UML and Its Use

Alex Apostolov

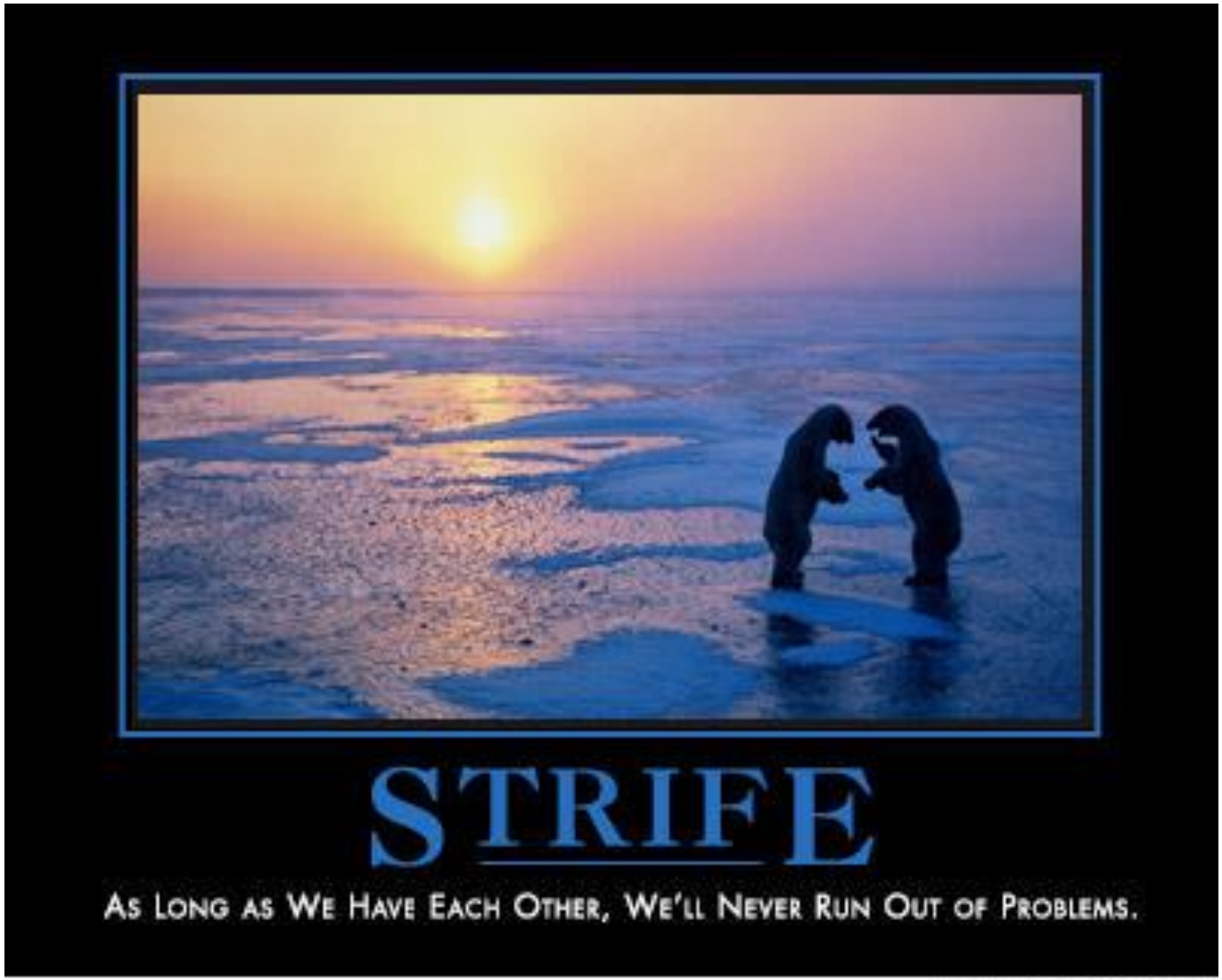
OMICRON electronics

Presented to the IEEE PES PSRC
13 January 2014 – New Orleans, USA

UML

- **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?



What is UML?

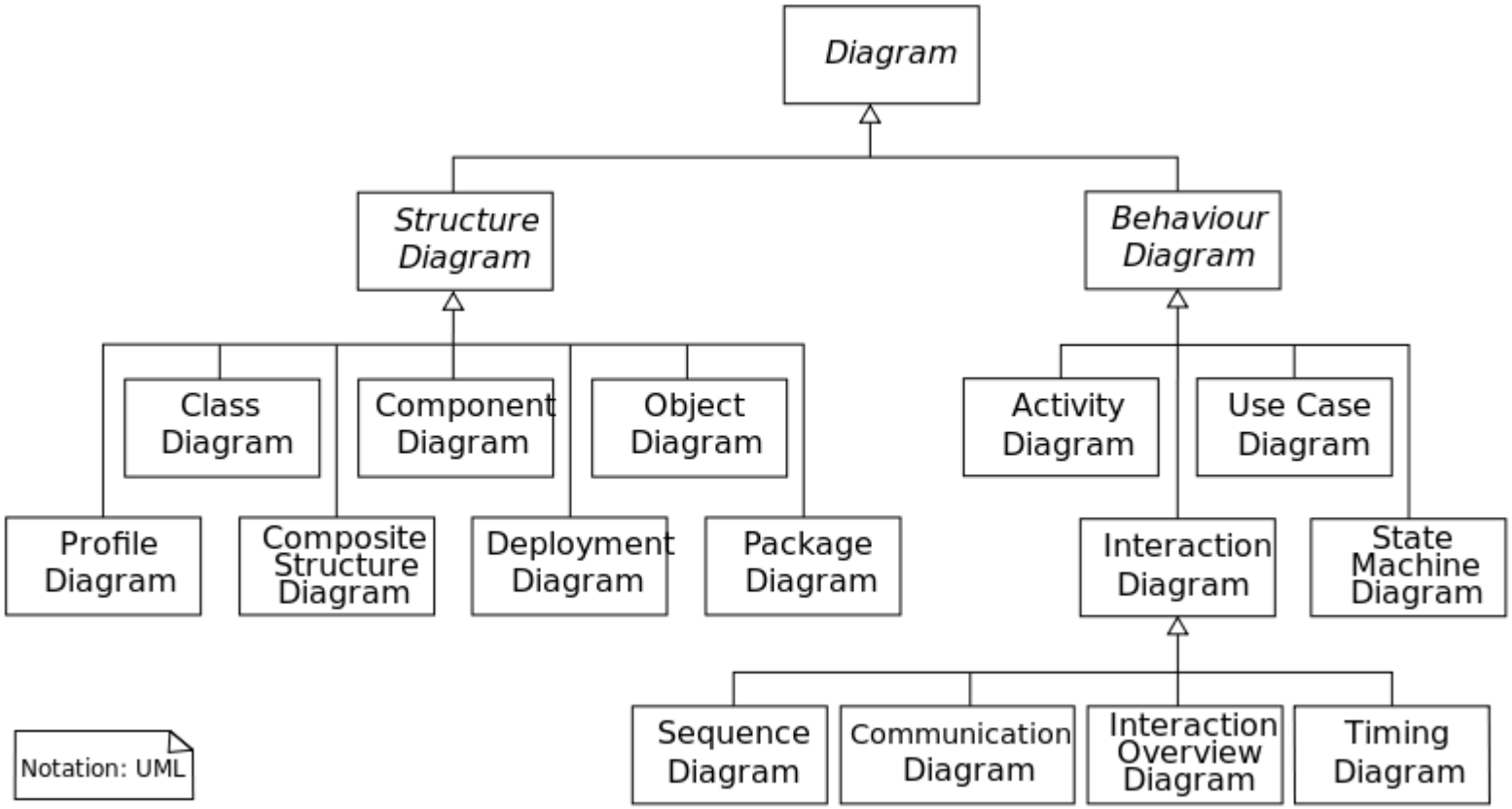
- **UML: Unified Modeling Language**
- **An industry-standard graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling.**
- **The UML uses mostly graphical notations to express the OO analysis and design of software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

What is UML?

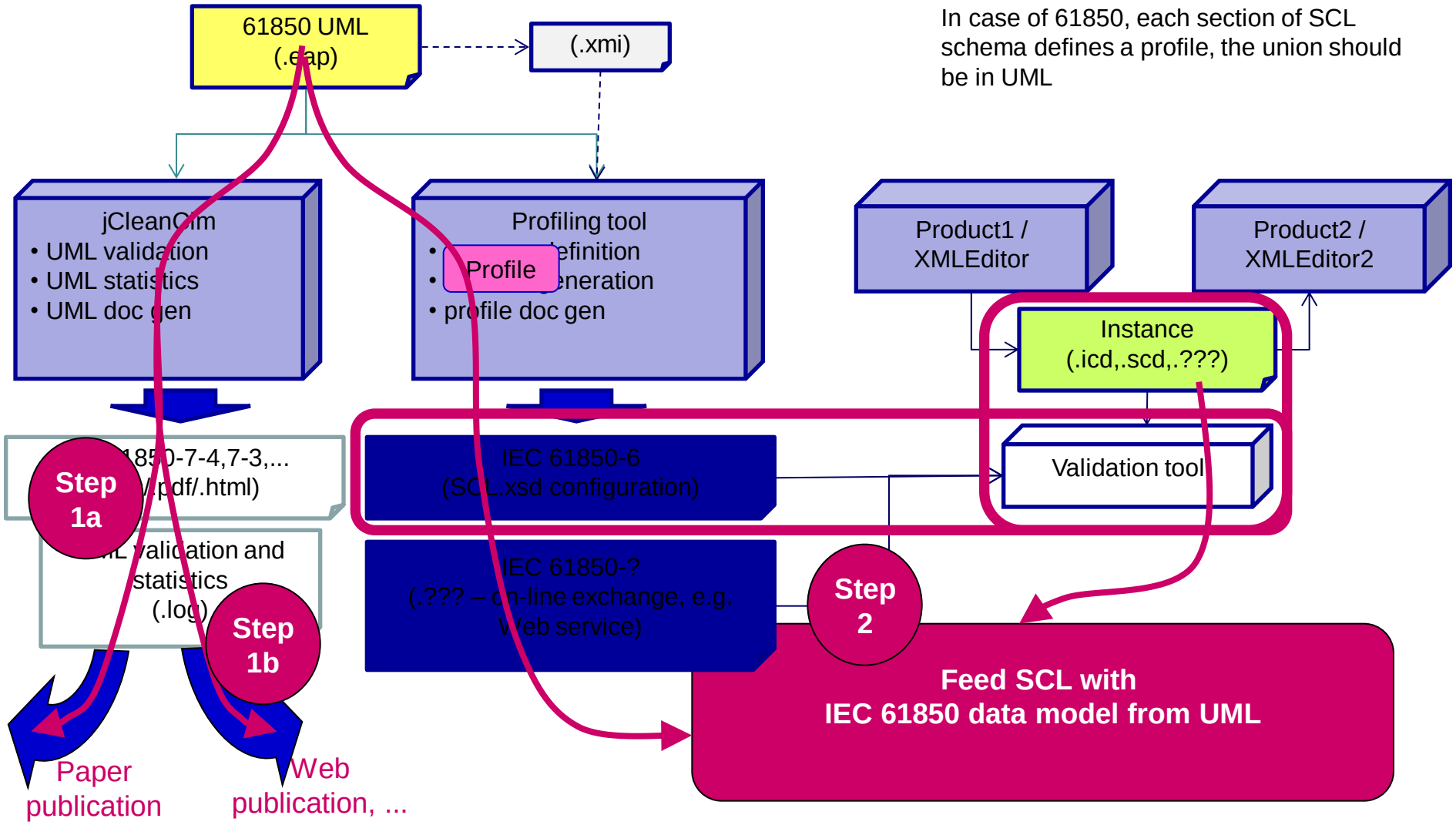
- The Unified Modeling Language was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in the 1990s
- It was adopted by the Object Management Group (OMG) in 1997, and has been managed by this organization ever since.
- In 2000 the Unified Modeling Language was accepted by 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.

UML Logo

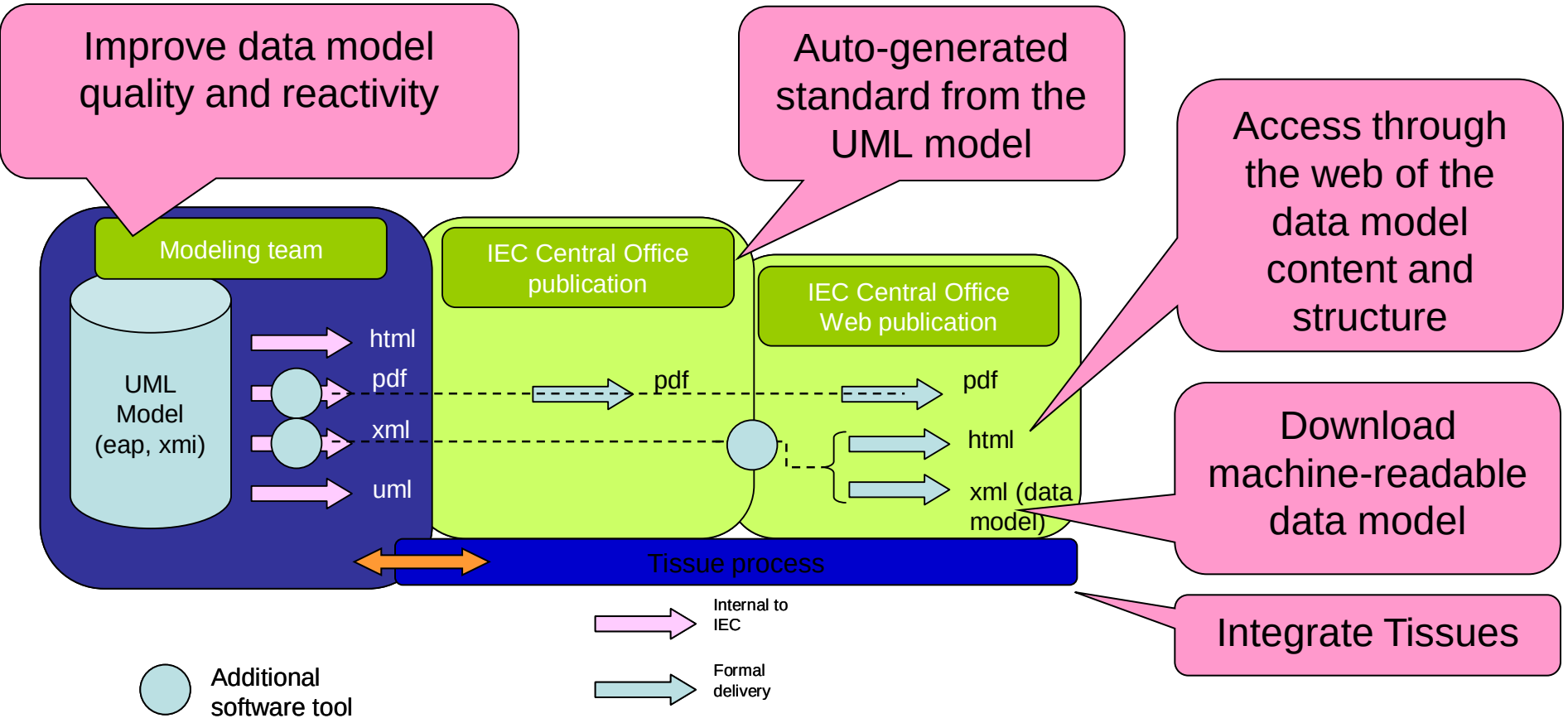
UML Diagrams Overview



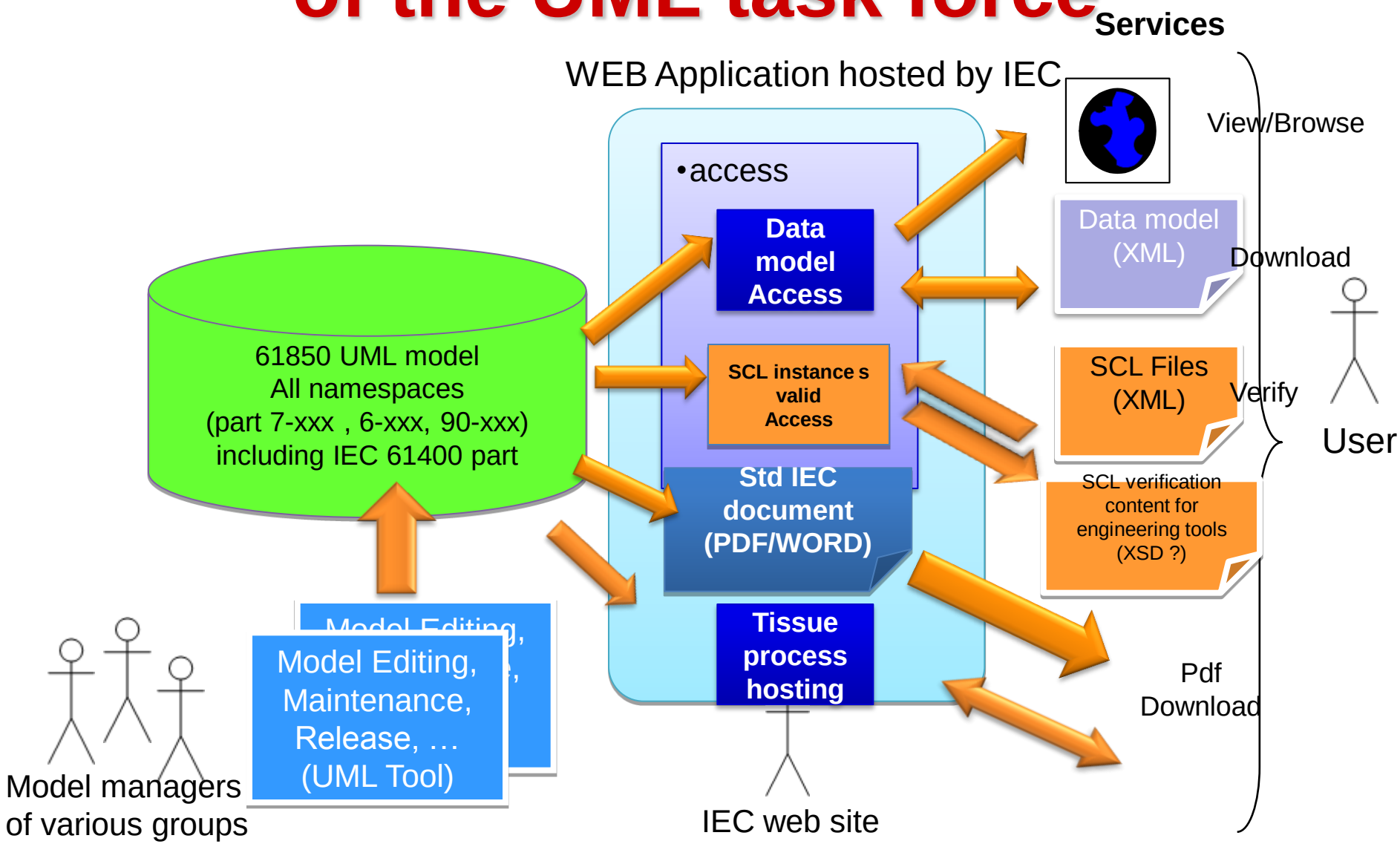
What can be derived from UML (parallel view CIM-61850)



Today's main usage objectives of the IEC 61850 UML model



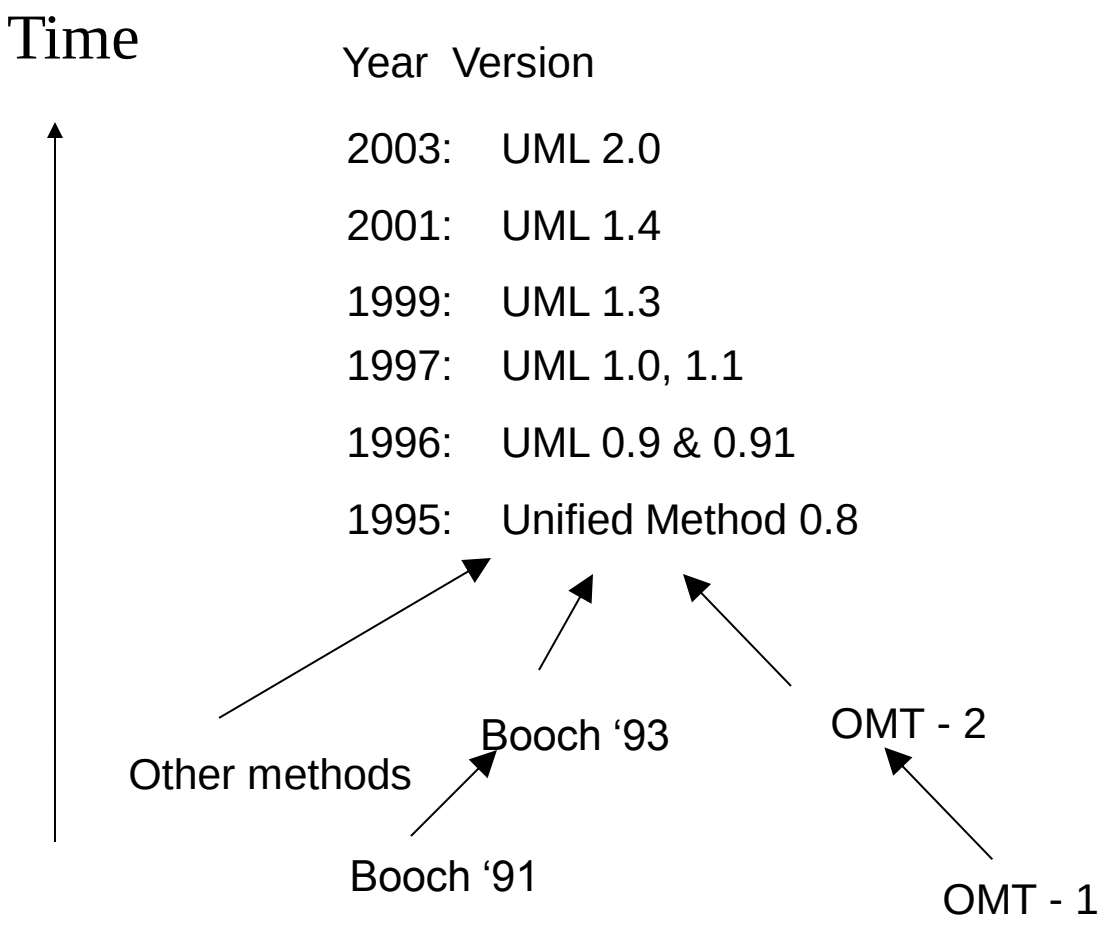
Ultimate objectives of the UML task force



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


Time
Year Version
2003: UML 2.0
2001: UML 1.4
1999: UML 1.3
1997: UML 1.0, 1.1
1996: UML 0.9 & 0.91
1995: Unified Method 0.8
Other methods
Booch '93
OMT - 2
Booch '91
OMT - 1

UML

- **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.**

UML

- 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).

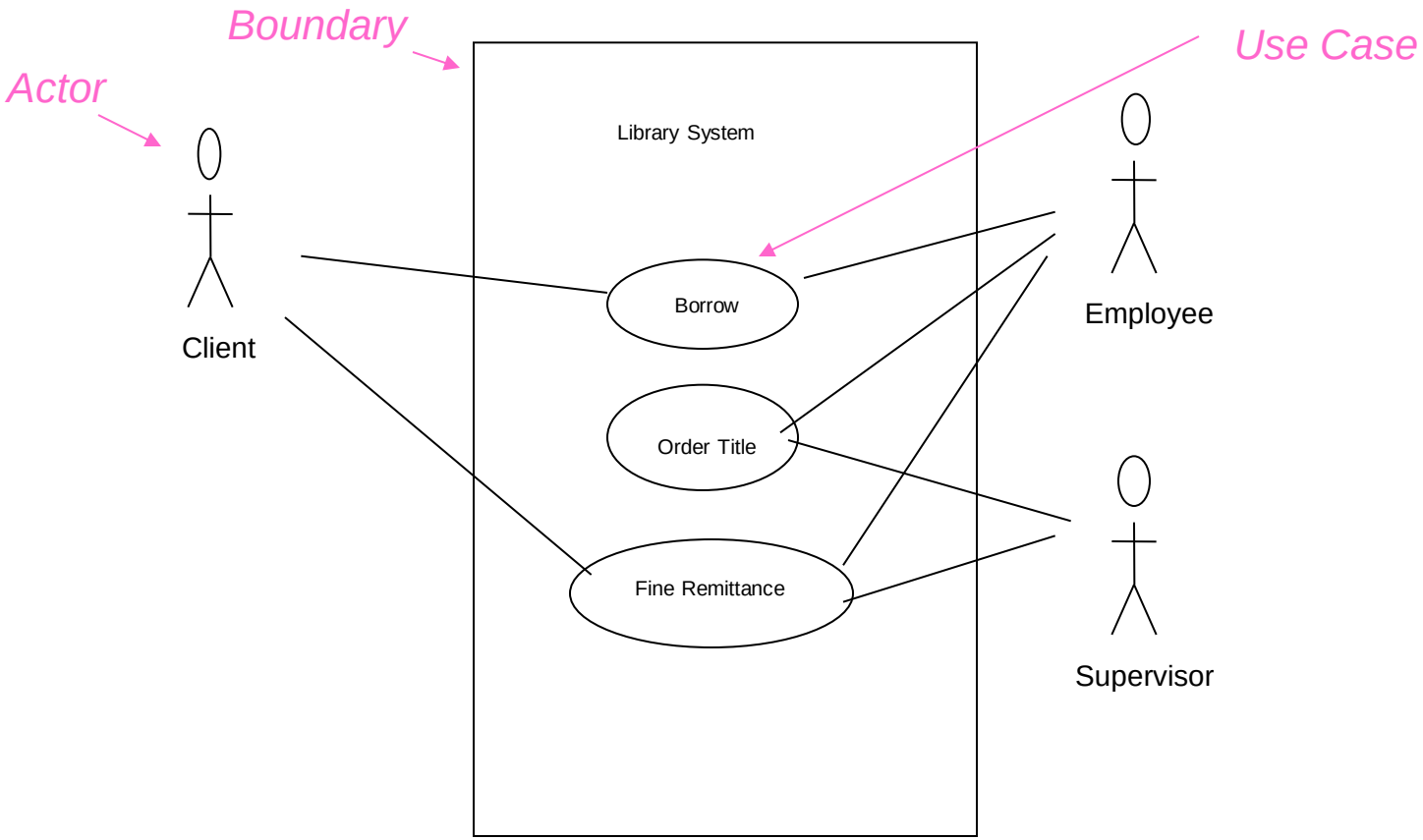
UML Modeling Diagrams

- 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.**

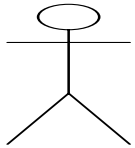
Use Case Diagrams



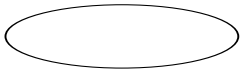
- A generalized description of how a system will be used.
- Provides an overview of the intended functionality of the system

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.**

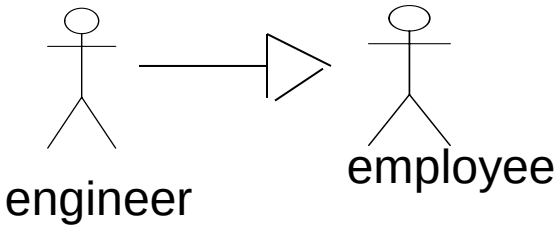
System boundary: a rectangle diagram representing the boundary between the actors and the system.

Use Case Diagram(core relationship)

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

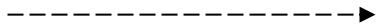
Generalization: relationship between one general use case and one specific use case.

Represented by a line with a triangular arrow head toward the parent use case, the more general modeling element.



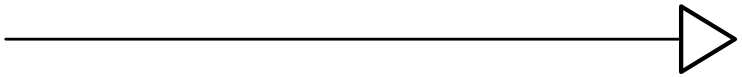
Use Case Diagram(core relationship)

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.



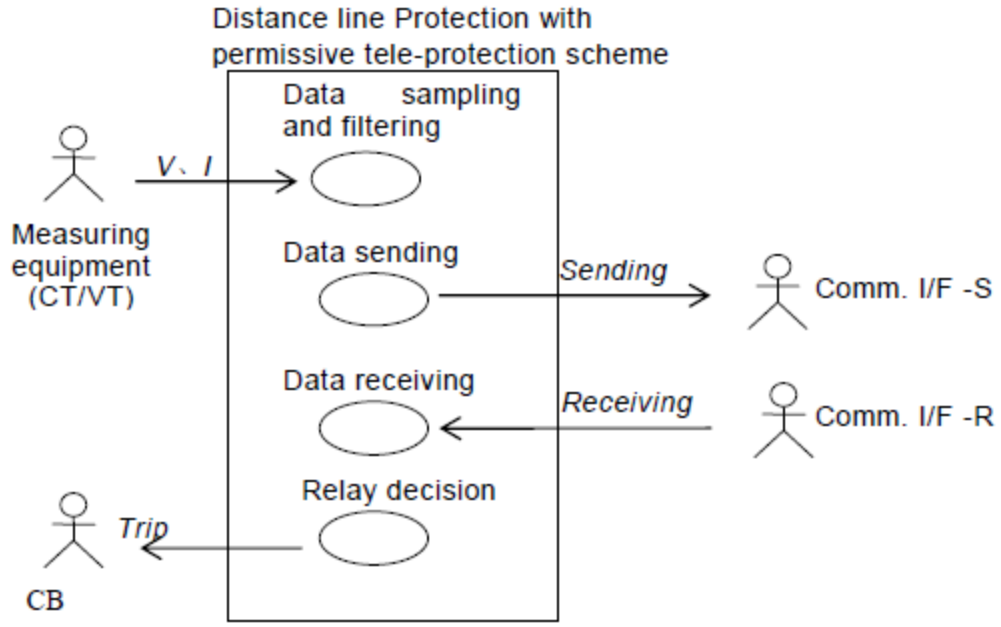
<<include>>

<<uses>>

or in MS Visio 

Use Case Diagram Use

Use case diagram:



Actor(s):

Name	Role description
Measuring equipment	Measures current and voltage from protected line
Comm. I/F –S	Receives data from the local relay and sends the data to the remote end
Comm. I/F -R	Receives data from the remote end and gives the data to the local relay
CB	Disconnects the protected line from other system (Circuit Breaker)

Use Case Diagram Use

Use case(s):

Name	Services or information provided
Data sampling and filtering	Samples current and voltage data from measuring equipment and filters them
Data sending	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).
Data receiving	Receives the permissive signal from Comm. I/F –R (the remote end).
Relay decision	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

Basic flow:

Data sampling and filtering

Use Case Step	Description
Step 1	Current and Voltage are given to Distance protection by Measuring equipment
Step 2	Distance Protection samples an analogue value and converts it to digital data
Step 3	Distance Protection removes any unwanted frequency components from the sampled data using a digital filter

Use Case Diagram Use

Data sending

Use Case Step	Description
Step 1	Distance Protection stores the filtered instantaneous data
Step 2	Distance Protection calculates a distance to the fault using filtered data.
Step 3	When a distance protection detects a forward fault to a pre-determined distance, a distance protection sends the permissive signal to Comm. I/F –S (in order to send the data to a remote end relay)
Step 4	Comm. I/F –S send the information to remote end

Data receiving

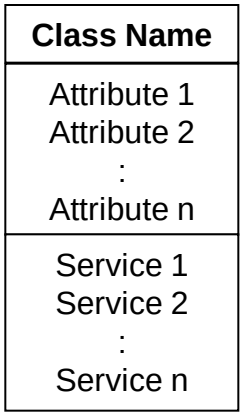
Use Case Step	Description
Step 1	Comm. I/F –R receives the data from the remote end
Step 2	Comm. I/F –R gives the received data to Distance Protection
Step 3	Distance Protection receives the data

Relay Decision

Use Case Step	Description
Step 1	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

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.**



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.

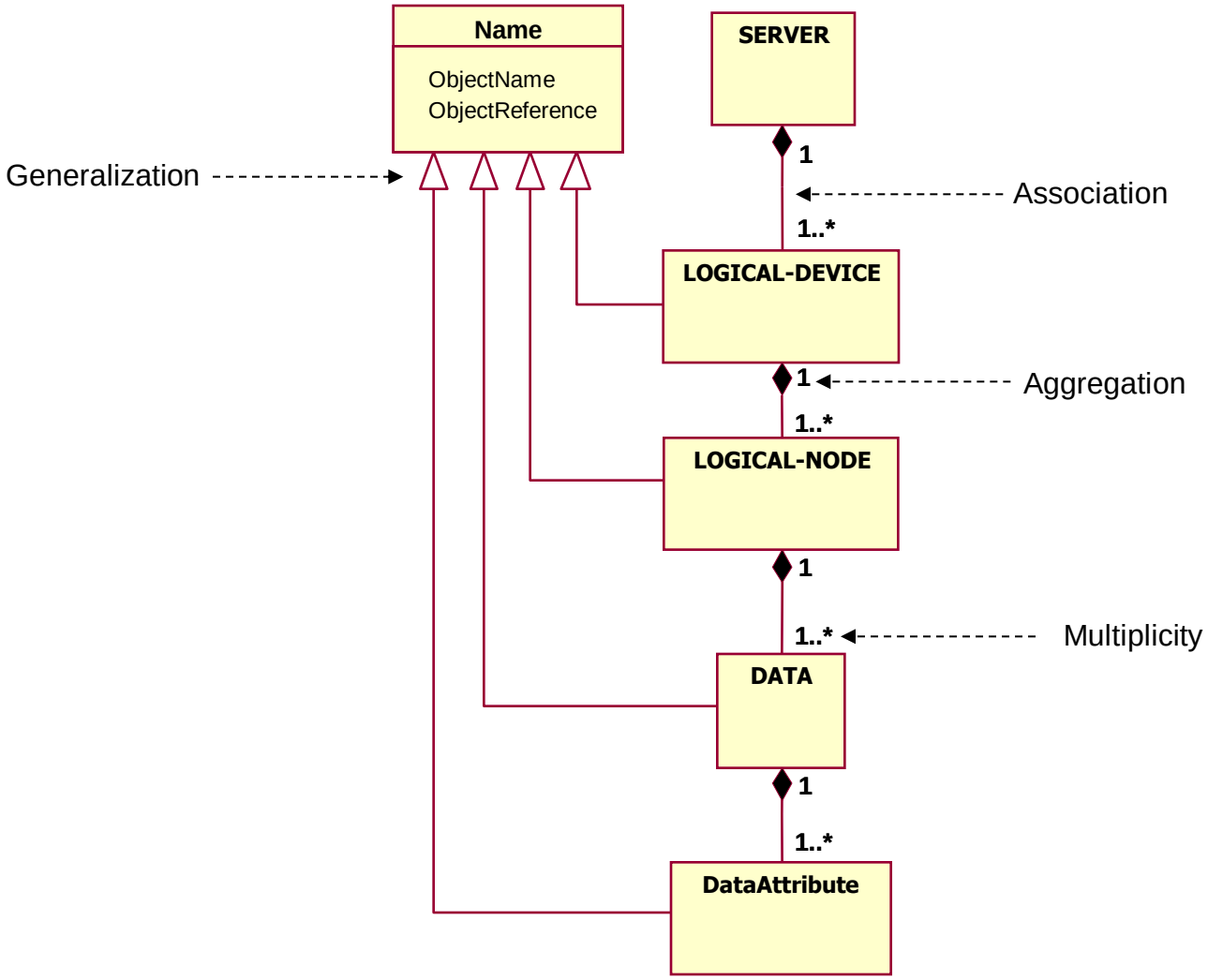
UML: Class Diagrams

- **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.**

UML: Class Diagrams

Multiplicities	*Meaning*
0..1	*0 or 1 instance. The notation n . . m indicates n to m instances.*
0.. or * *	*No limit on the number of instances (including none).*
1	*exactly one instance*
*1..**	*at least one instance*

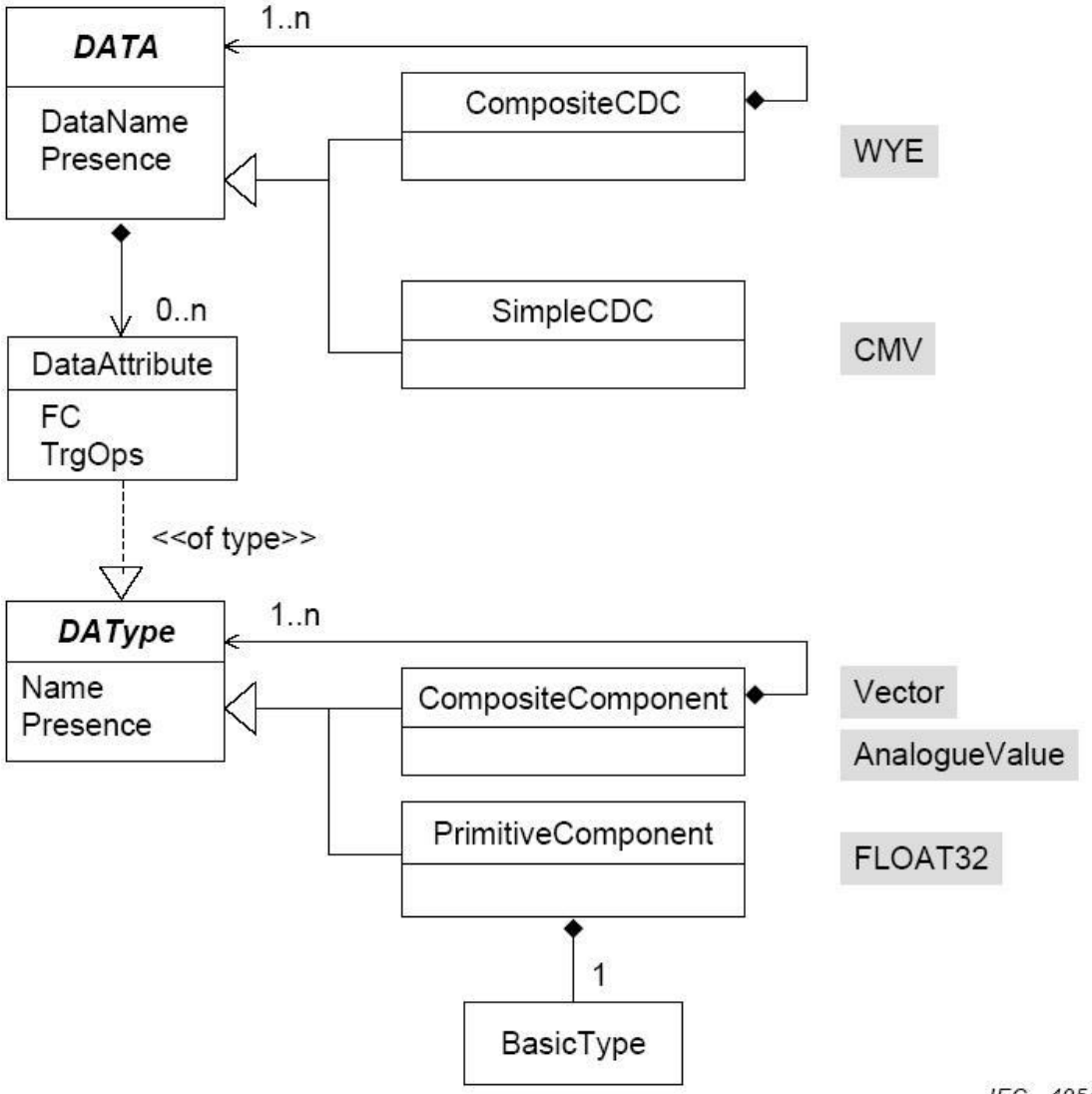
UML: Class Diagrams



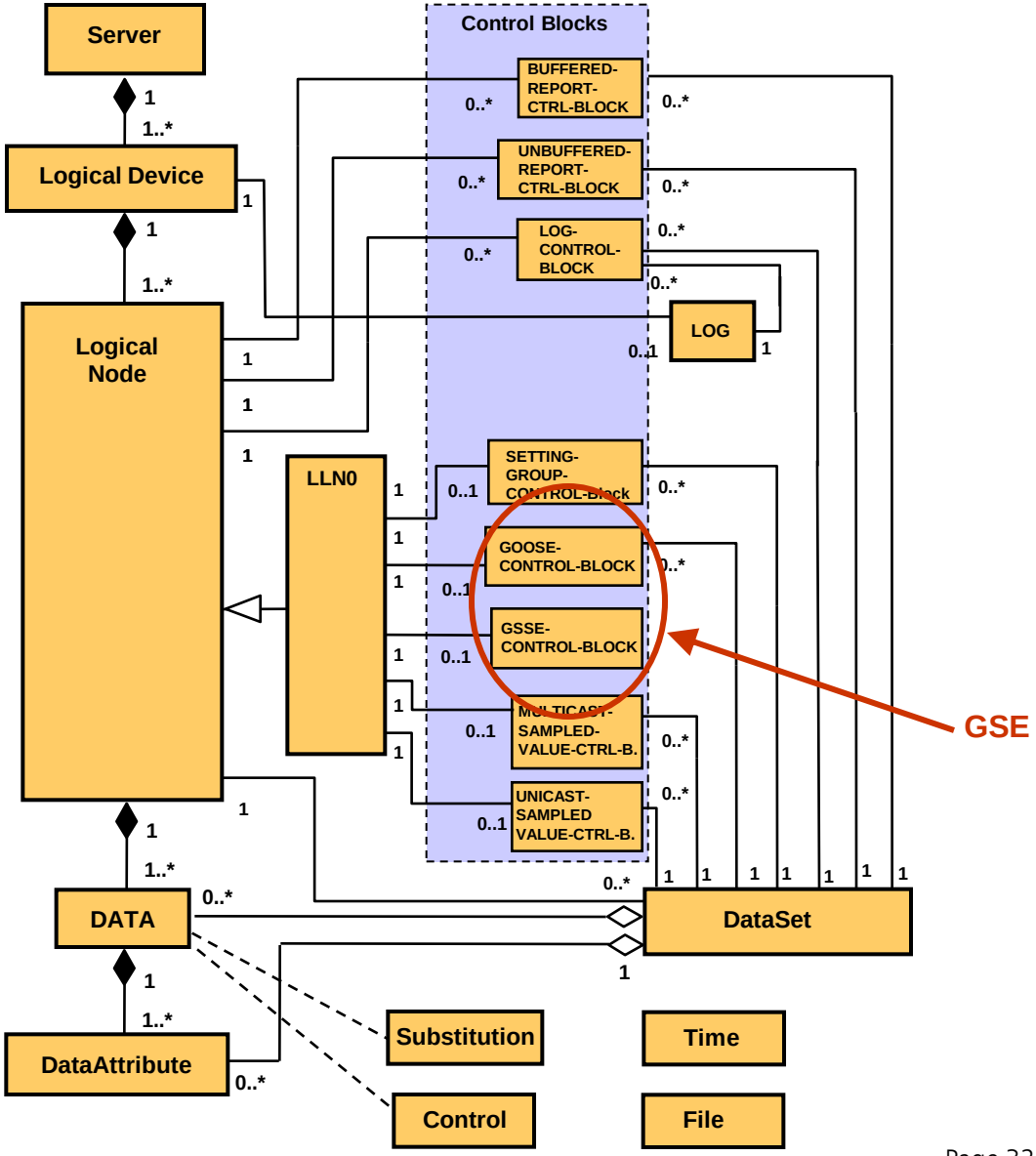
IEC 61850 Logical Node Class

LOGICAL-NODE class		
Attribute name	**Attribute type**	**Explanation**
LNName	ObjectName	Instance name of an instance of LOGICAL-NODE
LNRef	ObjectReference	Path-name of an instance of LOGICAL-NODE
Data [1..n]	DATA	
DataSet [0..n]	DATA-SET	
BufferedReportControlBlock [0..n]	BRCB	
UnbufferedReportControlBlock [0..n]	URCB	
LogControlBlock [0..n]	LCB	
IF compatible LN class defined in IEC 61850-7-4 equals LLN0		
SettingGroupControlBlock [0..1]	SGCB	
Log [0..1]	LOG	
GOOSEControlBlock [0..n]	GoCB	
GSSEControlBlock [0..n]	GsCB	
MulticastSampledValueControlBlock [0..n]	MSVCB	
UnicastSampledValueControlBlock [0..n]	USVCB	
Services GetLogicalNodeDirectory GetAllDataValues		**IEC 61850-7-2**
NOTE 1 IEC 61850-7-4 defines specialized logical node classes – the compatible logical node classes, for example, XCBR representing circuit-breakers.		

IEC 61850 Data Class

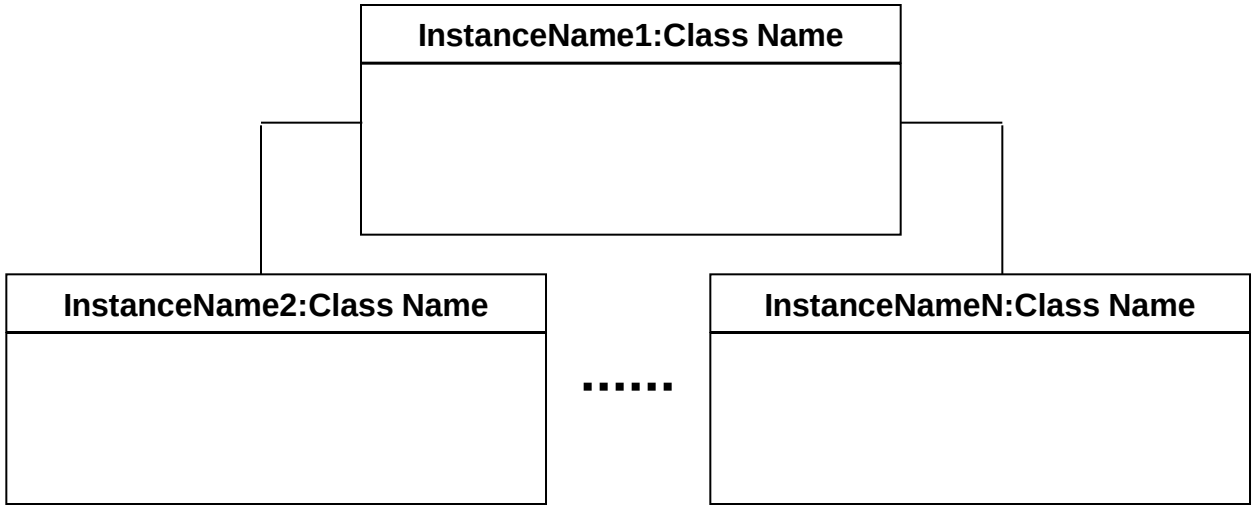


IEC 61850 Services



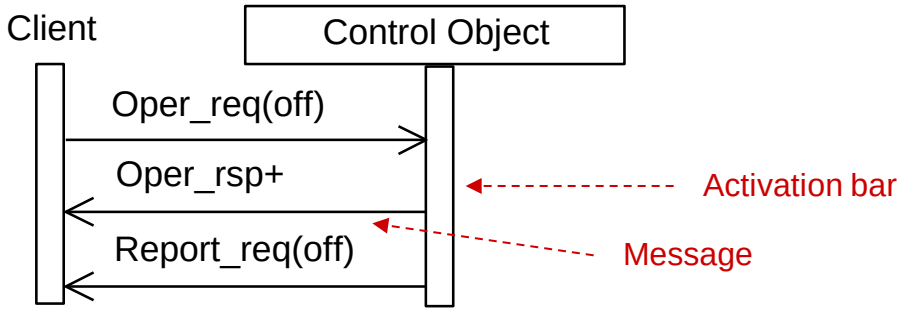
UML: Object Diagrams

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



UML: Sequence Diagrams

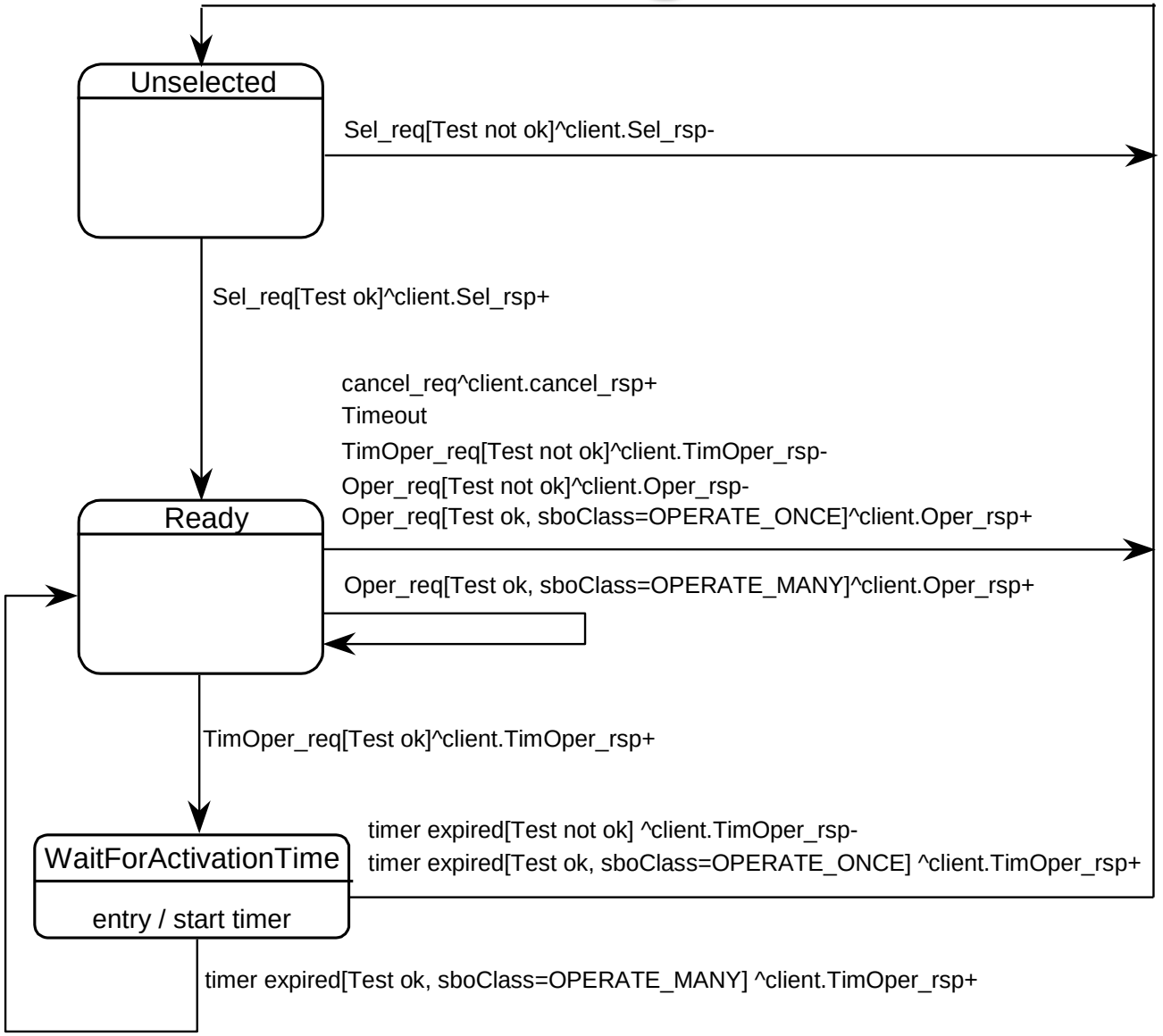
- **Class and object diagrams are static model views.**
- **Interaction diagrams are dynamic. They describe how objects collaborate.**
- **A sequence diagram is an interaction diagram that details how operations are carried out -- what messages are sent and when.**
- **Sequence diagrams are organized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



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.**

UML: Statechart Diagrams - SBO



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).**

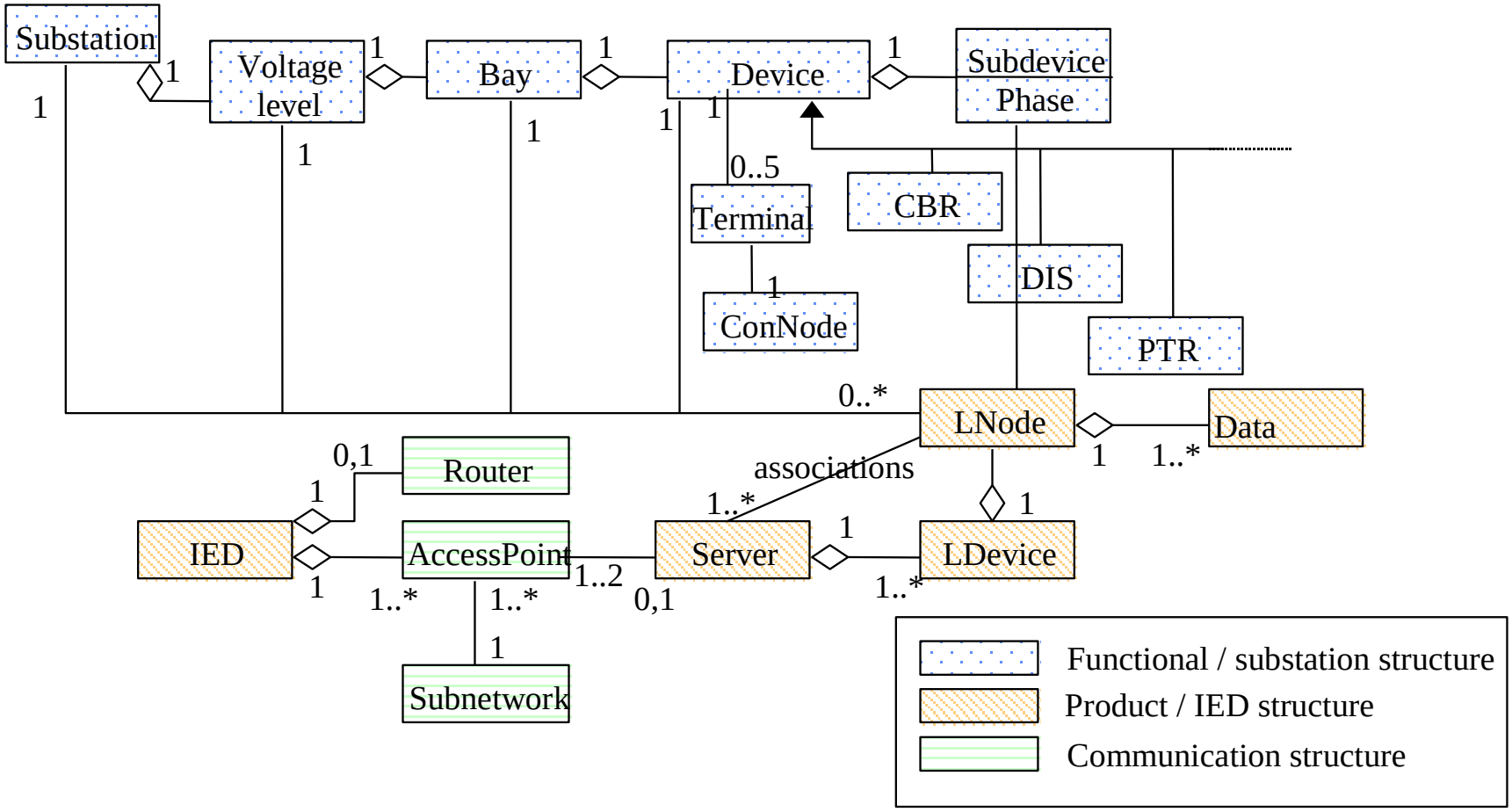
Substation Configuration Language SCL

- **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.**

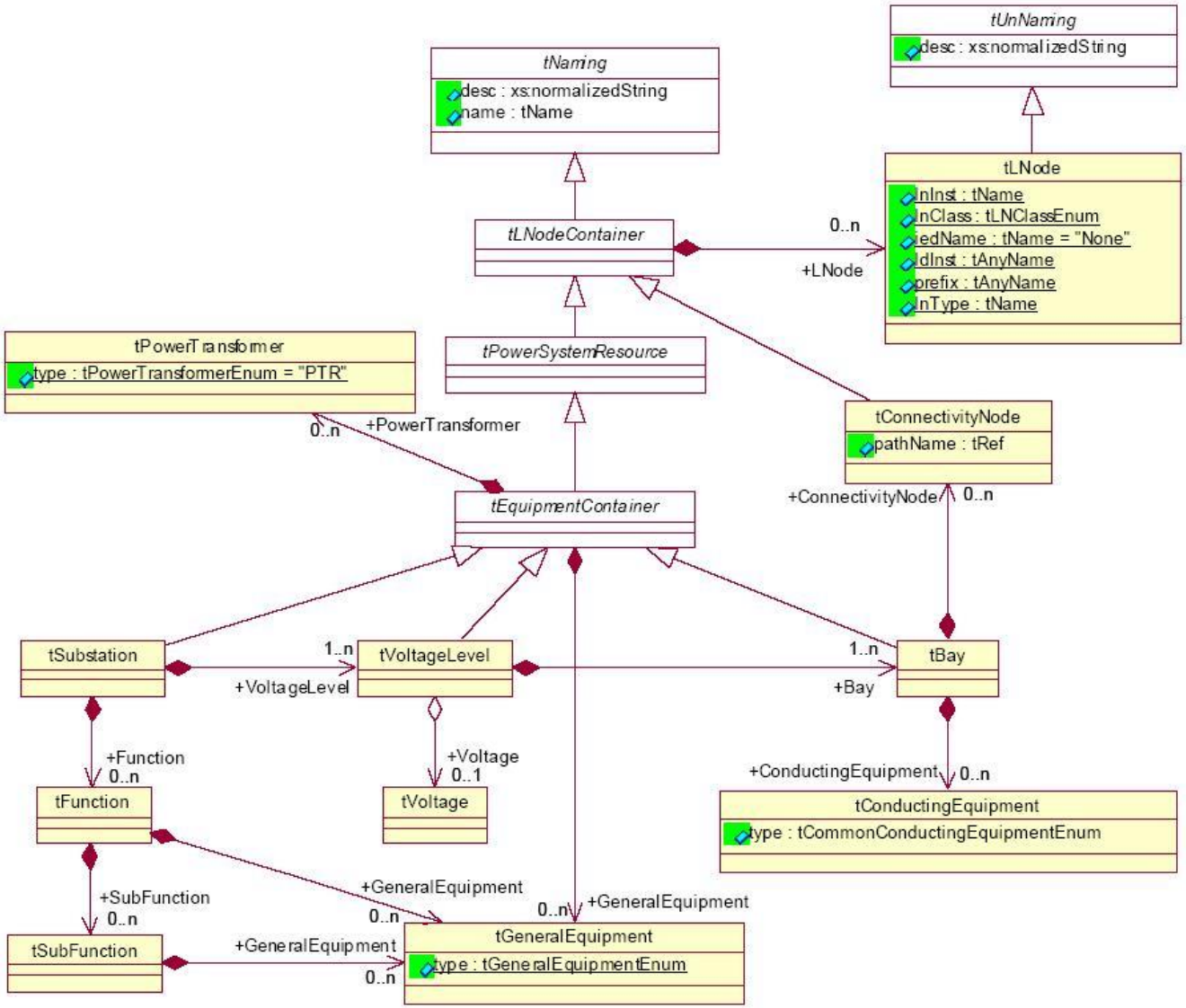
Substation Configuration Language SCL

- **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.**

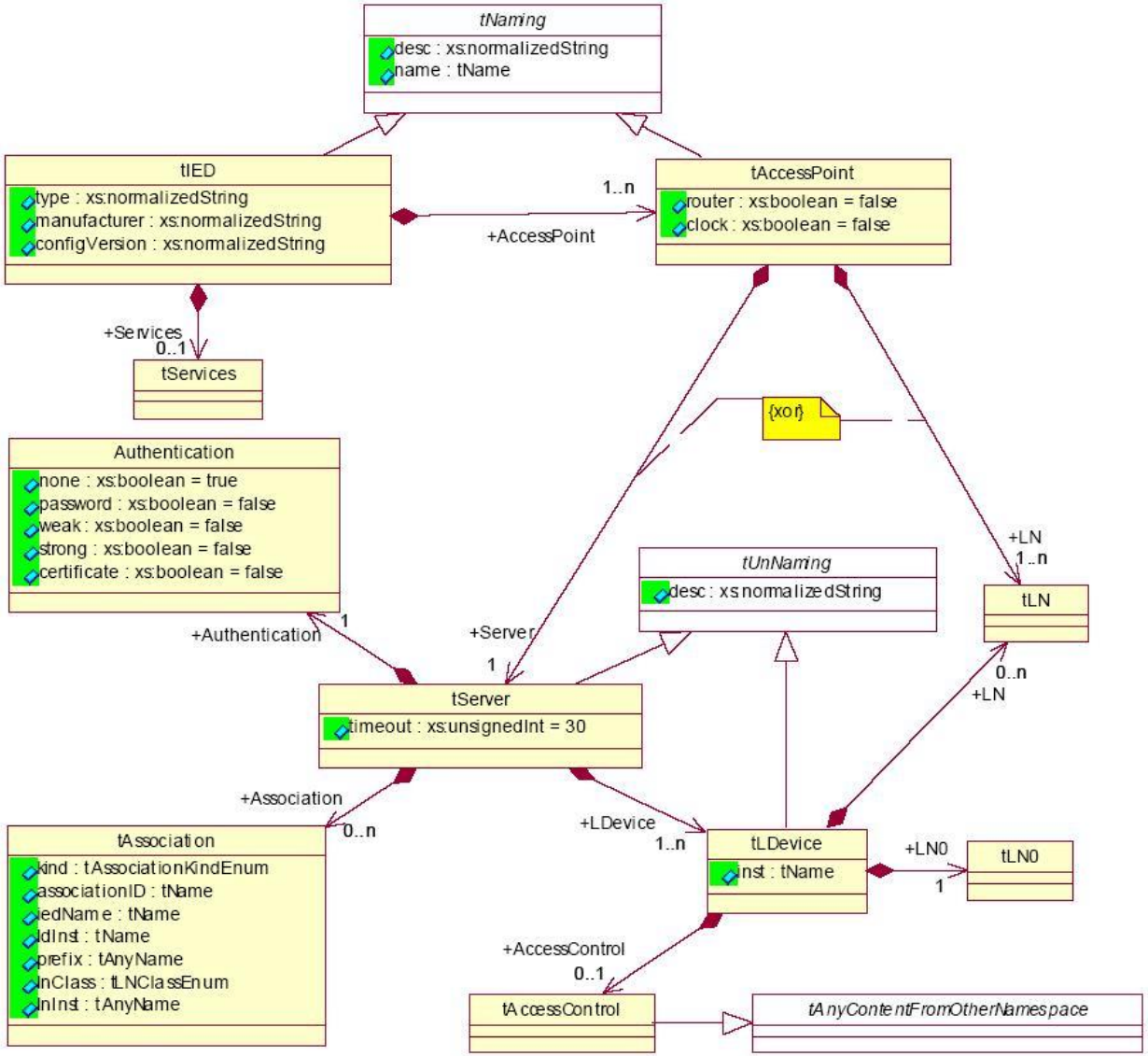
Objectives


Substation
Voltage level
Bay
Device
Subdevice
Phase
Terminal
CBR
DIS
PTR
ConNode
LNode
Data
Router
IED
AccessPoint
Server
LDevice
Subnetwork
associations
0..5
0..*
1..*
1..2
0,1
Functional / substation structure
Product / IED structure
Communication structure

SCL UML Diagram Example



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**