February 2017

Use of UML in Tranmodel

Public Transport Reference Data Model (Transmodel)

CEN TC278/WG3/SG4 PT0302

Table of Contents

Introduction

This document represents a short guide to describe how the UML (unified modelling language is used in the Public Transport Reference Data Model (Transmodel).

Typographic conventions

This Standard makes use of specific typographic conventions that have been adopted for previous and related Standards and Technical Specifications. Unless the context dictates otherwise:

- Terms wholly in CAPITAL LETTERs refer to a concept which is defined in the Data Dictionary in the relevant part or in a part with a lower number, i.e. capitalised concepts in Part 1 are defined in the Data Dictionary of Part 1, capitalised concepts in Part 2 are defined either in the Data Dictionary of Part 2 or of Part 1, etc. Note that pluralisation of such an entity is indicated by the addition of a lower case "s". It is planned that a complete Data Dictionary will be issued as a separate document, updated as additional Parts of this Standard are published.
- Terms wholly in CAPITAL LETTERs and in *italic characters* appearing mainly in the diagrams concern abstract classes, i.e. classes which cannot be instantiated directly. They represent common characteristics of all their sub-classes (specialisations).
- Terms wholly in lower case letters refer to the use of those words in their normal everyday context.
- Terms in *italic characters* are used for explanatory text, particularly related to the context in which a defined entity may be found.
- Terms in UpperCamelCase are class attributes, such as PersonCapacity, AtCentre, IsExternal, etc.
- The use of colours helps the reader to link the different classes with similar semantical meaning to a particular package.
- The word "model" is written either "model", or "Model", or "MODEL". The diagram notes marked MODEL refer to the corresponding conceptual diagrams of the NeTEx documentation.

Methodology for conceptual modelling

General

Notation UML 2 is object–oriented modelling notation and is used for describing (specifying, documenting and visualizing) the conceptual data model in Transmodel. The UML specification has proved efficient because it facilitates common understanding and use of conceptual data model.

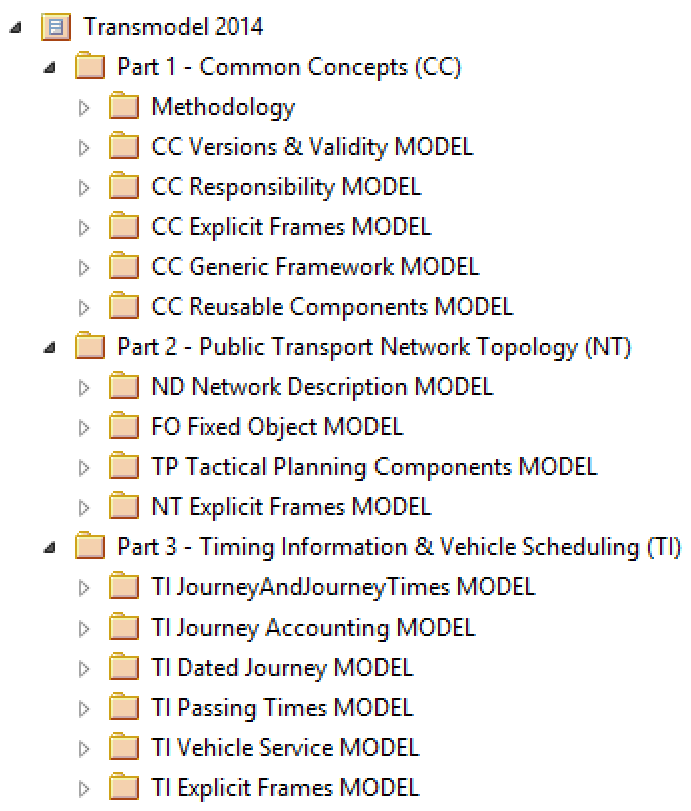
Transmodel uses a notation that bears some features of UML 1 (or E/R conceptual modelling), in particular as regards the labelling of roles/relationship names.

The following section summarises the UML features used in Transmodel and illustrates them with corresponding example diagrams. Diagrams in Transmodel documents are designed with the modelling tool Enterprise Architect version[1] 10.0 (EA).

Packages

Transmodel EA model is structured into main packages corresponding to the different parts (Part 1, Part 2 , etc) containing sub-packages (models), which group classes according to a common functional objective. Specific packages "Explicit Frames" in the different parts are created and details of the models contained in them will be discussed in the relevant parts. The hierarchical modular structure is shown in Figure 1.

[1] A useful reference may be found at the following address: http://www.sparxsystems.eu/resources/project-development-with-uml-and-ea/



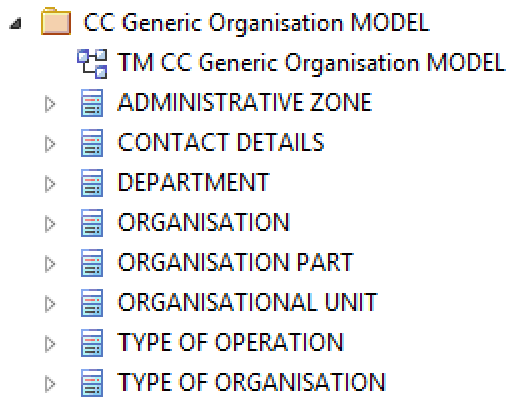
- Transmodel 2014
  - Part 1 - Common Concepts (CC)
    - Methodology
    - CC Versions & Validity MODEL
    - CC Responsibility MODEL
    - CC Explicit Frames MODEL
    - CC Generic Framework MODEL
    - CC Reusable Components MODEL
  - Part 2 - Public Transport Network Topology (NT)
    - ND Network Description MODEL
    - FO Fixed Object MODEL
    - TP Tactical Planning Components MODEL
    - NT Explicit Frames MODEL
  - Part 3 - Timing Information & Vehicle Scheduling (TI)
    - TI JourneyAndJourneyTimes MODEL
    - TI Journey Accounting MODEL
    - TI Dated Journey MODEL
    - TI Passing Times MODEL
    - TI Vehicle Service MODEL
    - TI Explicit Frames MODEL

Figure 1 - Transmodel Hierarchy of Packages

A prefix in front of each package name indicates the part if the standard where this package has been introduced and described first, e.g.:

- CC stands for Common Concepts
- NT stands for Network Topology
- ND stands for Network Description
- FO stands for Fixed Objects
- TP stands for Tactical Planning Components
- TI: Timing Information & Vehicle Scheduling

The classes are grouped together in a package for a specific task or functional purpose. Figure 2 shows content of the package “Generic Organisation Model”, which contains 8 classes. Each class has one and only one “home” package.



CC Generic Organisation MODEL
TM CC Generic Organisation MODEL
ADMINISTRATIVE ZONE
CONTACT DETAILS
DEPARTMENT
ORGANISATION
ORGANISATION PART
ORGANISATIONAL UNIT
TYPE OF OPERATION
TYPE OF ORGANISATION

Figure 2 - Package Content Example

Class diagrams

Class diagram is a visual representation of the structure of a system by showing the system's classes, their attributes, operations and the relationships among the classes. Class diagram shows how objects in a system interact with each other. Figure 3 shows an example class diagram from the package "Generic Organisation Model" (described in the Common Concepts part).

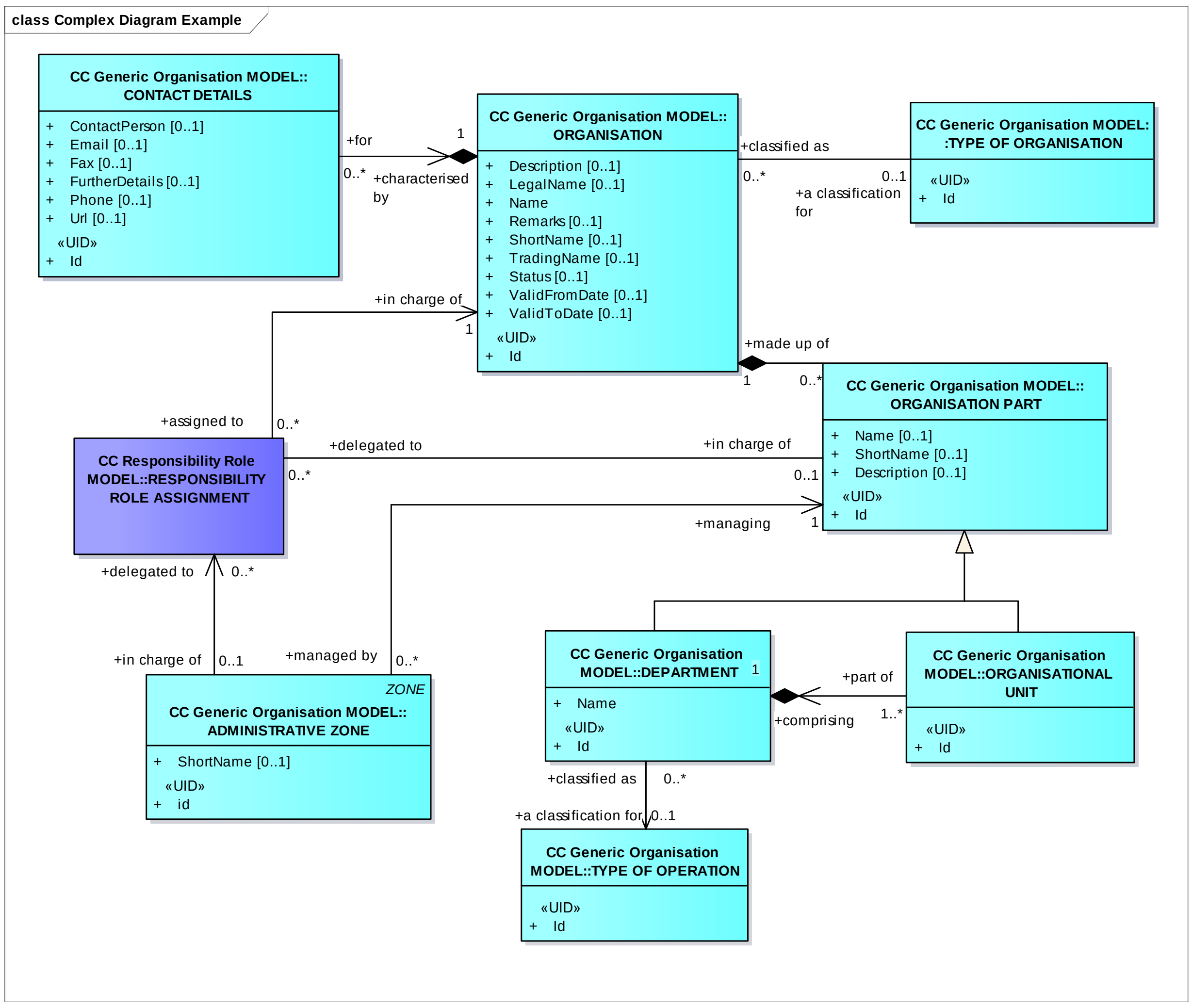


Figure 3 - Complex Diagram Example - Generic Organisation model

Classes and attributes

Classes are represented by boxes that are divided into three parts: the top part contains name of the class, the middle part contains the class's attributes and the bottom part shows possible operations that are associated with the class. In Transmodel only the top and middle parts are used for class name and attributes, respectively.

Figure 4 shows a class diagram containing a single class ORGANISATION with its attributes.

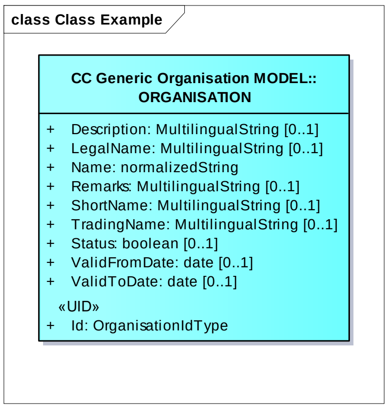

class Class Example
CC Generic Organisation MODEL:: ORGANISATION
+ Description: MultilingualString [0..1]
+ LegalName: MultilingualString [0..1]
+ Name: normalizedString
+ Remarks: MultilingualString [0..1]
+ ShortName: MultilingualString [0..1]
+ TradingName: MultilingualString [0..1]
+ Status: boolean [0..1]
+ ValidFromDate: date [0..1]
+ ValidToDate: date [0..1]
«UID»
+ Id: OrganisationIdType

Figure 4 - Class Example - ORGANISATION

Table 1 describes some of the elements from the class "ORGANISATION":

Table 1 : Elements in the class ORGANISATION

Notation	Semantics
CC Generic Organisation Model	Name of the package "Generic Organisation Model", described in the Common Concepts (CC) part.
ORGANISATION	Name of the class "ORGANISATION" defined in the package "Generic Organisation Model".
Description: MultilingualString [0..1]	Attribute "Description" of type "MultilingualString" is optional (mutiplicity: 0 or 1) for the class "ORGANISATION"
Name: normalizedString	Attribute "Name" is mandatory
‹‹UID››	Stereotype indicating that a particular attribute (in general named id) is a unique identifier for this class.
+	Scope of the attribute is "Public" : in general all attributes introduced are public
~	Scope of the attribute is "Package"

The attributes are indicated by at least their name. The full syntax is:

[Visibility] [Name [:Type] [Multiplicity]

Visibility (scope) is indicated by a

- '+' if visibility is public
- '~' if visibility is limited to its package

Each class in Transmodel contains a UID (Unique Identifier) named “id”. The id guarantees uniqueness for instances of the class.

Visibility of attribute types (example: MultilingualString[0..1]) is subject to the layout of the diagram. However, attribute types are always described in the class documentation.

The multiplicity indicates whether the attribute is

- Optional: marked as [0..1] or
- Mandatory: marked as [1] (or omitted).

Figure 5 shows a class diagram with three classes. The two (internal) classes LOCATION and LOCATING SYSTEM are defined in the package “Location Model”, while the (external) class POINT is defined in another package called “Generic Point & Link Model”.

For internal classes the package name is not mentioned in front of the class names.

The class POINT is inserted as a link from another (external) package named “Generic Point & Link Model”.

For the classes defined in external packages, the package name appears as a stereotype in front of the class name (e.g. Generic Point & Link Model :: POINT). Attributes of external classes are hidden.

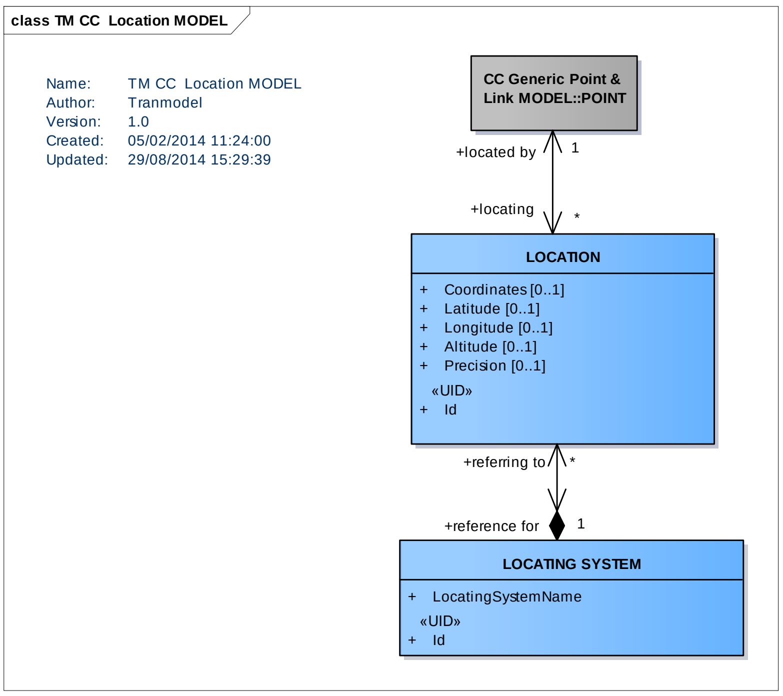


Figure 5 - Simple Diagram Example

Table 2 describes some elements of the class diagram:

Table 2 : Elements in a class diagram

Notation	Semantics
Location Model::LOCATION	Internal class "LOCATION" defined in the package "Location Model"
Generic Point & Link Model:: POINT	Class "POINT" linked from the external package "Generic Point & Link Model"
located by	Role name "located by" for the class POINT, which means: "each POINT is located by"
1	Multiplicity of the class POINT
locating	Role name "locating" for the class LOCATION, which means "each LOCATION is locating"
*	Multiplicity of the class LOCATION

The associations on the diagram present the following relationships between the classes LOCATION, POINT and LOCATING SYSTEM:

- A LOCATION is locating one and only one POINT
- A POINT may be located by many LOCATIONs
- A LOCATION is referring to one and only one LOCATING SYSTEM.

This means in particular that each POINT may be located through different types of LOCATIONs depending on the LOCATING SYSTEM.

In a class diagram multiple classes can be in specific relation to each other. Different notations are used for different types of relationships. In the following subsections relationship types relevant for Transmodel are explained.

Association relationships

Association is the general relationship type between classes represented by a solid line connector. The connector may include role names at each end, cardinality (multiplicity), direction (arrowheads) and constraints. A relationship can be named to describe the nature of the relationship between the two classes.

Figure 5 shows a class diagram with two associations; one general association relationship and one composite association relationship. Each side of the relationship connector has a role name and a multiplicity (cardinality) number.

Reflexive association relationship

A reflexive (also called recursive) relationship is represented by a solid line connector that connects a single class to itself.

Figure 6 shows a class with reflexive relationship named "is adjacent to". A topographic place in PT network may have zero or many adjacent topographic places, which in turn may be adjacent to other topograhic places as well.

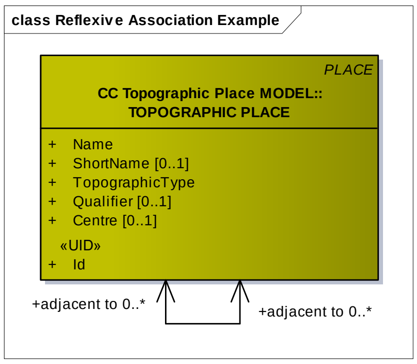


Figure 6 - Reflexive Association Example

Composition association relationship

A composition relationship is a strong form of association represented by a solid line with a filled (black) diamond at the relationship end, which is connected to the composite class. In a composition relationship component class depends on the composite class. If a composite object is removed, the component object is also removed.

Figure 5 shows a composition relationship between the classes LOCATION and LOCATING SYSTEM, which means:

- A LOCATING SYSTEM is a reference for zero or more (*) LOCATIONs
- A LOCATION must be referring to one and only one (1) LOCATING SYSTEM.

Aggregation association relationship

An aggregation relationship is a weak form of association represented by a solid line with a white diamond at the relationship end, which is connected to the aggregate class. In an aggregation relationship an aggregate class represents an assembly of component classes. If one aggregate object is removed, the component object may still exist.

Figure 7 shows an aggregate relationship between two classes, which means:

- A TIME BAND may be (optional relationship) in one GROUP OF TIME BANDS or A TIME BAND is in “0 or 1” GROUP OF TIME BANDS
- A GROUP OF TIMEBANDS is made up of “0 to n” TIME BANDs.

This means in particular that a GROUP OF TIMEBANDs may still exist even if a TIME BAND is suppressed.

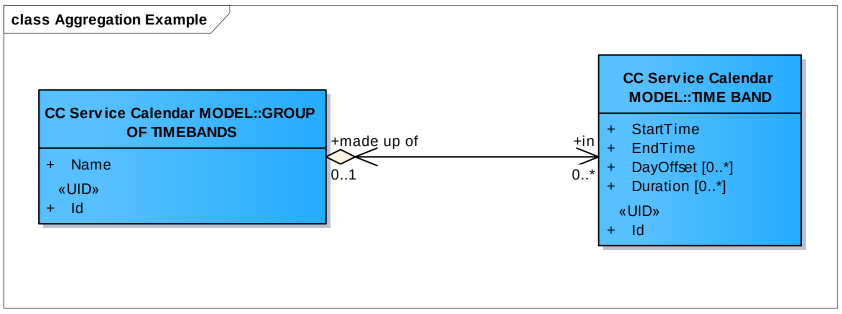

class Aggregation Example
CC Service Calendar MODEL::GROUP OF TIMEBANDS
+ Name
«UID»
+ Id
+made up of
0..1
+in
0..*
CC Service Calendar MODEL::TIME BAND
+ StartTime
+ EndTime
+ DayOffset [0..*]
+ Duration [0..*]
«UID»
+ Id

Figure 7 - Aggregation Example

Generalisation association relationship

A generalisation relationship indicates inheritance and is represented by a solid line with a white arrowhead at the relationship end. In the generalisation relationship a child class is based on a parent class. The child class captures and inherits attributes and relationships in the parent class. Child classes define only the attributes and relationships that are distinct from the parent class. Generalisation relationships do not have names.

Figure 8 shows generalisation relationship where "AUTHORITY and OPERATOR inherit from ORGANISATION".

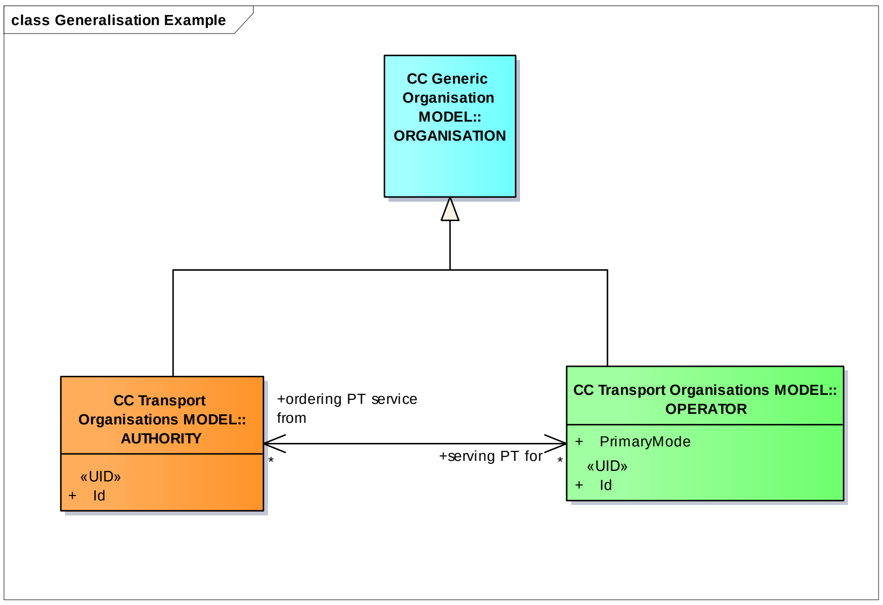

class Generalisation Example
CC Generic Organisation MODEL:: ORGANISATION
CC Transport Organisations MODEL:: AUTHORITY
«UID»
+ Id
+ordering PT service from
+serving PT for
*
*
CC Transport Organisations MODEL:: OPERATOR
+ PrimaryMode
«UID»
+ Id

Figure 8 - Generalisation Example

The "parent class ORGANISATION" may also appear on the diagram in the upper right corner of the corresponding class(es):

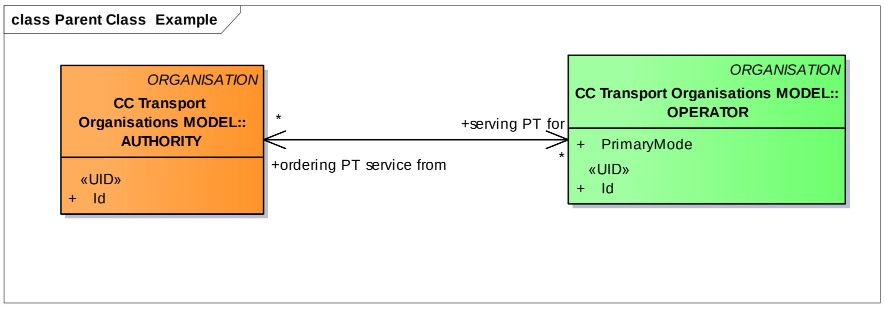

class Parent Class Example
ORGANISATION
CC Transport Organisations MODEL:: AUTHORITY
«UID»
+ Id
*
+serving PT for
*
+ordering PT service from
ORGANISATION
CC Transport Organisations MODEL:: OPERATOR
+ PrimaryMode
«UID»
+ Id

Figure 9 - Parent Class Example

Summary of Rules for Transmodel Representation

Rules for use of classes are shown in Table 3 :

Table 3 : Rules for the use of classes

Rule	Description
R1.1	Class names in class diagrams are written in UPPER CASE LETTERS.
R1.2	External class in a class diagram is named with its home package followed by double colon and its class name. Pattern is HOME-PACKAGE::CLASS-NAME.
R1.3	External class in a class diagram does not show its attributes.

Rules for use of **role names** in relationships are shown in Table 4 :

Table 4 : rules for the use of role names in relationships

Rule	Description
R2.1	Role name and multiplicity (cardinality) number belonging to a class are displayed on side of the class.
R2.2	Role names may be verbs in the present continuous/progressive tense form. Examples are: “containing”, “locating”, “including”, “composing”, “referring to”, etc.
R2.3	Role names may be verbs in the passive tense form. Examples are: “contained in”, “located by”, “included in”, “composed of”, “referenced in”, etc.
R2.4	Pair of role names of the two connected classes must be mutual in meaning. Examples are: “containing/contained in”, “locating/located by”, “including/included in”, “composing/composed of”, “referring to/referenced in” etc.
R2.5	If a relationship between classes is named then role names are not necessary.
R2.6	If role names are used then a relationship name is not necessary.

Rules for use of **multiplicity** (cardinality) in relationships are shown in Table 5 :

Table 5 : rules for the use of multiplicity / cardinality in relationships

Rule	Multiplicity	Description
R3.1	1 or 1..1	“exactly one”
R3.2	* or 0..*	“zero or more”, “none to many”
R3.3	0..1	“zero or one”
R3.4	1..*	“at least one”, “one or many”
R3.5	n..m	“at least n, but no more than m”