Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 19 (2015) 651 – 656

Procedia Technology

8th International Conference Interdisciplinarity in Engineering, INTER-ENG 2014, 9-10 October 2014, Tirgu-Mures, Romania

Object-oriented modelling applied to electricity critical infrastructures

Olga Bucovetchi[a]*, Cristina Petronela Simion[a], Radu D. Stanciu[a]

[a]*University POLITEHNICA of Bucharest, 313 Spl. Independentei, 060042 Bucharest, Romania*

Abstract

Given the current social and economic environment, critical infrastuctures and their protection are major debate topics. The present paper provides a methodological framework for risk assessement within the energy sector as critical infrastructure sector. The paper embedded the key concepts and key mechanisms of object-oriented modelling in order to build a risk multimodel that can be used as starting point in future protection strategies.

Peer-review under responsibility of "Petru Maior" University of Tirgu Mures, Faculty of Engineering

Keywords: electricity critical infrastructure; risk multimodel; risk assessment; critical infrastructures protection.

1. Introduction

Critical infrastructures have always been a sensitive domain. The sensitivity derives from their special role in the structure, stability and function of a system, no matter what system or process is taken into consideration. Their vulnerability is proportional to the role played by each infrastructure. Due to this vulnerability, critical infrastructures are the first to be targeted when aiming to destabilize and even destroy a system or a process. Identifying, optimizing and protecting critical infrastructure is an unquestionable priority for both systems and processes management, as well as for their opponents, i.e. for those who seek to attack, destabilize and destroy the systems and processes. Critical infrastructures are not becoming critical only due to attacks, but also from other triggers, some of it being difficult to detect and analyze.

* Corresponding author. Tel.: +40-21-402-3818.
E-mail address: olga.bucovetchi@upb.ro

Peer-review under responsibility of "Petru Maior" University of Tirgu Mures, Faculty of Engineering
doi:10.1016/j.protcy.2015.02.092

Although there are different approaches on the definition of critical infrastructures concept when looking for the regulations and the literature surveys in USA [1], Canada [2], Australia [3] on one hand and in the European Union [4] area on the other hand, energy sector is considered to be critical infrastructure sectors in all cases.

The present paper provides a methodological framework for risk assessment within the energy sector as critical infrastructure sector. The paper embedded the key concepts and key mechanisms of object-oriented modelling in order to build a risk multimodel that can be used as starting point in future protection strategies.

2. Object-oriented modelling and programming

Object-Oriented Modelling (OOM) may be seen as a collection of cooperating objects, as opposed to a traditional view in which a program may be seen as a list of instructions to the computer. In Object Oriented Programming (OOP), each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent little machine with a distinct role or responsibility.

It also use several concepts/techniques from previously established paradigms, including inheritance, modularity, polymorphism and encapsulation. These concepts give support to the development of efficient class structures. The aim is to approximate the behavior of the real world elements within the software environment [5].

OOP works with some key concepts (class, object, and message passing).

A *class* defines the abstract characteristics of a thing, including the thing's characteristics (its attributes or properties) and the things it can do (its behaviors or methods). It is a template for defining similar objects, providing the basis for abstracting the common characteristics of real-world objects.

A particular instance of a class and executable software representations of real-world concepts are *objects*. Most broadly defined, an object is a software package that includes all the necessary data and procedures to represent a real-world object for a specific set of purposes. Objects are not only physical they can also be conceptual.

Objects interact with each other by sending requests for services known as *messages*. A message is a request to a particular object to invoke a specified procedure, which is typically called a method. It is universal communication medium through which objects interact with one another.

According to Taylor [6], in OOP objects support some mechanisms: encapsulation, messages support polymorphism, and classes support inheritance.

Encapsulation is the mechanism by which related data and procedures are bound together within an object. It conceals the exact details of how a particular class works from objects that use its code or send messages to it. In principle an object is a piece of software and it reacts to the massages coming from other objects according to its own rules and does not disclose itself by making all variables to be private, or hidden from view outside of the object. In this way, encapsulation supports and extends the proven principle of information hiding. Information hiding is valuable because it prevents local changes from having global impact.

The fact that different objects can respond to the same message in different ways is known as *polymorphism*, a Greek term meaning "many forms". Polymorphism is a behavior that varies depending on the class in which the behavior is invoked, that is, two or more classes can react differently to the same message. The power of polymorphism is that it greatly simplifies the logic of programs. Polymorphism is often seen as the cutting edge of object technology and regarded as a very intuitive mechanism [7].

In some cases, a class will have ''subclasses'', more specialized versions of a class. *Inheritance* is the mechanism that allows classes to be defined as special cases, or subclasses, of each other. If a department class is declared to be a subclass of organization, the department class automatically incorporates all the methods and variables defined in the organization class. Inheritance can cascade down over any number of levels, allowing deeply nested class hierarchies to be constructed.

Abstraction is a subset of Inheritance capability of Object-Oriented Programming Paradigm. Abstraction is a method for simplifying complex reality by modelling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem. The use of inheritance with overriding allows the declarative capture and implementation of two fundamental cognitive processes — generalization and specialization.

Inheritance represents a further enhancement of the concept of type in object technology. In addition to defining new types that package data and procedures together, developers can define type hierarchies in which the scope and generality of a type is determined by its position within the hierarchy.