Objects and Communication Services Model Development for the Automatic Speed Regulator of a typical Itaipu dam Hydraulic Turbine in conformance with the IEC 61850

by
David Daniel Pérez Sosa

Submitted to the Department of Electrical Engineering
in partial fulfillment of the requirements for the degree of
Electrical Engineer
at the
UNIVERSIDAD NACIONAL DEL ESTE
December 2010

Author .....
Department of Electrical Engineering
December, 2010

Certified by .....
Ladislao Aranda Arriola
Electrical Engineer, Itaipu Binacional
Thesis Supervisor

Certified by .....
Rodrigo Ramos
Electrical Engineer, Itaipu Binacional
Thesis Supervisor

Certified by .....
Juan Manuel Ramirez Duarte
Electrical Engineer, Facultad Politécnica
Thesis Supervisor

Accepted by .....
Professor Anastasio Sebastián Arce Encina
Thesis coordinator, Department of Electrical Engineering

Objects and Communication Services Model Development for the Automatic Speed Regulator of a typical Itaipu dam Hydraulic Turbine in conformance with the IEC 61850

by

David Daniel Pérez Sosa

Submitted to the Department of Electrical Engineering
on December, 2010, in partial fulfillment of the
requirements for the degree of
Electrical Engineer

Abstract

Forthcoming.

Thesis Supervisor: Ladislao Aranda Arriola
Title: Electrical Engineer, Itaipu Binacional

Thesis Supervisor: Rodrigo Ramos
Title: Electrical Engineer, Itaipu Binacional

Thesis Supervisor: Juan Manuel Ramirez Duarte
Title: Electrical Engineer, Facultad Politécnica

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

For example:

I would like to thank to my parents for supporting me though all of this.

Also to Leslie Lamport and Donald Knuth for LaTeX and TeX, without which I would have had much less fun formatting my thesis.

This research was supported in part by the Fundación Parque Tecnológico Itaipu supplying the needs of runaway geniuses everywhere.

Contents

61850, parte6, cl6.1, ¶3

Glossary

Intelligent Electronic Device

microprocessor based device with network capabilities. 22, 51

LN

This abbreviation of *Logical Node* is used by IED xml child node in the SCL. 50

LNode

This abbreviation of *Logical Node* is used by Substation xml child nodes in the SCL. 50

O-O

Object Oriented should be: Object oriented system, object oriented programming, object oriented model, object oriented technology. 25, 26, 29, 50

SCL

The SCL is a XML based language defined as a standard (IEC 61850-6 [1]) and is used for IED configuration. 48, 50, 60

Substation Automation System

Substation Automation System. 50

System Specification Description

SCL file that contains the power system functions, i.e, the substation xml node of the SCL, but without great details, not including descriptions such as substation LNodes correspondences with IED LN objects. 50

Unified Modeling Language ™ - UML

Unified Modeling Language ™ [2] is a specification to describes the model of a application structure, behabiour and system architecture. 25, 50

XML Schema Definition

XML Shema is used to describe the structure and restrictions of the XML documents. In the IEC 61850-6 XSD files are asigned to SCL object models.. 48, 50, 60

List of Figures

List of Tables

Chapter 1

Work Research Overview

1.1 Introduction

For comming...

1.2 Aim of this Research

For comming...

1.3 Research Methodologies and Techniques

Falta traducir al inglés toda esta sección. Esta sección la extraje de mi anteproyecto, que originalmente fue escrita en español.

1.3.1 Investigación bibliográfica

Incluye la investigación y estudio de la norma IEC 61850, en especial las definiciones de modelos de información definidos en la parte IEC 61850-7-4-10 *Basic communication structure for substation and feeder equipment Compatible logical node classes and data classes - Hydro Power Plants*, los servicios de intercambio de información para diferentes funciones (por ejemplo, control, reporte, *getters* y *setters*) definidos en

el apartado IEC 61850-7-2 *Basic communication structure for substation and feeder equipment Abstract communication service interface (ACSI)*, la implementación de dichos servicios de comunicación a través de protocolos especificados en la parte IEC 61850-8-1 *Specific Communication Service Mapping (SCSM) Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3* para las necesidades de la unidad generadora, y otros documentos, normas, y tecnologías relacionadas.

1.3.2 Investigación de campo

pregunta: como se traduce al ingles la palabra campo en el contexto de ingenieria electrica -¿ *switchyard* puede ser?, como podria decir *"investigacion de campo" ?*

Incluye el levantamiento de informaciones del generador de Itaipú, la identificación de las funciones de automatización inherentes regulador de velocidad de la unidad generadora y otras documentaciones y estadísticas relacionadas al caso en estudio. También incluye la simulación en una red IEC 61850 utilizando las herramientas de ingeniería disponibles en el mercado para tal efecto.

1.3.3 Creación de los objetos

Diseño de los nodos lógicos normalizados en la parte IEC 61850-7-4, e IEC 61850-7-4-10 mediante UML, implementaciones en XML, Java, C++, o mediante las herramientas de ingeniería especializadas en la norma IEC 61850 disponibles en el mercado.

1.3.4 Identificación de servicios de información

Modelado completo de los servicios de comunicación necesarios para el regulador de velocidad de una unidad generadora típica de Itaipu utilizando la parte IEC 61850-7-2.

1.3.5 Propuesta de extensión/complementación de los nodos lógicos

Por ejemplo, ZAXN, insuficiente para la alimentación AC y DC de los servicios auxiliares de Itaipu.

este nodo logico es solo un ejemplo, la idea es que, si se encuentran funcionalidades no contempladas por la norma, pero son necesarias, sea propuesta en los tissues. El ZAXN es solo un ejemplo, pero en realidad voy a cambiar este ejemplo, o dejo asi, no se todavia, dado que no voy a modelar nada de los servicios auxiliares. esto debo revisarlo mas adelante, no tiene prioridad inmediata y es un punto que dejare pendiente para esta primera mitad del tiempo de mi investigacion

1.3.6 Metodología de aplicación

Estructuración de una metodología de aplicación de la norma IEC 61850 en la automatización de hidroeléctricas: Elección de las herramientas de ingeniería disponibles en el mercado y que mejor se adapten a la automatización de unidades generadoras aplicando la norma IEC 61850 y la creación de procedimientos padronizados utilizando dichas herramientas, para satisfacer los requerimientos de ingeniería definidos en el apartado IEC 61850-4 subsección 3, y para identificar las funciones de automatización que son necesarias para el modelado de los nodos lógicos, pero no están contempladas en la norma IEC 61850.

1.4 Delimitations

To achieve the objectives of this research, some delimitations was defined. The description of theses delimitations are important for a mayor understandability of this document. The delimitations are:

- Una de ellos es que me preocupo mas por la parte de modelado de objetos, no tocando mucho la parte de redes de comunicacion. Los protocolos y las redes

se tratan en forma muy basica. Mas bien se trabaja con las APIs de alto nivel (los objetos, las clases) definidas en los modelos SCL. Bueno, de cierta forma trabajo con redes dado que tengo que modelar los servicios de comunicacion, pero ese modelado se realiza en alto nivel, a travez de la instanciacion de las clases especificadas en los esquemas de la parte 6. Osea, no uso sniffers ni nada parecido, no trato esa parte de la red, hablo poco de la parte 8-x y 9-x, mas bien hablo de la parte 7-2 y parte 6 para hacer el proyecto de modelado (por eso el titulo tentativo de la tesis es -Object and services model development for...-).

traducir al ingles

- Se haran algunas simplificaciones para eliminar las dependencias de regulador de velocidad con otras partes del sistema, de modo a que se pueda modelar solamente esa parte de la unidad generadora y no toda la unidad generadora.
- se agregaran mas limitaciones cuando se crea oportuno, en caso de que sean necesarias
- The object modeling in conformance with the IEC 61850 is limited to the Automatic Speed Regulator of a typical Itaipu dam Hydraulic Turbine.
- This thesis focuses on the power system automation engineering ,

en vez de engineering, no es mejor decir development?

en realidad yo no queria decir aca *power system automation*, yo queria decir SAS, pero como no es subestacion y es hidroelectrica, entonces no se si esta bien decir SAS, por eso deje en forma medio generica nomas (que a lo mejor presta a ambiguedad, pero no se cual es la palabra correcta)

not covering the maintenance and operation of them.

1.5 Phases of the research

This research process follow the steps mentioned here in the respective order:

traducir al ingles

Esto lo quite de mi anteproyecto, debo pulirlo bastante, las fechas estan mal, debo colocar las fechas indicadas en mi anteproyecto aprobado. Yo reaproveche este texto que ya habia redactado en LaTeX a finales del año pasado.

1.5.1 Object oriented paradigm

Agosto del 2009 Clases, métodos, atributos. Objeto. Instanciación. Herencia. Abstracción. Encapsulación. Polimorfismo. Interfaces. Recursividad. Tipos de datos.

1.5.2 Septiembe del 2009

Unified Modeling Language - UML: Diagrama de clases. Representación de objetos mediante UML.

1.5.3 Octubre del 2009

XML - Extensible Markup Language: Lenguajes de marcación. XML. DTD (*Document Type Definition*). XSD (*XML Schema Definition*)

1.5.4 Noviembre del 2009

Análisis de los conceptos de la automatización de Sistemas Eléctricos de Potencia

Sistema de automatización de hidroeléctricas, en especial de una unidad generadora. Topología de red del sistema de automatización. Funciones del sistema de automatización de una unidad generadora: comando, adquisición de datos, protecciones, supervisión, alarmas, secuencia de eventos, enclavamientos y bloqueos.

Identificación de las comunicaciones en el Sistema de Automatización Eléctrico

Redes de área local. Tecnologías de red, en especial las implementaciones de la norma IEEE 802.3. Jerarquías de protocolos. Servicios: Connection-Oriented y Connectionless. Relaciones entre servicios y protocolos. Modelo de referencia *Open System*

Interconnection (OSI). Medios de transmisión de información. Control, flujo, corrección y detección de errores. Algoritmos de enrutamiento: broadcast y multicast. Arquitecturas cliente-servidor, *peer-to-peer*.

1.5.5 Diciembre del 2009 y enero del 2010

Lectura e interpretación de la norma IEC 61850: Incluyendo las partes 1, 2, 4 (solo la sub parte 5), 6, 7-1, 7-2, 7-3, 7-4.

1.5.6 Febrero del 2010

Desglosar la arquitectura, los elementos y modelos de comunicación de un Sistema de Automatización de Usinas

1.5.7 Marzo del 2010

Conocer el funcionamiento y clasificar las funcionalidades generales necesarias en el sistema de automatización de un regulador de velocidad: Comandos, adquisición de datos, protecciones, supervisión, alarmas, secuencia de eventos, enclavamientos, secuencias automáticas, controles de velocidad, sincronización, informes, valores hidroenergéticos, entre otros.

1.5.8 Abril del 2010

Estudio del funcionamiento y características particulares del regulador de velocidad de una unidad generadora típica de Itaipu: Identificación de las funciones de automatización con ayuda de especialistas de la máquina. Desglosar cada función de automatización en los nodos lógicos correspondientes.

1.5.9 Mayo del 2010

Breve estudio de la arquitectura de red necesaria para la automatización del regulador de velocidad implementando la norma IEC 61850.

1.5.10 Junio y Julio del 2010

Modelado de los nodos lógicos normalizados del apartado IEC 61850-7-4-10 *Hydro Power Plants* utilizando herramientas de ingeniería disponibles en el mercado.

1.5.11 Agosto a Septiembre del 2010

Identificación y modelado de los servicios de comunicación necesarios para el regulador de velocidad. Apartado 7-2 de la norma IEC 61850.

1.5.12 Octubre a Diciembre del 2010

Validación y corrección de errores del trabajo mediante simulación por software en una red IEC 61850 y posterior revisión por pares.

1.6 Parties

This section introduces the stakeholders or parties that have supported the development of this research.

falta completar

1.6.1 CIAC - Parque Tecnológico Itaipu - Paraguay

The *Centro de Innovación en Automatización y Control - Parque Tecnológico Itaipu* , Paraguay is a .....

agregar mas informacion sobre PTI

1.6.2 Itaipu Binacional

La Central Hidroeléctrica Itaipu Binacional .....

agregar mas informacion sobre Itaipu

1.6.3 Universidad Nacional del Este

La Universidad Nacional del Este .....

agregar mas informacion sobre la UNE

1.7 Originality of this Research

For comming...

1.8 Organization of this Research Document

traducir al ingles - organizacion de la tesis

This text is organized as follows: For comming...

- Chapter 1: revisin de la tesis
- Chapter 2: ...

Captulo: hacer una revisin de la tesis

capitulo: capitulo introductorio a la norma IEC 61850, al estilo de la parte 1 y la parte 7-1, sin entrar en puntos muy tecnicos, solo para dar una nocion de lo que trata la norma, asi como los famosos powerpoints que te presentan para la norma y de dan una nocion super super breve sobre que trata la norma.

Captulo: hablar sobre programacin orientada a objetos, uml y xml demasiado bien va a caer comenzar con este captulo, dado que mi tema trata sobre modelado de objetos.

Captulo: hablar sobre subestaciones, lo bsico para poder entender la norma.

Captulo: hablar sobre la automatizacin de usinas (basado en la tesis brasilera que me paso el Ing. Aranda)

Captulo: hablar sobre redes, de absolutamente todos los conceptos necesarios.

Captulo: se puede estandarizar a UML las notaciones y explicaciones de arquitecturas que explica la norma. Posible contribucion: utilizar OCL para mejorar la especificacion.

Captulo: hablar sobre la turbina itaip

Captulo: hablar sobre la norma en si. (esto se puede subdividir otra vez en ms captulos especializados si es necesario)

Captulo: se puede hablar sobre las funciones de automatizacin aplicadas a la turbina (o al regulador directamente)

y los siguientes captulos... veremos despus... tambien es probable de que varios de estos futuros capitulos se fusionen en uno.

hay capitulos que ya tengo bien aprendido, pero aun no escribi, en las vacaciones de invierno voy a escribir bien sobre redes y las demas teorias basicas de automatizacion. Por el momento, he creado mapas mentales que apareceran en los capitulos inconclusos que ya tengo aprendido pero solo me falta escribir. Para que pueda escribir, esos capitulos con un buen nivel tecnico voy a apoyarme en papers de varias asociaciones, principalmente el de la IEEE. Aun no estoy pudiendo suscribirme y desde la CAPES no se accede, apenas pueda me suscribire y escribire esos capitulos. Siempre mi material principal es la norma en si, y en base a eso ya tengo varios capitulos en mente (aprendidos pero por documentar) o ya escritos.

Chapter 2

Introduction to the IEC 61850 series

2.1 IEC 61850: A standard for the whole electrical systems

The IEC 61850 is a standard for communication networks and systems and the related system requirements for the whole electrical system used by Intelligent Electronic Devices (IEDs) to perform the required functions for the secundary equipments.

2.2 Objectives of the IEC 61850 standard

2.3 IEC 61850 Parts

Debo ordenar estas partes y colocar las descripciones de las partes

o por considerar

List of the IEC 61850 parts considered in this work:

IEC61850-1:2003 [3]

IEC61850-2:2003 [4]

IEC61850-3:2002 [5]

IEC61850-4:2002 [6]

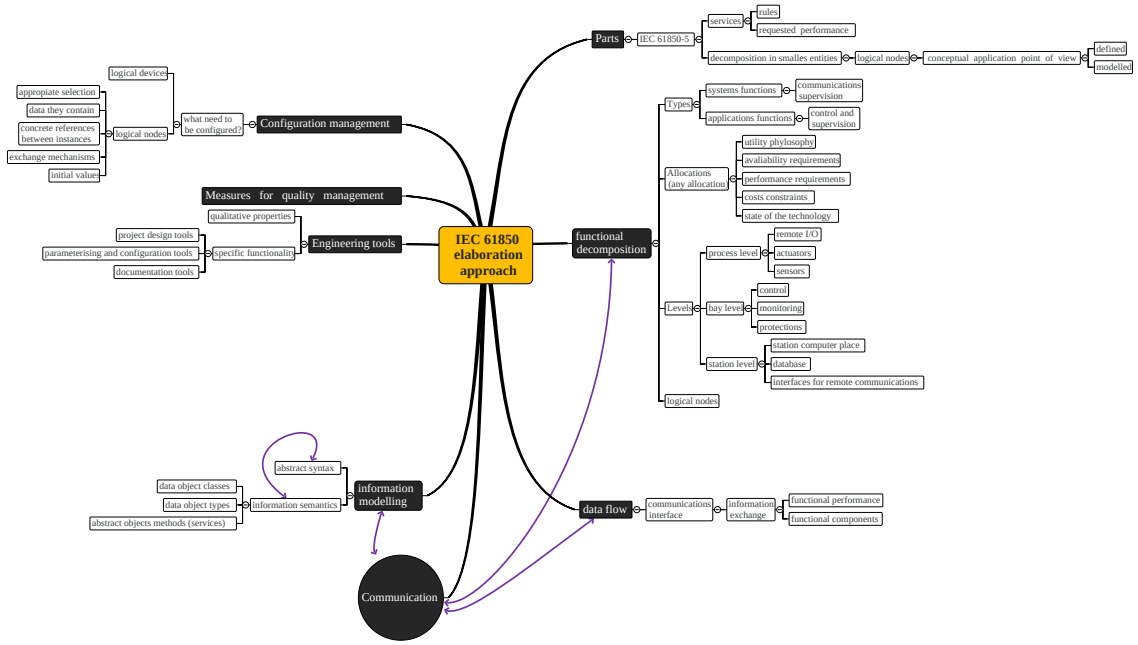

Figure 2-1: Borrador - Esquema del futuro capitulo

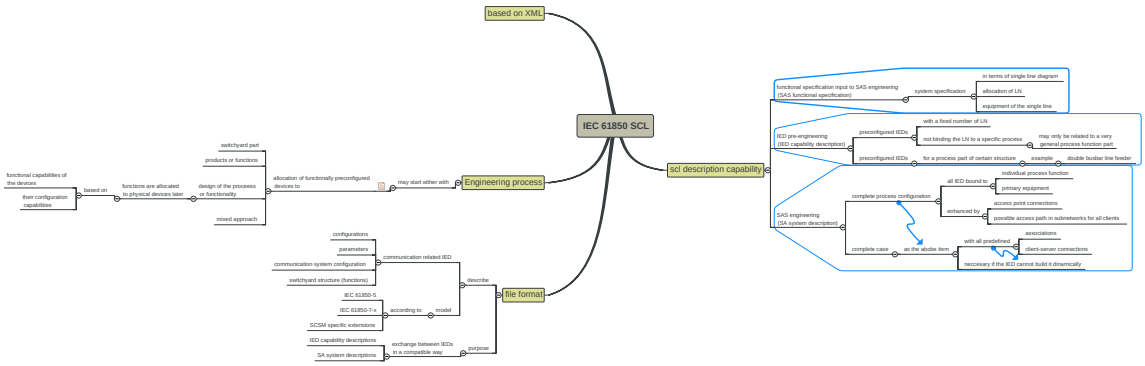

Figure 2-2: Borrador - Esquema del futuro capitulo

IEC61850-5:2003 [7]

IEC61850-6:2004 [1]

IEC61850-7-1:2003 [8]

IEC61850-7-2:2003 [9]

IEC61850-7-3:2003 [10]

IEC61850-7-4:2003 [11]

IEC61850-7-410:2007 [12]

IEC61850-7-420:2009 [13]

IEC61850-7-430:200-X[14]

IEC61850-7-5:2010 [15]

IEC61850-7-500:2010 [16]

IEC61850-7-510:2009 [17]

IEC61850-7-520:2010 [18]

IEC61850-7-10:2009 [19]

IEC61850-8-1:2004 [20]

IEC61850-9-1:2003 [21]

IEC61850-9-2:2004 [22]

debo arreglar las bibliografias del bibtex referenciadas en esta seccion, no se estan viendo como se debe. Algunas deben ir como reporte tecnico y no como estandard porque aun no son oficiales sino borradores

Chapter 3

Object-oriented system construction

3.1 Introduction

The IEC 61850 information model is classified as a O-O system. Is necessary to have a clear understanding of Object-Oriented (O-O) technology to dive into IEC 61850 object modelling. For this reason this chapter describes the O-O technology on which the IEC 61850 information model has been standarized

No se si esta bien escrita la palabra standarized

traduccion:just focuses or focuses just?

.

The chapter does not describe all the O-O principles, just focuses on the necessary background to understand the IEC 61850 information model. The sections which follow will define the background which the IEC 61850 object oriented information system was constructed, both formally and through examples. A detailed description of the O-O fundamentals and reference materials are provided. To achieve a robust concepts comprehension a practical implementation with theUnified Modeling Language (UML) [2] and Java [23] are provided. This chapter is very practical in the sense that uses language programmings to model very carefully selected models used by the IEC 61850 standard, (but not explaining explicitly nothing about the standard yet!). This chapter is destined for electrical engineers and professionals of related ar-

agregar al glossaries

eas readers which the O-O software construction is not part of the curriculum. For this reason, the practical examples uses electricity area elements. The readers with a strong knowledge of programming could skip this chapter

debería agregar esta ultima frase?: *The readers with a strong knowledge of programming could skip this chapter*

En este captulo voy a agregar los graficos a continuacion:
(los ejemplos que voy usando son con estas clases)

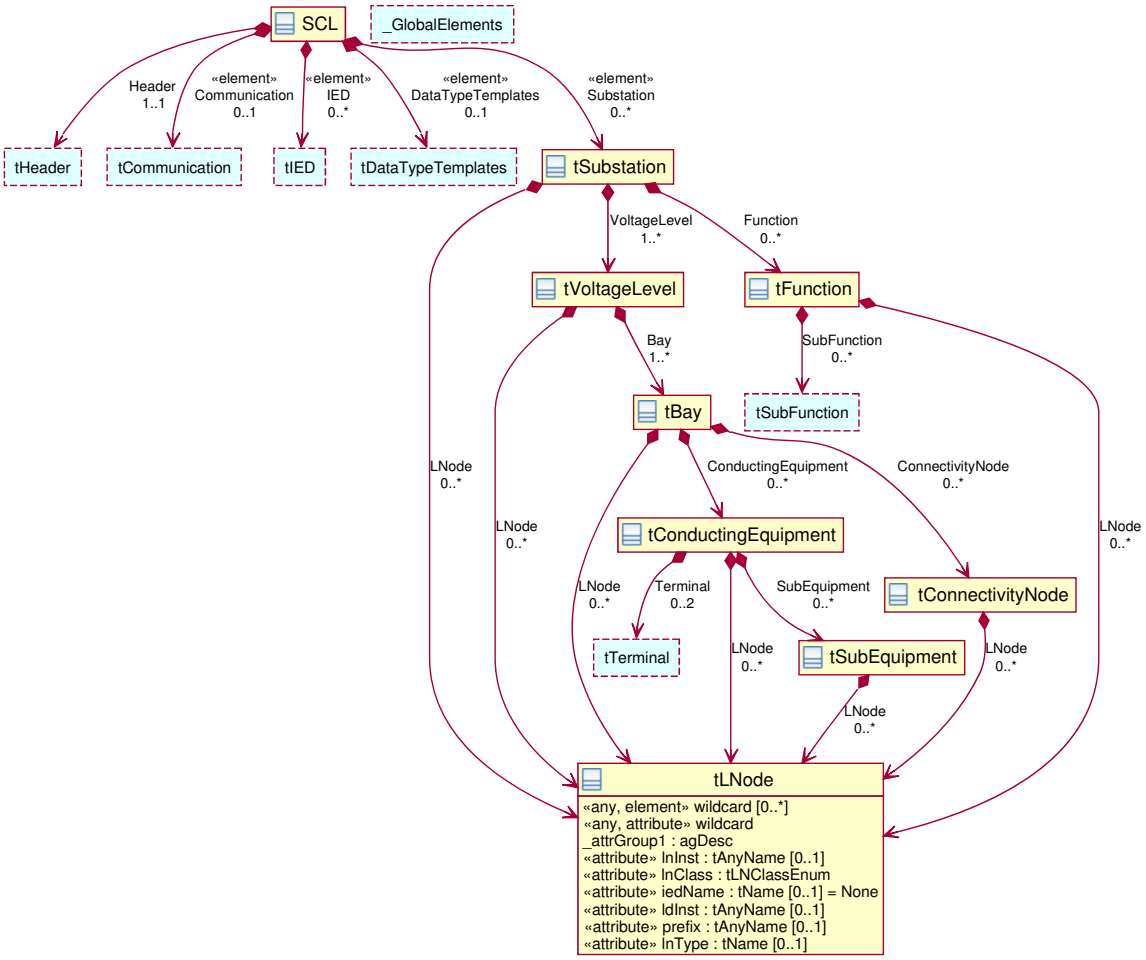

Figure 3-1: Logical Node and their role at the substation level

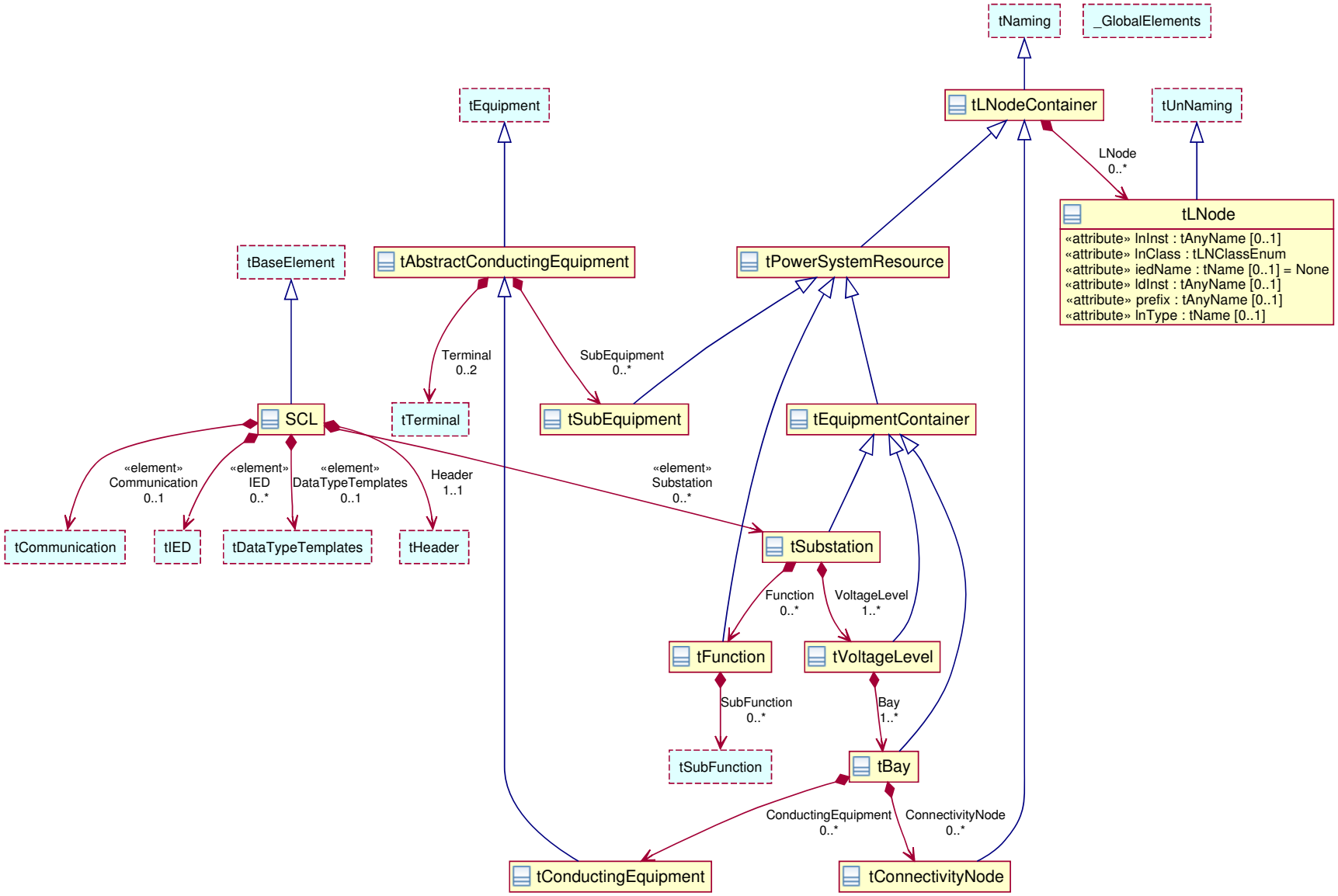

Figure 3-2: Logical Node and their role at the substation level, depicted with the heritance details

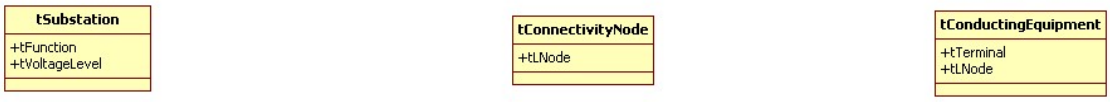

Figure 3-3: Real world object and their representation in software engineering

esta figura esta incompleta, deberia conseguir unas buenas fotos (en itaipu hay muchas fotos lindas y originales) de estos equipamientos electricos de la Itaipu y colocar unas flechitas que unen a estos diagramas con los objetos reales y palabras indicativas...

3.2 Object Oriented systems

Many autors formulate precise definition of O-O paradigm [24] [25] [26] [27], and the definition of Lastly, Wegner [28] has been the most widely accepted [29]. Wegner defines the O-O paradigm in terms of objects, classes and inheritance.

"Objects are autonomous entities that respond to messages or operations and share a state. Classes classify objects by their common operations. Inheritance serves to classify classes by their shared behavior. Data abstraction hides the representation of data and the implementation of operations" [28]. That is:

$$\text{object oriented} = \text{objects} + \text{classes} + \text{inheritance}$$

In the next sections theses concepts are explained.

The O-O system programming metodology defines an approach to code organization for objects creation. The objects store the data and have their own behabiour with a particular information grouping by common functionality and common information structures. O-O systems benefits to bundle actions together, manage a few quantity of variables rather than multiples ones, organizing the common behavior together and structure programs in a way that matches closely the real world [30].

The figure 3-3 shows a real world substation, their equipments, and the virtual substation software engineering representation through the class *tSubstation* that responds with the same behabiour as in the real world, but running by software.

3.3 Classes

A class is a static template from which objects are created. Classes are used to classify objects that define common operations and a data structure, and the data types

that the object can store.

To give a concrete example, this section focuses on data acquisition model of the current transformer and their (conveniently abbreviated TCTR for this chapter). The datas of switchyard functionalities, such as TCTR, depends of each specific situation. Then, the TCTR should be modelled as a class, i.e., a static template which diferents current transformer objects are created. The same reasoning might be applied to the Disconnecting Switches (by convention, here called XSWI) and circuit breakers (XCBR).

aqui ira el codigo de estos nodos logicos, escrito en java

Listing 3.1: TCTR class in Java

```
public class TCTR_v1 {

}
```

Listing 3.2: XSWI class in Java

```
public class XSWI_v1 {

}
```

Listing 3.3: XCBR class in Java

```
public class XCBR_v1 {

}
```

Listing 3.4: Voltage Level class in Java

```
public class TVoltageLevel_v1 {

}
```

3.3.1 Attributes

completar esta parte

Listing 3.5: Class with attributes and methods in Java

```
public class TSubstation_v2 {

  public String name;

  public TVoltageLevel_v1 voltageLevel;

  public TLNode lNode;

}
```

3.3.2 Methods

Methods are functions that are part of a class definition. Once an instance of the class is created, a method is bound to that instance.

Listing 3.6: Class with attributes and methods in Java

```
public  class SERVER_v2 {

  private String[] ServiceAccessPoint;

  public String[] getServiceAccessPoint() {
    return ServiceAccessPoint;
  }
```

```
}
```

Get and set accessor methods

Get and set accessor functions, also called getters and setters, allow you to adhere to the programming principles of information hiding and encapsulation while providing an easy-to-use programming interface for the classes that you create. Get and set functions allow you to keep your class properties private to the class, but allow users of your class to access those properties as if they were accessing a class variable instead of calling a class method. The advantage of this approach is that you can avoid having two public-facing functions for each property that allows both read and write access.

Listing 3.7: Class with attributes

```
public  class SERVER_v4 {

  private String[] ServiceAccessPoint;

  public String[] getServiceAccessPoint() {
    return ServiceAccessPoint;
  }

  public void setServiceAccessPoint(String[] serviceAccessPoint) {
    ServiceAccessPoint = serviceAccessPoint;
  }

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1
   * Explicado en la clusula 6.
   */
  public String GetServerDirectory(String param) {
    return null;
```

```
    }
}
```

Constructor methods

Constructor methods, sometimes simply called constructors, are functions that share the same name as the class in which they are defined. Any code that you include in a constructor method is executed whenever an instance of the class is created with the new keyword.

Listing 3.8: Class with attributes

```
public  class SERVER_v5 {

  public SERVER_v5() {
    // Default Constructor
  }

  public SERVER_v5(String[] serviceAccessPoint) {
    // Constructor using fields
    ServiceAccessPoint = serviceAccessPoint;
  }

  private String[] ServiceAccessPoint;

  public String[] getServiceAccessPoint() {
    return ServiceAccessPoint;
  }

  public void setServiceAccessPoint(String[] serviceAccessPoint) {
    ServiceAccessPoint = serviceAccessPoint;
  }

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
```

```
     * pgina 21, clusula 5.4, tabla 1
     * Explicado en la clusula 6.
     */
    public String GetServerDirectory(String param) {
      return null;
    }
}
```

3.4 Objects

A object is a agrupation of datas and operations with autonomous memory which should inside in the behabiour of each invocation. The object datas have a state that remember the effect of the operation [28]. and unlike a class, a object just exist on a running proccess.

A object are created from a class, in other words, a object is a instance of a class.

A O-O program interaction is realized by objects invoking methods.

3.4.1 Instance

por completar podrian servir como referencia los siguientes links (antes debo verificar si estan bien)

3.4.2 Reference

For comminf...

3.5 Diference betwen Classes and Objects

They are a clear separation between class and object, and they are (together with inheritance) the most important concepts of O-O programming languages [31].

A class is a static off-line template to generate objects. A object is a runtime data or a group of datas according to the class. The objects are independend datas with

its own behavior, and they are classified by the classes which generated.

ESP:por las clases q la han generado

3.6 The objects relationships

esto no va en la seccion de objetos, pues antes necesite explicar la sección 3.5 que trata sobre la diferencia entre objeto y clase. Esto es importante como una previa para esta seccion dado que las subsecciones a continuacion se presentan como clases pero describen las relaciones de los objetos. por eso es importante entender bien la diferencia primero. Esto es por el lector, para que le sea mas facil entender.

3.6.1 Association

completar - association

3.6.2 Aggregation

completar - aggregation

3.6.3 Composition

completar - composition

3.6.4 Other relationships

por el momento aca solo hay uno, por eso uso una subsubsection, pero es altamente probable que coloque mas cosas si es que encuentro que son utilizados en la norma.

Dependency

completar - dependency: la flechita del uml

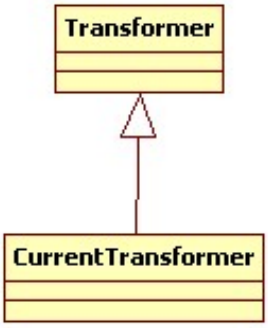

Figure 3-4: Inheritance: *Transformer* is the parent of *CurrentTransformer*

3.7 Inheritance

"The inheritance permit to a class to be used in a modified form when a sub-class is derived from it" [29]. This O-O resource can be used as a way to define new classes based in more general classes that have been defined previously, acquiring their characteristics. If a class *CurrentTransformer* directly inherits from class *Transformer* we say that *Transformer* is the parent of *CurrentTransformer* and *CurrentTransformer* is the child of *Transformer* [32]. The UML representation is given in Figure 3-4.

The inheritance is a great resource to create reusable code and common informations and methods thanks to hierarchical classes. Various techniques to write reusable code are avaliable [33] [34] [35] [36].

mejorar la abreb. de estas 2 primeras bibliografias. Los nombres completos pueden encontrarse en (ver link comentado en el codigo fuente)

Listing 3.9: Generic Substation Event Class

```
public class GSE{

}
```

Listing 3.10: GOOSE Class in Java

```
public abstract class GOOSE extends GSE {

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1 <br>
   * Explicado en la clusula 15.
   */
  public abstract String SendGOOSEMessage();

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1 <br>
   * Explicado en la clusula 15.
   */
  public abstract String GetGOReference();

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1 <br>
   * Explicado en la clusula 15.
   */
  public abstract String GetGOOSEElementNumber();

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1 <br>
   * Explicado en la clusula 15.
   */
  public abstract String GetGOCBValues();

  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1 <br>
   * Explicado en la clusula 15.
   */
  public abstract String SetGOCBValues();
```

```
}
```

3.8 Information hidding and encapsulation

3.8.1 Abstract data types

Abstraction and information hidding form the foundation of all object-oriented development [37]. An abstraction is a simplified description, or specification, of a system that emphasizes some of the system's details or properties while suppressing others [38]. Information hidding, as first promoted by Parnas, goes on to suggest that we should decompose systems based upon the principle of hidding design decisions about our abstractions [39] [40].

explicar mas c/mis palabras

The abstraction and information hidding are very common in electrical equipments and mathematical representations of the electrical world. The models are abstracted and is possible to identify the object and operations that exist at each level of integrations Thus, when

ver si las palabras estan bien escritas en este ejemplo

we work with phasors to represent a current, which leave just the static amplitude and phase information. The time space are hidden with the purpose to manage the information at a more hight level, thereby skipping the trigonometric calculations derived from the time dependence of the sine wave, and the information are combined just algebraically, simplifying certain kinds of complex calculations. [40]

The use of abstract data types on a object oriented system help to a more precise and at the same time simple on the specification taking adventage of the information hidding provided by abstract data types. []

aca debo citar mi paper sobre ACSI

Listing 3.11: Goose Abstract Class with attributes

```
public abstract class GOOSE extends GSE {
```

```
    /**
     * Definido en IEC 61850 parte 7-2 (2003)
     * pgina 21, clusula 5.4, tabla 1 <br>
     * Explicado en la clusula 15.
     */
    public abstract String SendGOOSEMessage();

    /**
     * Definido en IEC 61850 parte 7-2 (2003)
     * pgina 21, clusula 5.4, tabla 1 <br>
     * Explicado en la clusula 15.
     */
    public abstract String GetGOReference();

    /**
     * Definido en IEC 61850 parte 7-2 (2003)
     * pgina 21, clusula 5.4, tabla 1 <br>
     * Explicado en la clusula 15.
     */
    public abstract String GetGOOSEElementNumber();

    /**
     * Definido en IEC 61850 parte 7-2 (2003)
     * pgina 21, clusula 5.4, tabla 1 <br>
     * Explicado en la clusula 15.
     */
    public abstract String GetGOCBValues();

    /**
     * Definido en IEC 61850 parte 7-2 (2003)
     * pgina 21, clusula 5.4, tabla 1 <br>
     * Explicado en la clusula 15.
     */
    public abstract String SetGOCBValues();

}
```

3.8.2 Intefaces

A external interface is a group of methods declarations of a class that allows that serves to comunicate the class with other classes and thus between the designer of the class with another classes. The other clases should exist at the moment of the interface definition or could be created in the future [30, pp. 90-105] [32].

The interface of a class serves to know how to interact with a determined behabiour of the object, their input, ouput, and optionally the error handling parameters but not knowing how the behavior is implemented.

3.8.3 Encapsulation

The encapsulation is a technique for avoid dependencies among classes and modules by defining strict interfaces, and help for software maintenance, by information hidding, understandability of code, localization of information, narrow interfaces, and information restriction [41].

Listing 3.12: Goose Abstract Class with attributes

```
public class TimeStamp{

  private int SecondSinceEpoch;

  private int FractionOfSecond;

  private java.sql.Timestamp timeStamp;

  public TimeStamp(long arg0) {
    timeStamp = new java.sql.Timestamp(arg0);
  }


  public int getSecondSinceEpoch() {
```

```
    this.SecondSinceEpoch = timeStamp.getNanos();
    return this.SecondSinceEpoch;
  }

  public void setSecondSinceEpoch(int secondSinceEpoch) {
    this.SecondSinceEpoch = secondSinceEpoch;
  }

  public int getFractionOfSecond() {
    /*
     * TODO falta implementar, ver la parte 7-2,
     * clusula 5.5.3.7.3.2
     */
    return this.FractionOfSecond;
  }

  public void setFractionOfSecond(int fractionOfSecond) {
    this.FractionOfSecond = fractionOfSecond;
  }

}
```

3.8.4 Interfaces and implementation

modificar, reelaborar con mis palabras

Interfaces are based on the distinction between a methods interface and its implementation. A methods interface includes all the information necessary to invoke that method, including the name of the method, all of its parameters, and its return type. A methods implementation includes not only the interface information, but also the executable statements that carry out the methods behavior. An interface definition contains only method interfaces, and any class that implements the interface is responsible for defining the method implementations. [30, pp. 90-105]

agregar aca lo que subraye y resum en la segunda pagina del paper snyder:1986

Hacer la implementacion de esta interfaz

Listing 3.13: Class with attributes and methods in Java

```
/**
 * The SERVER class represents the external visible behaviour of
 * a device. All other ACSI models are part of the server.
 *  @category IEC 61850 parte 7-2 (2003)
 *  pgina 17, clusula 5.2 , prrafo
 *
 * @author David Prez
 *
 */

public abstract class SERVER_interface {

  private String[] ServiceAccessPoint;

  public String[] getServiceAccessPoint() {
    return ServiceAccessPoint;
  }
  public void setServiceAccessPoint(String[] serviceAccessPoint) {
    ServiceAccessPoint = serviceAccessPoint;
  }
  /**
   * Definido en IEC 61850 parte 7-2 (2003)
   * pgina 21, clusula 5.4, tabla 1
   * Explicado en la clusula 6.
   */
  public abstract String GetServerDirectory(String param);
}
```

3.9 Events

completar - event definition

For comming...

3.9.1 Event Dispatching

For comming...

3.9.2 Event Listener

For comming...

3.10 Exception handling

The exception is an unwanted and unexpeceted event which occours at runtime, and are caused by an unusual situation of the software that stop the normal secuence of software operations.

When a exception event are dispatched by the system, the error could be handled by a specific modelled object if the software developer anticipated it as a posibility.

The exception could be modelled with different handling stategies, [42] [43] [**?**] [44]

buscar mas papers de referencia para colocar aqui por si alguien tenga interes de profundizar en esto (buscar en la acm)

for example, could be represented by classes specifically designed for this purpose of which concrete exceptions would be some referenceable instances that handles the event of error.

Listing 3.14: Service Error Class in Java

```
public class ServiceError extends Exception {

  /**
   * Error de la respuesta al consumirse el
   * servicio de Logical Device
```

```
     */
    private static final long serialVersionUID = 1L;

    public ServiceError(String message) {
      super(message);
    }
}
```

61850, parte6, cl6.1, ¶3

3.11 Serialization

este parrafo a continuacion lo explique con mis palabras, debo buscar algun paper que habla con un lenguaje mas tecnico para apoyar mi explicacion y agregar las referencias correspondientes

Serialization, or marshalling is the process of persist the object in memory or for transmission purpose, by creating a human readable archive that containts the object structure and datas in a saveable way. A object just exist at runtime, and it is serializad to be saved over the time or to be alternativatelly retransmited by the network to another host.

3.12 Object oriented develoment approach

The object-oriented development should follow the steps proposed by Aboott [45]:

- Identify the objects and their attributes.
- Identify the operations suffered by and required of each object.
- Establish the visibility of each object in relation to other objects.
- Establish the interface of each object.
- Implement each object.

Chapter 4

IEC 61850: Technical overview

4.1 Objects for distributed systems

The effective distribution of Logical Nodes on diferents IEDs is a reality thanks to researches about the structure of distributed systems. More than 20 years ago emerged requirements for the object paradigm to suport the design and development of distributed systems.

Theses quotes were extracted from Jazayeri 1988 research:

"An object on one node can send a (multicast) message to several other objects ..." (MCAA)

completar y ver si esta bien

" ... The ability to group a set of objects and address them as one entity is important in many applications both from an efficiency point of view and from a program structuring point of view ..." (DO, DATA-SET, FCD, FCDA)

completar y ver si esta bien

" ... a final difference is that our objects are active and not reactive, in the sense that they can start up spontaneosly performing operations, not necessarily only in response to method invocations. Such a facility is useful, for example, to allow objects to monitor the enviroment and change their behavior based on changes in the enviroment ..." (Some active objects are GOOSE, URCB and some passive object are)

completar y ver si esta bien

[46].

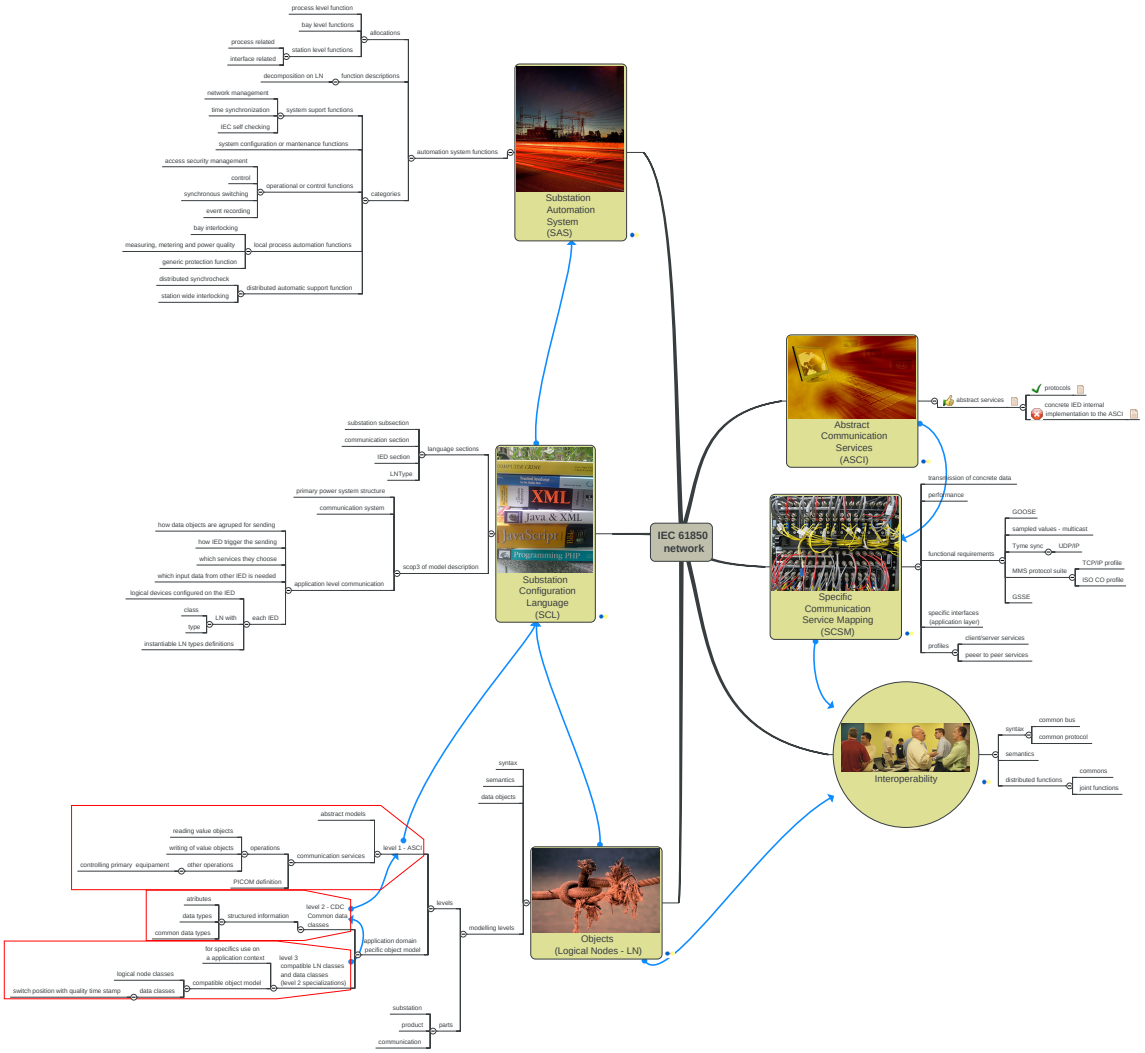

Figure 4-1: Borrador - Parte del esquema del futuro capitulo

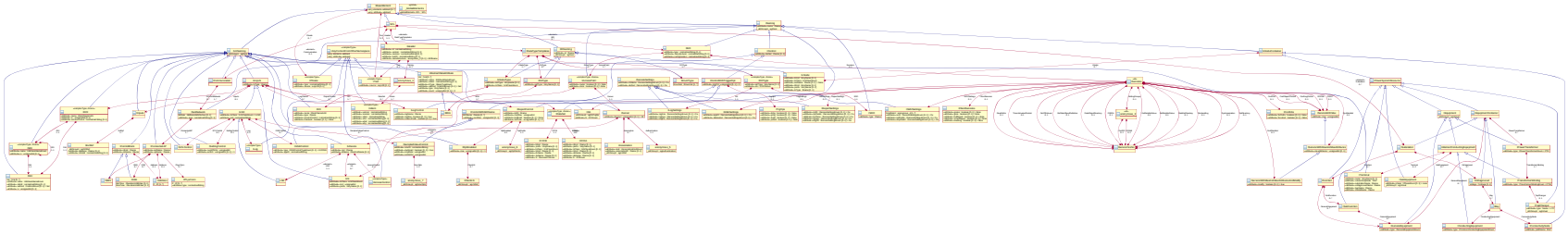

Figure 4-2: Borrador - Parte del esquema del futuro capitulo

Chapter 5

Substation Configuration Language

5.1 SCL object model

As explained in the section **??**, the SCL describes the SAS by modelling objects. The standard IEC 61850-6 [1] object model structure and constraints are described in terms of the XML Schema Definition (XSD).

The XSD of the Substation Configuration description Language (SCL) are structured as depicted in the figure .

Basically, it is composed by:

61850, parte6, cl6.1, ¶5

- The substation model: the object model of the primary power structure (a instance of tSubstation) with their designations structured according to IEC 61346-1 [47].
- The communication model: the object model of the IED communication system configuration, i.e., the the networks, subnetworks, ports informations , the communication connection relations of IEDs to subnetworks, the routing for another subnetworks informations, and clocks configuration information and locations for time synchronisation (the gateways are not considered here, a gateway has to be modelled as another IED) .

61850, parte6, cl6.1, ¶5

61850, parte6, cl6.1, ¶8

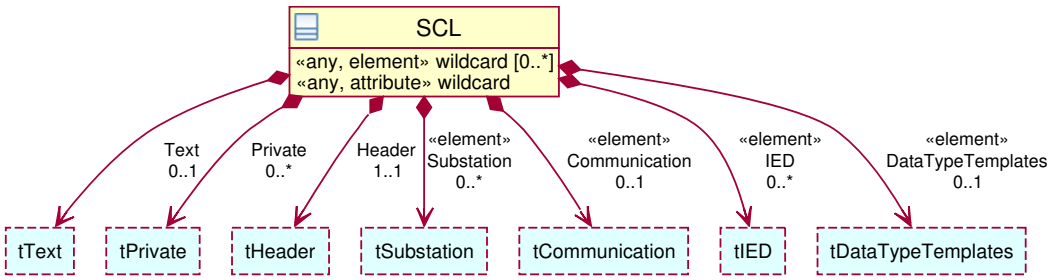

Figure 5-1: SCL object model

- The product model: Contains the IEDs objects and their logical node implementations.

  averiguar la funcion del DataTypeTemplates para agregar aca si corresponde

All the structures are derived from differents parts depicted in the figure 5-1. In this figure, each class represent the top level of their structure, e.g., the substation class mentioned here has voltage levels, connectivity nodes and tranformers. Theses classes are completely decoupled (the most notable decoupling are the diferentiation of Logical node classes on the IED -tLogical Node (LN)- and Substation -tLogical Node (LNode)- structure), allowing the separation of the SCL development by the differents engineering process described in the section **??**.

The SCL relationship are based in a ring O-O architecture between Substation, IED, Header, Communication, and DataTypeTemplate classes, and it are based in a hierarchical structure under each of the mentioned classes.

5.1.1 SCL modelling steps

esta seccion la dejo pendiente, debo escribirlo con detenimiento para que queden en concordancia la figura 5-1 y lo que tengo escrito en esta seccion

The figure **??** show the detailed SAS engineering process depicted with the autor interpretation of the step mentioned in the standard by using an UML secuence diagram.

The more usual approach for the Substation Automation System (SAS) description with SCL begin with the single line drawing process where the substation topology are defined using a system configurator tool. The system configurator tool, described in the section **??**, creates the System Specification Description (SSD) file containing the LNodes and their allocation in the substation. The class diagram provided by the author are based on the SCL XSD's of the substation. (figures 5-2 and 5-3).

The LNodes objects SCL models (from the substation section) is used to map the

IED object model references (figure **??**) that describes the IED objects are used to mapping

apuntar a labelfig:pdf-SCL-uml-IED-Deept2

a este le falta expandir la clase correspondiente hasta que muestre el transformador

a este le valta expandir la clase correspondiente hasta que muestre el transformador

terminar bien este parrafo...

aqui falta el uml del IED con detalles de herencia

debo agregar aca la explicacion de los demas modelos descriptos mediance los XSDs, pues esos tres items que menciona la norma son solo los mas importantes, pero no da para entender y saber leer los archivos scl sabiendo solo eso. Los tres items mencionados arriba son solo la base. Falta principalmente el DATypeTemplate que corresponden a los objetos a serializar correspondientes a las instancias de las clases de la parte 7-x

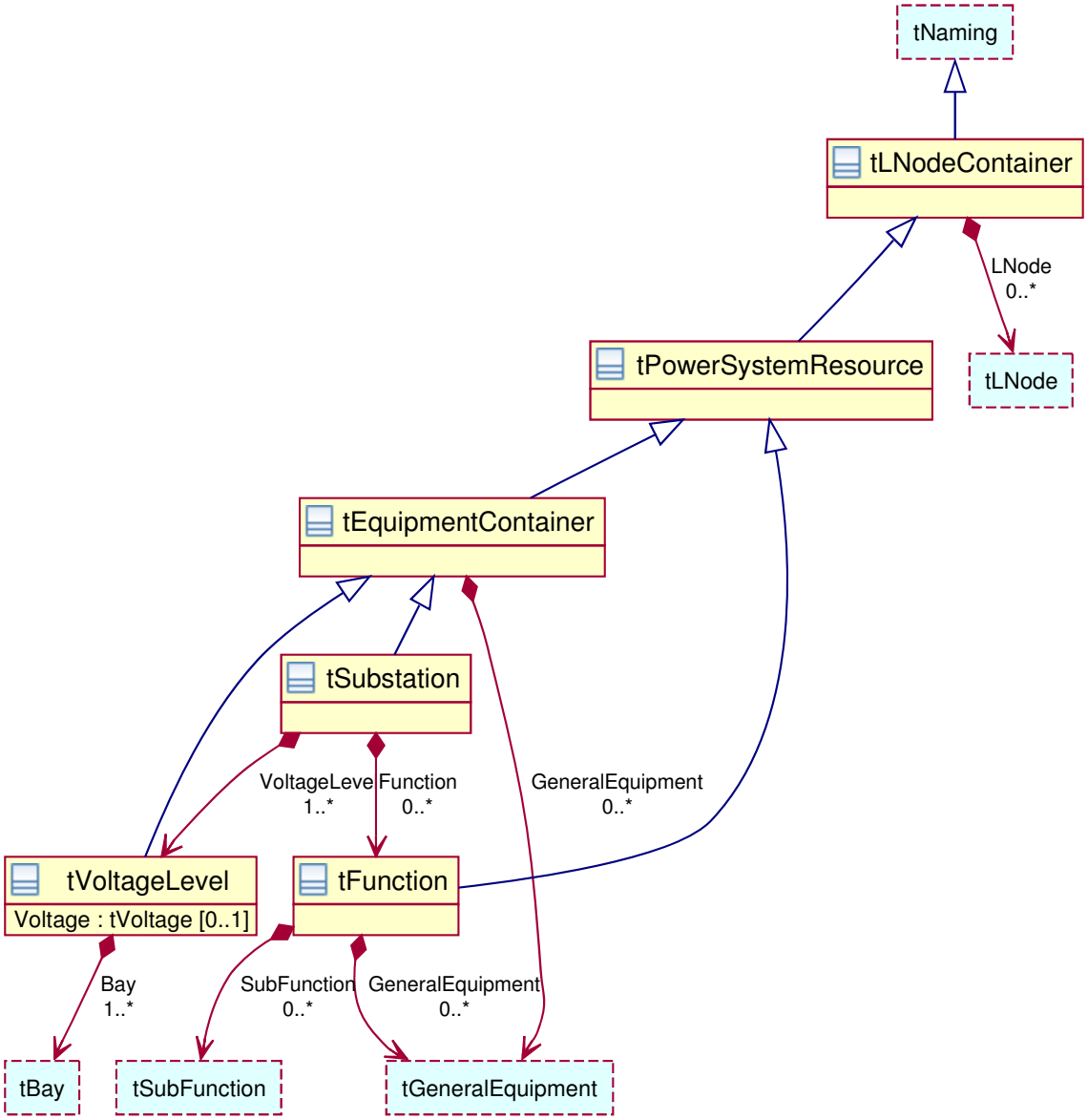

Figure 5-2: SCL Substation class diagram with heritance details

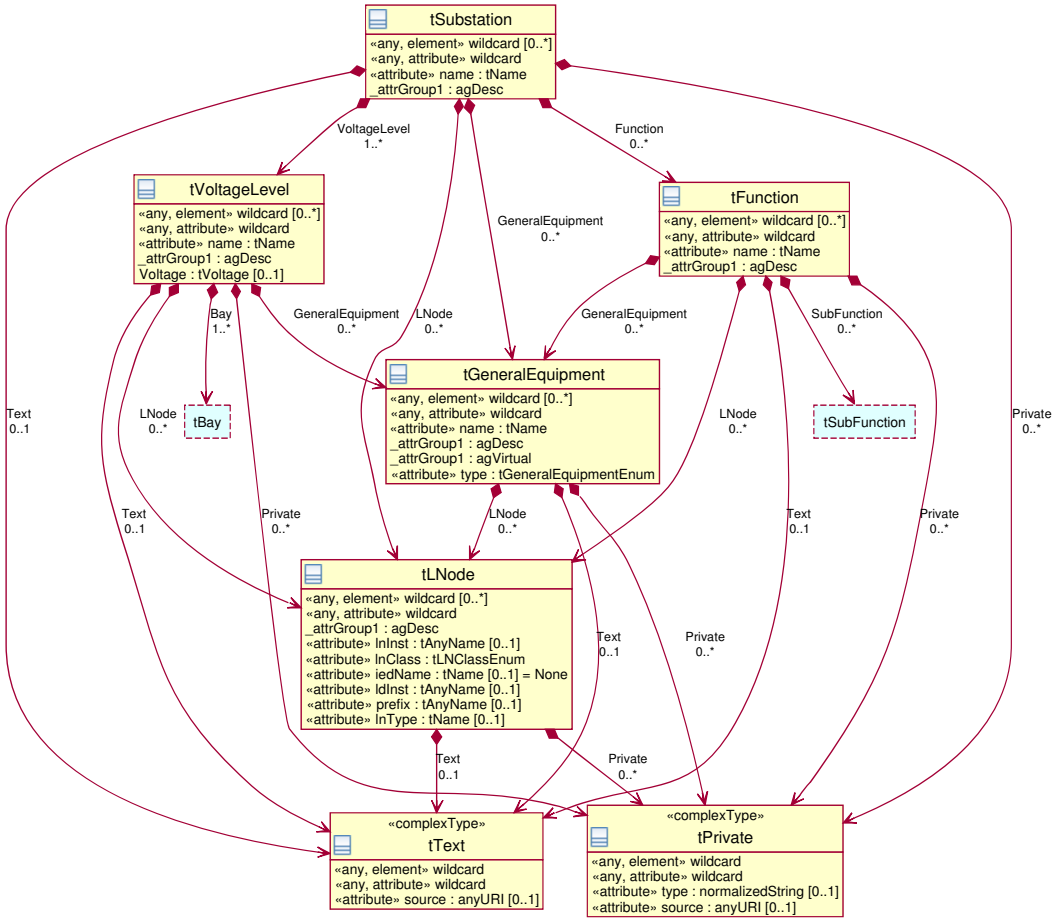

Figure 5-3: SCL Substation class diagram inherited

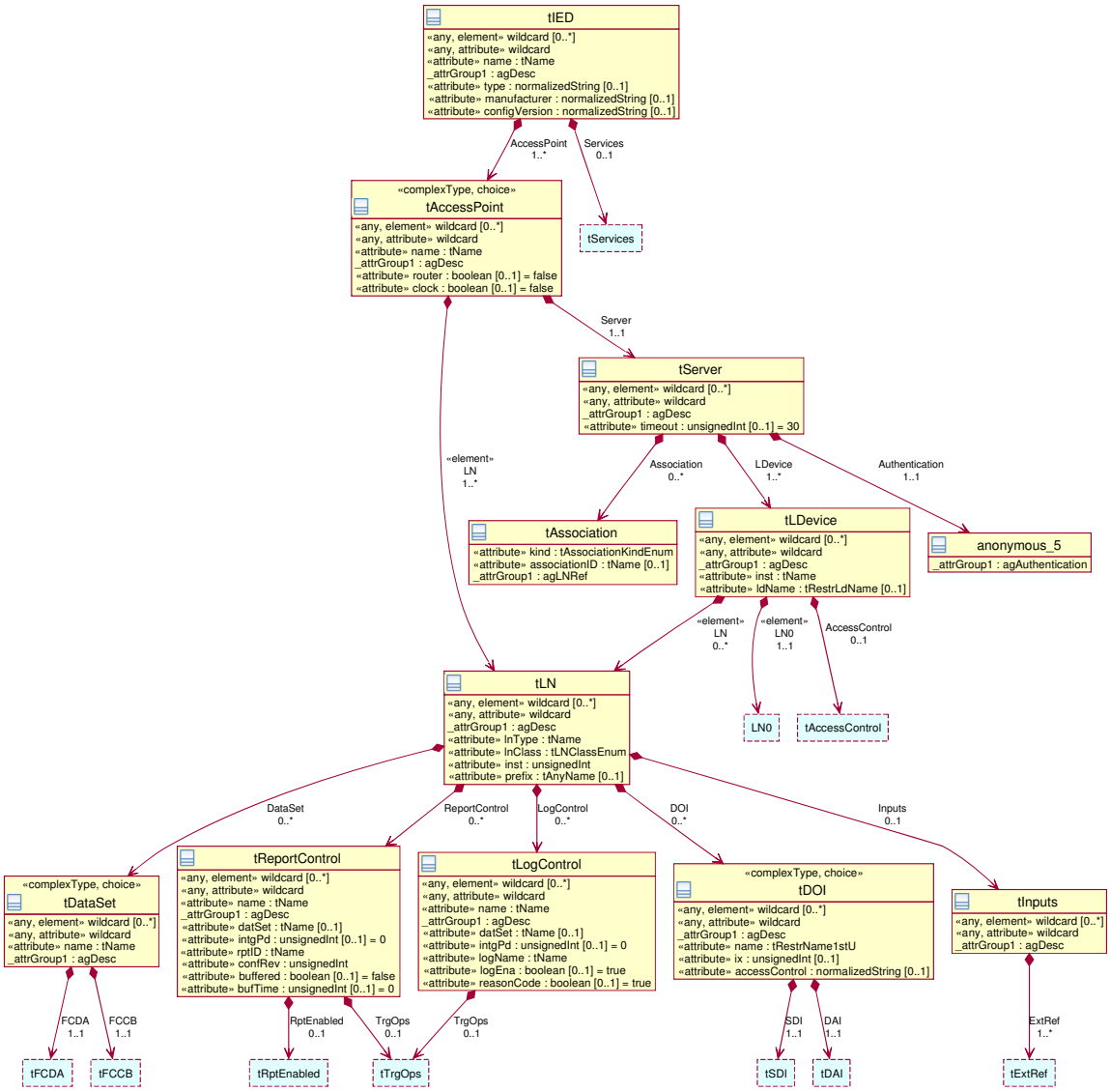

Figure 5-4: SCL Substation class diagram inherited

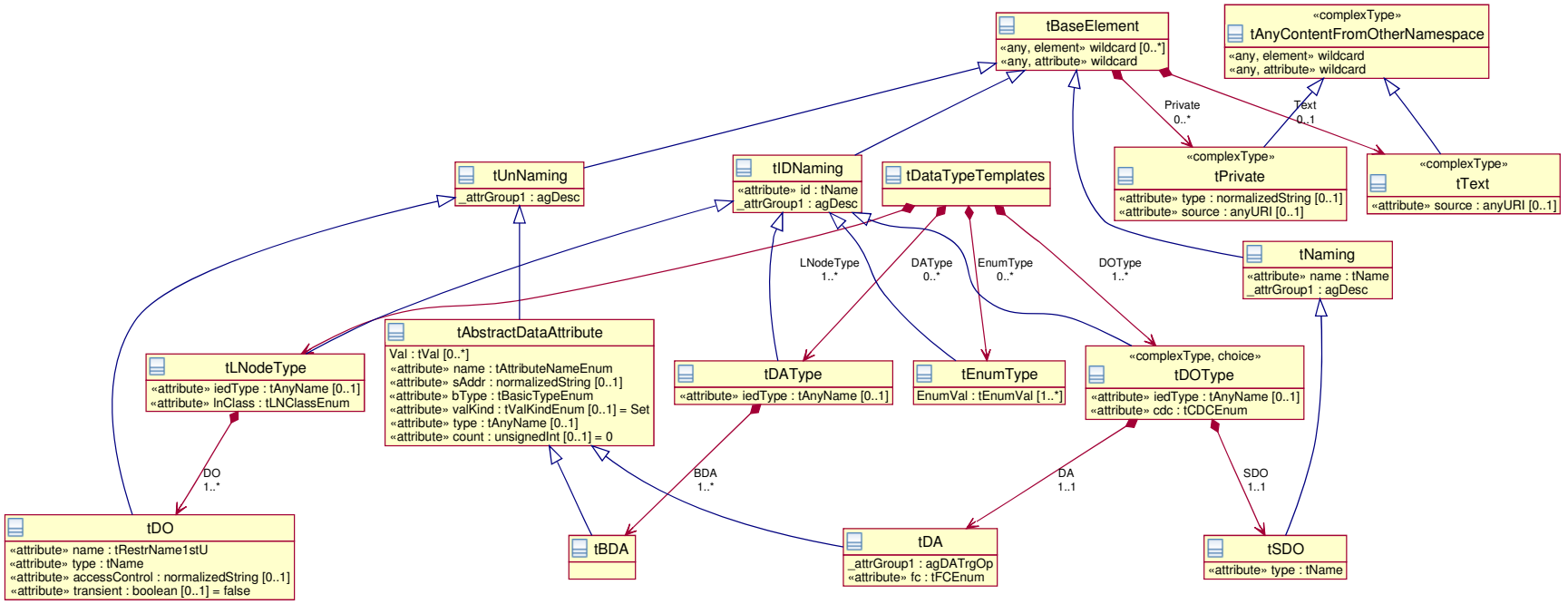

Figure 5-5: DAType class diagram template with heritance details

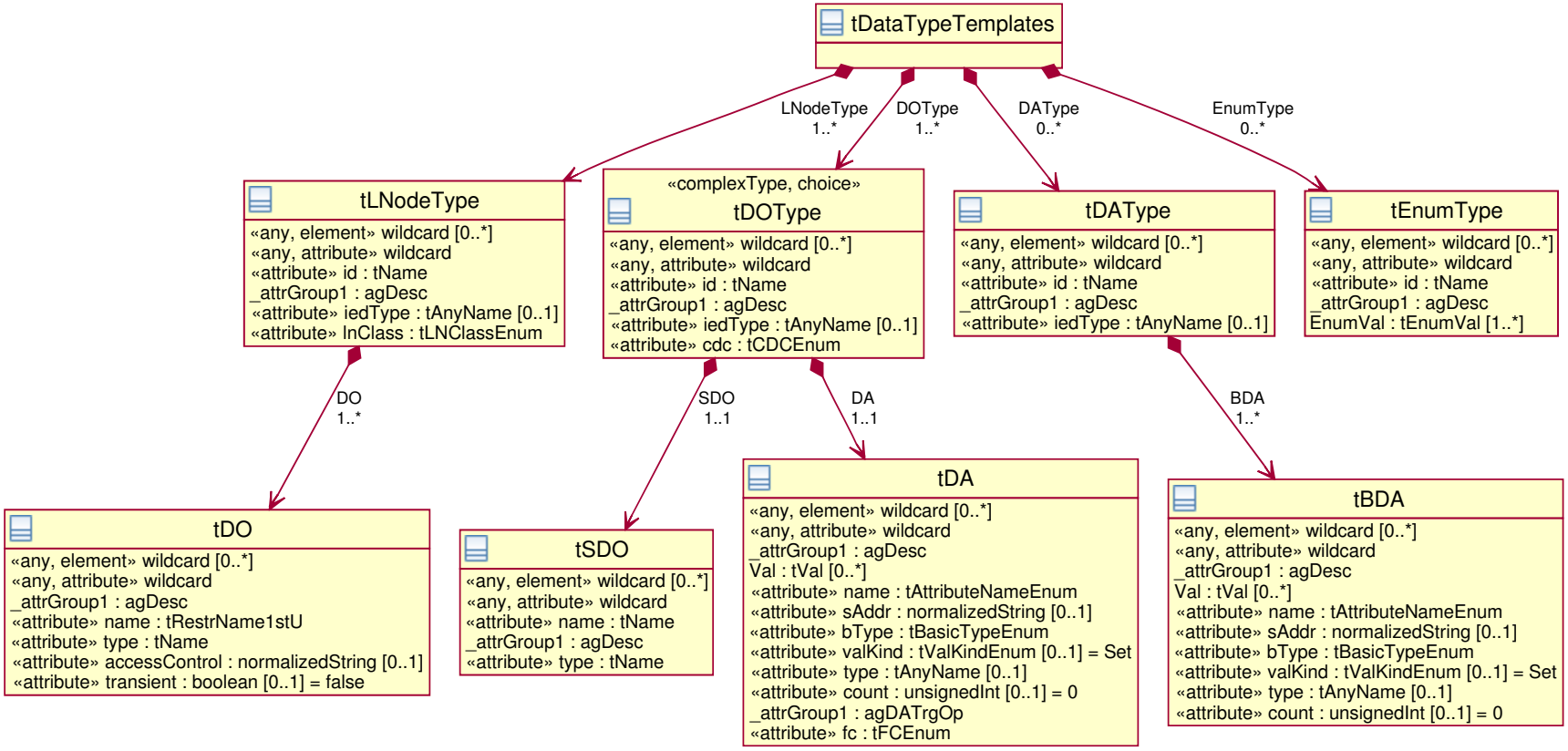

Figure 5-6: DAType class diagram template inherited

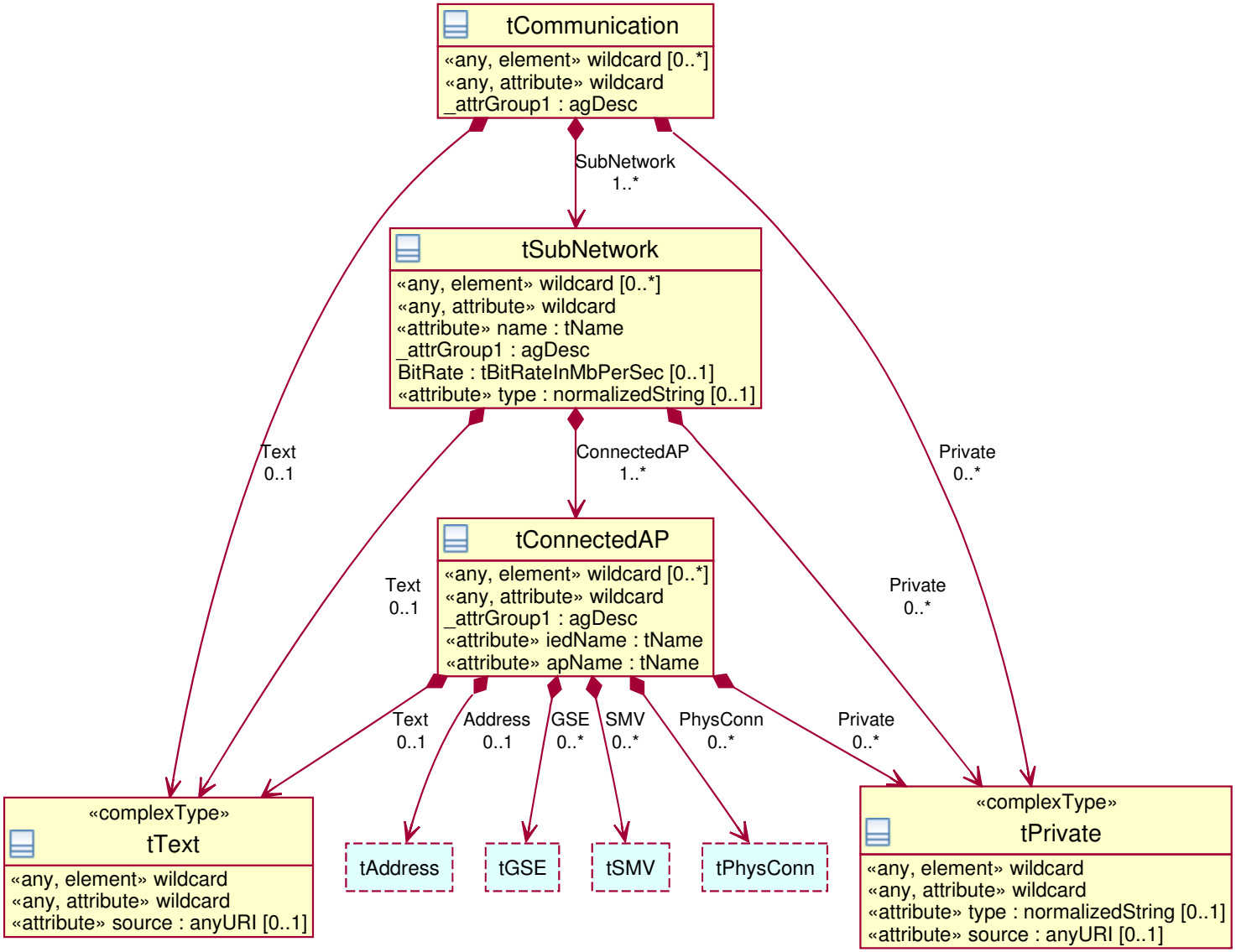

Figure 5-7: SCL Communication class diagram

The UML of the figure 5-8 is a resume of the SCL model, where is evident the key importance of the Logical Node for the information topology description (see the composition of the Substation class). The Logical Node is the transition object to connect the diferent structures of the SAS that are defined by IEC 61341-1 [47].

61850, parte6, cl6.1, ¶9

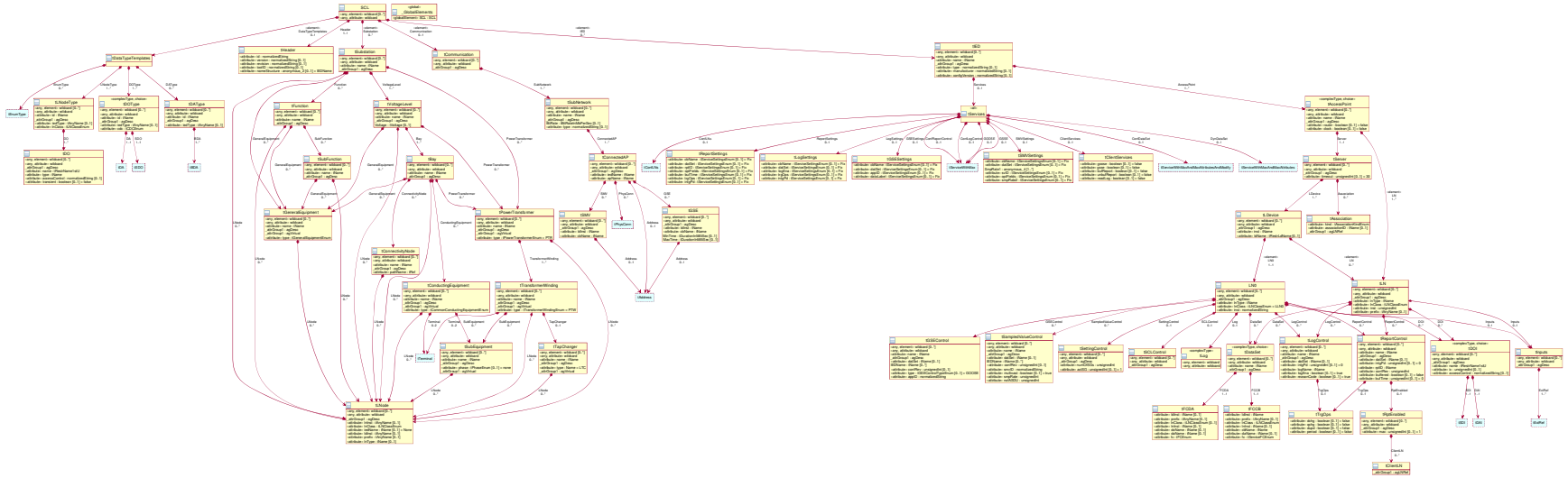

Figure 5-8: Resume of SCL shema represented by a UML class diagram

5.2 A practical object and communication services modelling example with SCL

5.2.1 Namespaces

This element must appear first among all sub-elements, and provide information about the SCL's XSD to be assigned in the SCL engineering process.

Listing 5.1: IEC 61850 SCL Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<SCL xmlns="http://www.iec.ch/61850/2006/SCL"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.iec.ch/61850/2006/SCL ../../scl/SCL
      .xsd">
```

5.2.2 Header

The header identify a SCL file, gives information about the version of the SCL file, and the tool used for the SCL engineering.

Listing 5.2: SCL header

```
<Header id="SCL Example T1-1"
    nameStructure="IEDName"
    revision="R01.1"
    toolID="XMLSpy"
    version="0.1"/>
```

5.2.3 Substation

The substation xml node group the primary power system topology and their respective functions. The substation model are described in the section 5.1.

Listing 5.3: SCL Substation node

```
<Substation name="S12" desc="Baden">
  <!-- childs here -->
</Substation>
```

5.2.4 Voltage Level

The VoltageLevel tag allow the agrupation of various levels of switchyard equipments in the same potential. In the listing 5.4 are defined a first level of voltage called "*D1*":

Listing 5.4: SCL Substation voltage level D1

```
<VoltageLevel name="D1" desc="Voltage level of the substation">
  <!-- childs here -->
</VoltageLevel>
```

and the second level of voltage of this example is provided in the listing 5.5:

Listing 5.5: SCL Substation voltage level E1

```
<VoltageLevel name="E1" desc="Voltage level of the substation">
  <!-- childs here -->
</VoltageLevel>
```

grouping theses voltage levels into the substation:

Listing 5.6: SCL Substation voltage levels

```
<Substation name="S12" desc="Baden">
  <VoltageLevel name="D1" desc="Voltage level of the substation">
    <!-- childs here -->
  </VoltageLevel>
  <VoltageLevel name="E1" desc="Voltage level of the substation">
    <!-- childs here -->
  </VoltageLevel>
</Substation>
```

and finally, assigning determined voltages to the diferents voltage levels:

Listing 5.7: SCL Substation voltage levels containing their respective voltage information

```
<Substation name="S12" desc="Baden">
  <VoltageLevel name="D1" desc="Voltage level D1 of the substation">
    <Voltage multiplier="k" unit="V">220</Voltage>
    <!-- more childs here -->
  </VoltageLevel>
  <VoltageLevel name="E1" desc="Voltage level E1 of the substation">
    <Voltage multiplier="k" unit="V">132</Voltage>
    <!-- more childs here -->
  </VoltageLevel>
</Substation>
```

5.2.5 Bay

A bay object modelling is analogue to the voltage level. A voltage level can contain several bays. The bay object are represented here:

Listing 5.8: Bay

```
<Bay name="Q1">
  <!-- childs here -->
</Bay>
```

And here several bays of a unique voltage level.

Listing 5.9: Bay as child of the voltage level D1

```
<VoltageLevel name="E1" desc="Voltage level of the substation">
  <!-- childs here -->
  <Bay name="Q1">
    <!-- childs here -->
  </Bay>
  <Bay name="Q2" desc="Turgi">
    <!-- childs here -->
  </Bay>
```

```
  <Bay name="Q3" desc="London">
    <!-- childs here -->
  </Bay>
</VoltageLevel>
```

---

and 4 bays in a substation. The references to theses bays are:

- S12/D1/Q1
- S12/E1/Q1
- S12/E1/Q2
- S12/E1/Q3

Note that the bay S12/D1/Q1 is different that the bay S12/E1/Q1. Both of them have the name Q1, but they have different paths. Then, theses references do not point to the same object.

Listing 5.10: 2 Bays in the Substation context.

```
<Substation name="S12" desc="Baden">
  <!-- childs here -->
  <VoltageLevel name="D1" desc="Voltage level D1 of the substation">
    <!-- childs here -->
    <Bay name="Q1">
      <!-- childs here -->
    </Bay>
  </VoltageLevel>
  <VoltageLevel name="E1" desc="Voltage level E1 of the substation">
    <!-- childs here -->
    <Bay name="Q1">
      <!-- childs here -->
    </Bay>

    <Bay name="Q2" desc="Turgi">
      <!-- childs here -->
    </Bay>
```

```
      <Bay name="Q3" desc="London">
        <!-- childs here -->
      </Bay>
    </VoltageLevel>
  </Substation>
```

5.2.6 Conducting Equipment

The conducting equipment are completar...

5.2.7 Complete SSD

The complete SSD are provided here:

Bibliography

[1] IEC-TC57, *IEC 61850-6: Communication networks and systems in substations - Part 6 - Configuration description language for communication in electrical substations related to IEDs*, IEC Std., Rev. IS Ed1:2004-03, 2004.

[2] O. M. G. ™, *Unified Modeling Language*, OMG Std., Rev. 2.2, 2009.

[3] IEC-TC57, *IEC 61850-1: Communication networks and systems in substations - Part 1 - Introduction and overview*, IEC Std., Rev. TR Ed1:2003-04, 2003.

[4] ——, *IEC 61850-2: Communication networks and systems in substations - Part 2 - Glossary*, IEC Std., Rev. TS Ed1:2003-08, 2003.

[5] ——, *IEC 61850-3: Communication networks and systems in substations - Part 3 - General requirements*, IEC Std., Rev. IS Ed1:2002-02, 2002.

[6] ——, *IEC 61850-4: Communication networks and systems in substations - Part 4 - System and project management*, IEC Std., Rev. IS Ed1:2002-01, 2002.

[7] ——, *IEC 61850-5: Communication networks and systems in substations - Part 5 - Communication requirements for functions and device models*, IEC Std., Rev. IS Ed1:2003-07, 2003.

[8] International Electrotechnical Commission - Technical Committee 57: Power systems management and associated information exchange, *IEC 61850-7-1: Communication networks and systems in substations - Part 7-1 - Basic communication structure Principles and models*, IEC Std., Rev. IS Ed1:2003-07, 2003.

[9] ——, *IEC 61850-7-2: Communication networks and systems in substations - Part 7-2 - Basic communication structure Abstract communication service interface (ACSI)*, IEC Std., Rev. IS Ed1:2003-05, 2003.

[10] ——, *IEC 61850-7-3: Communication networks and systems in substations - Part 7-3 - Basic communication structure Common data classes*, IEC Std., Rev. IS Ed1:2003-05, 2003.

[11] ——, *IEC 61850-7-4: Communication networks and systems in substations - Part 7-4 - Basic communication structure Compatible logical node classes and data classes*, IEC Std., Rev. IS Ed1:2003-05, 2003.

[12] ——, *IEC 61850-7-410: Communication networks and systems in substations - Part 7-410 - Hydroelectric power plants - Communication for monitoring and control*, IEC Std., Rev. IS Ed1:2007-08, 2007.

[13] ——, *IEC 61850-7-420: Communication networks and systems in substations - Part 7-420 - Communications systems for distributed energy resources (DER) - Logical nodes*, IEC Std., Rev. IS Ed1:2009-03, 2009.

[14] ——, *IEC 61850-7-430: Communication networks and systems in substations - Part 7-430 - Communication system for distribution feeder and network equipment*, IEC Std., Rev. 57/954/NP.

[15] ——, *IEC 61850-7-5: Communication networks and systems in substations - Part 7-5 - Basic communication structure Usage of information models for substation automation applications*, IEC Std., Rev. DC 2010-08, 2010.

[16] ——, *IEC 61850-7-500: Communication networks and systems in substations - Part 7-500 - Use of logical nodes to model functions of a substation automation system*, IEC Std., Rev. DC 2010-08, 2010.

[17] ——, *IEC 61850-7-510: Communication networks and systems in substations - Part 7-510 - Use of logical nodes to model functions of a hydro power plant*, IEC Std., Rev. DC 2009-12, 2009.

[18] ——, *IEC 61850-7-520: Communication networks and systems in substations - Part 7-520 - Use of logical nodes to model functions of distributed energy resources*, IEC Std., Rev. Draft 2010, 2010.

[19] ——, *IEC 61850-7-10: Communication networks and systems in substations - Part 7-10 - Web-based and structured access to the IEC 61850 information models*, IEC Std., Rev. DC 2009-12, 2009.

[20] ——, *IEC 61850-8-1: Communication networks and systems in substations - Part 8-1 - Specific communication service mapping (SCSM) Mappings to MMS (ISO/IEC 9506-1 and ISO/IEC 9506-2) and to ISO/IEC 8802-3*, IEC Std., Rev. IS Ed1:2004-05, 2004.

[21] ——, *IEC 61850-9-1: Communication networks and systems in substations - Part 9-1 - Specific communication service mapping (SCSM) Sampled values over serial unidirectional multidrop point to point link*, IEC Std., Rev. IS Ed1:2003-05, 2003.

[22] ——, *IEC 61850-9-2: Communication networks and systems in substations - Part 9-2 - Specific communication service mapping (SCSM) Sampled values over ISO/IEC 8802-3*, IEC Std., Rev. IS Ed1:2004-04, 2004.

[23] S. Microsystems. (1996) The java language specification, third edition. [Online]. Available: http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html

[24] T. Rentsch, "Object oriented programming," *SIGPLAN Not.*, vol. 17, no. 9, pp. 51–57, 1982.

[25] P. G. A., "Elements of object-oriented programming," *Byte*, vol. 11, no. 8, pp. 139–144, 1986.

[26] K. Nygaard, "Basic concepts in object oriented programming," in *Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming*. New York, NY, USA: ACM, 1986, pp. 128–132.

[27] O. L. Madsen and B. Møller-Pedersen, "What object-oriented programming may be - and what it does not have to be," in *ECOOP '88: Proceedings of the European Conference on Object-Oriented Programming*. London, UK: Springer-Verlag, 1988, pp. 1–20.

[28] P. Wegner, "Dimensions of object-based language design," *SIGPLAN Not.*, vol. 22, no. 12, pp. 168–182, 1987.

[29] L. F. Capretz, "A brief history of the object-oriented approach," *SIGSOFT Softw. Eng. Notes*, vol. 28, no. 2, p. 6, 2003.

[30] *Programming Adobe®ActionScript®3.0*, flash_as3_programming.pdf, Adobe Systems Incorporated, 2008.

[31] O.-J. Dahl and K. Nygaard, "Simula67 common base language," *Oslo: Norwegian Computing Centre*, no. S-22, 1970.

[32] A. Snyder, "Encapsulation and inheritance in object-oriented programming languages," in *OOPLSA '86: Conference proceedings on Object-oriented programming systems, languages and applications*. New York, NY, USA: ACM, 1986, pp. 38–45.

[33] J. R. E. and F. B., "Designing reusable classes," *Journal of Object-Oriented Programming*, vol. 1, no. 2, pp. 22–35, 1988.

[34] M. J., "Encapsulation, reusability and extensibility in object-oriented programming languages," *Journal of Object-Oriented Programming*, vol. 1, no. 1, pp. 12–36, 1988.

[35] S. Gossain and B. Anderson, "An iterative-design model for reusable object-oriented software," in *OOPSLA/ECOOP '90: Proceedings of the European conference on object-oriented programming on Object-oriented programming systems, languages, and applications*. New York, NY, USA: ACM, 1990, pp. 12–27.

[36] L. F. Capretz and P. A. Lee, "Reusability and life cycle issues within an object-oriented methodology," in *TOOLS 8: Proceedings of the eighth international conference on Technology of object oriented languages and systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992, pp. 139–150.

[37] H. Levy, *Capability Based Computer Systems.* Bedford, MA: Digital Press, 1984.

[38] M. Shaw, "Abstraction techniques in modern programming languages," *IEEE Software*, vol. 1, pp. 10–26, 1984.

[39] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," pp. 139–150, 1979.

[40] G. Booch, "Object-oriented development," *IEEE Trans. Softw. Eng.*, vol. SE-12, Nro 2, Feb./Oct. 1986.

[41] K. Lieberherr, I. Holland, and A. Riel, "Object-oriented programming: an objective sense of style," *SIGPLAN Not.*, vol. 23, no. 11, pp. 323–334, 1988.

[42] D. Weinreb, D. Moon, and R. M. Stallman, *Lisp Machine Manual*, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1983.

[43] C. Dony, "An object-oriented exception handling system for an object-oriented language," in *ECOOP '88: Proceedings of the European Conference on Object-Oriented Programming.* London, UK: Springer-Verlag, 1988, pp. 146–161.

[44] G. T. Leavens, "Introduction to the literature on object-oriented design, programming, and languages," *SIGPLAN OOPS Mess.*, vol. 2, no. 4, pp. 40–53, 1991.

[45] R. J. Abbot, "Report on teaching ada," Science applications, Inc., Tech. Rep. 129, 1980.

[46] M. Jazayeri, "Objects for distributed systems," in *OOPSLA/ECOOP '88: Proceedings of the 1988 ACM SIGPLAN workshop on Object-based concurrent programming.* New York, NY, USA: ACM, 1988, pp. 117–119.

[47] IEC, *IEC 61346-1 (1996): Industrial systems, installations and equipment and industrial products Structuring principles and reference designations*, Std., 1996.