UML and Its Use

Alex Apostolov
OMICRON electronics

Presented to the IEEE PES PSRC
13 January 2014 – New Orleans, USA
The heart of object-oriented problem solving is the construction of a model. The model abstracts the essential details of the underlying problem from its usually complicated real world. Several modeling tools are wrapped under the heading of the UML™, which stands for Unified Modeling Language™.
Why Do We Need UML?

STRIFE
As Long as We Have Each Other, We’ll Never Run Out of Problems.
What is UML?

- **UML**: Unified Modeling Language
- An industry-standard graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling.
- The UML uses mostly graphical notations to express the OO analysis and design of software projects.
- Simplifies the complex process of software design
What is UML?

- The Unified Modeling Language was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in the 1990s.
- It was adopted by the Object Management Group (OMG) in 1997, and has been managed by this organization ever since.
- In 2000 the Unified Modeling Language was accepted by the International Organization for Standardization (ISO) as industry standard for modeling software-intensive systems.
- The current version of the UML is 2.4.1 published by the OMG in August 2011.
What can be derived from UML (parallel view CIM-61850)

In case of 61850, each section of SCL schema defines a profile, the union should be in UML.

- **jCleanCim**
  - UML validation
  - UML statistics
  - UML doc gen

- **Profiling tool**
  - Profile definition
  - Profile generation
  - Profile doc gen

- **Product1 / XML Editor**

- **Product2 / XML Editor2**

- **IEC 61850-6**
  - (SCL.xsd configuration)

- **IEC 61850-?**
  - (.?? – on-line exchange, e.g. Web service)

- **Validation tool**

- **Instance**
  - (.icd, .scd, ??)

- **Validation tool**

- **Feed SCL with IEC 61850 data model from UML**

- **Step 1a**
  - Paper publication

- **Step 1b**
  - Web publication, ...

- **Step 2**
Today’s main usage objectives of the IEC 61850 UML model

- **Improve data model quality and reactivity**
- **Auto-generated standard from the UML model**
- **Access through the web of the data model content and structure**
- **Download machine-readable data model**
- **Integrate Tissues**

Today’s main usage objectives of the IEC 61850 UML model:

- Today’s main usage objectives of the IEC 61850 UML model
  - Auto-generated standard from the UML model
  - Access through the web of the data model content and structure
  - Download machine-readable data model
  - Integrate Tissues

Tissue process

IEC Central Office

- HTML
- PDF
- XMI
- UML

Modeling team

UML Model (eap, xmi)

Additional software tool
Ultimate objectives of the UML task force

WEB Application hosted by IEC

Services

View/Browse

User

Download

Verify

Data model (XML)

SCL Files (XML)

SCL verification content for engineering tools (XSD ?)

Tissue process hosting

SCL instance valid Access

Std IEC document (PDF/WORD)

Model Editing, Maintenance, Release, … (UML Tool)

IEC web site

Data model Access

• access

Model managers of various groups

61850 UML model
All namespaces (part 7-xxx, 6-xxx, 90-xxx) including IEC 61400 part
Why UML for Modeling?

- A diagram/picture = thousands words
- Uses graphical notation to communicate more clearly than natural language (imprecise) and code (too detailed).
- Makes it easier for programmers and software architects to communicate.
- Helps acquire an overall view of a system.
- UML is *not* dependent on any one language or technology.
- UML moves us from fragmentation to standardization.
**History**

<table>
<thead>
<tr>
<th>Year</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>UML 2.0</td>
</tr>
<tr>
<td>2001</td>
<td>UML 1.4</td>
</tr>
<tr>
<td>1999</td>
<td>UML 1.3</td>
</tr>
<tr>
<td>1997</td>
<td>UML 1.0, 1.1</td>
</tr>
<tr>
<td>1996</td>
<td>UML 0.9 &amp; 0.91</td>
</tr>
<tr>
<td>1995</td>
<td>Unified Method 0.8</td>
</tr>
</tbody>
</table>

Other methods:
- Booch ‘91
- Booch ‘93
- OMT - 1
- OMT - 2
• The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of different simple or complex systems.

• UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the system.
The UML is applicable to object-oriented problem solving.

A model is an abstraction of the underlying problem.

The domain is the actual world from which the problem comes.

Models consist of objects that interact by sending each other messages.

Objects have things they know (attributes) and things they can do (services).
UML Modeling Diagrams

- Use case diagrams
- Class diagrams
- Object diagrams
- Sequence diagrams
- Collaboration diagrams
- Statechart diagrams
- Activity diagrams
- Component diagrams
- Deployment diagrams
UML: Use Case Diagrams

• Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis is on what a system does rather than how.

• A scenario is an example of what happens when someone interacts with the system.

• An actor is who or what initiates the events involved in that task. Actors are simply roles that people or objects play.

• The connection between actor and use case is a communication association.
• A generalized description of how a system will be used.
• Provides an overview of the intended functionality of the system
Use Case Diagram (core components)

**Actors:** A role that a user plays with respect to the system, including human users and other systems. e.g., inanimate physical objects (e.g. robot); an external system that needs some information from the current system.

**Use case:** A set of scenarios that describing an interaction between a user and a system.
Use Case Diagram (core components)

• A use case is a single unit of meaningful work – transmit, receive, initiate, etc.
• Each Use Case has a description which describes the functionality that will be built in the proposed system.

System boundary: a rectangle diagram representing the boundary between the actors and the system.
Use Case Diagram (core relationship)

**Association**: communication between an actor and a use case; represented by a solid line.

**Generalization**: relationship between one general use case and one specific use case. Represented by a line with a triangular arrow head toward the parent use case, the more general modeling element.
Include: a dotted line labeled <<include>> beginning at base use case and ending with an arrows pointing to the include use case. An “Include” relationship is used to indicate that a particular Use Case must include another use case to perform its function.

<<include>>
or in MS Visio <<uses>>
Use Case Diagram Use

Use case diagram:

Distance line Protection with permissive tele-protection scheme

- Data sampling and filtering
- Data sending
- Data receiving
- Relay decision

Actors:

<table>
<thead>
<tr>
<th>Name</th>
<th>Role description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measuring equipment</td>
<td>Measures current and voltage from protected line</td>
</tr>
<tr>
<td>Comm. I/F -S</td>
<td>Receives data from the local relay and sends the data to the remote end</td>
</tr>
<tr>
<td>Comm. I/F -R</td>
<td>Receives data from the remote end and gives the data to the local relay</td>
</tr>
<tr>
<td>CB</td>
<td>Disconnects the protected line from other system (Circuit Breaker)</td>
</tr>
</tbody>
</table>
## Use Case Diagram Use

### Use case(s):

<table>
<thead>
<tr>
<th>Name</th>
<th>Services or information provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data sampling and filtering</td>
<td>Samples current and voltage data from measuring equipment and filters them</td>
</tr>
<tr>
<td>Data sending</td>
<td>Calculates a distance to the fault using filtered data. When a distance protection detects a forward fault the distance protection sends the permissive signal to Comm. I/F –S (the remote end).</td>
</tr>
<tr>
<td>Data receiving</td>
<td>Receives the permissive signal from Comm. I/F –R (the remote end).</td>
</tr>
<tr>
<td>Relay decision</td>
<td>When the distance protection detects the forward faults and receives permissive signal from remote end, the distance protection issues a trip command to the CB</td>
</tr>
</tbody>
</table>

### Basic flow:

**Data sampling and filtering**

<table>
<thead>
<tr>
<th>Use Case Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Current and Voltage are given to Distance protection by Measuring equipment</td>
</tr>
<tr>
<td>Step 2</td>
<td>Distance Protection samples an analogue value and converts it to digital data</td>
</tr>
<tr>
<td>Step 3</td>
<td>Distance Protection removes any unwanted frequency components from the sampled data using a digital filter</td>
</tr>
</tbody>
</table>
# Use Case Diagram Use

## Data sending

<table>
<thead>
<tr>
<th>Use Case Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Distance Protection stores the filtered instantaneous data</td>
</tr>
<tr>
<td>Step 2</td>
<td>Distance Protection calculates a distance to the fault using filtered data.</td>
</tr>
<tr>
<td>Step 3</td>
<td>When a distance protection detects a forward fault to a predetermined distance, a distance protection sends the permissive signal to Comm. I/F –S (in order to send the data to a remote end relay)</td>
</tr>
<tr>
<td>Step 4</td>
<td>Comm. I/F –S send the information to remote end</td>
</tr>
</tbody>
</table>

## Data receiving

<table>
<thead>
<tr>
<th>Use Case Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Comm. I/F –R receives the data from the remote end</td>
</tr>
<tr>
<td>Step 2</td>
<td>Comm. I/F –R gives the received data to Distance Protection</td>
</tr>
<tr>
<td>Step 3</td>
<td>Distance Protection receives the data</td>
</tr>
</tbody>
</table>

## Relay Decision

<table>
<thead>
<tr>
<th>Use Case Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>When the distance protection detects the forward faults in a predetermined zone, and receives a permissive signal from the remote end, the distance protection issues a trip command to the CB</td>
</tr>
</tbody>
</table>
**UML: Class Diagrams**

- **Class diagram** gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static -- they display what interacts but not what happens when they do interact.

- **UML class notation is a rectangle divided into three parts: class name, attributes, and operations.**

  ![Class Diagram Notation](chart.png)

<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute 1</td>
</tr>
<tr>
<td>Attribute 2</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Attribute n</td>
</tr>
<tr>
<td>Service 1</td>
</tr>
<tr>
<td>Service 2</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Service n</td>
</tr>
</tbody>
</table>
UML: Class Diagrams

- **Association** -- a relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes.

- **Aggregation** -- an association in which one class belongs to a collection. An aggregation has a diamond end pointing to the part containing the whole.

- **Generalization** -- an inheritance link indicating one class is a superclass of the other. A generalization has a triangle pointing to the super-class.
UML: Class Diagrams

- **Multiplicity** of an association end is the number of possible instances of the class associated with a single instance of the other end.

- Multiplicities are single numbers or ranges of numbers.
<table>
<thead>
<tr>
<th>Multiplicities</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..1</td>
<td>0 or 1 instance. The notation n..m indicates n to m instances.</td>
</tr>
<tr>
<td>0..* or *</td>
<td>No limit on the number of instances (including none).</td>
</tr>
<tr>
<td>1</td>
<td>exactly one instance</td>
</tr>
<tr>
<td>1..*</td>
<td>at least one instance</td>
</tr>
</tbody>
</table>
UML: Class Diagrams

Name
ObjectName
ObjectReference

SERVER
1
1..*

LOGICAL-DEVICE
1
1..*

LOGICAL-NODE
1

DATA
1

DataAttribute
1..*

Generalization
Association
Aggregation
Multiplicity
## IEC 61850 Logical Node Class

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Attribute type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LName</td>
<td>ObjectName</td>
<td>Instance name of an instance of LOGICAL-NODE</td>
</tr>
<tr>
<td>LNRef</td>
<td>ObjectReference</td>
<td>Path-name of an instance of LOGICAL-NODE</td>
</tr>
<tr>
<td>Data [1..n]</td>
<td>DATA</td>
<td></td>
</tr>
<tr>
<td>DataSet [0..n]</td>
<td>DATA-SET</td>
<td></td>
</tr>
<tr>
<td>BufferedReportControlBlock [0..n]</td>
<td>BRCB</td>
<td></td>
</tr>
<tr>
<td>UnbufferedReportControlBlock [0..n]</td>
<td>URCB</td>
<td></td>
</tr>
<tr>
<td>LogControlBlock [0..n]</td>
<td>LCB</td>
<td></td>
</tr>
</tbody>
</table>

IF compatible LN class defined in IEC 61850-7-4 equals LLN0

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Attribute type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SettingGroupControlBlock [0..1]</td>
<td>SGCB</td>
<td></td>
</tr>
<tr>
<td>Log [0..1]</td>
<td>LOG</td>
<td></td>
</tr>
<tr>
<td>GOOSEControlBlock [0..n]</td>
<td>GoCB</td>
<td></td>
</tr>
<tr>
<td>GSSEControlBlock [0..n]</td>
<td>GsCB</td>
<td></td>
</tr>
<tr>
<td>MulticastSampledValueControlBlock [0..n]</td>
<td>MSVCB</td>
<td></td>
</tr>
<tr>
<td>UnicastSampledValueControlBlock [0..n]</td>
<td>USVCB</td>
<td></td>
</tr>
</tbody>
</table>

**Services**
- GetLogicalNodeDirectory
- GetAllDataValues

**NOTE 1** IEC 61850-7-4 defines specialized logical node classes – the compatible logical node classes, for example, XCBR representing circuit-breakers.
UML: Object Diagrams

• **Object diagrams** show instances instead of classes.
• They instantiate class diagrams

```
  InstanceName1:Class Name

  InstanceName2:Class Name

  ......       ......       ......

  InstanceNameN:Class Name
```
UML: Sequence Diagrams

- Class and object diagrams are static model views.
- **Interaction diagrams** are dynamic. They describe how objects collaborate.
- A **sequence diagram** is an interaction diagram that details how operations are carried out -- what messages are sent and when.
- Sequence diagrams are organized according to time.

![Sequence Diagram Example](image-url)
UML: Statechart Diagrams

- Objects have behaviors and state. The state of an object depends on its current activity or condition.
- A **statechart diagram** shows the possible states of the object and the transitions that cause a change in state.
- States are rounded rectangles.
- Transitions are arrows from one state to another.
- Events or conditions that trigger transitions are written beside the arrows.
Substation Configuration Language SCL

• Must be capable of describing:
  • A system specification in terms of the single line diagram, and allocation of logical nodes (LN) to parts and equipment of the single line to indicate the needed functionality.
  • Pre-configured IEDs with a fixed number of logical nodes (LNs), but with no binding to a specific process. may only be related to a very general process function part.
  • Pre-configured IEDs with a pre-configured semantic for a process part of a certain structure, for example a double busbar GIS line feeder.
Substation Configuration Language SCL

- Must be capable of describing:
  - Complete process configuration with all IEDs bound to individual process functions and primary equipment, enhanced by the access point connections and possible access paths in subnetworks for all possible clients.
  - As above, but additionally with all predefined associations and client server connections between logical nodes on data level. This is needed if an IED is not capable of dynamically building associations or reporting connections (either on the client or on the server side).
Substation Configuration Language SCL

• Substation section: describes the substation single line diagram, and its binding to logical nodes as well as the placement of logical nodes onto IEDs. Thus also the binding of IEDs to substation parts and substation devices is defined.

• Communication section: describes the communication connections between IEDs in terms of connecting communication links.
Substation Configuration Language SCL

- **IED section**: describes the capabilities (configuration) of one or more IEDs, and the binding to logical nodes on other IEDs.

- **LNType section**: defines which data objects are actually contained within the logical node instances defined for the IEDs.
Objectives
SCL UML Diagram Example: IED
Conclusions

• UML diagrams are widely used in different parts of IEC 61850 to present the abstract models of the substation domain.

• They represent the foundation of the definitions of the object models and services, as well as the Substation Configuration Language and the different types of files used to describe the functional hierarchy of the system and data used for exchange between IEDs and applications.
Conclusions

• Part 6 of the IEC 61850 standard specifies a description language for configurations of electrical substation IEDs – the Substation Configuration Language (SCL), based on UML and XML