INSPIRATION – Spatial Data Infrastructure in the Western Balkans

Introduction to Unified Modeling Language (UML)

3rd INSPIRATION Training
December 4-5, 2012

A multi-country project funded by the European Union and implemented by

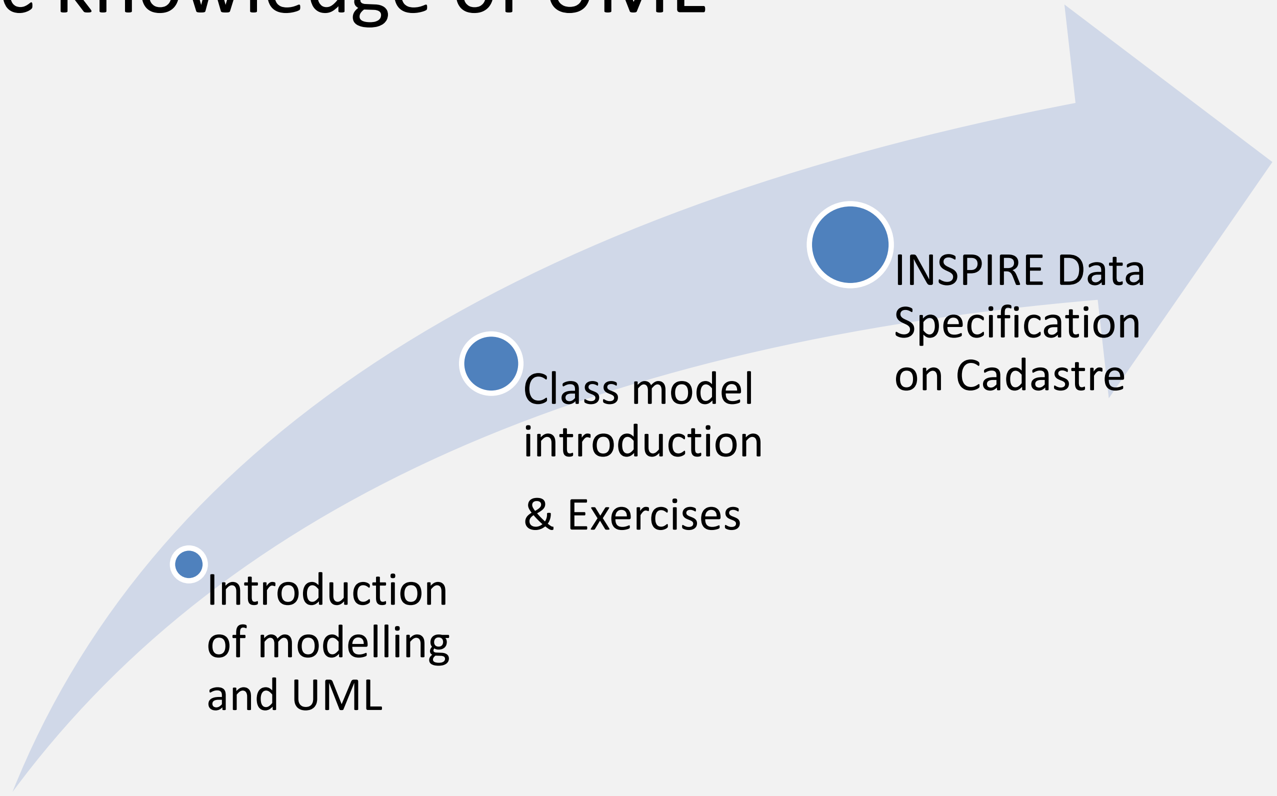
Content

- Basic introduction
  - Models
  - UML
    - Diagrams
- Exercises & Examples
  - Class diagram

A multi-country project funded by the European Union and implemented by

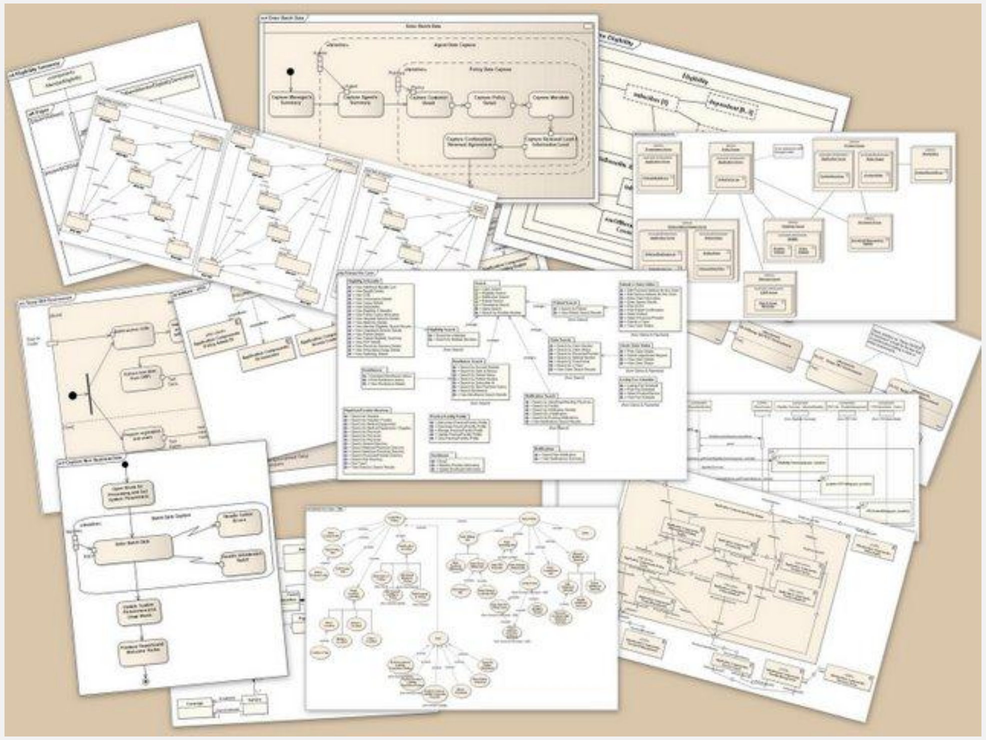
Scope

- Basic knowledge of UML


Introduction of modelling and UML
Class model introduction
& Exercises
INSPIRE Data Specification on Cadastre

UML Background

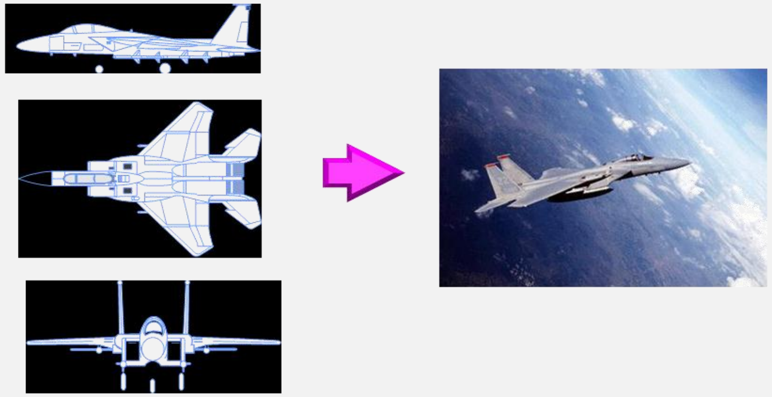
<u>What are models?</u>



UML Background

<u>What are models?</u>

- A complete description of a system from a particular perspective
- Simplification of reality

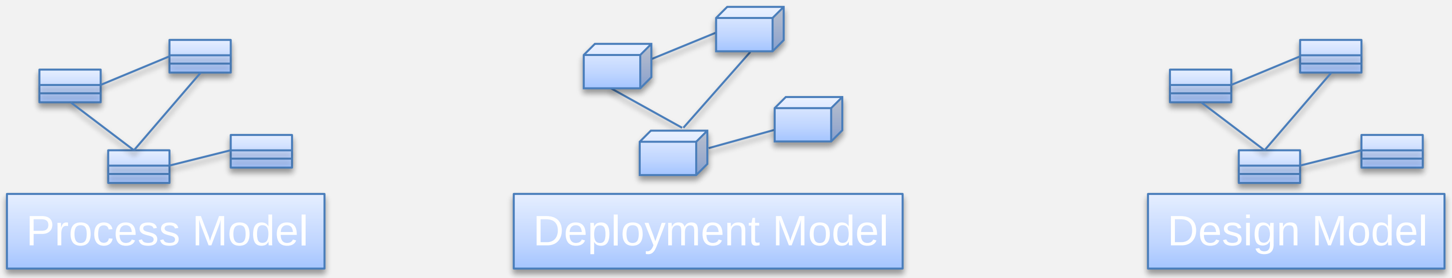


Why models?

- Modeling achieves four aims:
  - Helps you to **visualize a system** as you want it to be.
  - Permits you to **specify the structure or behavior of a system.**
  - Gives you a **template that guides you in constructing** a system.
  - **Documents the decisions** you have made.
- You build models of complex systems because you cannot comprehend such a system in its entirety.
- You build models to better understand the system you are developing.

Four Principles of Modeling

- The model you choose influences how the problem is attacked.



- Every model may be expressed at different levels of precision.
- The best models are connected to reality.
- No single model is sufficient.

UML Background

What is UML?

?

The OMG specification states:

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."

A multi-country project funded by the European Union and implemented by

UML Background

What is UML?

- UML is a language (Unified Modeling Language) for models
  - technical and graphical specification
  - Graphic notation to visualize models
  - Not a method or procedure
- Managed and created by the Object Management Group

UML Background

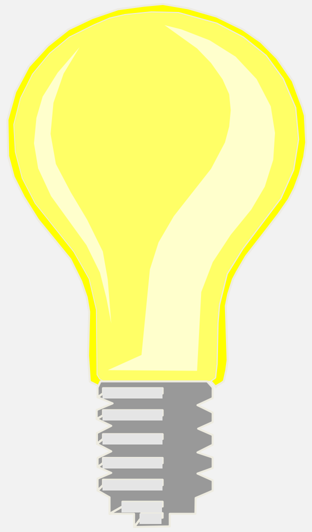
- The UML is a language for
    - Visualizing
    - Specifying
    - Constructing
    - Documenting

  the artifacts of a software-intensive system.

- The Unified Modelling Language (UML) is an industry standard for object oriented design notation, supported by the Object Management Group (OMG).

Language for Visualizing

- Communicating conceptual models to others is prone to error unless everyone involved speaks the same language.
- There are things about a software system you can't understand unless you build models.
- An explicit model facilitates communication.



Language for Specifying

- The UML builds models that are precise, unambiguous, and complete.

A multi-country project funded by the European Union and implemented by

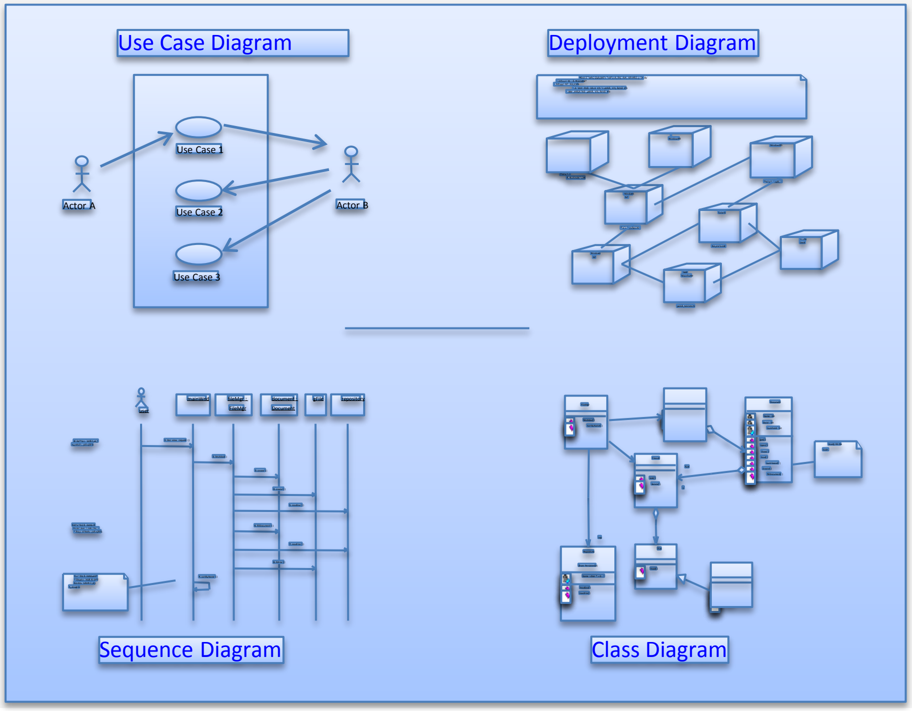
Language for Constructing

- UML models can be directly connected to a variety of programming languages.
  - Maps to Java, C++, Visual Basic, and so on
  - Tables in a RDBMS or persistent store in an OODBMS
  - Permits forward engineering
  - Permits reverse engineering

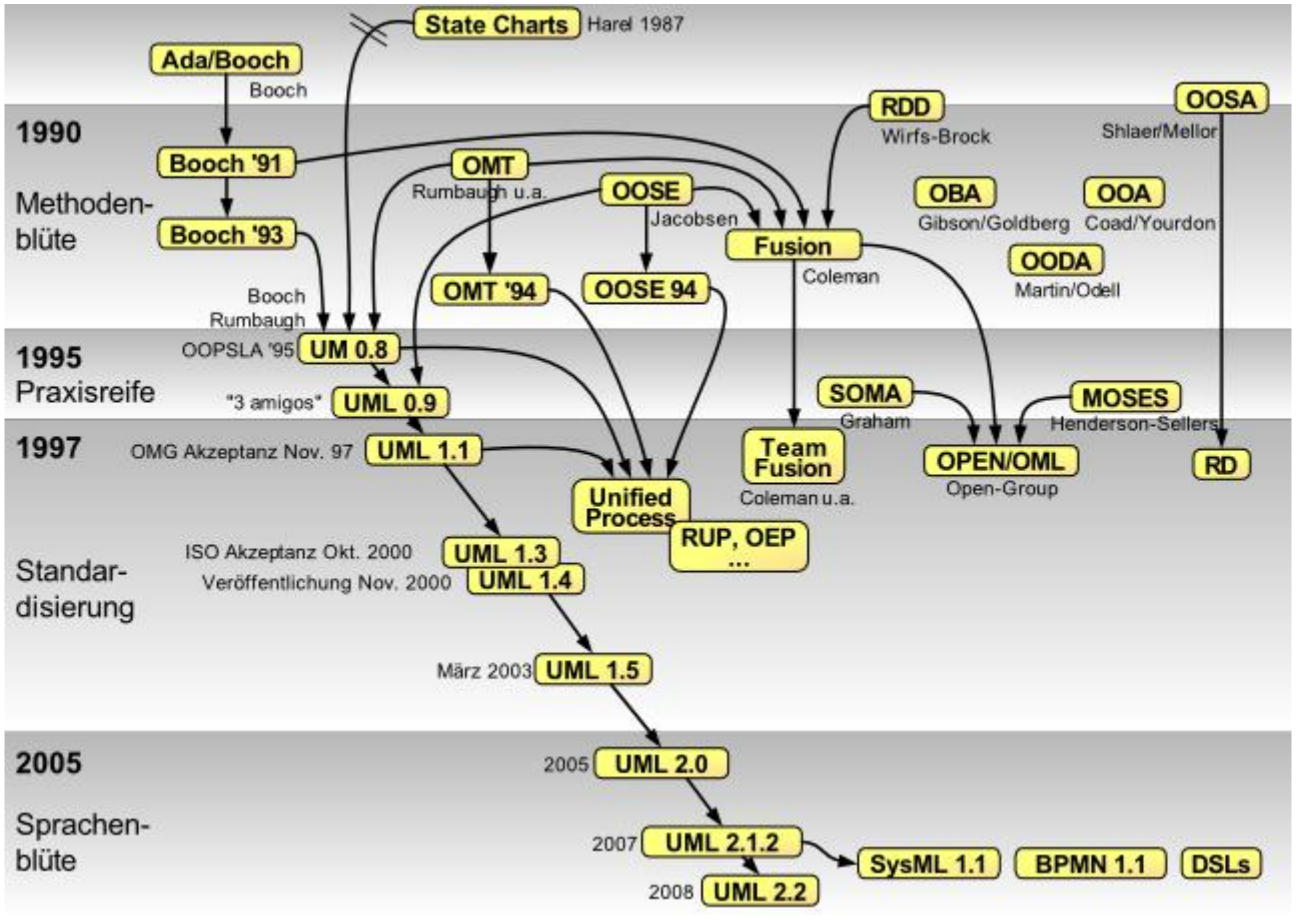
A multi-country project funded by the European Union and implemented by

Language for Documenting

- The UML addresses documentation of system architecture, requirements, tests, project planning, and release management.

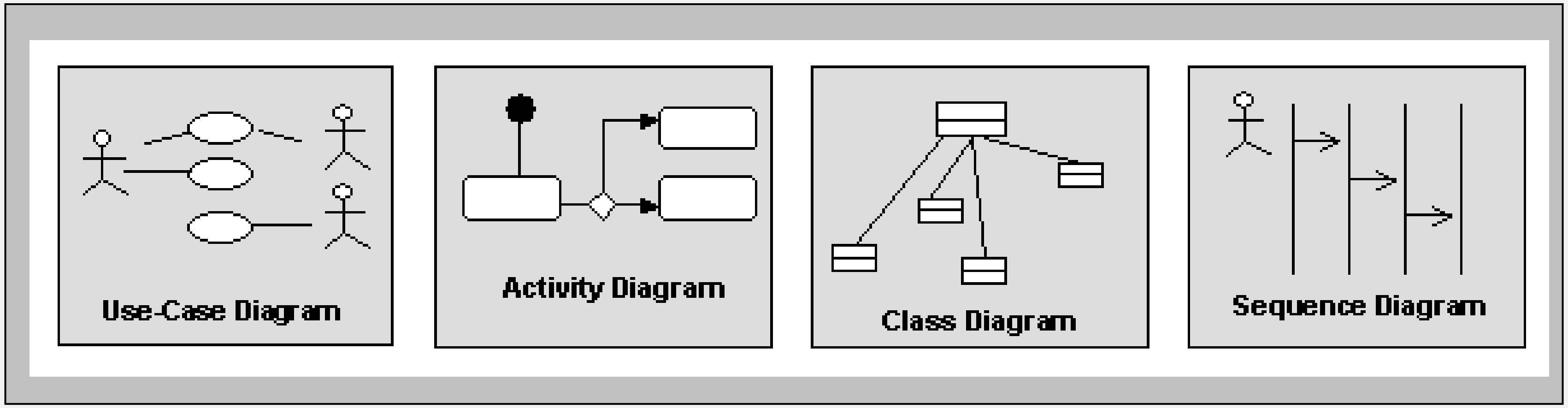


History of the UML



Diagrams

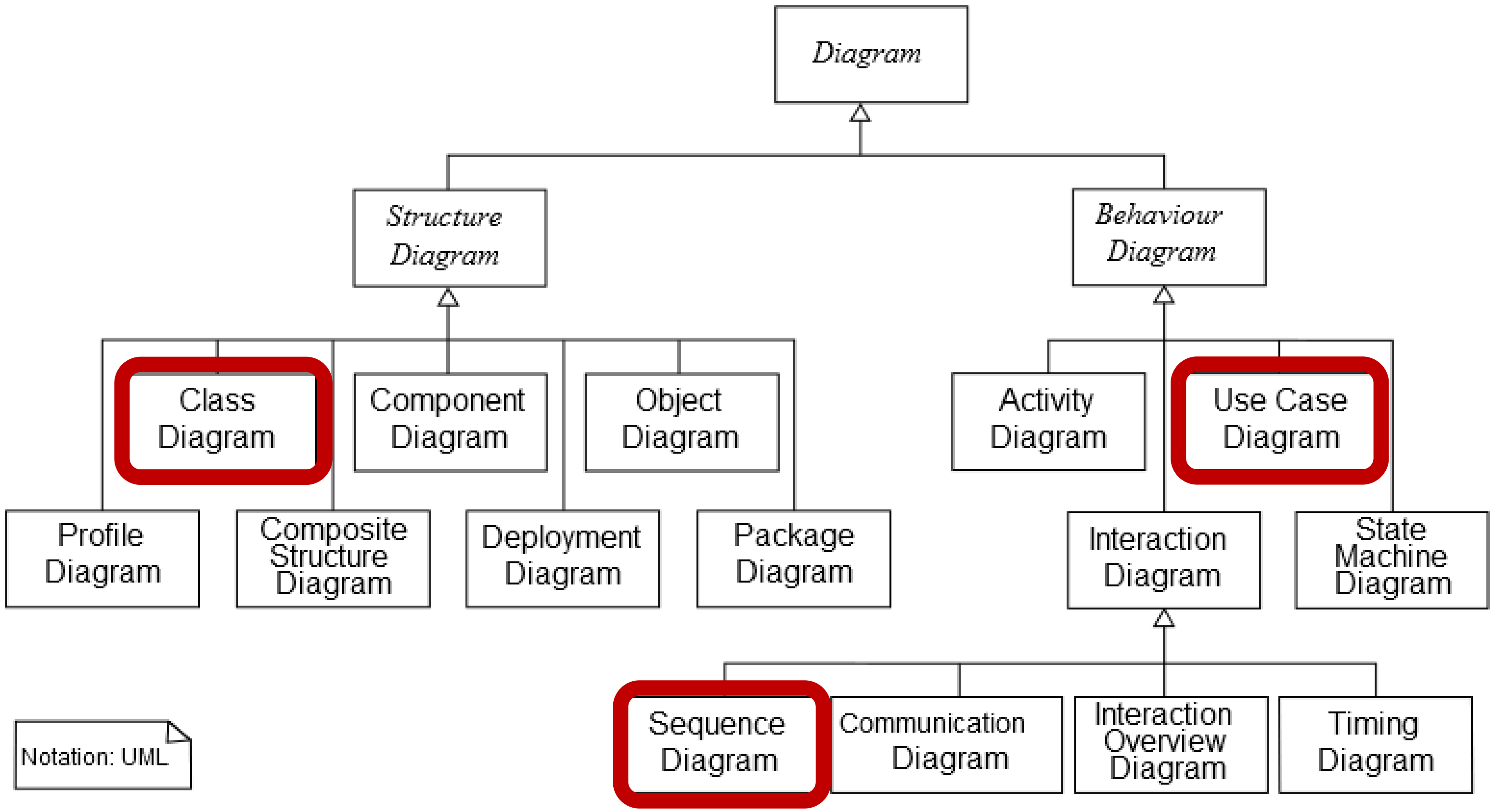
- Diagrams graphically depict a view of a part of your model.
- Different diagrams represent different views of the system that you are developing.
- A model element will appear on one or more diagrams.



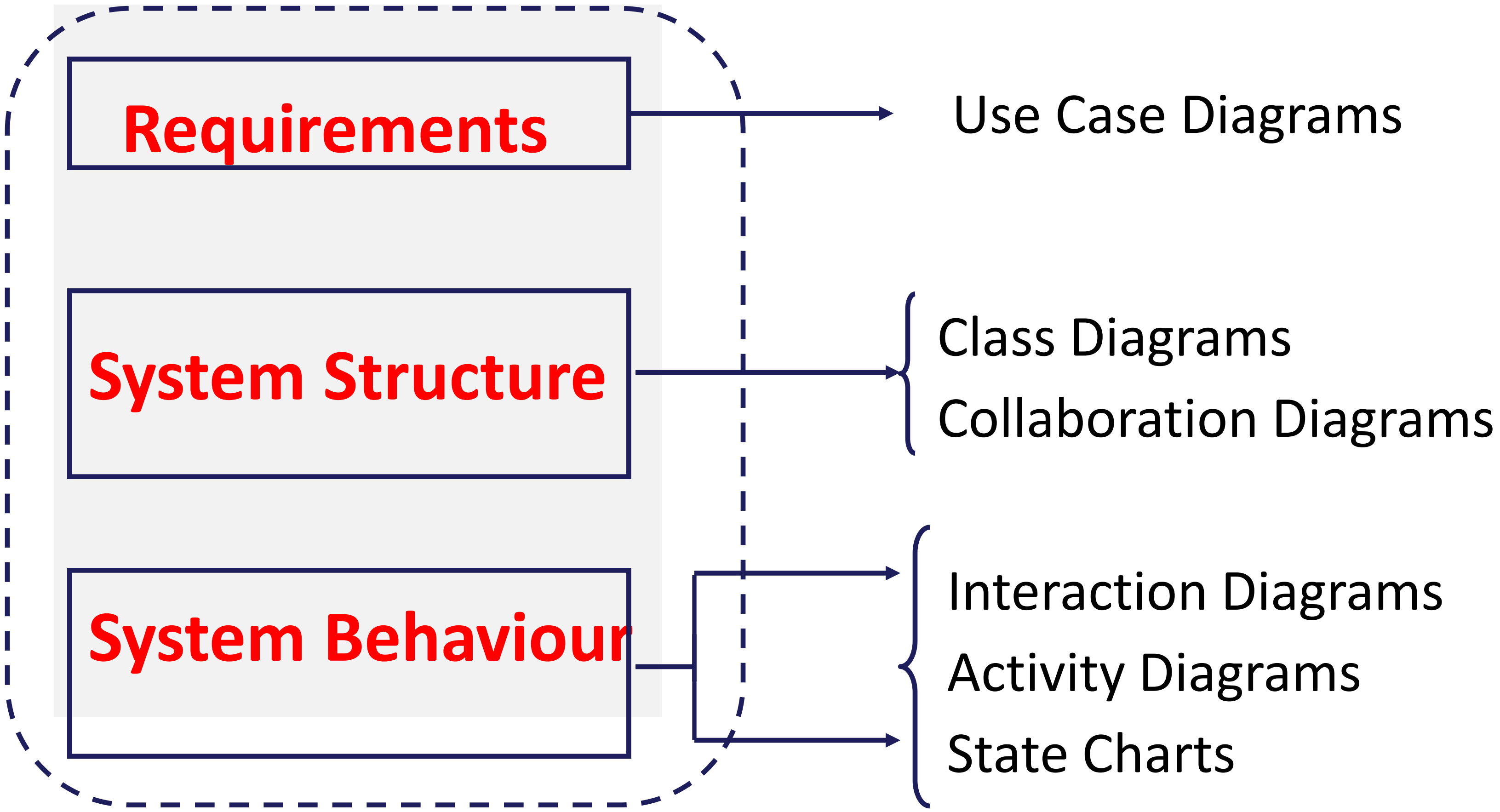
UML Diagrams

- UML 2.2
- 14 different types of diagrams
- 2 different groups
  - Behavior & Interaction models
  - Structural models

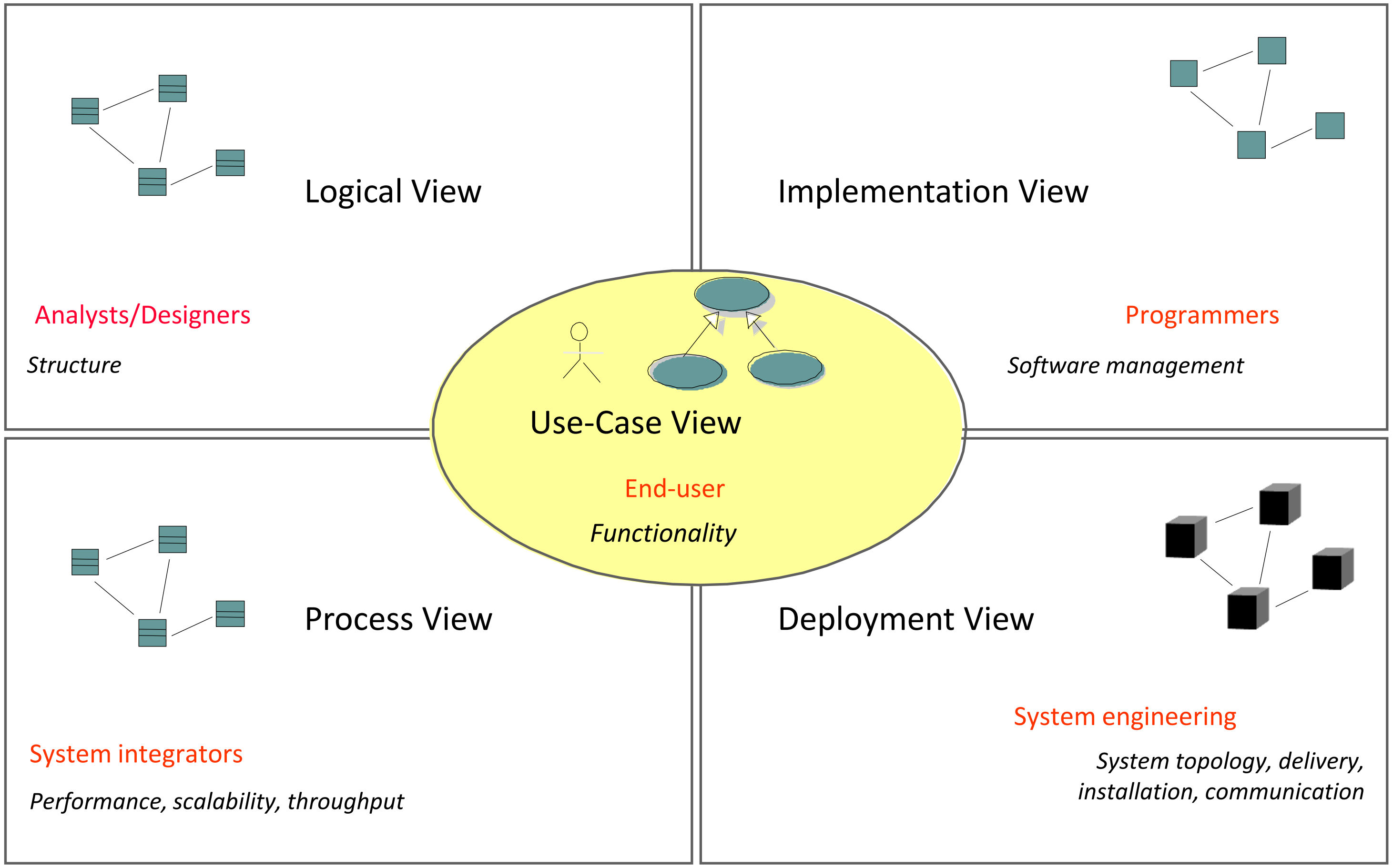
UML Diagrams



Key Diagrams in UML



Different diagrams of system for different people



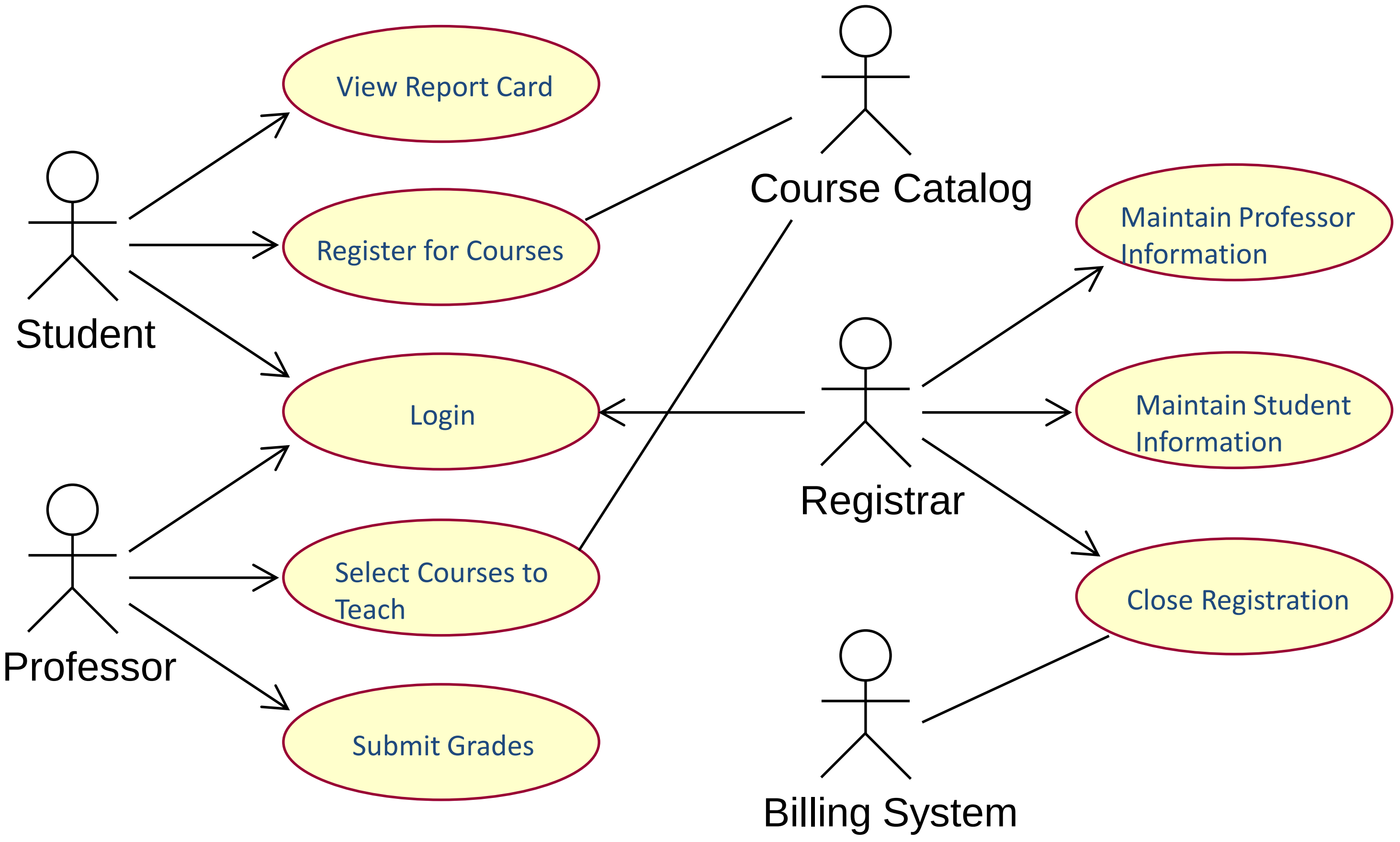
A multi-country project funded by the European Union and implemented by

What is a Use-Case Model?

A use-case model:

- Is a model of a system’s intended functions and its environment
- Serves as a contract between the customer and the developers
- Contains the following diagrams:
  - Use case: Shows a set of use cases and actors and their relationships
  - Activity: Shows the flow of events within a use case
  - Sequence: Shows how a use case will be implemented in terms of collaborating objects

Use-Case Diagram



Activity Diagram

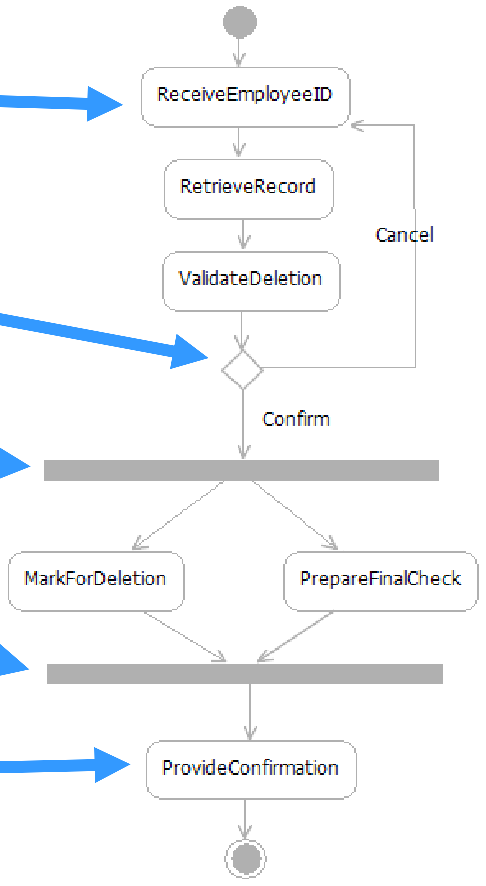
Action
A step in the flow of events

Decision
Flows split based on a guard condition

Fork
Beginning of concurrent flows

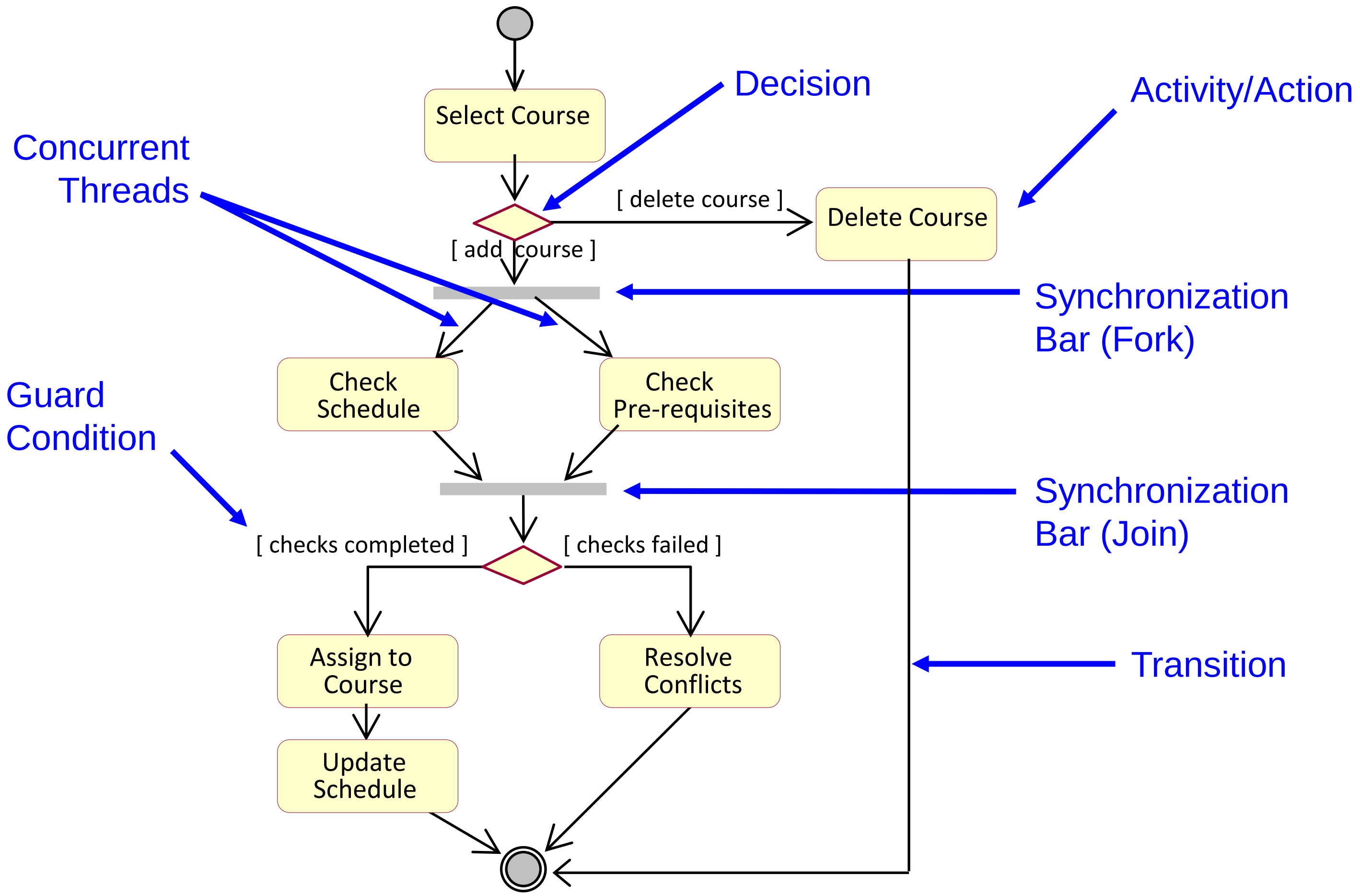
Join
End of concurrent flow

Flow
Show the sequence of activities



A multi-country project funded by the European Union and implemented by

Activity Diagram (Example)



What is a Design Model?

A design model:

- Describes the realization of use cases in terms of design elements
- Describes the design of the application
- Contains the following diagrams:
  - Class: Shows UML classes and relationships
  - Component: Shows the structure of elements in the implementation model
  - Communication and Sequence: Show how objects and classes interact
  - State Machine: Shows event-driven behavior

Class Diagram

- Class diagrams show the static structure of the model resp. system
  - Classes
  - Attributes
  - Relationships to other classes
- Class diagrams do not show temporal information
- → INSPIRE data specifications

Class Diagram

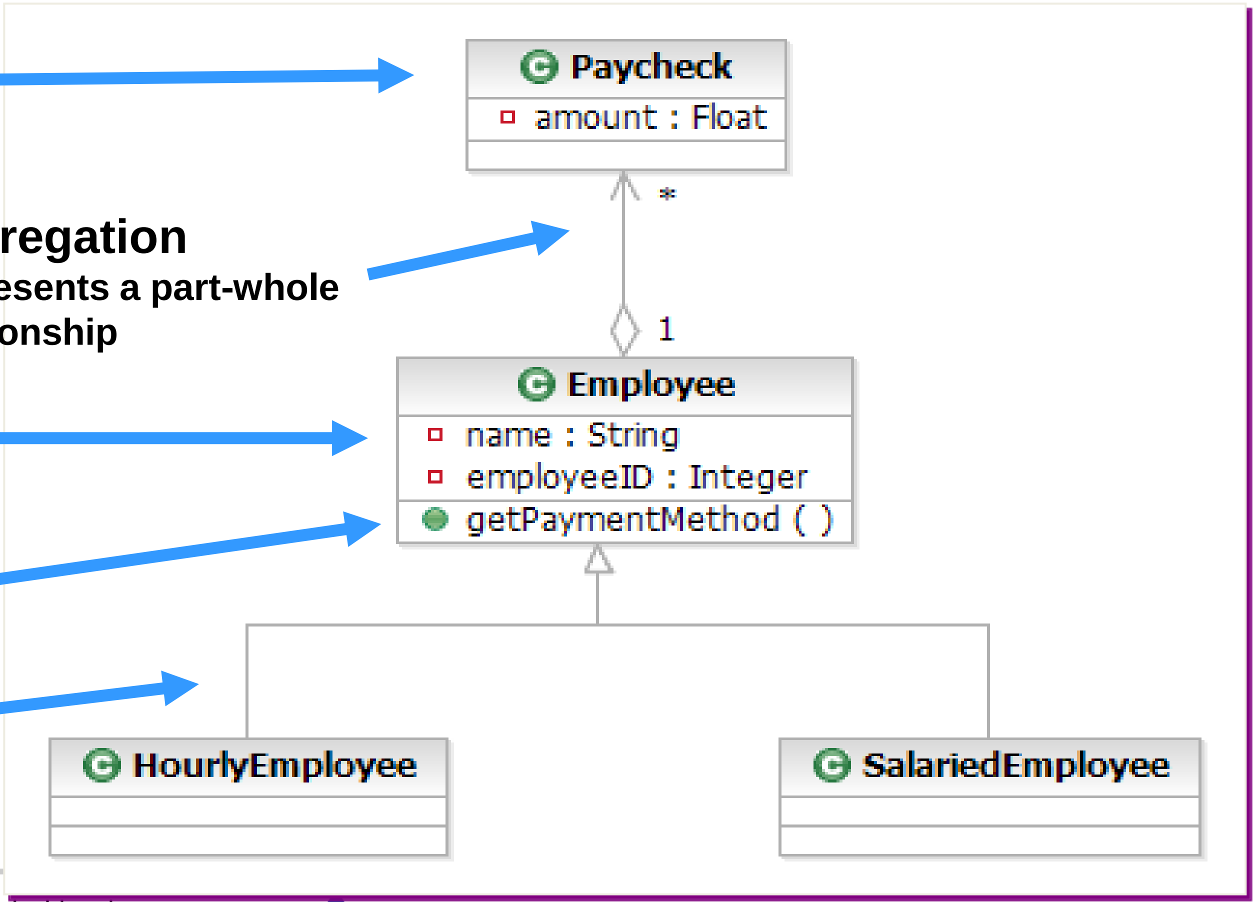
Class
A description of a set of objects

Aggregation
Represents a part-whole relationship

Attribute
Named property of a class

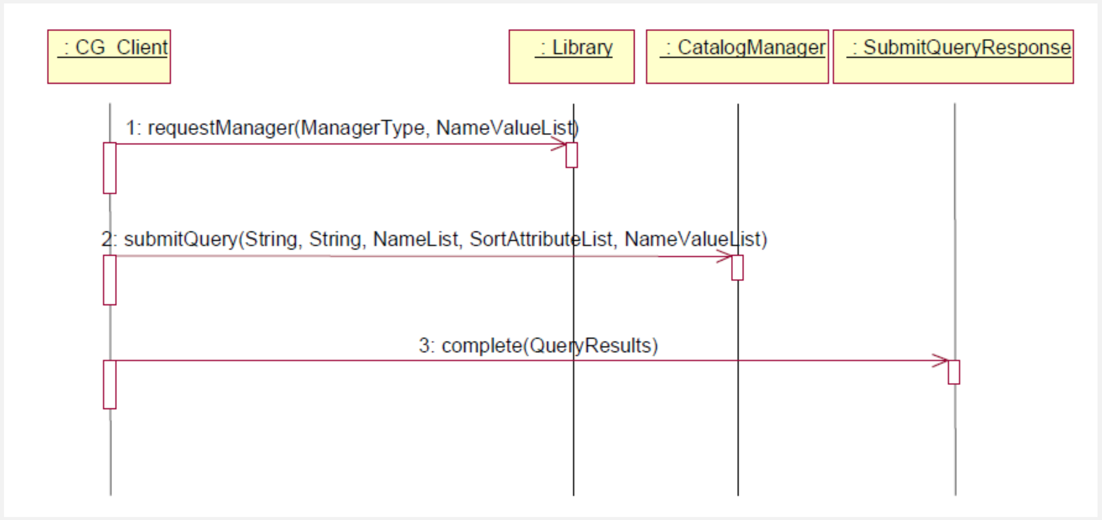
Operation
Class behavior

Generalization
Shows an inheritance relationship

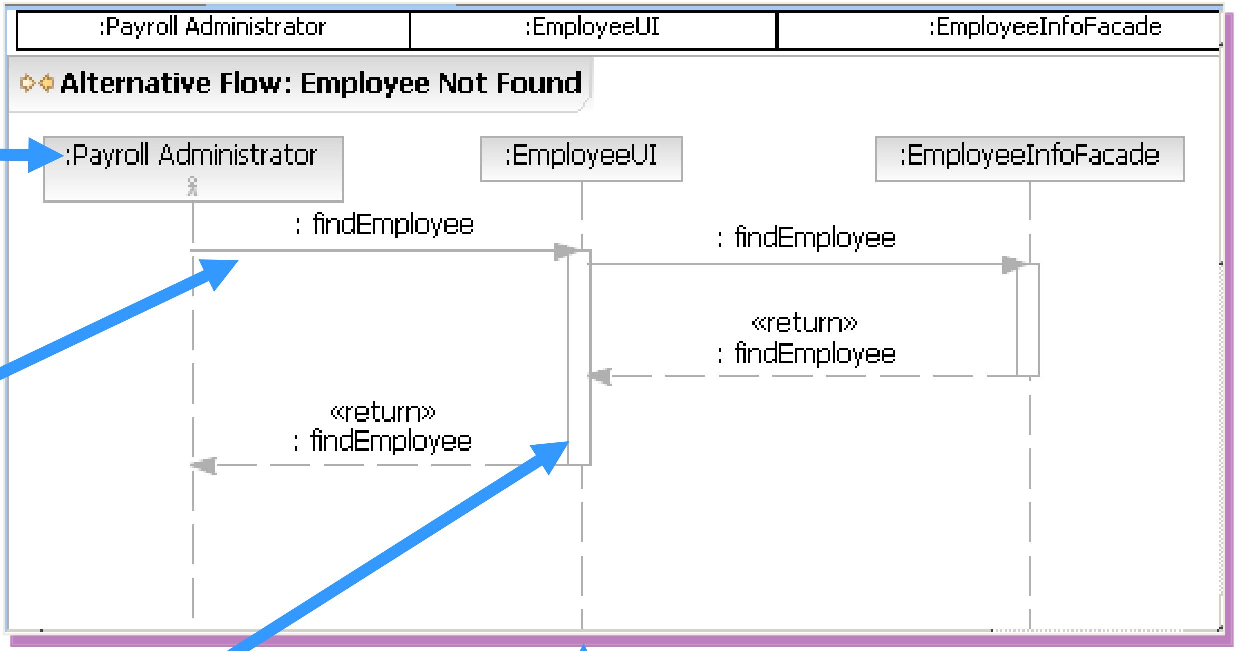


Sequence Diagram

- used to show how objects interact to perform the behavior of all or part of a use case as part of a use-case realization


: CG_Client
: Library
: CatalogManager
: SubmitQueryResponse
1: requestManager(ManagerType, NameValueList)
2: submitQuery(String, String, NameList, SortAttributeList, NameValueList)
3: complete(QueryResults)

Sequence Diagram



Object/Class
Shows the object/class involved in the interaction

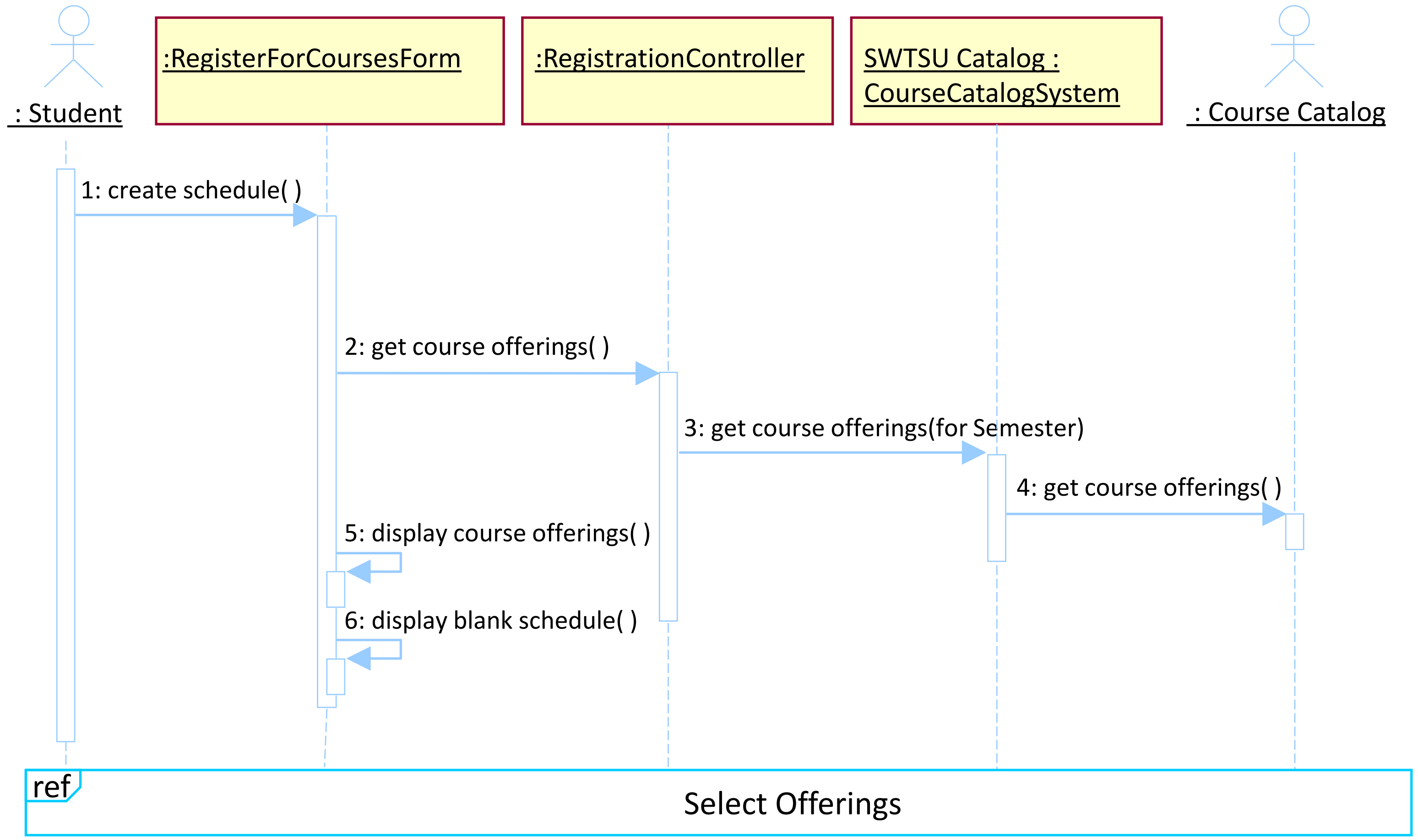
Messages
Show data exchanged between objects

Execution Occurrence
Shows object executing

Lifeline
Shows the life of the object

A multi-country project funded by the European Union and implemented by

Sequence Diagram (Example)


: Student
:RegisterForCoursesForm
:RegistrationController
SWTSU Catalog : CourseCatalogSystem
: Course Catalog
1: create schedule( )
2: get course offerings( )
3: get course offerings(for Semester)
4: get course offerings( )
5: display course offerings( )
6: display blank schedule( )
ref
Select Offerings

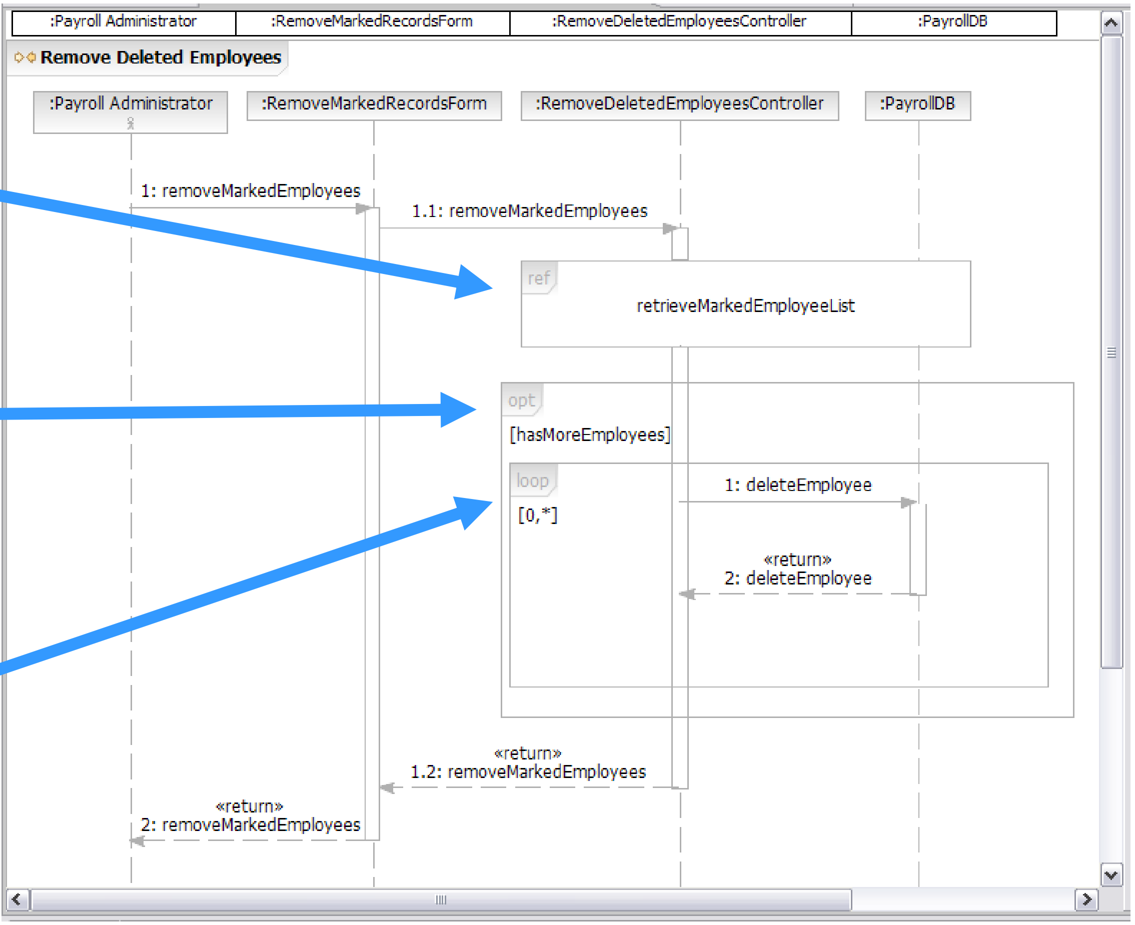
Sequence Diagram

Combined Fragments

Interaction Use (ref)
References another interaction

Optional Fragment (opt)
Executed if guard condition evaluates to true

Loop (loop)
Executed as long as the first guard condition evaluates to true



A multi-country project funded by the European Union and implemented by GFA Consulting Group, umweltbundesamt, GDi GISDATA, con terra

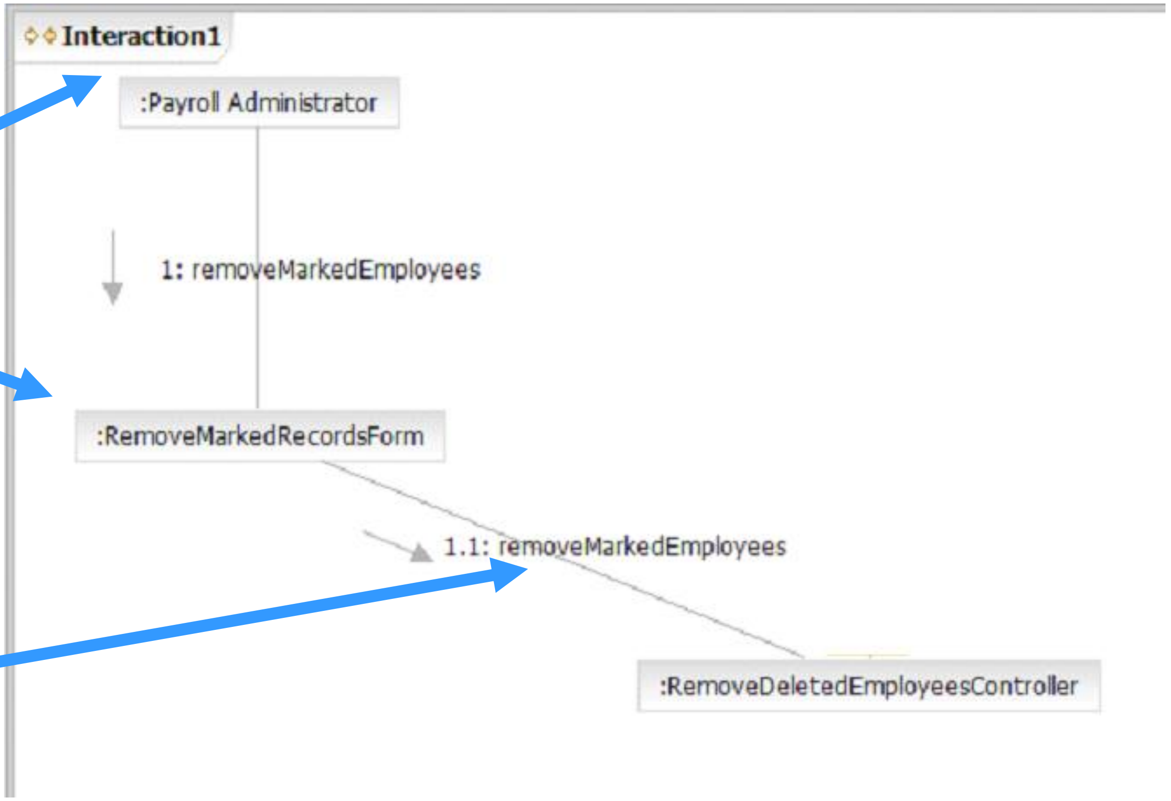
Communication Diagram

- Collaboration diagram
- provide another way to show how objects interact to perform the behavior of a particular use case or a part of a use case. Where sequence diagrams emphasize the interactions of objects over time, communication diagrams are designed to emphasize the relationships between objects

Communication Diagram

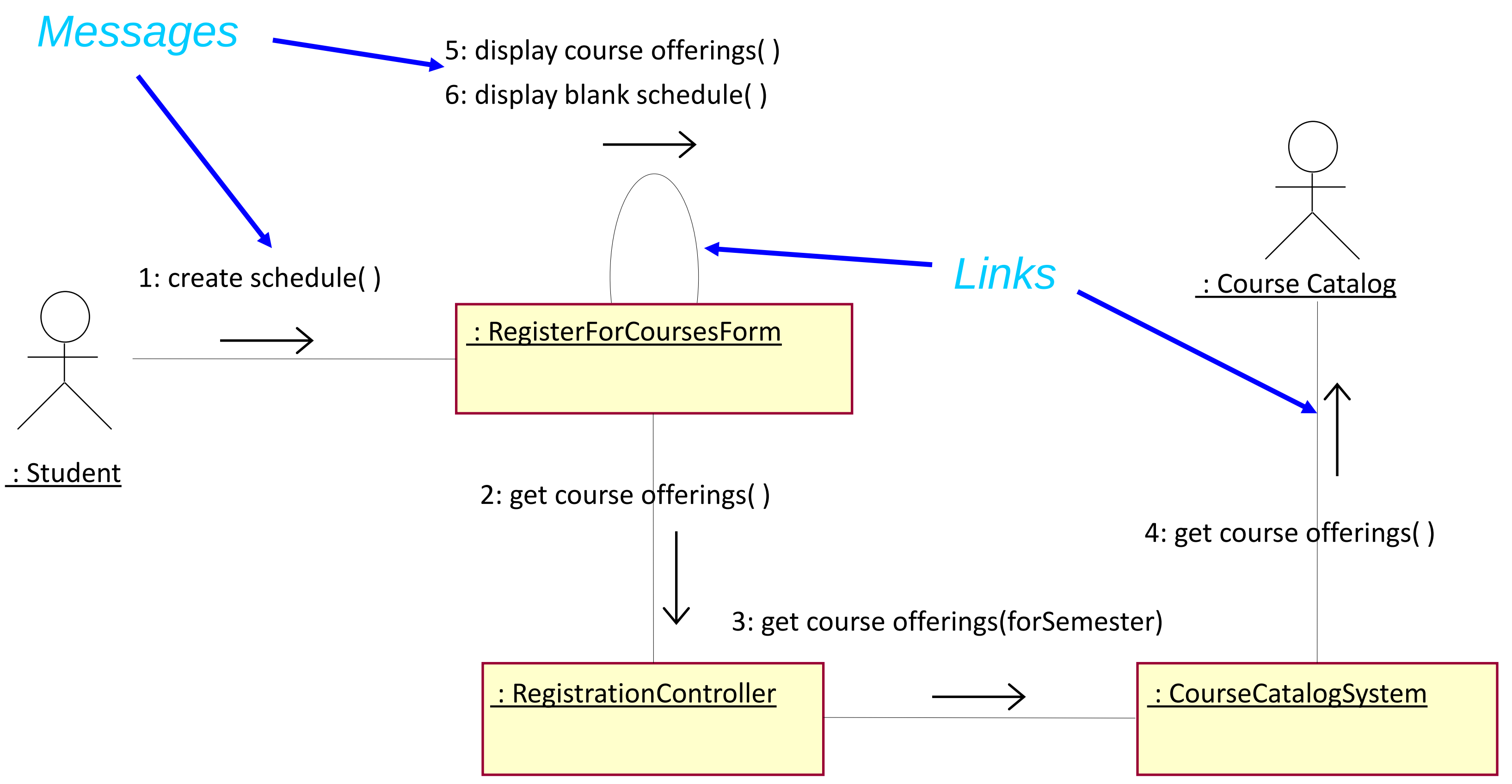
Object/Class
Shows the object/class involved in the interaction

Message
Shows data exchanged between objects



A multi-country project funded by the European Union and implemented by

Communication Diagram

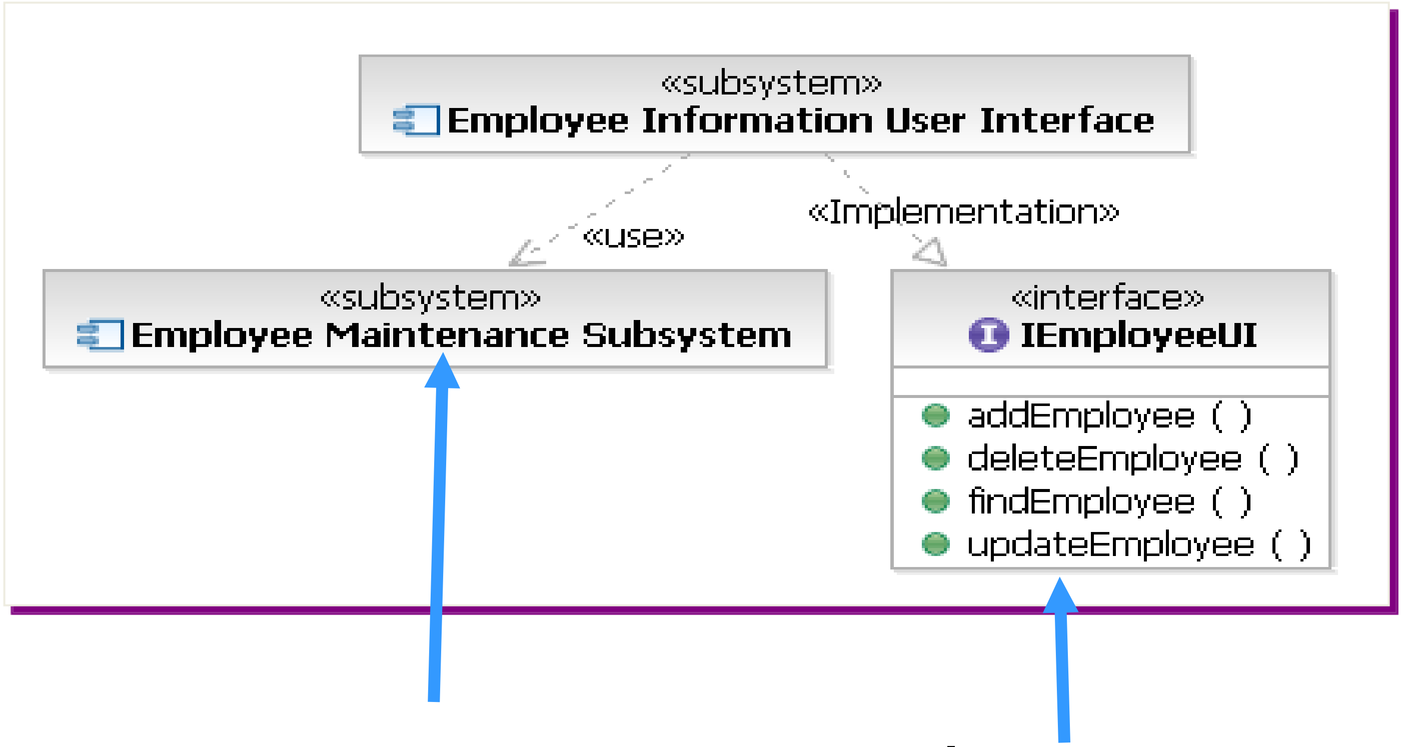

Messages
5: display course offerings( )
6: display blank schedule( )
Links
1: create schedule( )
: RegisterForCoursesForm
: Course Catalog
: Student
2: get course offerings( )
4: get course offerings( )
3: get course offerings(forSemester)
: RegistrationController
: CourseCatalogSystem

A multi-country project funded by the European Union and implemented by

Component Diagram

- It shows the runtime structure of the system at the level of software components. Components are the modular parts of the system and are made up of groups of related objects that are hidden behind an external interface.

Component Diagram



Component
Modular parts of the system

Class
Included to show implementation relationships.

Deployment Diagram

- Deployment diagrams show the deployment architecture of the system, that is, which of the system's software artifacts reside on which pieces of hardware.

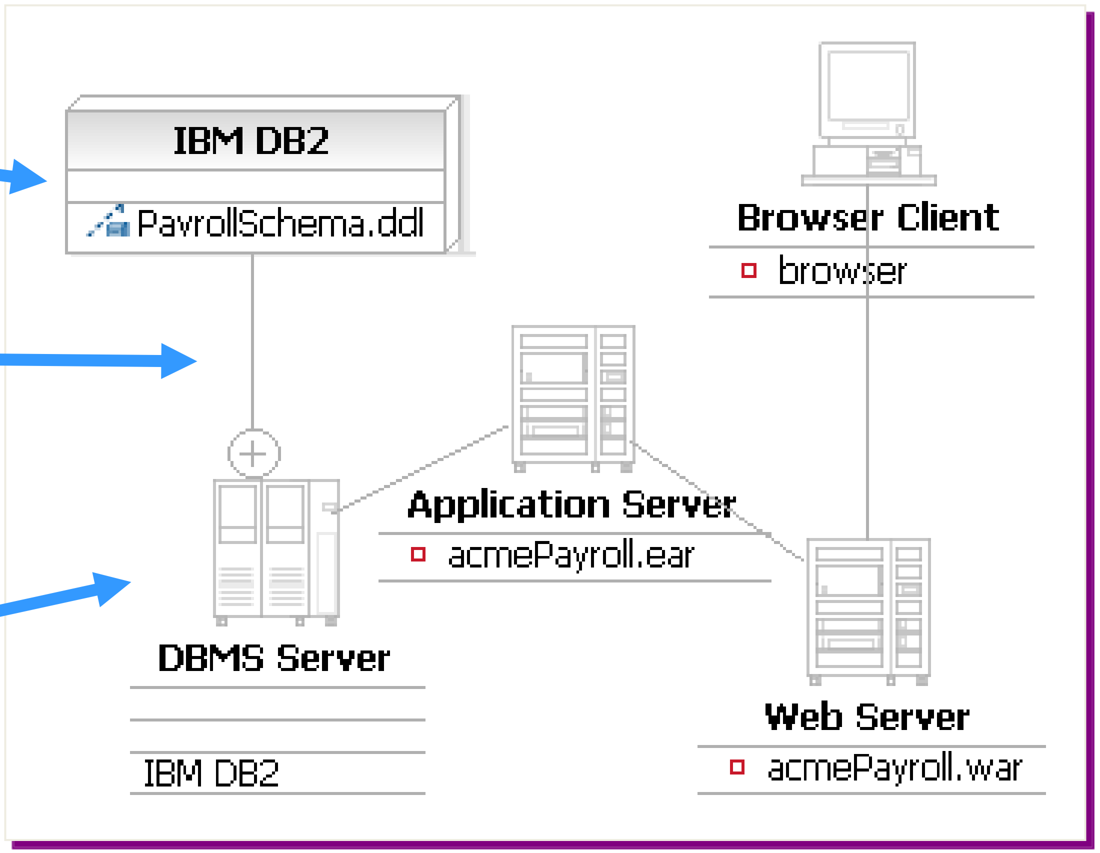
A multi-country project funded by the European Union and implemented by

Deployment Diagram

Artifact
Represents a physical file

Owned Element Relationship
Shows another way of showing nested elements

Node
Represents a physical machine



A multi-country project funded by the European Union and implemented by

How Many Diagrams?

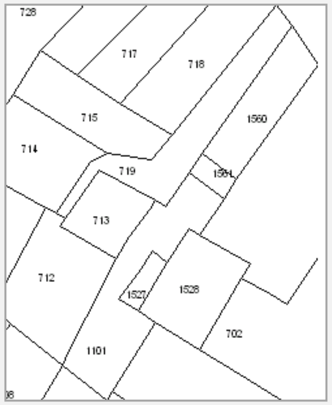
- Depends:
  - You use diagrams to visualize the system from different perspectives.
  - No complex system can be understood in its entirety from one perspective.
  - Diagrams are used for communication
- Model elements will appear on one or more diagrams.
  - For example, a class may appear on one or more class diagrams, be represented in a state machine diagram, and have instances appear on a sequence diagram.
  - Each diagram will provide a different perspective.

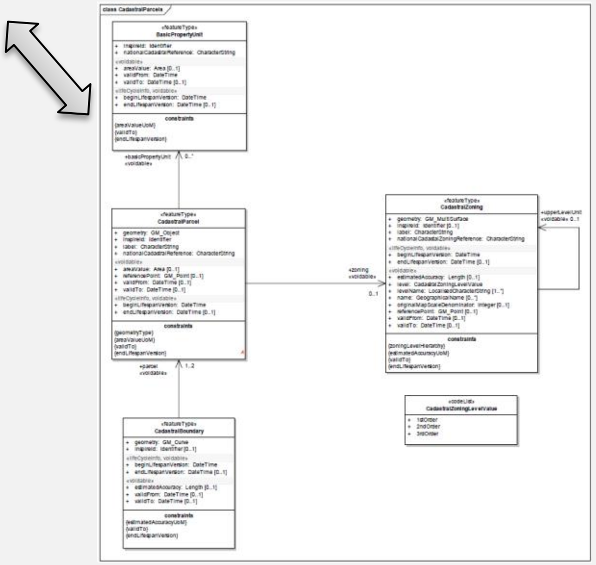
UML – Exercise

A multi-country project funded by the European Union and implemented by

UML – Exercise

- Class diagram
  - INSPIRE Data Specifications
  - Foundation for other structure diagrams
  - Classification of reality




class CadastralParcels
«featureType» BasicPropertyUnit
+ inspireId: Identifier
+ nationalCadastralReference: CharacterString
«voidable»
+ areaValue: Area [0..1]
+ validFrom: DateTime
+ validTo: DateTime [0..1]
«lifeCycleInfo, voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
constraints
{areaValueUoM}
{validTo}
{endLifespanVersion}
+basicPropertyUnit «voidable» 0..*
«featureType» CadastralParcel
+ geometry: GM_Object
+ inspireId: Identifier
+ label: CharacterString
+ nationalCadastralReference: CharacterString
«voidable»
+ areaValue: Area [0..1]
+ referencePoint: GM_Point [0..1]
+ validFrom: DateTime [0..1]
+ validTo: DateTime [0..1]
«lifeCycleInfo, voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
constraints
{geometryType}
{areaValueUoM}
{validTo}
{endLifespanVersion}
+zoning «voidable» 0..1
«featureType» CadastralZoning
+ geometry: GM_MultiSurface
+ inspireId: Identifier [0..1]
+ label: CharacterString
+ nationalCadastalZoningReference: CharacterString
«lifeCycleInfo, voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«voidable»
+ estimatedAccuracy: Length [0..1]
+ level: CadastralZoningLevelValue
+ levelName: LocalisedCharacterString [1..*]
+ name: GeographicalName [0..*]
+ originalMapScaleDenominator: Integer [0..1]
+ referencePoint: GM_Point [0..1]
+ validFrom: DateTime [0..1]
+ validTo: DateTime [0..1]
constraints
{zoningLevelHierarchy}
{estimatedAccuracyUoM}
{validTo}
{endLifespanVersion}
+upperLevelUnit «voidable» 0..1
+parcel «voidable» 1..2
«featureType» CadastralBoundary
+ geometry: GM_Curve
+ inspireId: Identifier [0..1]
«lifeCycleInfo, voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
«voidable»
+ estimatedAccuracy: Length [0..1]
+ validFrom: DateTime [0..1]
+ validTo: DateTime [0..1]
constraints
{estimatedAccuracyUoM}
{validTo}
{endLifespanVersion}
«codeList» CadastralZoningLevelValue
+ 1stOrder
+ 2ndOrder
+ 3rdOrder

UML Exercise

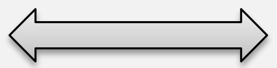
- The class diagram
  - Class (and objects)
  - Relationship
  - Package (advanced)
  - Interfaces (advanced)

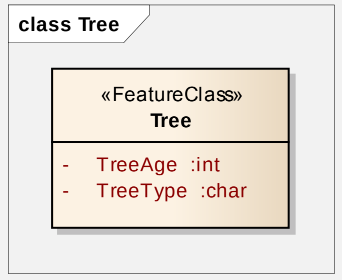
UML Exercise

- The class
  - Summarize a number of objects with the same behavior and semantics
  - Abstraction of entities
    - Semantic concept with common attributes and operations

UML Exercise

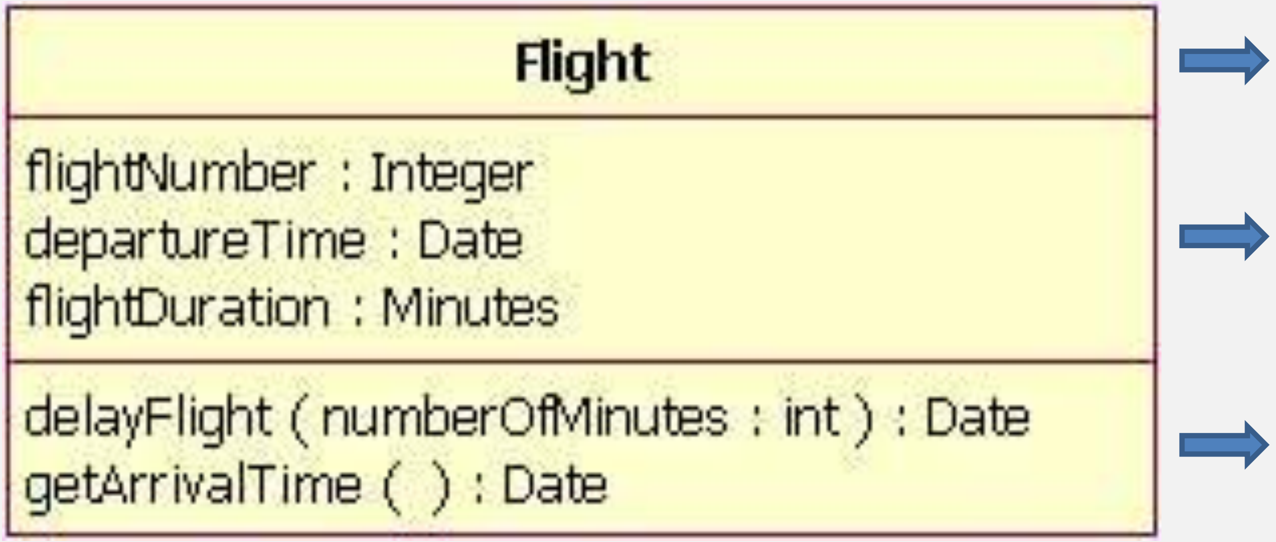
- The class
  - Abstraction of entities





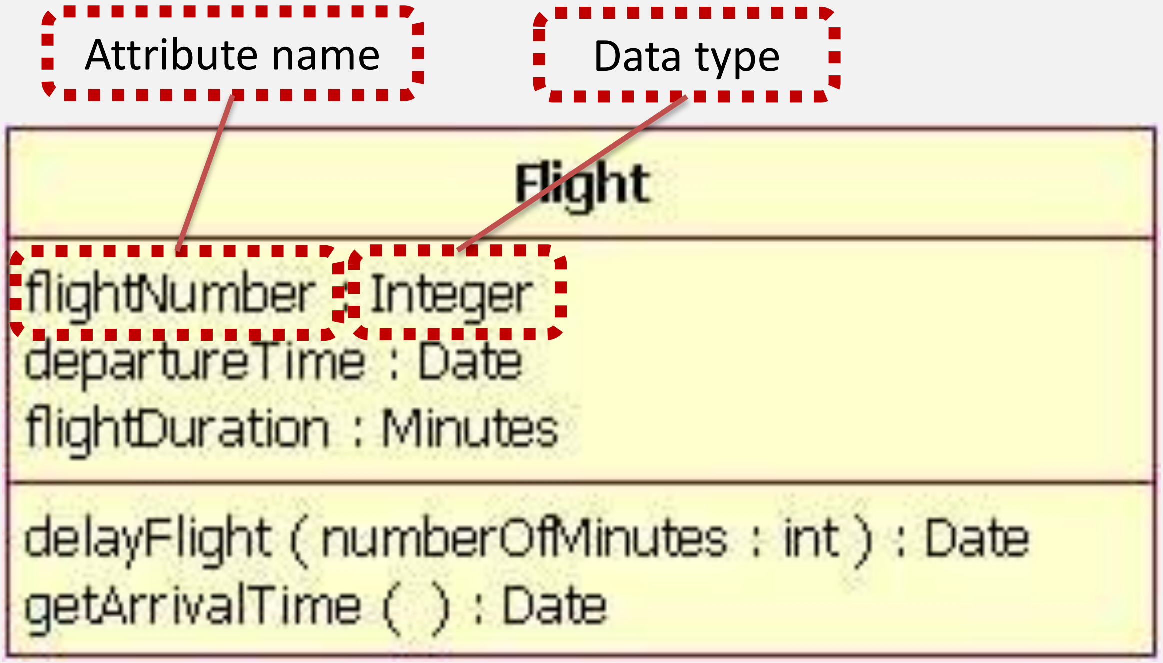
UML Exercise

- The class



UML Exercise

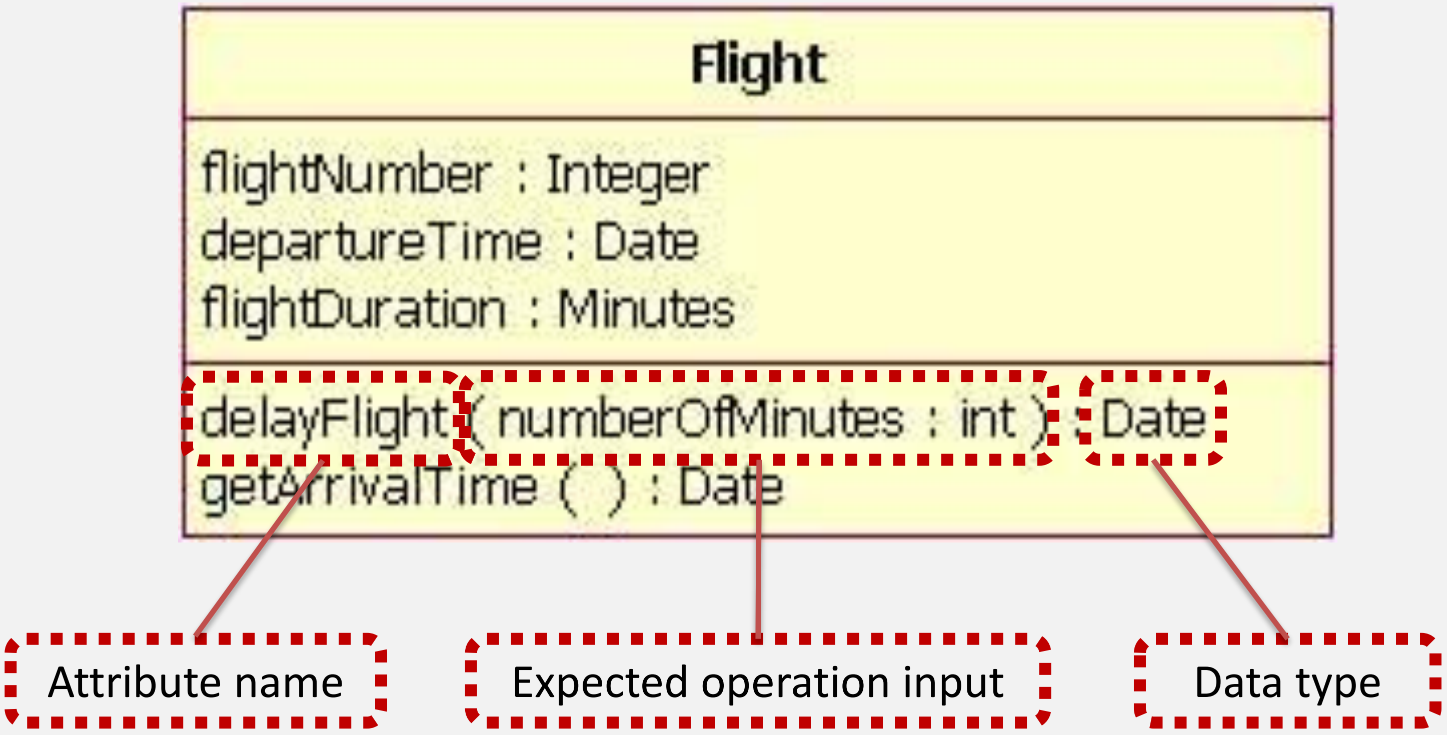
- The class attribute



E.g.:
-Integer
- LongInt
- Double
- Char
- Date
- Boolean
- String
- Geometry
- …

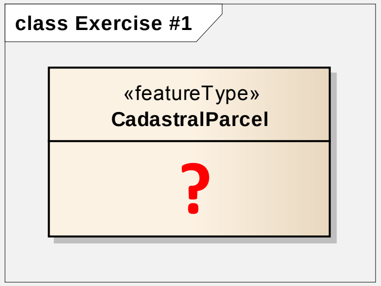
UML Exercise

- The class operations



UML Exercise

- Exercise #1 – The Class
  - Please develop/draw the class "Cadastral_Parcel"
    - What common characteristics (attribute: datatype) should the concept "Cadastral_Parcel" have?



Group 1

- Class: Parcel
- Attributes:
  - Object no.
  - Number
  - Cadastral municipality
  - Land use
  - Number of building
  - Address
  - Area
  - Owner

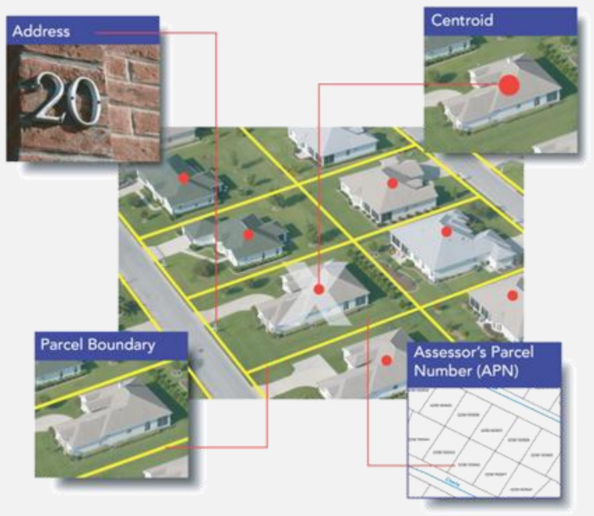
A multi-country project funded by the European Union and implemented by

Group 2

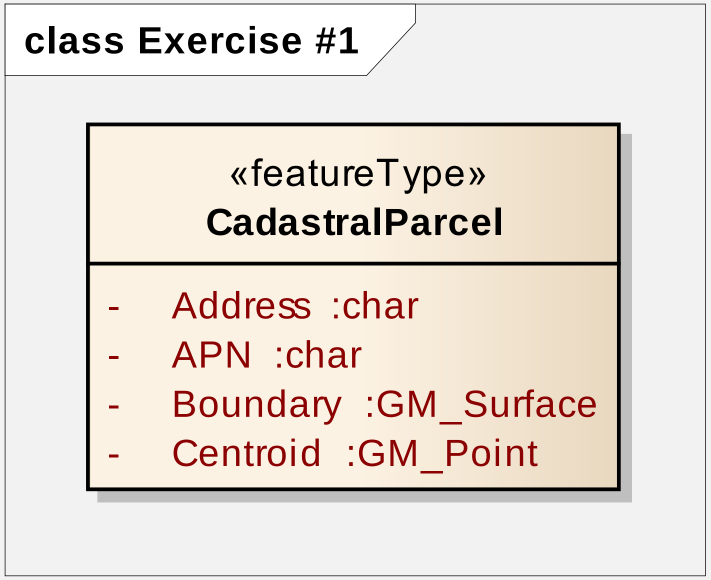
- The same as g1, just no address

UML Exercise

- Exercise #1 – The Class
  - Multiple solutions possible



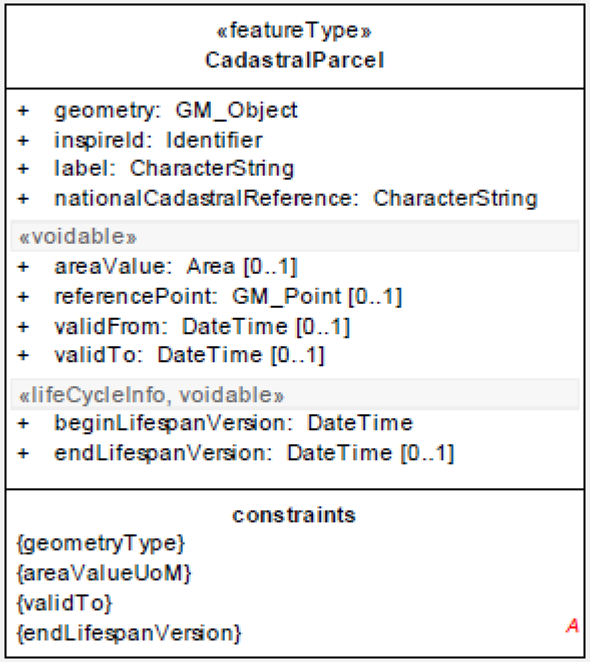
Four Elements for Parcels



A multi-country project funded by the European Union and implemented by

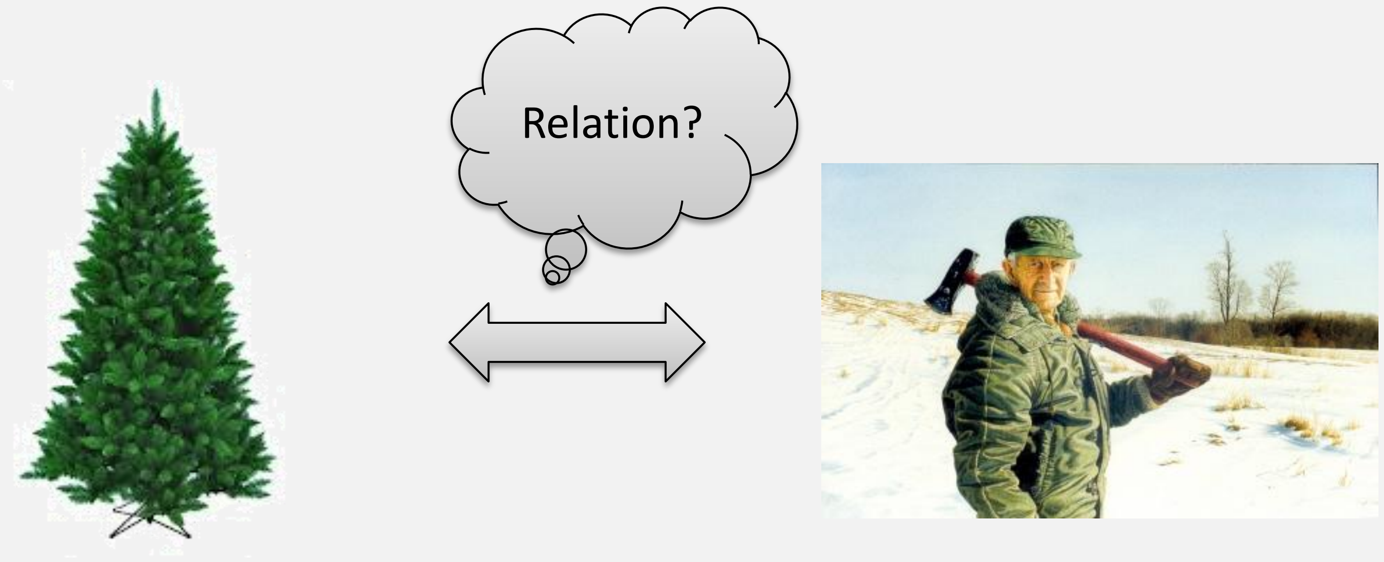
UML Exercise

- Exercise #1 – The Class
  - INSPIRE Data Specifications on Cadastral
    - Geometry
    - Label
    - National cadastral reference
    - Area value (optional)
    - Reference Point (optional)


«featureType»
CadastralParcel
+ geometry: GM_Object
+ inspireId: Identifier
+ label: CharacterString
+ nationalCadastralReference: CharacterString
«voidable»
+ areaValue: Area [0..1]
+ referencePoint: GM_Point [0..1]
+ validFrom: DateTime [0..1]
+ validTo: DateTime [0..1]
«lifeCycleInfo, voidable»
+ beginLifespanVersion: DateTime
+ endLifespanVersion: DateTime [0..1]
constraints
{geometryType}
{areaValueUoM}
{validTo}
{endLifespanVersion}
A

UML Exercise

- Relations



UML Exercise

- Relations
  - Associations
  - Generalisations
  - Aggregations
  - Compositions

A multi-country project funded by the European Union and implemented by

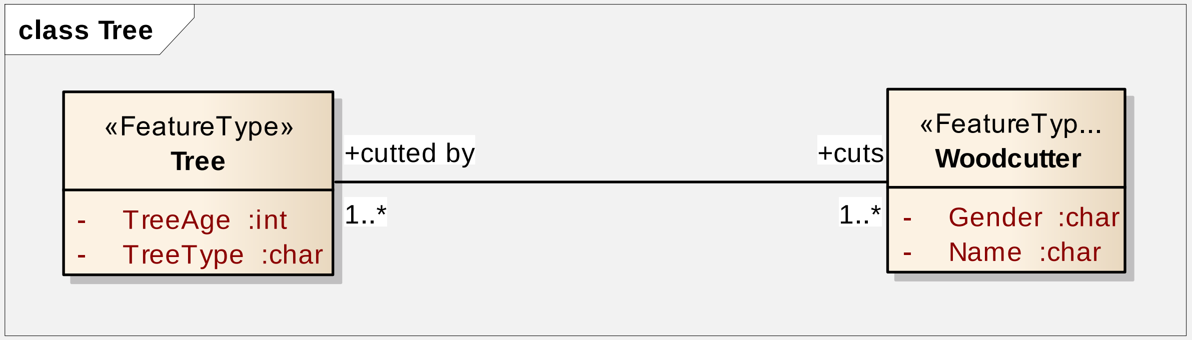
UML Exercise

- Associations
  - Implies that two classes have a relationship
  - General relationship connector
    - Target/Source roles
    - Cardinality
    - Directions
    - Constrains

A multi-country project funded by the European Union and implemented by

UML Exercise

- Associations



A multi-country project funded by the European Union and implemented by

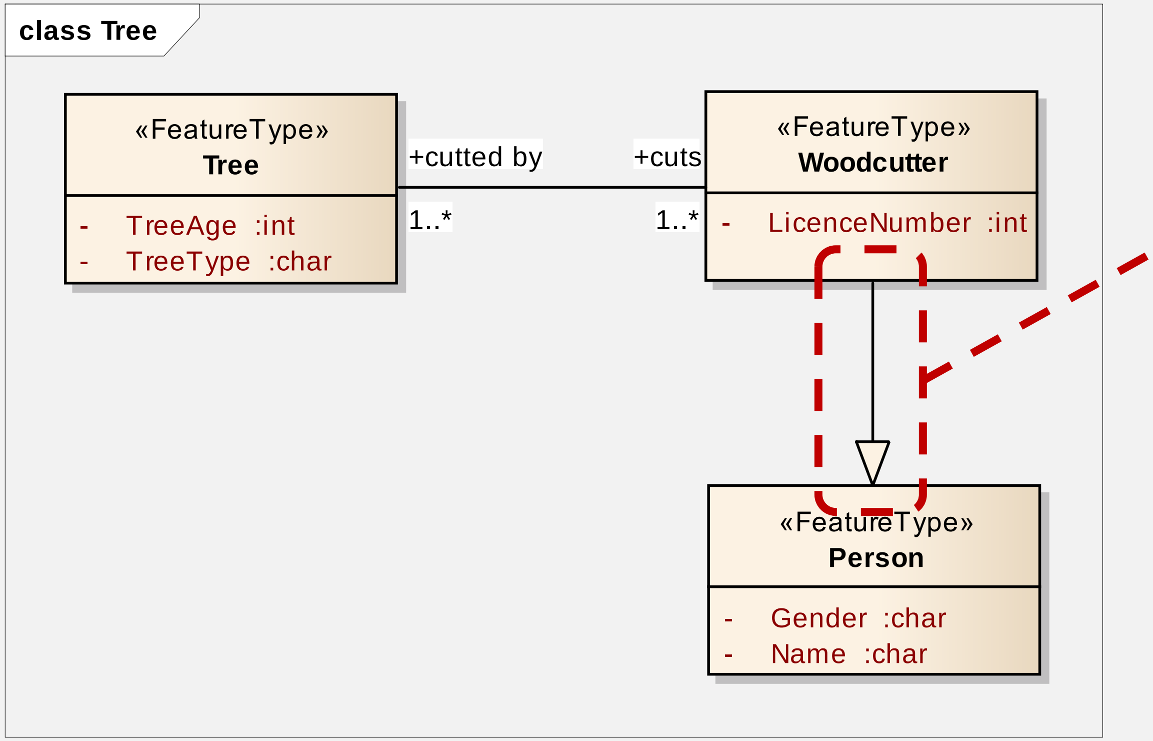
UML Exercise

- Generalisations
  - Indicated inheritance
    - Target/Source roles (e.g. isPartOf)
    - Cardinality
    - Constrains
  - Source inherits targets characteristic

A multi-country project funded by the European Union and implemented by

UML Exercise

- Generalisations



Woodcutter inherits attributes from Person

UML Exercise

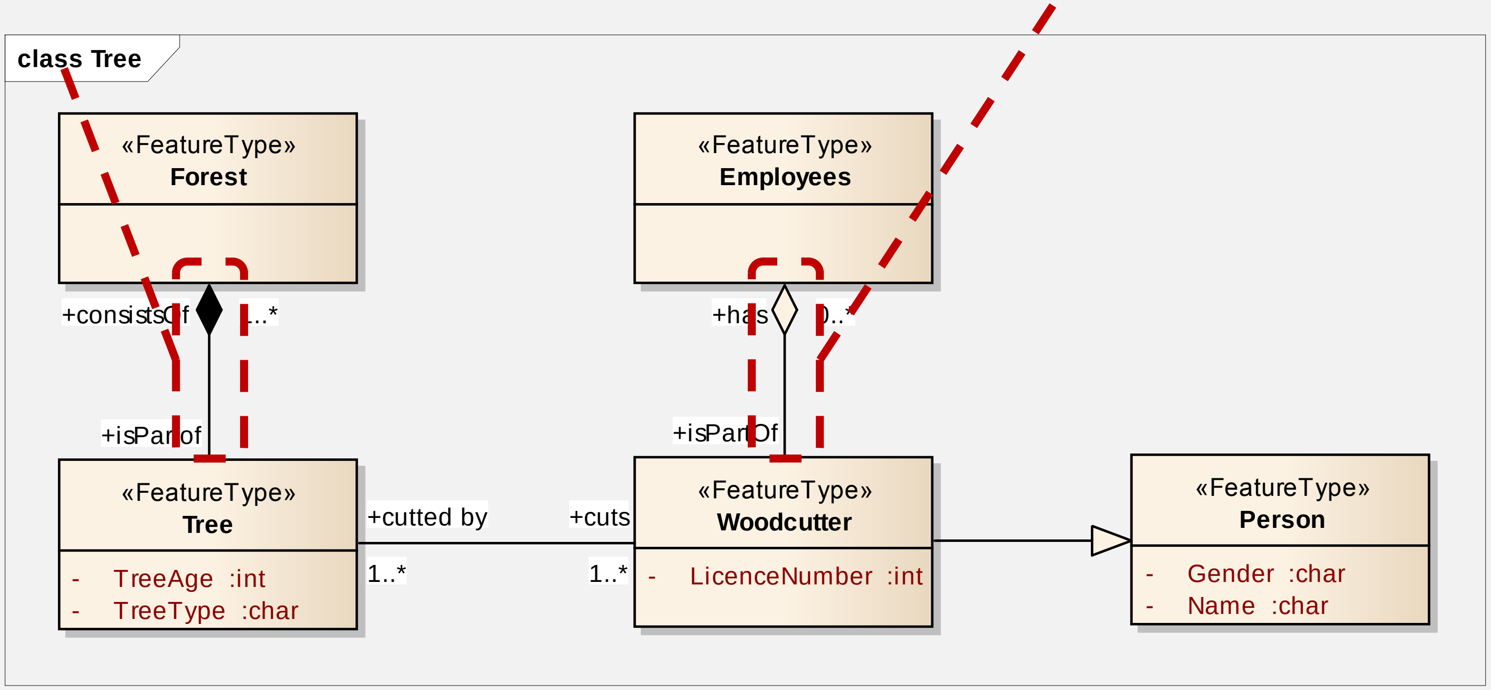
- Aggregations & Compositions
  - Indicates that the lower concept is part of a higher concept
    - Aggregation: Lower concept ISN'T necessary for existence of higher concept
    - Composition: Lower concept IS necessary for existence of higher concept

UML Exercise

- Aggregations & Compositions

Compositio
n

Aggregation

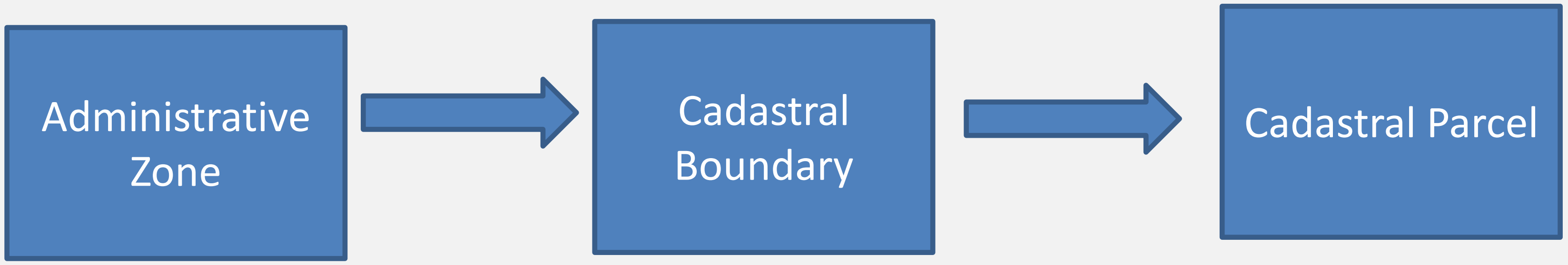


A multi-country project funded by the European Union and implemented by

UML Exercise

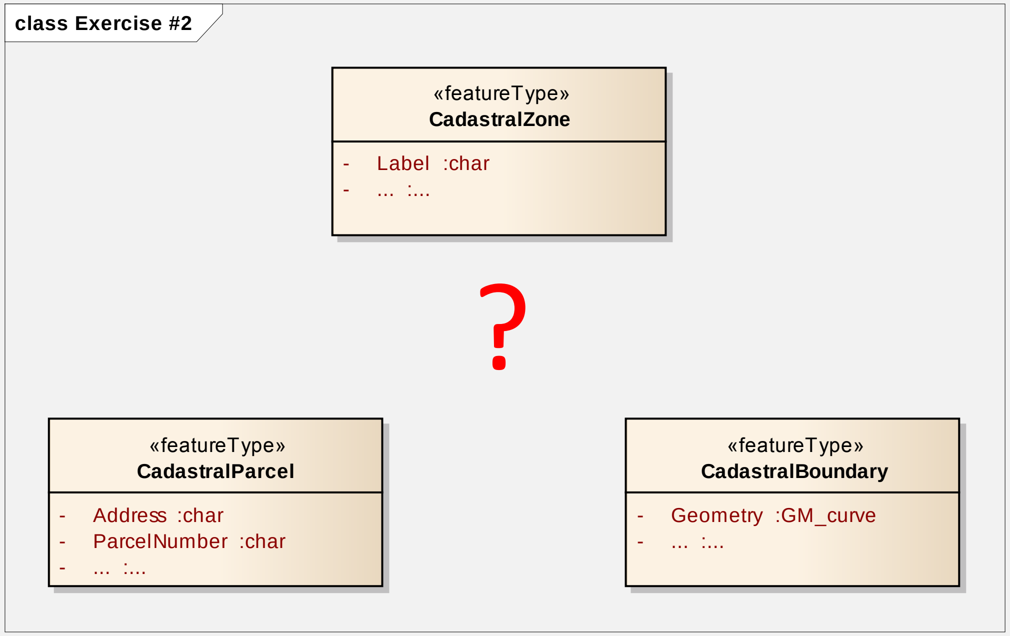
- Exercise #2 – The relationship types
  - Imagine you have 3 different classes
    - CadastralParcel
      - Core class
      - Is part of several(!) administrative zones (different levels of hierarchy)
    - CadastralBoundary
      - Indicates measured boundary of CadastralParcel
    - AdministrativeZone
      - Administrative zones with different hierarchal levels which existence doesn't depend on CadastralParcel
  - Please develop diagram using relationship types and classes with (some) attributes!

Exercise 2


Administrative Zone
Cadastral Boundary
Cadastral Parcel

UML Exercise

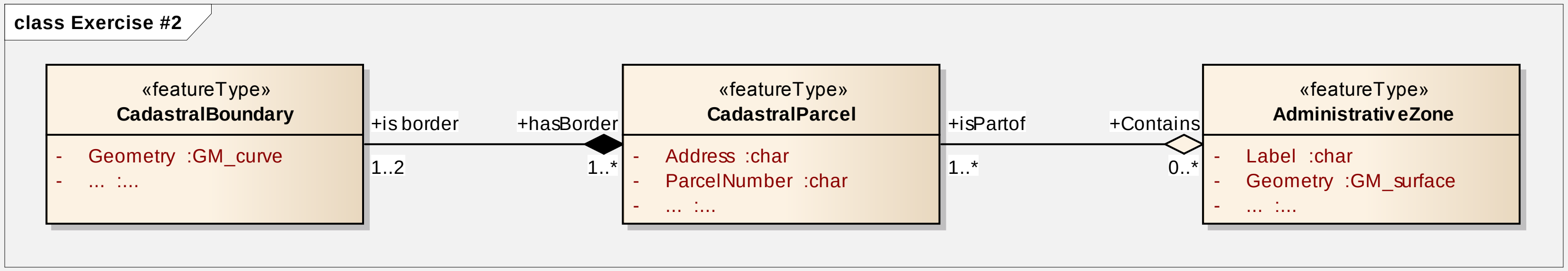
- Exercise #2 – The relationship types
  - Again there are multiple solutions



A multi-country project funded by the European Union and implemented by

UML Exercise

- Exercise #2 – The relationship types
  - There are multiple solutions
  - One example:



A multi-country project funded by the European Union and implemented by

INSPIRE Cadastre

- INSPIRE Data specifications on cadastral
  - http://inspire.jrc.ec.europa.eu/

A multi-country project funded by the European Union and implemented by

References

- OMG - UML
  - http://www.uml.org/
- Sparx Systems
  - http://www.sparxsystems.com/resources/uml2_tutorial/index.html
- Learners support publication
  - http://www.lsp4you.com/seminar.htm

Contact & Information

Christian Ansorge

+43-(0)1-313 04/3160

christian.ansorge@umweltbundesamt.at

Ivica Skender

+385-(0)91-33667-120

ivica.skender@gdi.net

Bečići ■ December 5, 2012

A multi-country project funded by the European Union and implemented by