UML design patterns in a Smart Grid

Prakash Ranganathan
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND, 58201
prakashranganathan@mail.und.edu

Kenneth Magel
Department of Computer Science
North Dakota State University
Fargo, ND, 58102
Kenneth.magel@ndsu.edu

Abstract

The work is focused on modeling Smart Grid functional requirements using use case descriptors and UML class diagrams. The design patterns help identify appropriate agents in the event of any outage in a microgrid. The paper presents selective UML diagrams for different layers in a multi agent design for a power system called UML-MAPS. The architecture is cost effective, easy to track, manage and document, and is highly beneficial in modeling grid management.

Index Terms. Smart Grid, Use cases and classes,; design patterns

1. INTRODUCTION

The electric grid infrastructure in the United States, including transmission, distribution and grid management has become obsolete, complex and prone to outages and failures. Automating software control processes is vital to improving the grid. The current US electric grid structure is semi-automatic in that most control is manual [1,3]. Many questions arise concerning modernizing the grid, including: How much energy can “smart meters” save? What problems can a common communications platform prevent? How can wind and solar renewable resources best be integrated? Although technological innovations are important, most of the inefficiencies have to do with the grid maintenance and operational processes.

In recent years, equipment failures, sub-system aging or employee miscommunication have been responsible for the catastrophic loss of transformers. Efforts are in progress by both industries and academic research communities to develop automatic control processes to modernize the grid; prevent outages; meet adequate energy demand to offset load fluctuations; and to avoid serious threats to the electric grid infrastructure. Data from the North American Electric Reliability Council (NERC) and analyses from the Electric Power Research Institute (EPRI), show that outages from 1984 to the present have affected nearly 700,000 customers annually [1]. Smaller outages occur much more frequently and also affect customers, and larger outages typically occur every two to nine years and affect millions. Preventing these outages is challenging, and much of the attention must be focused on grid automation. Examples include automating the identifying of changes in consumer demand and the protecting of the grid by employing knowledge and decision rules when outages occur. One way to address energy shortages during outage conditions is through the use of Distributed Energy Resources (DERs) that include renewable sources like solar and wind based power.

Power networks are complex and operate in real time with continuously changing or adverse environments, such as poor weather conditions, sudden transformer failures or, malfunctioning of a sub-system of a transmission or distribution network. Hence our motivation to develop software design patterns that can identify the risks, and to document the functional blocks that need attention. We employ a unique and rigorous multiagent design to identify and document the power system risk requirements, knowing that only a handful of limited MAS approaches have been proposed in earlier literatures applicable to power system disturbances [2,11]. We approach the system requirement through use cases and Unified Modeling Language (UML) diagrams. Use cases describe tasks that end users will accomplish using the system, and includes the responses of the system to each user's actions [4]. Team members from affected parts of the system participate. The use cases are then tested, documented and reviewed to ensure all stakeholders needs have been captured [5,6]. The documents directly translate into functional requirements (what a system must do), although more work may be needed to specify nonfunctional ones (what a system must be). Several case studies points out that use case approaches deliver significant cost reductions in the long-run [4,5]. They reduce the risk of rework and provide a path to create documentation that helps to target the best preventive strategies, vendors and equipment [4,5]. Furthermore, they can lead to integrated, open standards-based systems that are easier to maintain. We have developed generic “use cases,” for a microgrid. We argue that our UML design patterns enable us to design a Resource Oriented Agent Architecture (ROAA) in an efficient manner to capture the various process steps from requirements collection, data flow, resource data modeling, to the interface model. The process provides an example of how to design a
ROAA in a standardized way from a modeling perspective and how to link it to an existing Service Oriented Architecture (SOA) environment.

2. UML Diagrams for Smart Grid Processes

The unified modeling language (UML) provides a suite of constructs for modeling the architecture and design of Smart Grid software. We describe our use of these constructs.

2.1 UML Diagrams

The Unified Modeling Language (UML) is a graphical language for creating diagrams that are useful in a software development process. The UML provides a set of standard graphical symbols and rules for combining them. The UML is independent of process and implementation language and provides symbols to support various views of the system [7,8,9]. We apply UML to our Smart Grid application.

UML diagrammatic approaches that can be utilized to model a microgrid include those described below.

- The use-case diagram, which describes the relationships of actors to power system functions
- The class diagram, which points to static class structures within the system
- The object diagram, which relates object instances
- The state diagram, where the states of objects within a particular class are described
- The sequence diagram, which describes the Object message passing structure
- The collaboration diagram, which is similar to the sequence diagram but includes context (i.e., objects and their relationships)
- The activity diagram, which shows sequential flow of activities (i.e., action states)
- The component diagram, which refers to code structure
- The deployment diagram, which maps software to hardware

We utilize the use case and class modeling constructs to represent the hierarchy of functional unit requirements present within the microgrid [8,9]. Figure 1 illustrates a generic use-case diagram of a microgrid with dynamic (i.e., fluctuating) loads that are critical or non-critical and Distributed Energy Resource (DER) units with batteries and circuit breakers (CB).

Within the model, a DER agent is responsible for storing information associated with it, as well as monitoring and controlling DER power levels and its connect/disconnect status. DER information to be stored includes DER identification number, type (e.g., solar cells, microturbines, or fuel cells), a power rating (kW), local fuel availability, a cost function for modeling pricing at which users agree to sell, as well as DER availability, such as the planned maintenance schedule [11]. The circuit breaker (CB) functions as a switch that isolates a portion of the circuit in the event of a power disturbance such as an outage or malfunction of a piece of equipment within the subsystem. There are two modes of operation: the normal and islander. Normal mode refers to regular operation of the electric grid with predicted energy usage of the critical and non-critical loads. The Islander mode refers to preventive control mechanisms that occur, such as isolating a portion of a system in the event of any outage or change in energy demand for critical loads. When the microgrid is operated in island mode, the user and the DER agent balance the demand and supply within the microgrid. The total system load is reduced to its critical loads and the distributed generator produces appropriate power level internally to supply these critical loads. These behavior modes are illustrated in Figure 1.

2.2 UML Relationships

Here we describe a number of UML relationships that apply to modeling in the Smart Grid.

Association. An association is a relationship that specifies that the objects of one class are connected to objects of another class. An association is represented by a solid line connecting the two classes. For example, suppose you have a DER class and a microgrid class. An association is depicted by the following diagram

```
DER       ,       Microgrid
```

The connections between objects are labeled with their multiplicity, such as the asterisk that means “many.” In the example, an instance of the DER class can function for many power systems in the microgrid and an instance of the Microgrid class can have any number of DERs. In our application, connections between objects refer to an inflow of energy data swept up from the DER units by data acquisition processes. The data streams originate at multiple DER units and flow toward the data function units in the microgrid for pricing and data management.
**Aggregation.** In modeling a microgrid, multiple DERs aggregate critical and non-critical loads as shown in the UML diagram below.

The diamond at the end of the connection between the classes indicates that the entire class is precisely comprised of an aggregation of the parts on the opposite end of the connection. We note that an aggregation denotes a whole/part relationship, while an association does not.

**Generalization.** A generalization is a relationship between a general class and a more specific class, with inheritance supported. The diagram below with the open arrowhead shows that a Wind turbine for generating power is a specific instance of a DER. Other DERs could include solar or other types of generated energy. The Wind class inherits all properties of the DER class.

**Dependency.** A dependency is a relationship between two classes in which a change to an independent class causes a change in a dependent class. The diagram below shows a dashed line that specifies the relationship. Here the class Islander mode is dependent on DER agents, since its responsibility is to disconnect and isolate, and the DER is responsible to offset a certain level of energy during the active mode of operation.

**Observers.** Observers are people and agencies who operate within their specific area of interest and have roles for setting up power, instrumentation, and energy management systems in the grid that sense and record measurements of environmental conditions (both internal and external). Examples include substation controllers, DER users, and consumers, all which are actors in the grid.

The same people and utility agencies also typically have roles as consumers of the data produced by the systems they have created. When people and agencies are controlling the operation of the instrumentation systems, they are defined as Observers in the context of this analysis.

### 3. FUNCTIONAL REQUIREMENTS

We capture the functional requirements of the grid system using agents. The behaviors are expressed as services, tasks or functions that the system is required to perform, including those listed below.

- Self healing
- Motivating consumers to actively participate in operations of the grid
- Resisting attacks (cyber, environmental, terror)
- Providing high quality power that will save money wasted from outages
- Accommodating all generation and storage options (e.g., DERs)
- Enabling electricity markets to flourish (e.g., through negotiation and pricing)
- Efficiently run sub-systems
- Enabling high penetration of intermittent power generation sources

The top compartment of a class icon provides the name of the class, the middle contains a list of attributes (member variables), and the bottom contains a list of operations (member functions).

#### 3.1 The UML Multiagent Software System

A multiagent system (MAS) is a powerful tool for developing complex systems that utilize the agent properties of autonomy, sociality, reactivity and pro-activity [10,11]. Figure 2 shows a use case diagram depicting a Smart Grid in a MAS environment. The primary use cases employ agents for the DERs; residential and commercial end users; protection, assessing vulnerability; isolating faults; and generating power. The control station has lead responsibility for communicating and distributing energy to end users. When there is a sudden surge in demand, a demand agent automates a DER agent to seek adequate power levels and to later deactivate during non-peak or no-demand states. The fault isolation agents can deactivate the connections with a portion of the microgrid when a threat is detected, and communicates the information to the involved end users. The vulnerability assessment agent monitors the risk levels associated with any external and internal events.

The UML architecture has three layers: Top, Middle and bottom. The layers are categorized into reactive, collaborative and indicative [1]. We followed the approach discussed in [1][11], but provide a detailed UML representation as opposed to generalized functional modeling.
Figure 2. Smart Grid Use Cases

Figure 3 shows the primary classes for the Smart Grid UML model. The reactive agent layer (bottom) consists of protective agents and generation agents that perform preprogrammed self-healing actions that require an immediate response. These agents get the system input/output status using object-oriented programming methods `input()`, `output()`, which are public by default. Using `input()`, `protect()`, `generate()` and `output()` methods in Layer 1, the grid parameters are sensed and monitored. For ease of exposition, we set all methods as public (+) and members for accessibility except for the model update agent being private (-). Reactive agents, whose goal is independent and prompt control, are available in every local subsystem of the power network. The agents in the middle layer include heuristic knowledge to identify which triggering event from the reactive layer is urgent, i.e., higher priority, important, or resource consuming. The events could be a terror threat to the grid, sudden surge in energy demand, frequency instability during generation or transmission, malfunctioning of a transformer or a meter, unavailability of any DER’s or other resources which can isolate and categorize the various faults into critical and non-critical. Here we use methods `criticalfault()`, `non-critical()`, and action methods `isolate()` to separate the critical from non-critical events and `stability()` to stabilise the frequency of the newly generated energy from a DER. These methods also have timestamping function `gettime()` to keep track of the time of the event.

Finally, the model update agents (with private methods by default) in the top layer check for consistency by updating the system’s real-world facts using the command interpret and alarm filtering agents. They primarily check for the plans (or commands) from the top layer that represent the system’s current status or any presence of disturbance that is detected, using public methods such as `knowledge and decision agents()`, `vulnerability assessment()`, `planning agent()`, `interpret()`, `event identify()` and `restoration agent()`. If the plans do not match the real-world model, the agents in the middle layer trigger the top layer to modify the plans. The top layer consists of knowledge and decision-based agents that have goals and explicit plans that let them achieve their goals in an intelligent manner. The goals of agents in this layer are interoperability, dependability, robustness, and self-healing during these contingency scenarios or events.

![Figure 3. UML design for MAPS](image-url)

3.1.1 Stakeholders

The role of grid management becomes much more challenging when stakeholders are no longer seen as simple objects of managerial action but rather as subjects with their own objectives and purposes. To help understand the stakeholders and their goals, Table 1 and Figure 4 illustrate the stakeholders and their geographical distribution, their primary goals, and the associated Energy Research Power Institute (EPRI) communication and networking standards [13].
Table 1: Goals of Stakeholders in Grid Management

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Goals</th>
</tr>
</thead>
</table>
| Enterprise Agent     | 1. Track protocols, information exchange, and communication standards.  
2. Obtain information about particular computers, users, and assignments through web services.                                            |
| Control Center       | 1. Connects to various substations.  
2. Acquire, monitor and Control all distribution units.  
3. Schedule distribution units.  
4. Generate energy reports or document risk factors for any distribution unit.  
5. Maintain the control interface.                                                                                           |
| Substation           | 1. Perform Asset management of various bulk Distributed energy resources such as Wind, Solar, and Bio mass.  
2. Prepare a cost effective budget for maintaining these resources effectively  
3. Connect and deliver the energy from DERs during Contingency scenarios.                                                    |
| Field Area network   | 1. Provide connectivity to commercial users, local residential end users, DERs, bulk generation users and substations.  
2. Obtain the data from users and quickly connect to them wirelessly.                                                          |
| Commercial users     | 1. Minimize the time and money devoted to maintaining the data  
2. Minimize time and money devoted to maintenance of residential energy usage  
3. Obtain desired on usage, billing, and hourly pricing information during non-peak hours for dynamic pricing.  
4. Direct surplus energy back to the grid.                                                                                 |

Figure 4. EPRI Utility Communication Standards

Note that many of the goals require that the agents support a knowledge base. In addition to those shown in Table 1, there are other important stakeholders, such as substation controllers and the control center itself. Value proposition conflicts can occur. Techniques such as calibrated cost models and simplifier and complicator routines are available to support stakeholders in understanding goal-seeking behaviors that are infeasible with respect to constraints that arise from considerations pertaining to budgets, schedules, and technology [15]. Goal-driven approaches are evolutionary and provide a common language for analysts and stakeholders. The goal orientation, rather than details of specific requirements, allows inter-stakeholder communication using a language based on concepts. Goals are also likely to be more stable than requirements [14].

3.1.2 Risk Factors

Recognizing, documenting, and tracking risks provides a basis for prevention of recurrent problems [12]. Standardized formats for specifying and communicating risks and their status are essential. We adopt a risk template with primary components as shown below.

- Safety
- Security
- Location and Source of the risk
- Frequency
- Potential inimical impact
- Cost
- Personnel
- Probability
- Mitigation procedure

We capture relationships between requirements and development artifacts through tracing procedures. The traceability models have value traces that address risks from changing stakeholder utility values, including market pricing, and supply–demand negotiation procedures. Such changes can happen without changing requirements and can have major effects on overall utility value. Template tracing between
requirements and stakeholder value supports grid management through awareness of the linkages and monitoring for validity. The Grid Risk Tracking Template (GRIT) we employ is illustrated in Table 2. The effort of generating, synchronizing and maintaining GRIT value traces is central to risk management, but can also be beneficial more broadly in management and design decisions.

Table 2. Grid Risk Item tracking template (GRIT)

<table>
<thead>
<tr>
<th>ID: &lt;sequence number&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date opened: &lt;date the risk was identified, specify which distribution unit failed&gt;</td>
</tr>
<tr>
<td>Date Closed: &lt;date the risk was closed out&gt;</td>
</tr>
<tr>
<td>Description: &lt;description of the risk in form of “condition-sequence”&gt;</td>
</tr>
<tr>
<td>Location: &lt;identify the location of risk in the grid&gt;</td>
</tr>
<tr>
<td>Source: &lt;identify the source of the risk&gt;</td>
</tr>
<tr>
<td>Probability: &lt;the likelihood of this risk becoming a problem, example or are there any possibility of cascading phenomena of this risk to a failure of sub system&gt;</td>
</tr>
<tr>
<td>Impact: &lt;the potential damage if the risk does become a serious problem&gt;</td>
</tr>
<tr>
<td>Exposure: &lt;Probability multiplied by Impact&gt;</td>
</tr>
<tr>
<td>Safety: &lt;Any danger to public or property due to the risk&gt;</td>
</tr>
<tr>
<td>Mitigation Plan: &lt;one or more approaches to control, avoid, minimize or otherwise mitigate the risk&gt;</td>
</tr>
<tr>
<td>Owner: &lt;the individual responsible for resolving this task&gt;</td>
</tr>
<tr>
<td>Cost: &lt;cost involved in mitigating this risk&gt;</td>
</tr>
<tr>
<td>Personnel: &lt;number of persons if any needed to mitigate this risk&gt;</td>
</tr>
<tr>
<td>Date Due: &lt;date by which the mitigation actions are to be implemented&gt;</td>
</tr>
</tbody>
</table>

4. CONCLUSION

UML design patterns, such as class diagrams and use cases, have been developed for a Smart Grid application Using design patterns in a multi-agent environment with a layered architecture and a risk documenting procedure efficiently supports tracking of the disturbance and outage events that can happen in the electric grid environment. Given the challenges inherent in managing collections of grid events and scenarios, utilizing UML use case design patterns shortcut is a logical and powerful way to help design and architect a Smart Grid software system.

REFERENCES