ERASMUS+
SECTOR SKILLS ALLIANCES

Mu.SA: Museum Sector Alliance
575907-EEP-1-2016-1-EL-EPPKA2-SSA
www.project-musa.eu

# Video Script – Main concepts of modelling and data structures

Sara Fernandes and Filippe Correia

University of Porto

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

| | |
|---|---|
| Module: | Application Design |
| Unit: | Modelling and Data Structures |
| Learning Object: | Main concepts of modelling and data structures |
| Author: | Sara Fernandes, Filipe Correia, UPORTO |
| Technical Reviewer: | Panagiota Polymeropoulou, Christos Pierrakeas, HOU |
| Scientific Reviewer: | Katarzyna Leszczynska, LCU |

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

The document created in 2019 for the project Mu.SA – Museum Sector Alliance, funded under the Erasmus+/KA3 programme (project number 575907-EEP-1-2016-1-EL-EPPKA2-SSA).

Layout design:

DAISSY research group, Hellenic Open University (http://daissy.eap.gr)

Project coordinator: HELLENIC OPEN UNIVERSITY

Project partners:

More information about the project:

www.project-musa.eu

musa@daissy.eap.gr

@MuseumSectorAlliance

#MuseumSectorAlliance

**Sara Fernandes, University of Porto**

UML (Unified Modeling Language) is a language (notation with associated semantics) that allows to visualize, specify and document the data, easily.

With UML we can cover most or even all the application's data. UML is a standard for software modeling and structuring.

With this, we can have nine different standard diagrams:

- STATIC VIEW DIAGRAMS: Use Cases, Classes, Objects, Components and Deployment
- DYNAMIC VIEW DIAGRAMS: Sequence, Collaboration, State Chart and Activities

## CLASS DIAGRAM

A class diagram models the application's vocabulary, from the point of view of the user/problem or the implementer/solution.

In a class diagram, we can have:

- Classes;
- Attributes;
- Operations;
- Associations;
- Objects;
- …

CLASS: Representations of the different kinds of *things* in the system (person, car, property, etc), roles (student, teacher, pilot, etc), events (course, class, accident, etc) or data types (date, time range, complex numbers, vectors, etc).

In UML, a class is represented by a rectangle.

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

ATTRIBUTE: Attributes characterize a class. Each attribute should have a unique name inside the respective class and also a type.

OPERATIONS: Operations represent the behaviors associated to a class.

ASSOCIATIONS: Associations represent relationship between two different classes.

Associations have a multiplicity on each end:

- 1 – Exactly one;
- 0..1 – Zero or one;
- * - Zero or more;
- 1..* - One or more;
- 1,3..5 – One or three to five

There are also aggregation associations (“stronger” association) where one class is part of another, and composition associations, represented by a filled diamond, which represent even “stronger” relationships.

ASSOCIATION CLASSES: When a new association appears, it can introduce a new class that describes the connection.

GENERALIZATIONS: Generalizations serve to represent inheritance relationships between classes.

OBJECT: Instance of a class. While a class represents an abstract element, an object represents something more concrete. While attributes are set at the class level, attribute values are set at the object level. Objects of the same class have the same operations.

Different data can be structured through a markup language which consists in a group of words and symbols that describe the identity or function of a document component.

One example of a markup language is XML (Extensible Markup Language) which lets you create and draw markup languages with several possibilities, being why it’s an extensible language. With this, we create and draw markup languages with unlimited possibilities.

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

A well-formed XML should have:

1. One or more elements;
2. A “root” element;
3. Elements interconnected.

**An element is a representation of a piece of data that we want to structure.**
An XML element is represented using tags, and is expressed using this notation: **<element></element>**;

All XML elements must have a start and end-tag (exactly equal), to close the respective element.

Empty elements don’t need an end-tag. With these elements, we can only use “/” at the end of them.

An XML element can have **attributes**. While an element typically represents concrete data, attributes tend to represent metadata (data about data).

If we want to restrict XML since it’s a “free” language, we can use XML Schema Definition (XSD), for instance.

**END OF SCRIPT**

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.