.

.

.

.

## Agregar la sección del cover

Debo agregar una hoja donde indique que estos capítulos son el marco teorico de la investigación, dicho comando para agregar la palabra parte A por ejemplo, sin numeración de paginas en esta, voy a encontrar en el formato de tesis de la universidad de washintong basado en la del MIT, dicho template esta en mis archivos de Google Docs

# Chapter 1

# Object-oriented system construction

## 1.1 Introduction

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

No se si esta bien escrita la palabra standarized

.

The chapter does not describe all the O-O principles, just focuses on the neccesary background to understand the IEC 61850 engineering.

A detailed description of the O-O fundamentals and reference materials are provided. To achieve a robust comprehension of the concepts a practical implementation with the Unified Modeling Language (UML) and Java are provided. This chapter is destined for electrical engineers and professionals of related areas which the O-O software construction is not part of the curriculum. For this reason, the practical examples uses electricity area elements.

Me falta explicar que los ingenieros electricos generalmente no tienen base en programacin orientada a objetos, por eso decid dedicar todo un captulo para O-O systems. Sin un robusto conocimiento de esta etapa es casi imposible conocer a profundidad el modelo de informacin de la norma

## 1.2 Object Oriented systems

Aca debo contar la historia, y tambin explicar que es un OO system

In the decade of...

## 1.3 Abstract data types

explicar mas c/mis palabras

Abstraction and information hidding form the foundation of all object-oriented development [?]. An abstraction is a simplified description, or specification, of a system that emphasizes some of the system's details or properties while suppressing others [?]. 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 [?] [?].

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 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. [?]

The abstraction and information hidding are common in our activities, we abstract the models by identify the object and operations that exist at each level of integration.

Thus, when [?, inline] working a transformer, we consider the taps, the current on the low and hight side, the transformation relation [?].

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

## 1.4 Object oriented develoment approach

The object-oriented development steps, first proposed by Aboott [?] are as follows:

- 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.

## 1.5 Basics of object-oriented programming

### 1.5.1 Introduction to object-oriented programming

Object-oriented programming (OOP) is a way of organizing the code in a program by grouping it into objects-individual elements that include information (data values) and functionality. Using an object-oriented approach to organizing a program allows you to group particular pieces of information (for example, a automation function or a current value) together with common functionality or actions associated with that information (such as “switchgear actuation” or “voltage measurement”). These items are combined into a single item, an object (for example, an “Album” or “MusicTrack”). Being able to bundle these values and functions together provides several benefits, including only needing to keep track of a single variable rather than multiple

cambiar por un ejemplo eléctrico

ones, organizing related functionality together, and being able to structure programs in ways that more closely match the real world.

### 1.5.2 Common object-oriented programming tasks

In practice,

- Defining classes
- Creating properties, methods, and get and set accessors (accessor methods)
- Controlling access to classes, properties, methods, and accessors
- Creating static properties and methods
- Creating enumeration-like structures
- Defining and using interfaces
- Working with inheritance, including overriding class elements

## 1.6 Classes

A class is an abstract representation of an object. A class stores information about the types of data that an object can hold and the behaviors that an object can exhibit.

### 1.6.1 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.

#### 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

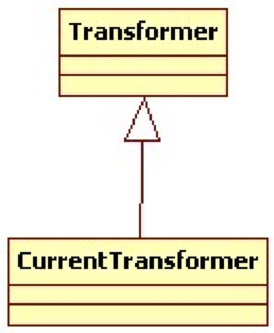

Figure 1-1: Inheritance: *Transformer* is the parent of *CurrentTransformer* and *CurrentTransformer* is the child of *Transformer*

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.

**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.

## 1.7 Inheritance

The inheritance 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*. The UML representation is given in Figure 1-1.

## 1.8 Intefaces

An interface is a collection of method declarations that allows unrelated objects to communicate with one another.

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. [1, pp. 90-105]

# Bibliography

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