

UML for LPIS

Baveno, 25 March, 2015

www.jrc.ec.europa.eu

Serving society
Stimulating innovation
Supporting legislation

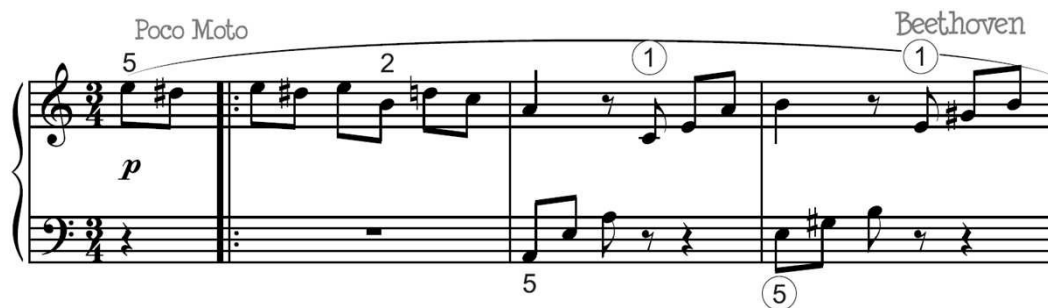


What is UML?

Unified Modelling Language

- A language that is used for model description, which has
 - Vocabulary
 - Grammar
 - Notation
- Purpose oriented compact presentation
- Interface between IT and domain experts

Other examples where the usage of specific notation is widely accepted:



$$c^2 = a^2 + b^2$$

UML “dialect” used for LPIS (CAP) models

- A limited number of possible modelling elements/constructs (to keep it simple)
- GML (Geography Markup Language) profile (extension) to handle spatial data
- The specific profile has been defined in the reference model

Rather abstract; the target audience are developers (business analysts, information architects, software developers)

The content is exemplary – does not do anything with the content of the LPIS (CAP) domain models

Has been reviewed by leading European modelling experts in June 2014

Can be downloaded from our ftp (but do not analyse the content!! It is fictitious)

- Instead of discussing the reference model, this tutorial goes through specific examples

Diagram types used in LCM

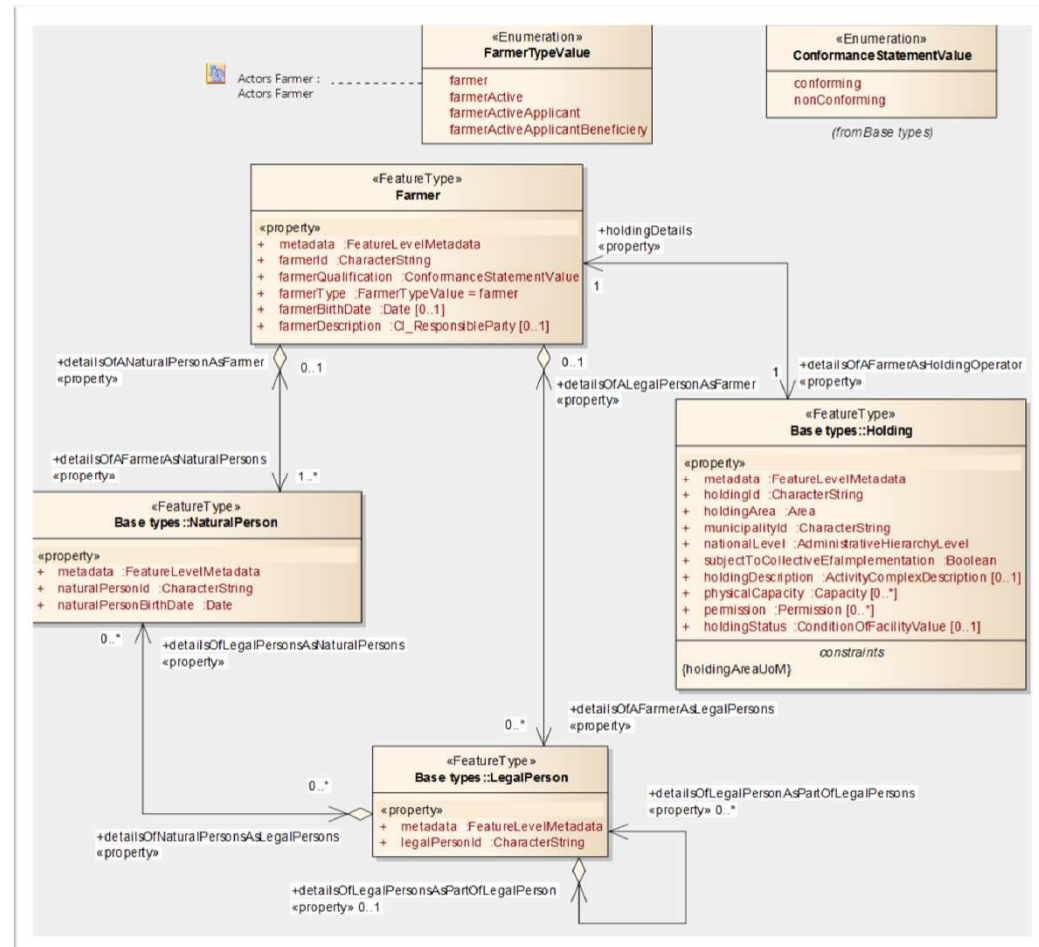
| Diagram type | Usage |
|-----------------------|---|
| Package diagram | Presentation of LPIS as subsystem within IACS. |
| Class diagram | Base types LPIS core model (LCM) LPIS QA conceptual model |
| Use case diagram | Perform LPIS upkeep (high level) Perform LPIS schema upgrade Perform LPIS data update Perform MTS Perform ETS Prepare for testing (high level) |
| Activity diagrams | Each of the above use cases are described as series of low level use cases or series of atomic (undividable) actions. |
| State machine diagram | Description of life-cycle states of schema elements or data instances |

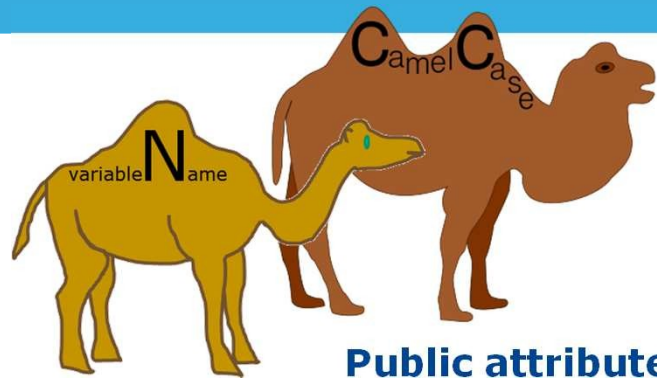
- Part of the conceptual model
- Tool for structuring
- Shows how the packages are related
- What information resides in which package



Class diagram

- Part of the conceptual model
- Shows how the information is structured main elements (classes) properties of classes
- Properties are presented as attributes or relationships

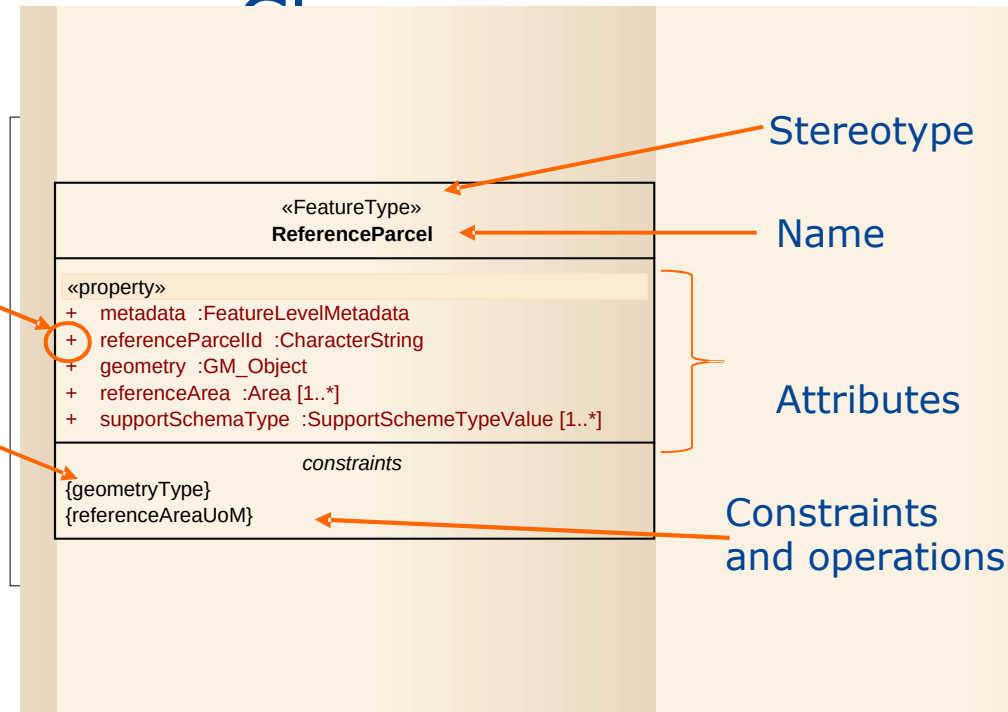




Public attribute

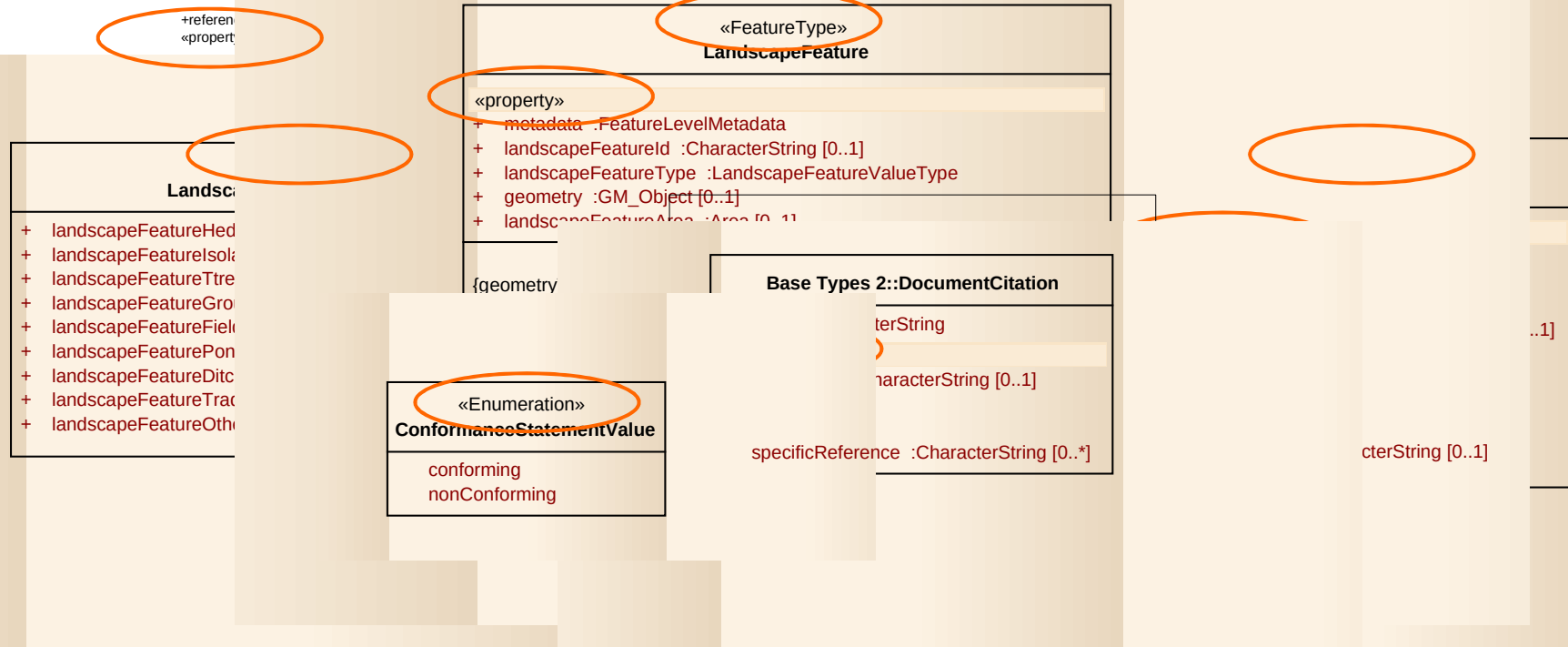
Details in text and Object Constraint Language (OCL)

```
/* Type of geometry shall be
GM_Surface or GM_Point */
inv: geometry.ocllsKindOf(GM_Surface)
or geometry.ocllsKindOf(GM_Point)
```



- Representation of a certain type of an object that reflects the structure and the behavior
- Class models usually describe the logical structure of the system (how the information is structured)
- Operations in LCM are not used

Stereotypes



- Code list: list of possible values that a property may take (can be extended)
- Enumeration: finite list of values that an attribute may take
- Property: Descriptive element related to feature and data types. Can be used for associations
- Leaf: the package cannot be further subdivided



European
Commission

Extension of code lists

rules:

new value

produced

: only

ly narrower

be

introduced

None : centrally

managed code list; only

owner of LCM (i.e.

services) may

duce a new value

Tagged Values

- Class (LandscapeFeatureValueType)
 - extensibility
 - vocabulary
- GML::CodeList (LandscapeFeatureValueType)
 - asDictionary
 - defaultCodeSpace

«CodeList»
LandscapeFeatureValueType

- + landscapeFeatureHedgesAndWoodedStrips
- + landscapeFeatureIsolatedTree
- + landscapeFeatureTreesInLine
- + landscapeFeatureGroupOfTrees
- + landscapeFeatureFieldMargin
- + landscapeFeaturePonds
- + landscapeFeatureDitches
- + landscapeFeatureTraditionalStoneWalls
- + landscapeFeatureOtherProtectedByGaecSmr

Tagged Values

- Class (AgriculturalAreaTypeValue)
 - extensibility
 - vocabulary
- GML::CodeList (AgriculturalAreaTypeValue)
 - asDictionary
 - defaultCodeSpace

«CodeList»
AgriculturalAreaTypeValue

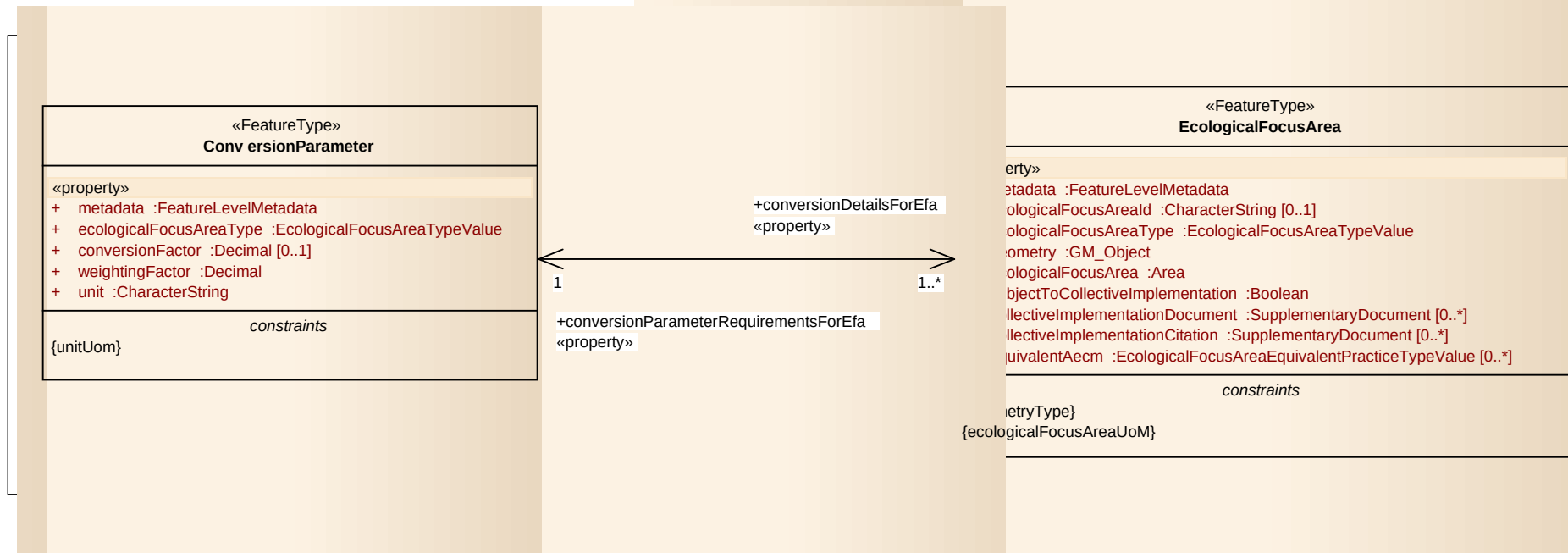
- + arableLand
- + permanentCrop
- + permanentGrassland
- + permanentGrasslandSensitive
- + permanentGrasslandElp

Tagged Values

- Class (TechnicalReasonValue)
 - extensibility
 - vocabulary
- GML::CodeList (TechnicalReasonValue)
 - asDictionary
 - defaultCodeSpace

«CodeList»
TechnicalReasonValue

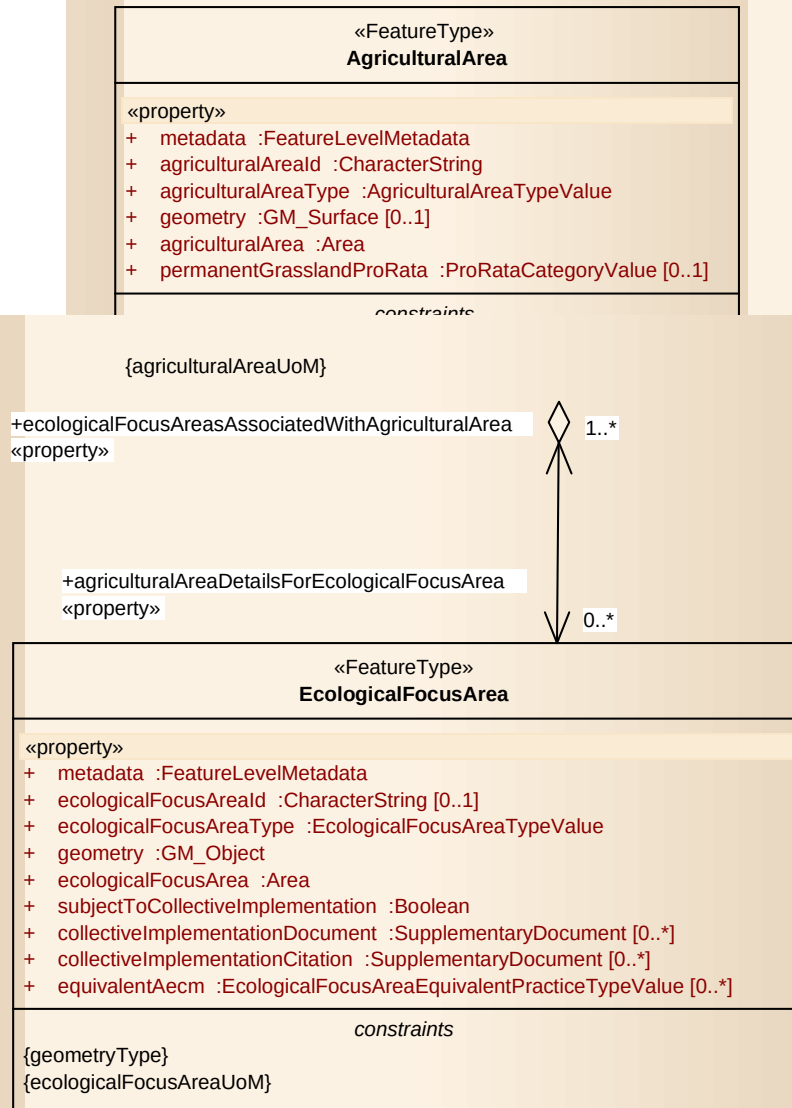
- + missedUpdate
- + missedUpgrade
- + incompleteProcessing
- + processingError
- + incompatibleDesign



- has named roles at each end, multiplicity, direction and constraints

Reading:

- From left to right: one conversion parameter may belong to one or more EFA types
- From right to left: an EFA type always has conversion parameter (i.e. at least the weighting factor).

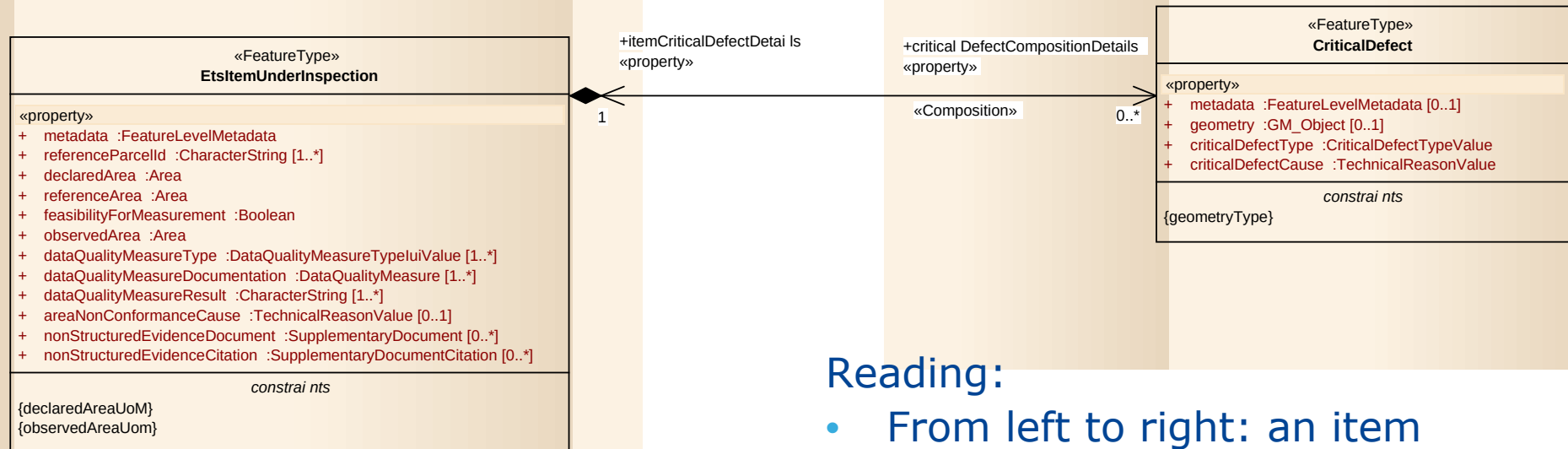


Aggregation

Aggregation that shows
that contains other

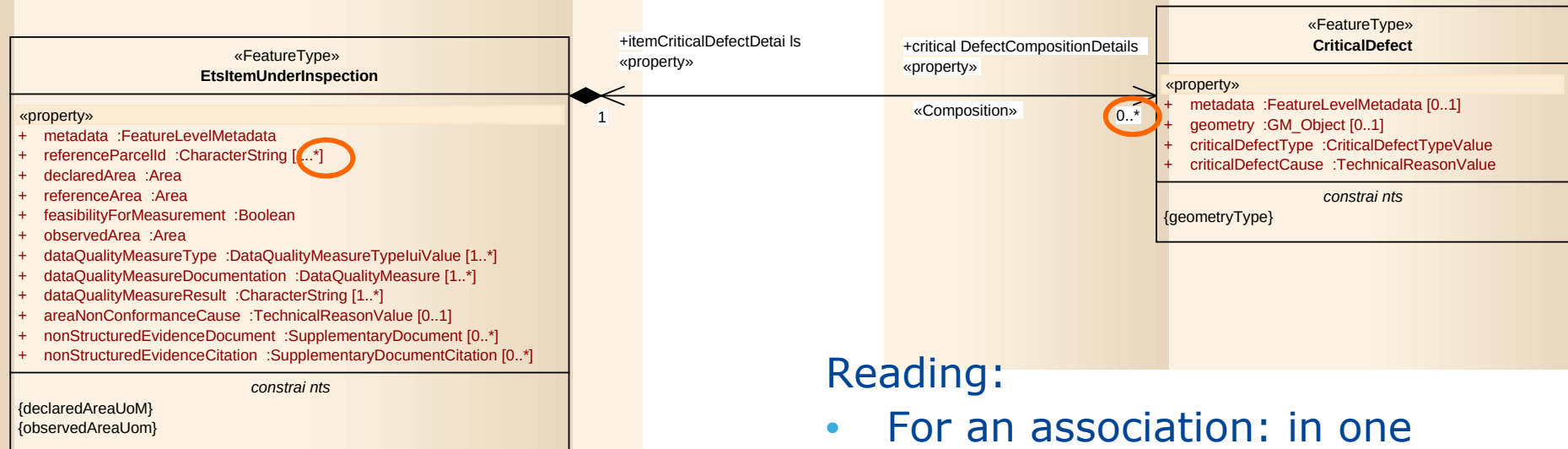
Diagram shows the
relationship.

Examples: an agricultural area
may not contain EFA
an EFA may belong to
an agricultural area
(if it is on the boundary)



Reading:

- A type of association that is used to depict an element that is made up of smaller components.
- It is a “strong” aggregation; when a composite is deleted, all of its parts are deleted with it.
- From left to right: an item under inspection consist of zero or more critical defects
- From right to left: a critical defect cannot exist without an item under inspection. When the item under inspection is deleted, the critical defect disappears, too.

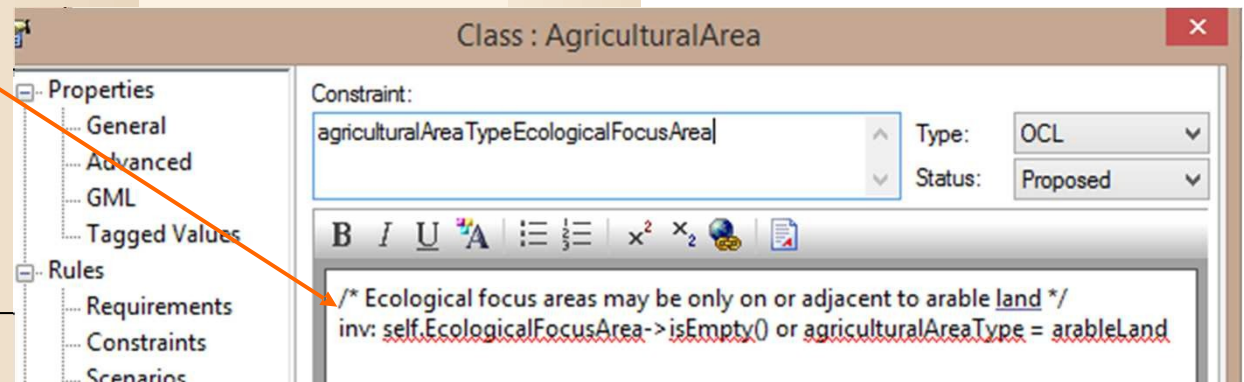
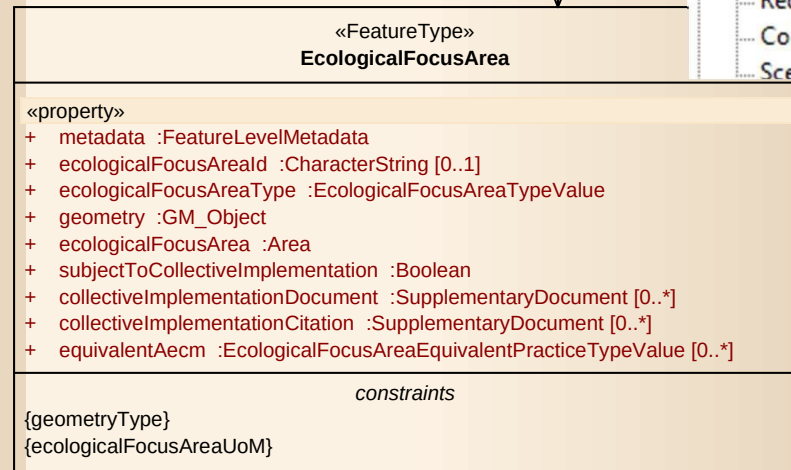
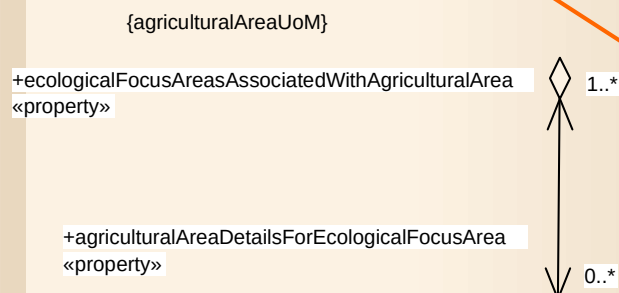
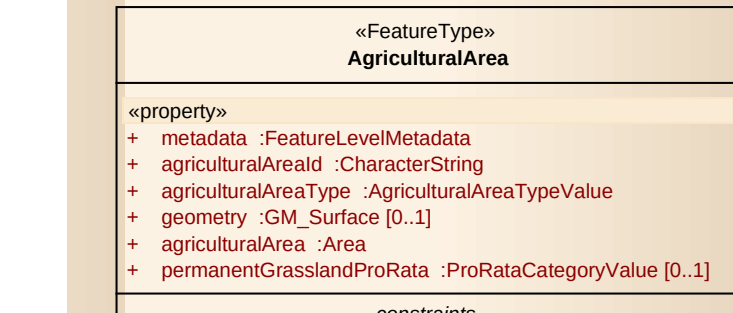


Reading:

- Number of occurrences that a property may take
- Applicable to attributes and associations
- For an association: in one inspected item the number of critical defects may be 0, 1, 2,... or any more
- For an attribute: the number of RP identifiers related to an inspected item is 1 (single parcel inspected) or more (an aggregate of parcels inspected)

Constraint

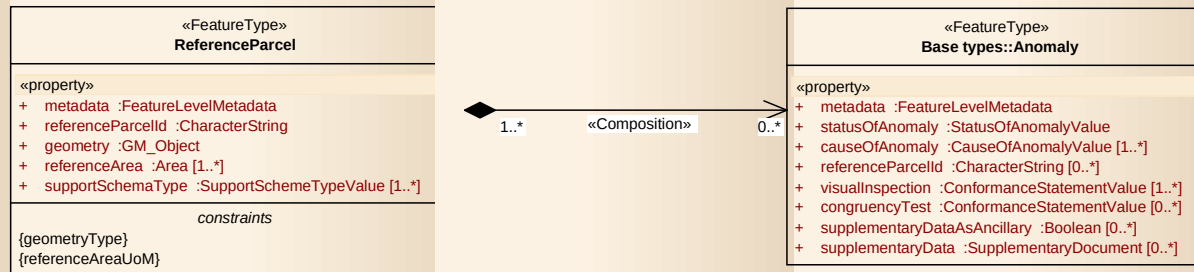
- Extention of UML that defines such conditions that cannot be described with standard UML elements



- Reading: if an instance of agricultural area does not belong to the arable type than no EFA instance can be defined for it

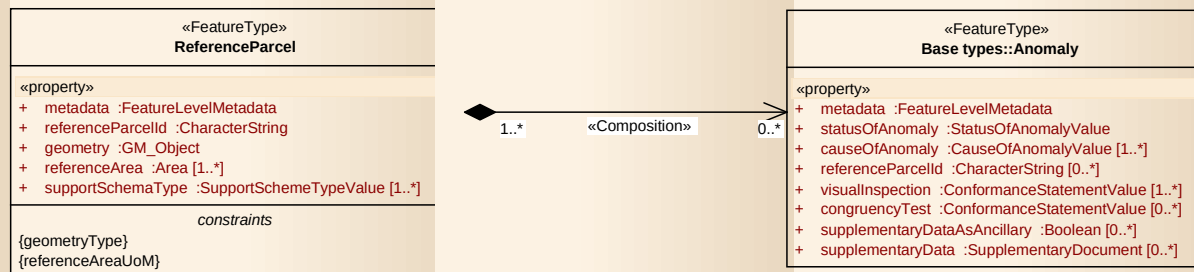
Quiz: what is wrong in this model?

GML Identification system for agricultural parcels



Quiz: answer

GML Identification system for agricultural parcels



- There is a redundant element: `referenceParcelId`

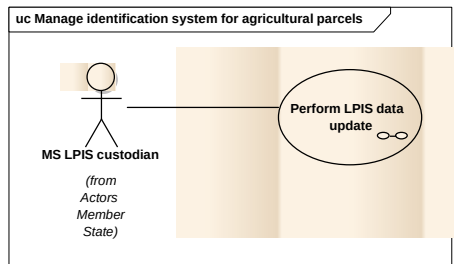
Either the connector has to be removed, or the attribute of the Anomaly feature type has to be deleted.

The way of correction depends on preference: whether all the attributes of ReferenceParcel feature types are necessary to deal with anomaly, or it is enough to have the ID.

- Use case diagram presents how the users (actors) interact with a system
- A use case is a discrete unit of work that the actor may perform using the system
- A high level use case can be subdivided in lower level use cases
- Even a low level use case has a meaning in terms of the business to be performed

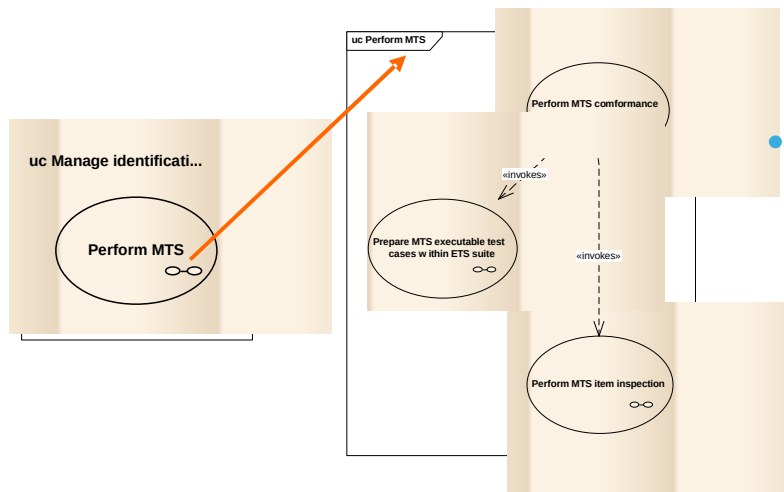
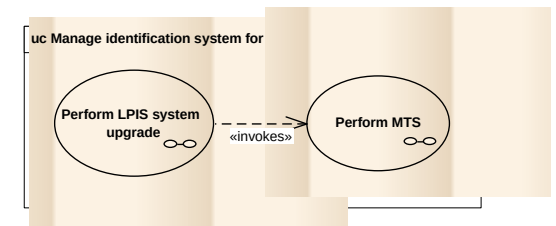


Navigation in use case diagrams (1)



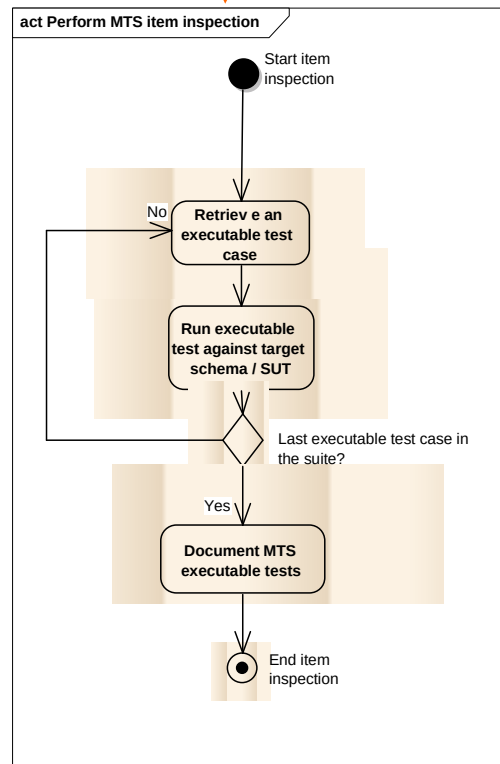
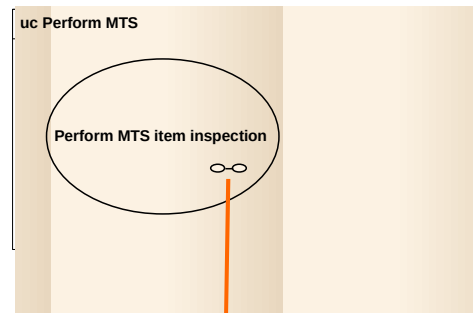
- Connector “use”: semantic relationship between an actor and a use case

- Connector “invoke”: indicates that the first use case causes the second to happen
- Reading: LPIS upgrade invokes MTS; i.e. whenever the LPIS schema is changed MTS has to be performed



- A high(er) level use case can be further detailed by low level use cases

Navigation in use case diagrams (2)

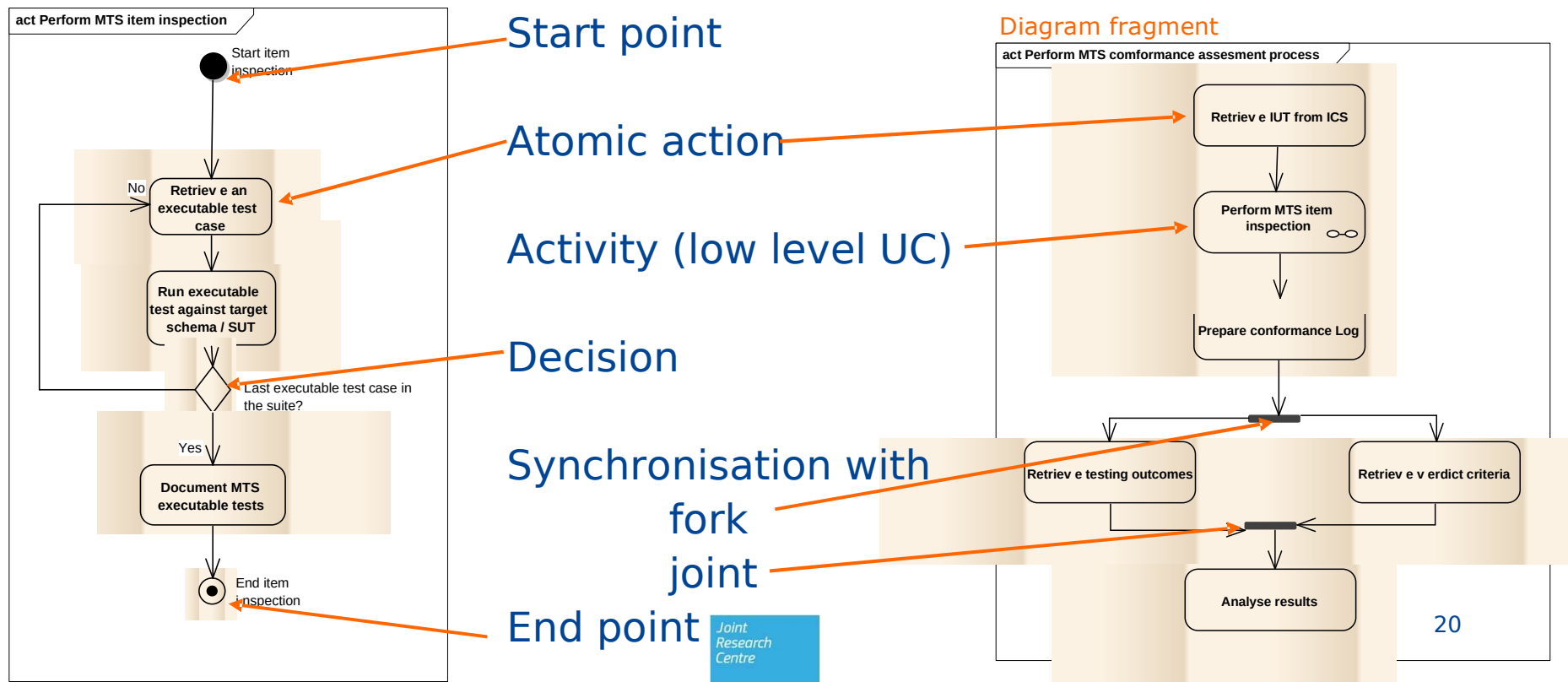


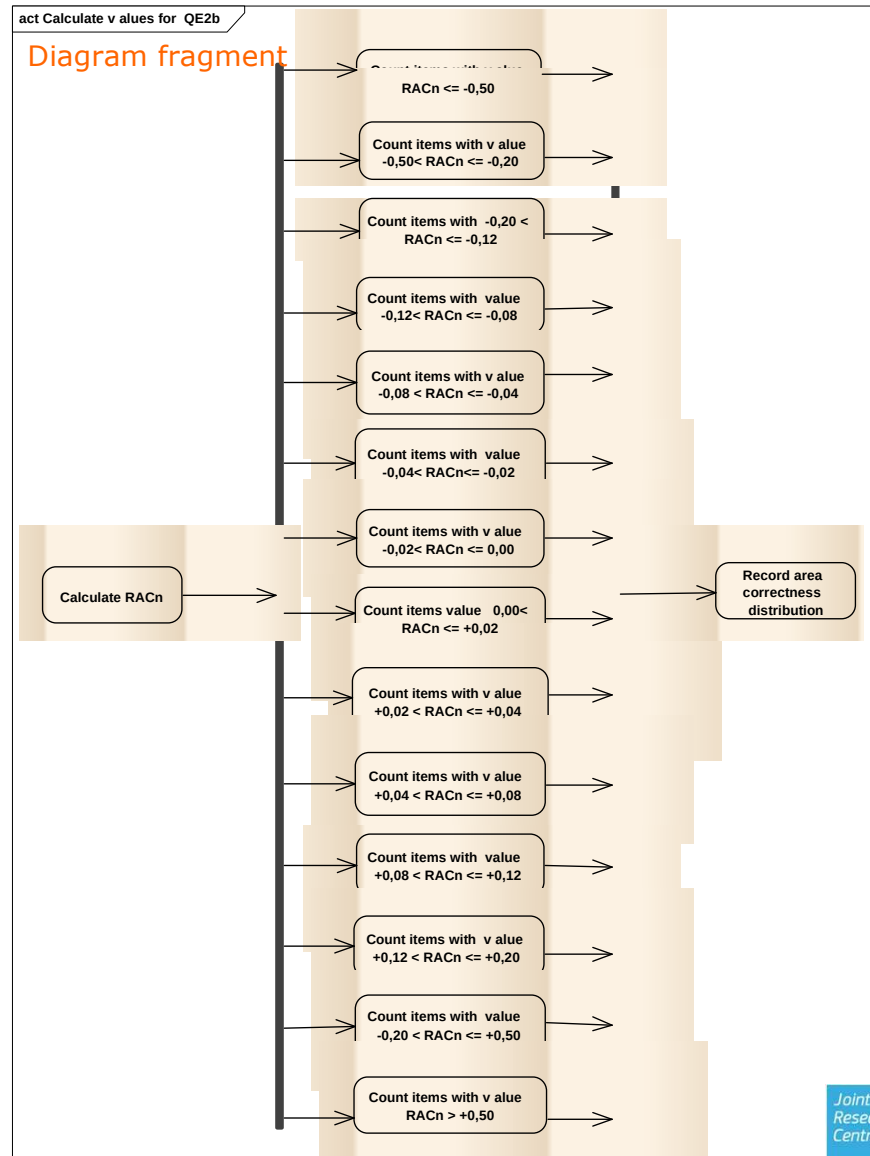
- A use case can be further detailed by an activity diagram

Activity diagram

- Describes the flow of events/actions within a use case

Includes

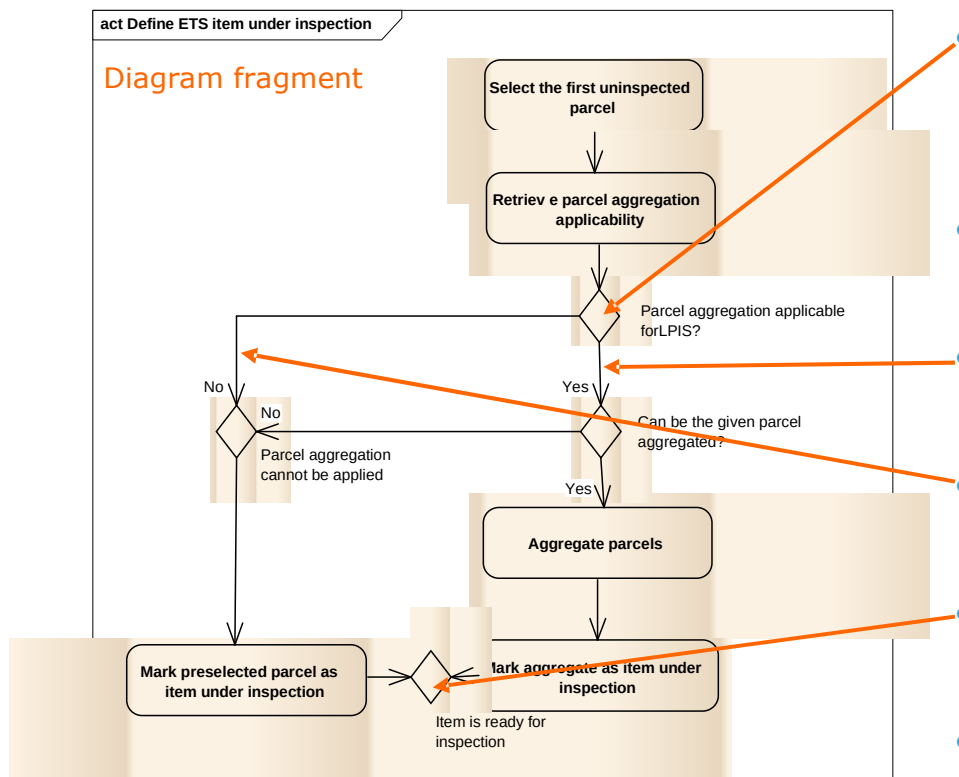




Activity diagram: synchronisation

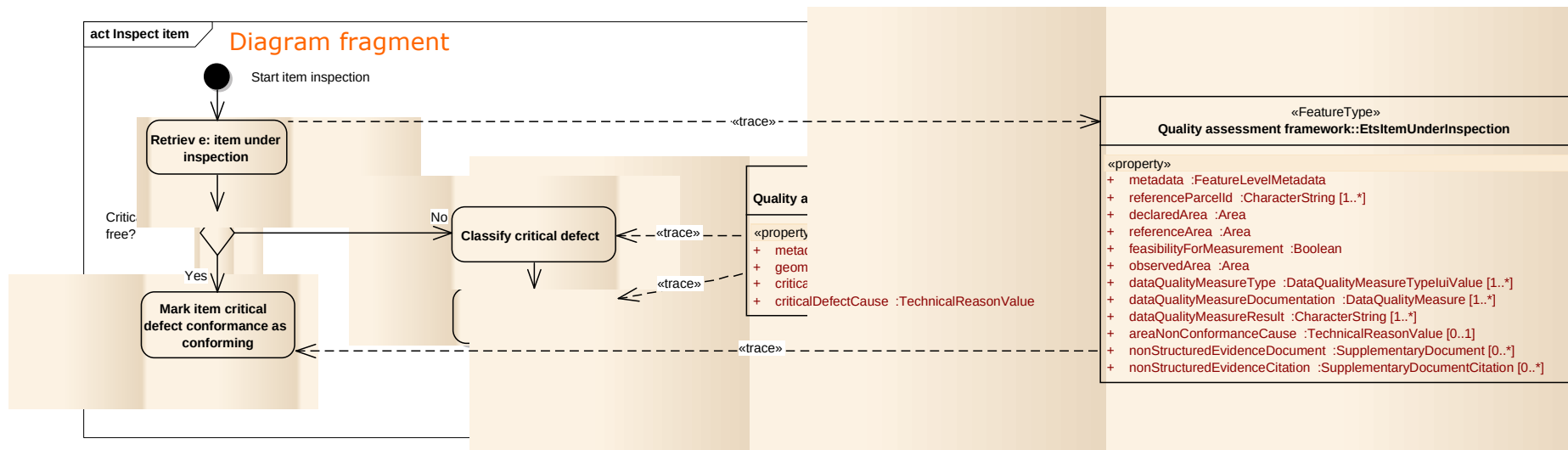
- At some moment of the activity a number of actions has to be take place in parallel in order to be able to move to the next action
- The start of synchronisation is noted by a “fork ”, the end by a “join ”

Activity diagram: decision



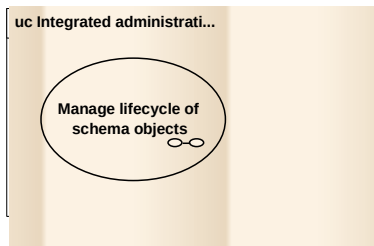
- At some point in the activity a decision has to be taken. This is described by a question.
- Based on the decision different flow of events is performed.
- “Yes” direction indicate the main flow of events
- “No” direction indicate alternative flow of events
- The main and alternative flows meet again in a merge point
- Only one of the flows can happen in one activity

Links between different diagram types: tracing (1)



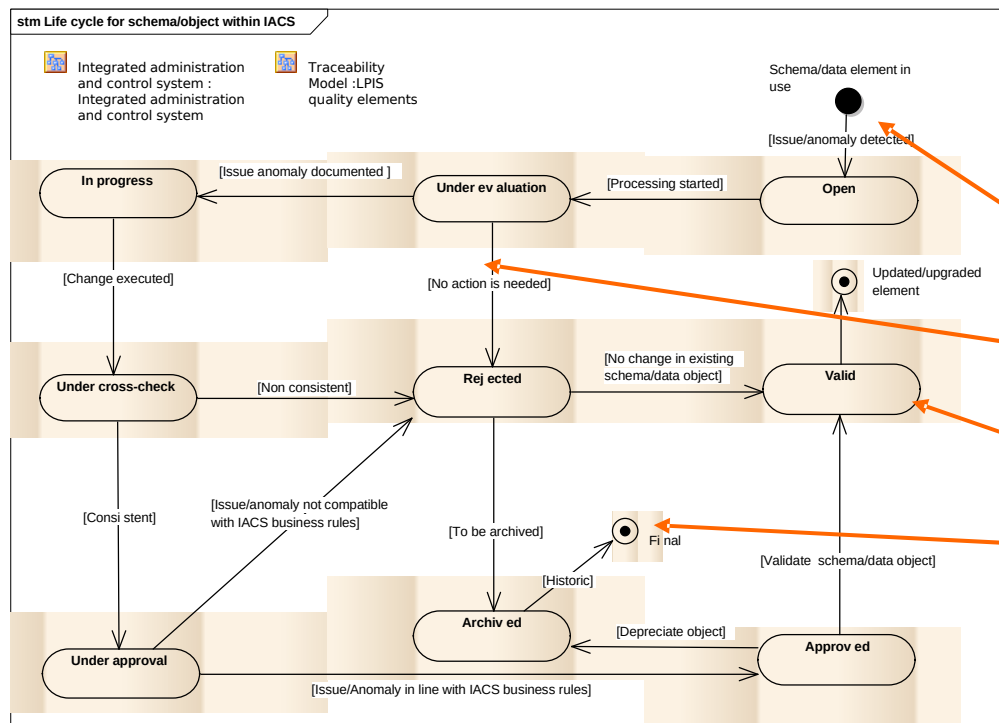
Tracing between activity and class

- Specifies which element of the class is being used in an action is being performed
- Tracing means that the object monitors the other one in order to retrieve input for an operation (to perform an action, or change the value of a bit of information (attribute)).



State machine diagram

Describes how an object changes its state when a use case is being performed

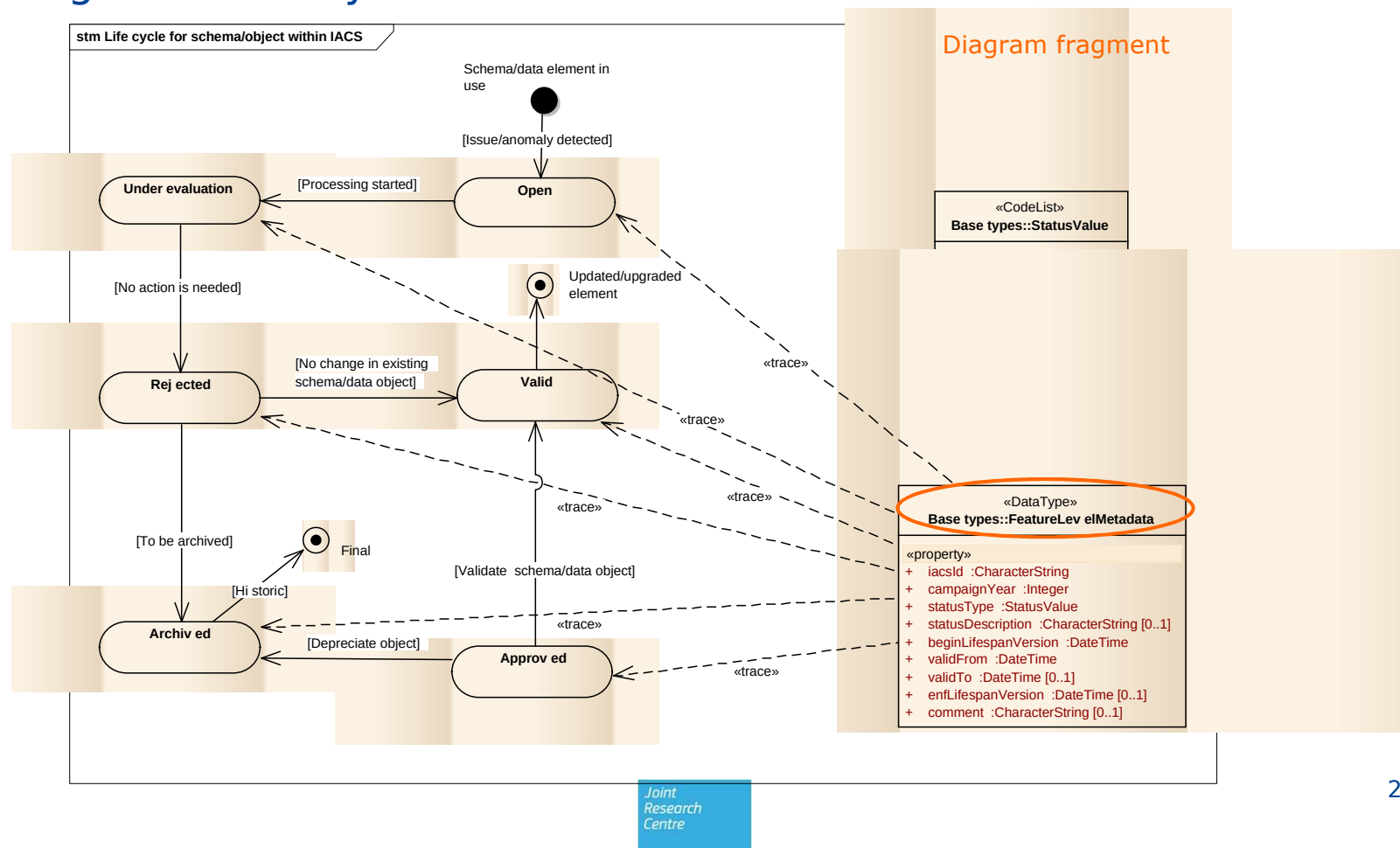


- The object can be a data instance (update) a schema element (upgrade)
- The diagram has
 - Initial state (only one)
 - Connectors (messages) that cause transition in the state
 - States
 - Final state (may be more)
- Is widely used to assign life-cycle metadata

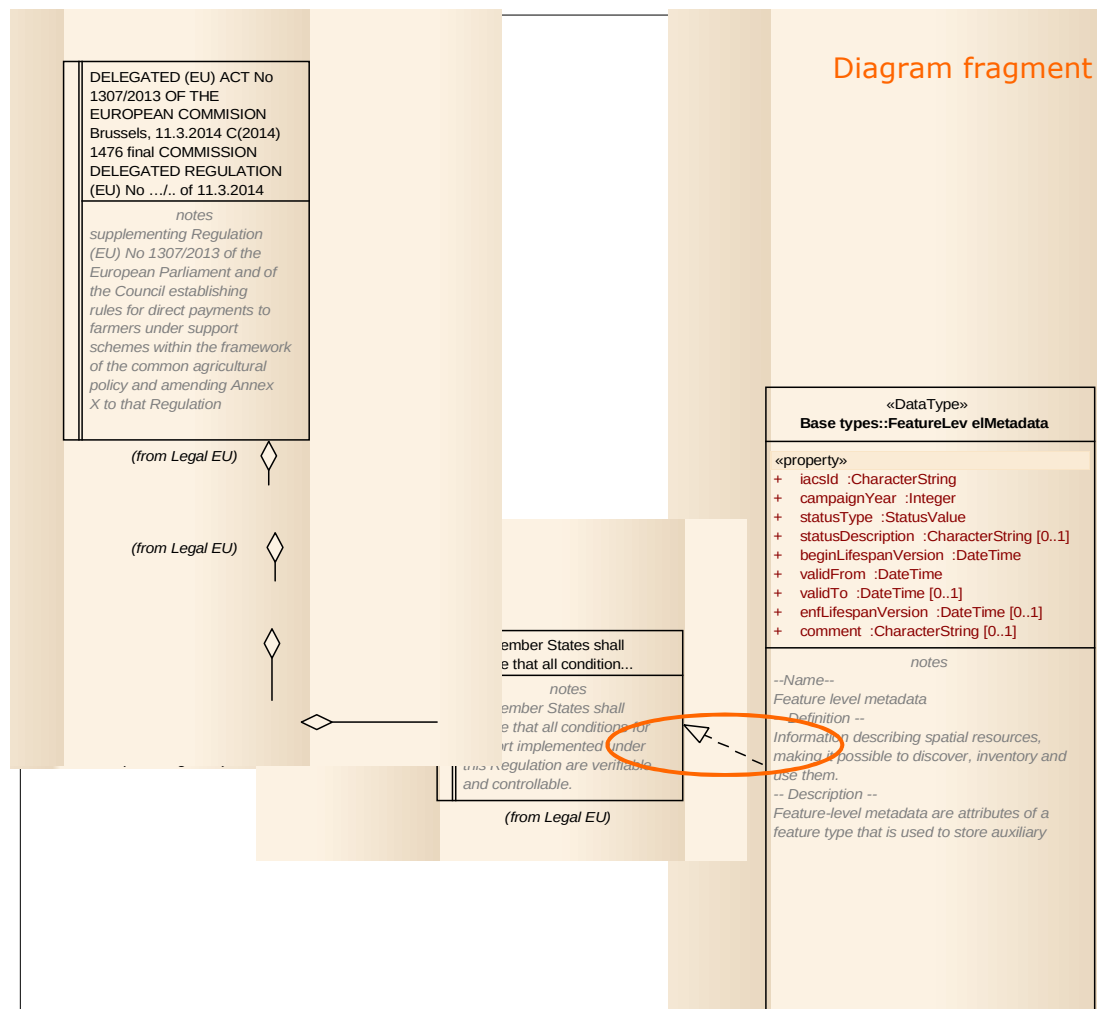
Links between different diagram types: tracing (2)

Tracing between state machine and class diagrams

- Assignment life cycle metadata values



Links between different diagram types: realisation



(connector) between
elements of
requirement model and
conceptual model
(as diagram)
With rigorous presentation
of requirement the
completeness of the
conceptual and business
models can be verified

Now...we all need a break!

