization, meaning that several types of hardware architecture are available to support the OAS design (distributed, centralized, hybrid). Each of these is then decomposed further in the analysis process.

Also note that each object represented in the SES can have attached attribute variables that further describe its properties. This feature is used to link the various artifacts of the analysis and design process for requirements traceability purposes. Each object in the SES contains references to one or more Use Case descriptions identifying all requirements directly influenced by this object. Each object also contains a reference to a class representation of one or more object diagrams, providing a cross reference between the two models. Where appropriate, the object also contains references to any interaction diagrams it participates in. In this way the SES serves as the anchor tying together use case descriptions, object diagrams, and interaction diagrams.

We feel that in order to facilitate the evaluation of the overall design and to ensure complete coverage of requirements, fundamental software tools are needed. These tools are used to check that each use case is linked to an object modeled in the SES. Correspondences between SES objects, object models and interaction diagrams are also evaluated and reported.

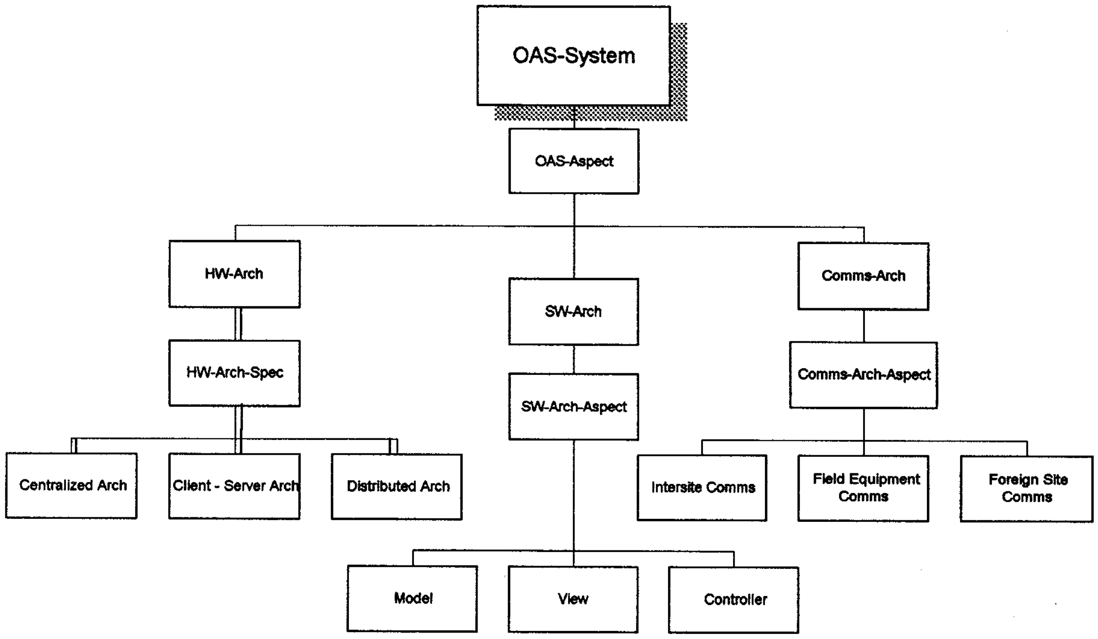

Figure 1: Operations Automation System top level SES

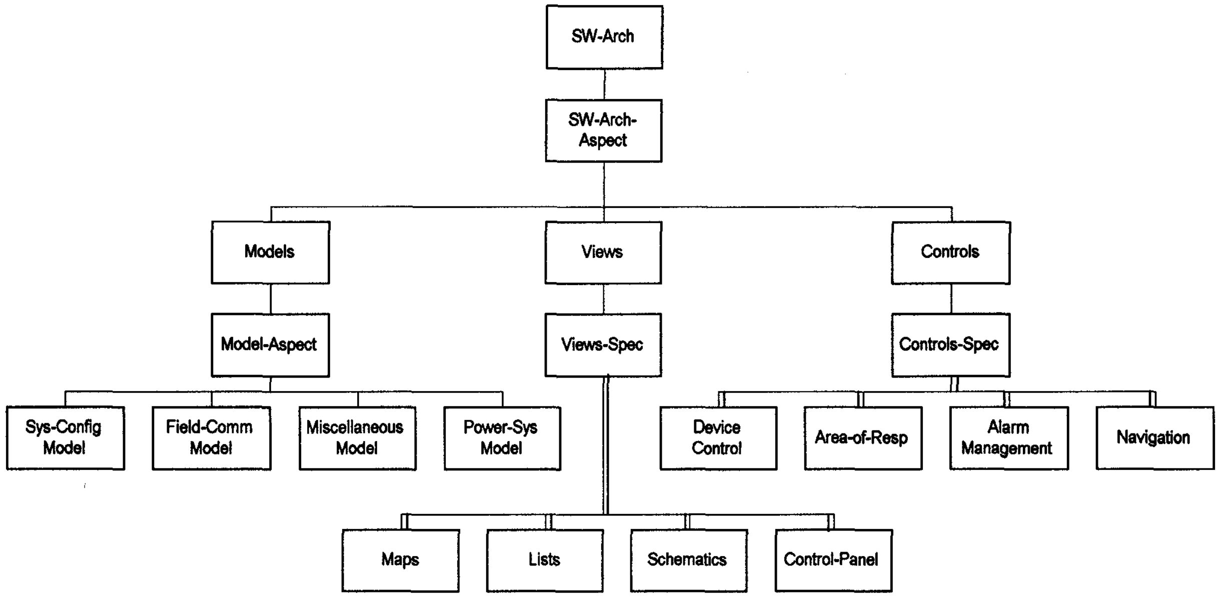

Figure 2: Operations Automation System Software Architecture SES

Software Architecture

The software architecture, shown in Figure 2, is based on a software design structure known as Model-View-Controller [7]. The idea is to decompose the software system into:

- model objects, which represent the real world objects the software system is meant to manipulate,
- view objects, which are the mechanisms used to visualize the models, and
- control objects, which allow users or other systems to interact with, or influence the behavior of the system. Controls, in many ways, model the behavior of the system by relating model objects together to accomplish a particular function.

Models. The overall system model consists of the system configuration model, field communication model, power system model and miscellaneous models. The System Configuration model represents the hardware and software entities that make up the OAS itself. Configuration management software interacts with these model components to monitor and control the state of the OAS so that redundant and non-redundant entities are properly managed under normal and failure modes of operation.

The Field Communication model represents the entities that make up communication circuits from a data acquisition processor to the field devices being monitored and controlled. Data acquisition software interfaces with these model components to monitor and control the state of field communication system.

The Miscellaneous models represent the software and physical entities used by various system functions. For example, access rights and access controls are defined as part of the Access Control function. Access Rights define the permissions given to a user or system making requests of the system. Access Controls define the privileges given each every modeled object in the system.

The Power System model represents the physical power system that is being monitored and controlled. The power system model is the heart of the OAS in that all of the end-user functionality is built upon this. The power system model holds the current state of the system and makes it accessible to the user interface (views), and to other system functions.

Views. View objects provide the mechanism for users of the system to visualize the states stored in the model. Most views reflect the state of the power system model, including historical, current, and planned states. Additional views are necessary to present the state of the OAS configuration itself as well as the OAS communication architecture.

The primary views required in the system include:

- Maps, providing a geographic view of the electrical system components as well as important non-electrical features. Map views are meant to replace or augment the paper wall board in the operations control room.
- Schematics, providing a more structured (non-geographic) view of system components. Schematic views are used to present dense areas of the power system not suited for a map view (i.e., the electrical system in a congested urban area).
- Lists, providing a tabular view of a group of model objects. Lists can be filtered in a variety of ways to show more specific groupings of model objects.
- Control Panel, providing a high-level view of the state of the OAS itself, as well as mechanisms to control the state of the OAS.

Controls. Control objects are used to model the inputs to a system, whether they are from users, sub-functions of the same system or entirely different systems. Control objects encapsulate the logic and business rules that correlate model objects to accomplish a desired function; in other words, the system behavior.

From a system maintenance standpoint, control objects are the most dynamic part of the system. As the business environment and computing requirements for a electric utility evolve, the control objects are the area of the system that must be modified.

Example control objects in the OAS are listed below.

- Area of Responsibility Control objects are used to encode behavior of all system objects relative to an input's access rights. For every system input, the requester's access rights are compared against the rights of the effected object. Based on this, the input request is either permitted or refused.
- Device Control objects are used to encode the behavior of all controllable power system devices in the system. Device Control objects provide control pre-check logic before executing the control. Logging of the control request is also provided by the device control object. Device controls make use of access rights checking from an associated area of responsibility control object.
- Alarm Management Control objects encode the behavior of the alarm subsystem. Inputs are from other subsystems that have detected an abnormal condition and from users of the system.
- Navigation Control objects encode the behavior of view objects in response to navigation requests. For example, when an alarm object is selected for navigation from an alarm list view, the map location showing the device in an alarm state is be presented.

There are many more control objects in the OAS than have been mentioned here. This portion of the software subsystem embodies essentially all system behavior. A more detailed analysis and presentation of these is beyond the scope of this paper.

Software Object Model

As mentioned above, the OAS software architecture is based on a software design structure known as Model-View-Controller. This section uses one of the control objects discussed above to illustrate the relationships between objects that cannot be easily shown in SES diagrams. This highlights the value of object modeling techniques in expressing relationships between objects or classes of objects.

Overview. The "Booch Notation" [8] is used in the class diagrams presented in this section. Figure 3 presents the key constructs of Booch Notation.

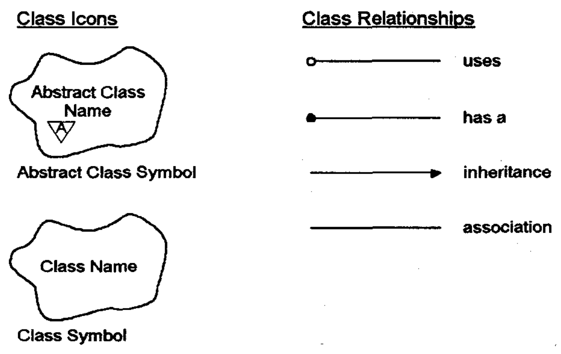

Figure 3: Booch Notation Overview

A diagram showing the Model-View-Controller structure is presented in Figure 4. Note that each of these classes is designated as "abstract," meaning the class itself is never instantiated; only classes that are descendants of the abstract class are instantiated.

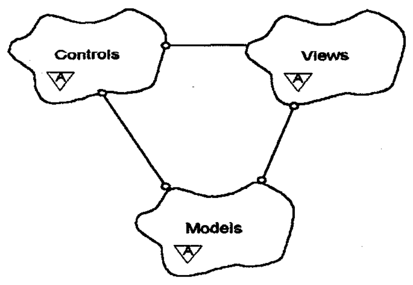

Figure 4: Model-View-Controller Structure

Controls Class Hierarchy. A more detailed illustration of the Controls class hierarchy is shown in Figure 5. The four classes shown all inherit from the root Controls class. Note that the "device-control" and the "alarm-management" classes both have "uses" relationships with the "area-of-resp" class. Each of these use the services of the area-of-resp class in providing services to their own clients.

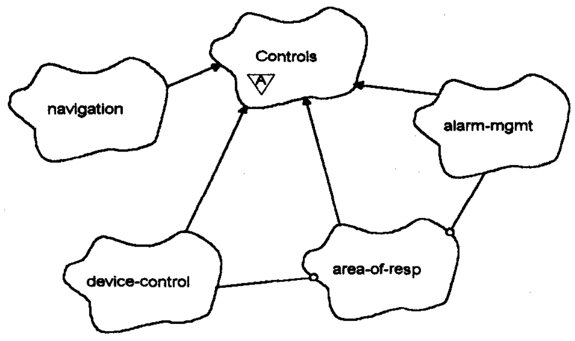

Figure 5: Controls Class Hierarchy

Area of Responsibility. The area of responsibility control class diagram is show in Figure 6. The "requester" class represents any client requesting a service from the system. For example the requester could be an individual user requesting to issue a control, place a tag, acknowledge an alarm or navigate to a region of a map. The requester could also be a foreign system making a request for selected real-time data.

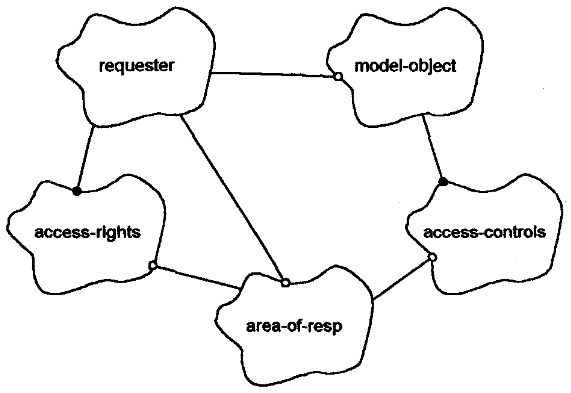

Figure 6: "Area of Responsibility" Class Diagram

Every valid requester in the system has an "access-rights" object associated with it. The access-rights object names the privileges granted to the requester.

The "model-object" class represents any of the model objects in the system. A model-object could be a power system device, another processor or a user definition. Each model object in the system has an "access-controls" object associated with it. The access-controls object names the privileges or operations that can be carried out on this model object. It also defines the details of each privilege. For example, the "operator" privilege may allow control of a device, while the "engineer" privilege may allow deletion and creation of new objects in the system configuration model.

System Dynamic Models

Dynamic models are used to lay out and analyze the time-based behavior of system functions. As with System Model objects, dynamic models can be created in a hierarchical, modular manner. Low-level dynamic models can be constructed and validated. The low-level models are then used to construct higher level dynamic models. This process continues until the behavior of the entire system has been modeled.

Interaction diagrams are used in this section to model system behavior. Jacobson [2] describes interaction diagrams as a mechanism to illustrate how behavior is realized through the interaction of objects or classes of objects.

Area of Responsibility

As described in the previous section, Area of Responsibility control class is used to verify that a requester has sufficient access rights to execute a control action. This process is illustrated in Figure 7, below.

The sequence of events in Area of Responsibility checking is as follows:

0. The sequence is started by a user or subsystem requesting an operation.
1. The requester notifies the target model-object that it intends to operate it. The model-object responds with the identifier of its access-controls object.
2. The requester sends a check_access message to the area-of-resp control object. Included in this message are the identifier of the requester and its access-rights identifier, the identifier of the target model-object and its access-controls identifier, and the operation to be carried out.
3. The area-of-resp control object consults (a) the access-control object and (b) the access-rights object to determine if the requester should be given permission to operate the model-object.
4. A reply message to the requester indicates whether access has been granted or denied for this request. The area-of-resp control object also logs the time and completion status of the access request. (The

alarm-mgmt object receives the log_action message).

5. If the request was refused the sequence is ended and the user or subsystem is informed that the request failed because of insufficient access rights.
6. If the access request was granted, the requester issues the control request to the model.
7. The sequence is ended with feedback to the requester that the request was serviced.

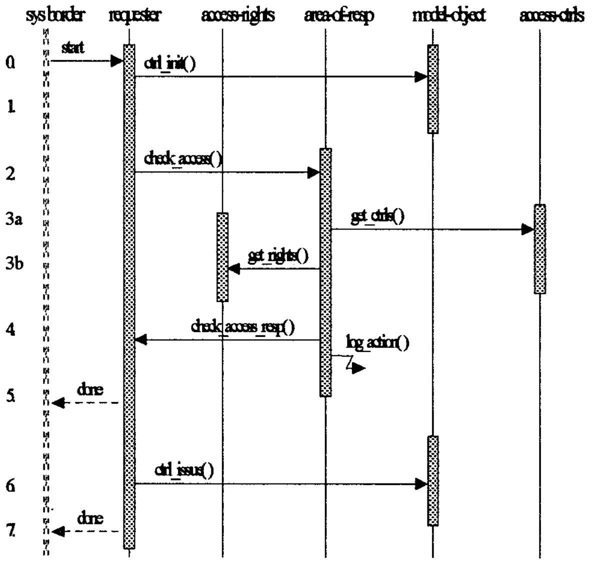

Figure 7: Area of Responsibility Interaction Diagram

Design Evaluation

Evaluation is a critical part of the design process. For large-scale systems such as the OAS, design evaluation is best achieved through a combination of simulation and prototyping. Prototyping is used to help understand requirements and get user feedback early in the process, while simulation is used to investigate performance aspects and verify correct operation under controlled, yet realistic conditions.

Prototyping

Prototypes can be classified as either functional or technical. As the names imply, functional prototypes are used to investigate system functionality, while technical prototypes are used to investigate specific technical issues.

Prototyping is most effective when it is structured to answer specific questions. When those questions have been answered, the prototype software can be retired or used for other prototyping exercises. The temptation to use prototype software to "jump start" the development of the production system must be resisted.

Typically, the customer or end user is very involved in functional prototypes, reviewing the effort and providing feedback to help focus the system requirements. There is less need for or benefit from customer involvement in technical prototypes.

In many cases the language and development environment used for functional prototyping is not the same used for development of the production system. The goal of the prototyping exercise is to generate a mock-up of the system quickly while ignoring many of the requirements for the final system (error handling, message logging, redundancy, performance, etc.).

Technical prototyping is used to evaluate issues in a specific area of the system for technical feasibility. For this type of prototype, it is important that the environment (the computing environment as well as the data and system loading), and tools used be as close to the production environment as possible. (In many ways, this type of prototype is similar to simulation, presented below.)

The evaluation of any third party tool must consider commercial issues in addition to technical issues. Issues to be evaluated include the cost of using the product (licensing expense, maintenance expense, training expense, etc.), the financial health of the product developer organization and the long term direction of the product developer organization.

Simulation

While prototyping is used to evaluate portions of the design in relative isolation, simulation is used to evaluate the design at a subsystem or system level. The goal of simulation is to provide a controlled and repeatable environment that generates realistic inputs and feedback to the software being evaluated.

Simulation provides an excellent mechanism for evaluating design trade-offs. Different designs and implementations can be executed in the simulation environment and the results can be compared.

Another feature of simulation is that it provides an excellent test bed for the system. If an adequate simulation environment exists, the system can be evaluated and tested throughout the development cycle and exercised for customer testing. Simulation scenarios can be developed for evaluating and tuning performance under different loading conditions as well as different functional conditions.

In the evaluation of the OAS design the following areas of simulation are necessary:

- Power System Simulation: A realistic simulation of the electrical characteristics of the monitored system. The power system simulation is used to evaluate functional aspects of the system in a realistic environment. The power system simulation is also used to generate all or part of the input for performance evaluations.
- RTU Simulation: This is used to provide realistic input to the telemetry system. The RTU simulation can be used to augment the power system simulation to provide inputs for system performance evaluation.
- Communication Network Simulation: This is used to simulate the network connections between sites and between user interface devices and server machines under different bandwidth and loading scenarios.

Conclusions

This paper presented an example of the knowledge-based design process in the design of an Operations Automation System. Use Case Scenarios were discussed for requirements modeling. System Entity Structure and Object Modeling techniques were used to structure the information describing the problem domain and the envisioned system. Dynamic models, based on Interaction Diagrams, were developed for a selected function of the system. Finally, the use of prototyping and simulation in design and system evaluation was discussed.

The integration of these different analysis and design techniques provides a methodology for effectively managing the design, development and operation through the entire system life cycle.

In the last few years, object-oriented techniques (OO) have been increasingly used in the design and modeling of complex, large scale systems. The OO paradigm is no longer applied primarily as a software development and implementation methodology [9]. In our approach, rather than to rely on a single, exclusive modeling technique, we reconcile various methods, concepts, and techniques. Although the concepts are different from a formal perspective, they are unified in that they operate on common domain entities — representing the system's objects — and are applied in both the structural or behavioral perspectives. The engineering process that employs these concepts in such perspectives leads to the overall, final system design.

Our choice of the knowledge representation (SES) allows us to capture the typical relationships necessary to construct object models. Interaction Diagrams have sufficient efficacy to support behavior modeling. Then, the use of prototyping and simulation is the basis for the system's model validation and testing.

Our initial experience using an integration of design modeling techniques has been positive. Whereas any single design methodology has inherent limitations, the integrated approach builds on the strengths of each to produce a solid overall design product. We believe this approach will be useful in the design of complex, large scale systems in the future.

References

1. R. Hoffman, "Effective User Interfaces for Distribution Management Systems," *DA/DSM Conference Proceedings, Palm Springs, CA 1995.*
2. I. Jacobson et al., *Object-Oriented Software Engineering - A Use Case Driven Approach.* ACM Press, Addison-Wesley Publishing Co., Wokingham, England, 1992.
3. C. Schaffer and J.W. Rozenblit, "Modular, Hierarchical Concepts for Support of Heterogeneous Systems Design," *Proceedings of the 1995 IEEE Symposium and Workshop on Systems Engineering of Computer Based Systems, 200-208, IEEE Cat. Number 95TH8053, 1995.*
4. J.W. Rozenblit and J.F. Hu, "Integrated Knowledge Representation and Management in Simulation Based Design Generation," *IMACS Journal of Mathematics and Computers in Simulation, 34(3-4), 262-282, 1992.*
5. J. Britton, "An Open, Object-Based Model As The Basis Of An Architecture For Distribution Control Centers," *1992 Winter Power Meeting Paper Number 92 WM 184-2 PWRS.*
6. B.P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, Inc., New York, 1984.
7. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software,* Addison-Wesley Publishing Co., Reading, MA, 1994.
8. G. Booch, *Object-Oriented Analysis and Design with Applications, Second Edition.* Benjamin Cummings Publishing Company, Inc., Santa Clara, CA, 1994.
9. J. Rumbaugh et al., *Object-Oriented Modeling and Design,* Prentice-Hall Publishing Co., 1991.