INTERNATIONAL STANDARD

ISO 19117

Second edition
2012-12-15

Geographic information — Portrayal

Information géographique — Présentation

Reference number
ISO 19117:2012(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 19117 was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*.

This second edition cancels and replaces the first edition (ISO 19117:2005), which has been technically revised.

Introduction

This International Standard specifies a conceptual schema for portrayal data, in particular symbols and portrayal functions. Portrayal functions associate features with symbols for the portrayal of the features on maps and other display media. This schema includes classes, attributes, associations and operations that provide a common conceptual framework that specifies the structure of and interrelationships between features, portrayal functions, and symbols. It separates the content of the data from the portrayal of that data, to allow the data to be portrayed in a manner independent of the dataset. This framework is derived from concepts found in existing portrayal implementations, and specifies a conceptual standard for use in future implementations (for example OGC Symbology Encoding and Styled Layer Descriptor Profile of WMS).

This International Standard provides an abstract model for developers of portrayal systems so that they can implement a system with the flexibility to portray geographic data to a user community in a manner that makes sense to that community.

The principal changes in this revision are to expand the concept of portrayal rules to more generic portrayal functions, include definitions for symbols (including parameterized symbols), include both portrayal functions and symbols in portrayal catalogues, and define a core portrayal schema, and extensions for specialized cases.

This revision for the most part expands on the concepts in ISO 19117:2005, but concepts for portrayal specifications (as a symbol instead of an operation), portrayal catalogue (also includes symbols), and rules-based portrayal (multiple rules allowed) have been changed.

Geographic information — Portrayal

1 Scope

This International Standard specifies a conceptual schema for describing symbols, portrayal functions that map geospatial features to symbols, and the collection of symbols and portrayal functions into portrayal catalogues. This conceptual schema can be used in the design of portrayal systems. It allows feature data to be separate from portrayal data, permitting data to be portrayed in a dataset independent manner.

This International Standard is not applicable to the following:

— standard symbol collection (e.g. International Chart 1 – IHO);

— a standard for symbol graphics (e.g. scalable vector graphics [SVG]);

— portrayal services (e.g. web map service);

— capability for non-visual portrayal (e.g. aural symbology);

— dynamic rendering (e.g. on the fly contouring of tides);

— portrayal finishing rules (e.g. generalization, resolve overprinting, displacement rules);

— 3D symbolization (e.g. simulation modeling).

2 Conformance

Any portrayal catalogue, portrayal function and symbol describing the portrayal of geographic information claiming conformance with this International Standard shall pass the relevant tests of the abstract test suite presented in Annex A, and those portrayal extension requirements that are applicable to the extension or extensions being used.

Conformance classes are defined for the portrayal core, and the core plus extensions. These extensions provide additional functionality, and are not mutually exclusive of each other.

Core portrayal conformance classes
Conformance class – portrayal core (general)
Conformance class – portrayal core – symbol
Conformance class – portrayal core – portrayal function
Conformance class – portrayal core – portrayal catalogue

Portrayal function extension conformance classes
Conformance class – portrayal core plus conditional function extension
Conformance class – portrayal core plus context extension

Conformance class – portrayal core plus function symbol parameter extension

Symbol extension conformance classes
Conformance class – portrayal core plus compound symbol extension
Conformance class – portrayal core plus complex symbol extension
Conformance class – portrayal core plus reusable symbol component extension
Conformance class – portrayal core plus symbol parameter extension

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/TS 19103:2005, *Geographic information — Conceptual schema language*

ISO 19107:2003, *Geographic information — Spatial schema*

ISO 19109:2005, *Geographic information — Rules for application schema*

ISO 19110:2005, *Geographic information — Methodology for feature cataloguing*

ISO 19111:2007, *Geographic information — Spatial referencing by coordinates*

ISO 19115:2003, *Geographic information — Metadata*

ISO/TS 19139:2007, *Geographic information — Metadata — XML schema implementation*

ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
annotation
any marking on illustrative material for the purpose of clarification

Note 1 to entry: Numbers, letters, **symbols** (4.31), and signs are examples of annotation.

4.2
class
description of a set of objects that share the same attributes, operations, methods, relationships and semantics

Note 1 to entry: A class may use a set of interfaces to specify collections of operations it provides to its environment. See: interface.

[SOURCE: ISO/TS 19103:2005, definition 4.27]

4.3
complex symbol
symbol (4.31) composed of other symbols of different types

EXAMPLE A dashed line symbol with a **point** (4.19) symbol repeated at an interval.

4.4
compound symbol
symbol (4.31) composed of other symbols of the same type

EXAMPLE A **point** (4.19) symbol that is composed of two point graphics.

4.5
conditional feature portrayal function
function (4.11) that maps a geographic **feature** (4.8) to a **symbol** (4.31) based on some condition evaluated against a property or attribute of a feature

4.6
curve
1-dimensional **geometric primitive** (4.13), representing the continuous image of a line

[SOURCE: ISO 19107:2003, definition 4.23]

4.7
dataset
identifiable collection of data

Note 1 to entry: A dataset may be a smaller grouping of data which, though limited by some constraint such as spatial extent or **feature** (4.8) type, is located physically within a larger dataset. Theoretically, a dataset may be as small as a single feature or **feature attribute** (4.9) contained within a larger dataset. A hardcopy map or chart may be considered a dataset.

[SOURCE: ISO 19115:2003, definition 4.2]

4.8
feature
abstraction of real world phenomena

Note 1 to entry: A feature may occur as a type or an **instance** (4.14). Feature type or feature instance shall be used when only one is meant.

[SOURCE: ISO 19101:2002, definition 4.11]

4.9
feature attribute
characteristic of a **feature** (4.8)

EXAMPLE 1 A feature attribute named “colour” may have an attribute value “green” which belongs to the data type “text”.

EXAMPLE 2 A feature attribute named “length” may have an attribute value “82.4” which belongs to the data type “real”.

Note 1 to entry: A feature attribute has a name, a data type, and a value domain associated to it. A feature attribute for a feature **instance** (4.14) also has an attribute value taken from the value domain.

Note 2 to entry: In a feature catalogue, a feature attribute may include a value domain but does not specify attribute values for feature instances.

[SOURCE: ISO 19101:2002, definition 4.12]

4.10
feature portrayal function
function (4.11) that maps a geographic **feature** (4.8) to a **symbol** (4.31)

4.11
function
rule that associates each element from a domain (source, or domain of the function) to a unique element in another domain (target, co-domain, or range)

[SOURCE: ISO 19107:2003, definition 4.41]

4.12
geographic information
information concerning phenomena implicitly or explicitly associated with a location relative to the Earth

[SOURCE: ISO 19101:2002, definition 4.16]

4.13
geometric primitive
geometric object representing a single, connected, homogeneous element of space

[SOURCE: ISO 19107:2003, definition 4.48]

4.14
instance
object that realizes a **class** (4.2)

[SOURCE: ISO 19107:2003, definition 4.53]

4.15
layer
basic unit of **geographic information** (4.12) that may be requested as a map from a server

[SOURCE: ISO 19128:2005, definition 4.6]

4.16
metadata
data about data

[SOURCE: ISO 19115:2003, definition 4.5]

4.17
parameterized feature portrayal function
function (4.11) that maps a geographic **feature** (4.8) to a **parameterized symbol** (4.18)

Note 1 to entry: A parameterized **feature portrayal function** (4.10) passes the relevant attribute values from the feature **instance** (4.14) for use as input to the parameterized **symbol** (4.31).

4.18
parameterized symbol
symbol (4.31) that has dynamic parameters

Note 1 to entry: The dynamic parameters map to the attribute values of each **feature** (4.8) **instance** (4.14) being portrayed.

4.19
point
0-dimensional **geometric primitive** (4.13), representing a position

[SOURCE: ISO 19107:2003, definition 4.61]

4.20
portrayal
presentation of information to humans

Note 1 to entry: Within the scope of this International Standard, portrayal is restricted to the portrayal of geographic information.

4.21
portrayal catalogue
collection of defined **portrayals** (4.20) for a **feature** (4.8) catalogue

Note 1 to entry: Content of a portrayal catalogue includes **portrayal functions** (4.23), **symbols** (4.31), and **portrayal context** (4.22) (optional).

4.22
portrayal context
circumstances, imposed by factors extrinsic to a geographic **dataset** (4.7), that affect the **portrayal** (4.20) of that dataset

EXAMPLE Factors contributing to portrayal context can include the proposed display or map scale, the viewing conditions (day/night/dusk), and the display orientation requirements (north not necessarily at the top of the screen or page) among others.

Note 1 to entry: Portrayal context can influence the selection of **portrayal functions** (4.23) and construction of **symbols** (4.31).

4.23
portrayal function
function (4.11) that maps geographic **features** (4.8) to **symbols** (4.31)

Note 1 to entry: **Portrayal** (4.20) functions can also include parameters and other computations that are not dependent on geographic feature properties.

4.24
portrayal function set
function (4.11) that maps a **feature** (4.8) catalogue to a **symbol set** (4.35)

4.25
portrayal rule
specific type of **portrayal function** (4.23) expressed in a declarative language

Note 1 to entry: A declarative language is rule-based and includes decision and branching statements.

4.26
portrayal service
generic interface used to portray **features** (4.8)

4.27
render
conversion of digital graphics data into visual form

EXAMPLE Generation of an image on a video display.

4.28
simple symbol
symbol (4.31) that is neither compound nor parameterized

4.29
spatial attribute
feature attribute (4.9) describing the spatial representation of the **feature** (4.8) by coordinates, mathematical **functions** (4.11) and/or boundary topology relationships

4.30
surface
2-dimensional **geometric primitive** (4.13), locally representing a continuous image of a region of a plane

[SOURCE: ISO 19107:2003, definition 4.75]

4.31
symbol
portrayal (4.20) primitive that can be graphic, audible, or tactile in nature, or a combination of these

4.32
symbol component
symbol (4.31) that is used as a piece of a **compound symbol** (4.4)

4.33
symbol definition
technical description of a **symbol** (4.31)

4.34
symbol reference
pointer in a **feature portrayal function** (4.10) that associates the feature type with a specific **symbol** (4.31)

4.35
symbol set
collection of **symbols** (4.31)

Note 1 to entry: Symbol sets are usually designed for a community of interest to portray information of interest to the community.

5 Abbreviated terms

CRS Coordinate Reference System

URL Uniform Resource Locator

UML Unified Modeling Language (ISO 19501)

6 Portrayal mechanism

6.1 Introduction

This International Standard is organized as a core portrayal model and a series of extensions.

The core portrayal model uses portrayal functions to map geospatial features to symbols. A portrayal function set maps a feature catalogue to a symbol set. A Feature Portrayal Function maps a geospatial feature to a symbol. A Portrayal Catalogue that can be used to transmit symbols and portrayal functions is also a part of the portrayal core.

There are two extensions to the portrayal function. The Conditional Function Extension extends the basic portrayal function to enable conditions to be applied in the function. Conditions can test for feature attributes, geometry, and other properties of the feature. The Context Extension extends the basic portrayal function to

enable contextual information such as display scale, viewing conditions, and other factors external to the application schema of the geospatial dataset to be utilized in portrayal functions.

The core symbol model provides for the definition of a basic symbol, which includes building up a symbol made from multiple components. There are three extensions of the symbol model, the Compound Symbol Extension (9.4), the Complex Symbol Extension (9.5), and the Reusable Symbol Component Extension (9.6) that allow a symbol to be stored at an external URL.

Finally, symbols can be parameterized by use of a Symbol Parameter Extension, and a Function Symbol Parameter Extension. Whereas conditional extensions allow a portrayal function to point to a specific symbol based on an attribute condition, the parameterized symbol uses feature attribute information as input to a symbol definition. The Function Symbol Parameter Extension allows that feature attribute information to be passed to the symbol by the Feature Portrayal Function.

This International Standard defines a feature-centred function-based portrayal mechanism. Instances of features are portrayed based on portrayal functions, which make use of geometry and attribute information. The relationship between the feature instances, attributes and the underlying spatial geometry is specified in an application schema according to ISO 19109. Spatial geometry and associated topological relationships are defined in ISO 19107.

Portrayal information is needed to portray a dataset containing geographic data. The portrayal information is handled as symbol references selected according to specific portrayal functions. The portrayal mechanism makes it possible to portray the same dataset in different ways without altering the dataset itself.

The portrayal mechanism is illustrated by Figure 1.

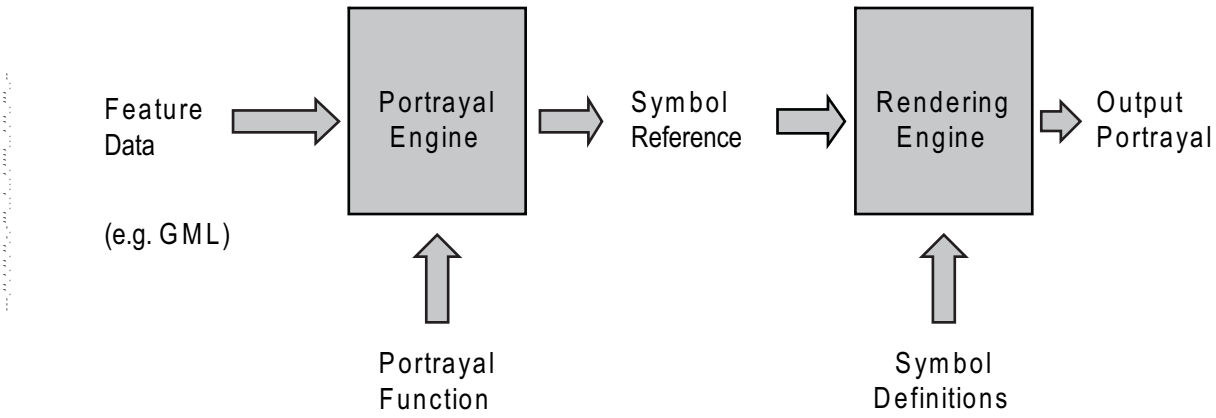

Figure 1 — Portrayal mechanism

The symbol definitions and portrayal function shall not be part of the dataset. The portrayal functions and symbol shall be able to be transferred in a portrayal catalogue. The symbols shall be referenced from portrayal functions. The feature portrayal functions shall be specified for the feature class or feature instances they will be applied on. The symbol definitions may be stored externally and referenced using a universal reference standard such as a network based URL. Portrayal information may be specified either by sending a portrayal catalogue with the dataset, or by referencing an existing portrayal catalogue from metadata.

In addition, the user may want to apply a user defined portrayal function and symbol definition. The model in Figure 2 shows how the portrayal catalogue is referenced by the dataset metadata. Only the metadata reference is shown and not the contents of the portrayal catalogue (see ISO 19115).

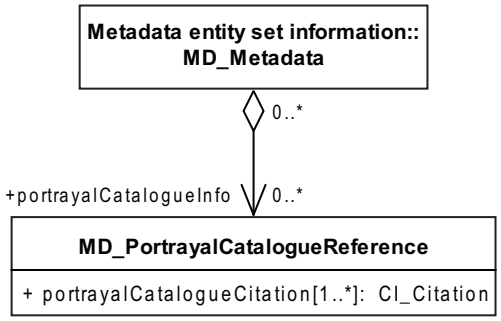

Figure 2 — UML model of the portrayal part of ISO 19115

6.2 Portrayal functions

A function is a rule that associates each element from a domain (source, or domain of the function) to a unique element in another domain (target, co-domain, or range) (ISO 19107]. In the portrayal of geographic information, a portrayal function can be considered as the assignment of a symbol instance to each geographic feature instance in a geographic dataset.

A function from a set *A* into a set *B* is defined as a rule that assigns to each element $a \in A$ a unique element $b \in B$ [4]. The set *A* is called the domain of the function while the set *B* is called the codomain. The subset of the codomain *B* that is assigned to from the domain *A* by the function *f* is called the range of *f* (see Figure 3).

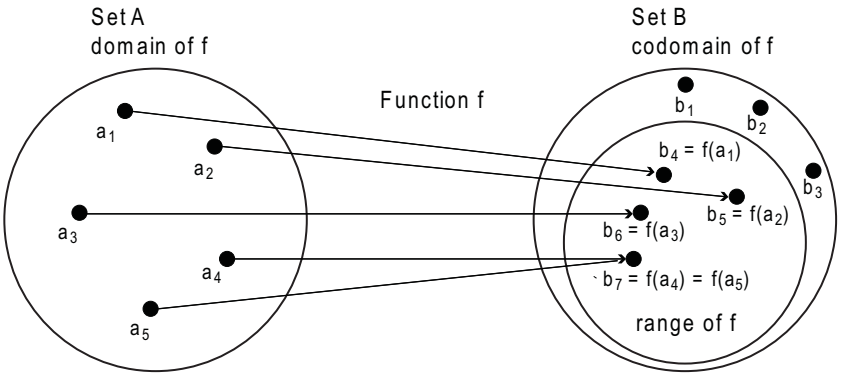

Figure 3 — Mapping from set A to set B

A geographic feature in a dataset is a member in the set of the domain. The geographic dataset is the domain. A symbol is a member in the set of the codomain. Those symbols that are actually assigned to a feature by the portrayal function are the range of the domain. Similar to features, symbols can be templates (corresponding to feature types) or instances. A symbol is a template defining, for example, the symbol used to represent a bridge, and a symbol is an instance defining, for example, the symbol that represents the Chesapeake Bay Bridge on a map.

The portrayal function is illustrated in the following equations. If domain G = a set of geographic features and codomain S = a set of symbols, then the function

$$\Phi : G \rightarrow S$$

is the **portrayal function** that maps geographic features to symbols.

If k is a feature type, and there is a function

$$t : G \rightarrow G_k$$

that maps geographic feature instances to geographic feature types, then the function

$$\Phi_k : G_k \rightarrow S$$

is a feature type dependent portrayal function or a **feature portrayal function**.

If i is a specific symbol definition, then the function

$$\Phi_k{}^i : G_k \rightarrow S_i$$

is the feature portrayal function for a symbol definition. A **portrayal function** set is a set of feature portrayal functions, over all feature types in a dataset.

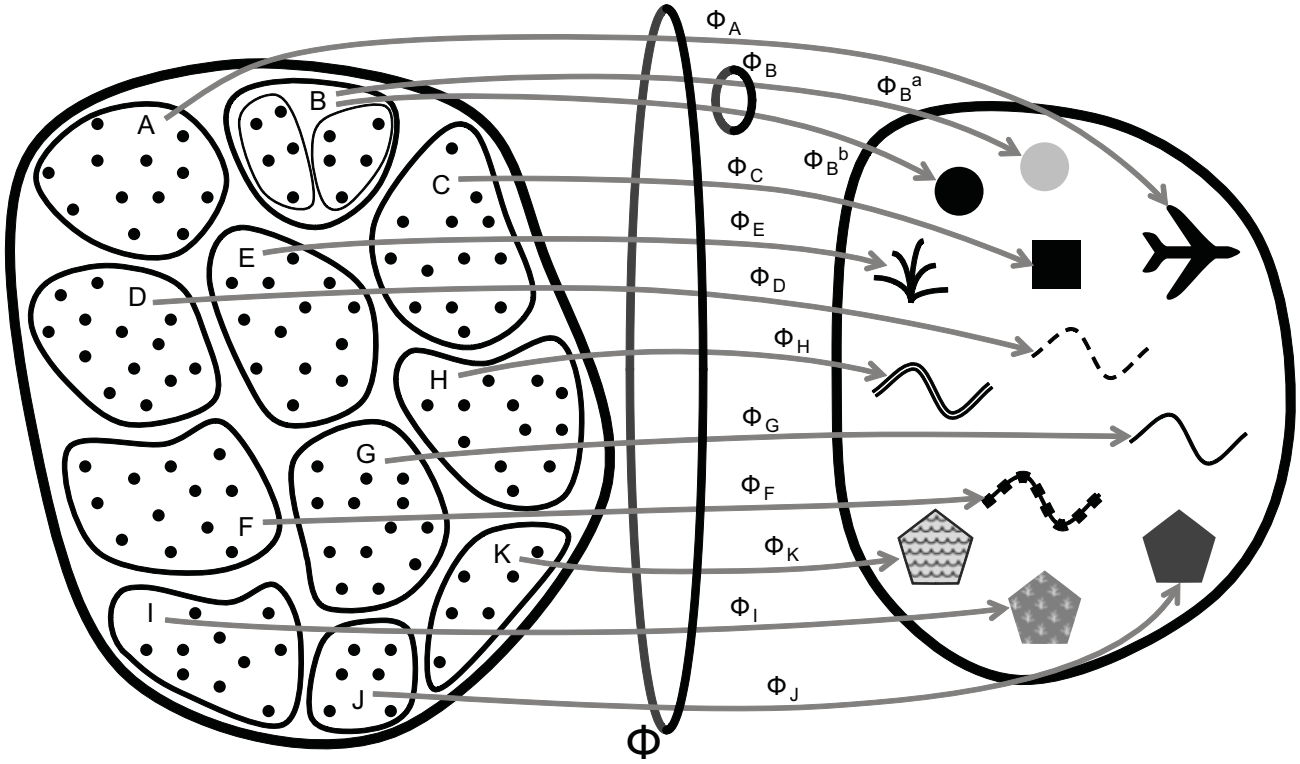

Figure 4 — Portrayal functions

Figure 4 illustrates the mapping of the portrayal function set Φ and its component feature portrayal functions Φ_A – Φ_K. The set on the left represents the domain of the function, collecting feature instances (black dots) of a dataset. This set is divided into subsets by feature type, labelled A – K. The set on the right is the range of the function and is the collection of symbols to which the features on the left are mapped. Each feature portrayal function maps the instances of a feature type to a single symbol except for ΦB which maps the instances of type B conditionally to one of two symbols.

Geographic features have properties, and so do symbols. Portrayal functions can also map feature properties to symbol properties.

Portrayal functions may also consider the context or parameters and other computations that are not dependent on the feature properties, or are external to the geospatial dataset's application schema in association of symbols to geographic features. This is particularly important where the portrayal context may determine symbolization. Information such as viewing conditions, medium, and rendering scale may influence symbolization and are thus part of the context. The context may determine which portrayal function to apply but may also be used as input to the portrayal function to achieve the same result. In the former case, a condition attached to the function is used to test the function's applicability to a context. In the latter case,

multiple, conceptually separate functions are combined into one using the context to choose symbolization in the function. These two approaches may also be combined, using the context to select a portrayal function and then using the context in the function to choose symbolization.

The portrayal function is analogous to a schema mapping from the application schema of the geospatial dataset to the application schema defined by the collection of symbol definitions that are available to portray the dataset. Application schema mapping is broader in applicability than just geospatial information portrayal, and therefore will not be addressed as a requirement in this International Standard.

A portrayal rule is a specific type of portrayal function that is expressed in some declarative language (rule language with an if/else, switch/case, etc.). Portrayal rules (a rules-based portrayal function) were described in a general sense in the previous version of this International Standard (ISO 19117:2005). An elementary portrayal rules-based portrayal function is defined in Annex B of this International Standard to provide users with a basic schema mapping method if they choose to use it, but since portrayal rules are not the only way to define a portrayal function, it is not a mandatory part of this International Standard.

6.3 Portray nothing

For a feature instance that is not to be portrayed, a feature portrayal function shall map to an empty symbol, i.e. a symbol with no symbol components.

6.4 Default portrayal

The default symbol shall be applied according to at least one of the spatial attributes of the feature instance, and shall only be applied when no feature portrayal function is applicable for a feature instance.

A default symbol reference shall be present to ensure that no feature instance is left unportrayed by mistake. The provider of the portrayal function set specifies the value of the default symbol reference but shall not portray nothing (6.3). Contextual information shall not be used in the default symbol definition. If the application fails to portray the data for some reason, the failure shall be handled by the application.

6.5 Annotation

The information that is to be portrayed shall be defined in an application schema. Typically there are two types of information included in a dataset: geographic information and annotation. Annotation includes text, grids, legends and special features such as a compass rose.

6.6 Overview of portrayal

Portrayal is illustrated by Figure 5. The diagram is not part of the portrayal schema and not for implementation. It is intended as an explanatory aid only.

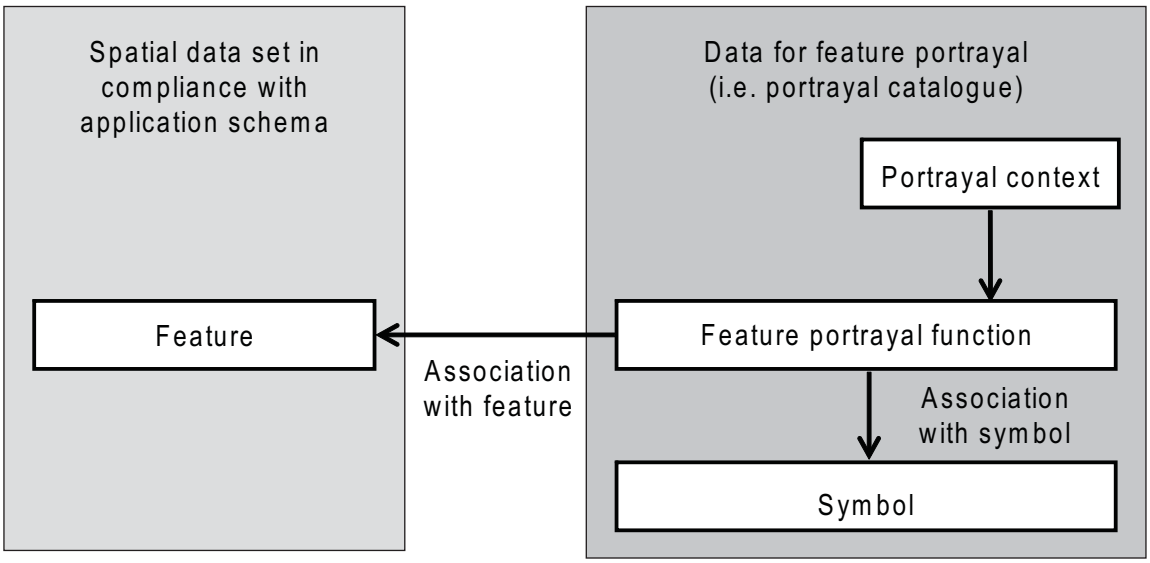

Figure 5 — Overview of portrayal

The portrayal catalogue consists of the feature portrayal functions, and symbols as shown in Figure 5. To produce different products, several portrayal catalogues may exist, portraying one or more datasets. Dataset is explained in ISO 19109. The portrayal catalogue relates to one or more symbols, and one symbol may be used in one or more portrayal catalogues.

7 Package — ISO 19117 Portrayal

7.1 Introduction

This International Standard presents a conceptual schema for describing portrayal functions, symbols, and symbol collections, all intended for use in the portrayal of geographic data. The conceptual schema is specified using the Unified Modeling Language (UML) (ISO 19501), following the guidance of ISO/TS 19103.

The schema is contained in a series of UML packages, defining the portrayal core, and extensions for conditional functions, context, compound symbols, complex symbols, reusable symbol components, parameterized symbols and the portrayal functions that use parameterized symbols. The Portrayal Catalogue package specializes the Feature Cataloguing package for portrayal. The Portrayal Function package defines a root for the mapping of features to symbols and several specializations. The Presentation package defines a root for presentation and several specializations. Classes in the packages of this schema are derived from classes included in this schema as well as classes included in the package Catalogues (ISO/TS 19139) carrying the prefix “CT_”. Classes in the packages of this schema reference and use as data types classes included in this schema as well as classes included in the packages Basic Types (ISO/TS 19103), Feature Cataloguing (ISO 19110) carrying the prefix “FC_”, Coordinate Reference Systems (ISO 19111) carrying the prefix “SC_”, Citation and responsible party information (ISO 19115) carrying the prefix “CI_”, Identification information (ISO 19115) carrying the prefix “MD_”, and Reference system information (ISO 19115) also carrying the prefix “MD_”.

Names of classes included in these packages carry the prefixes “PF_” for the portrayal function related classes and “SY_” for the symbol related classes.

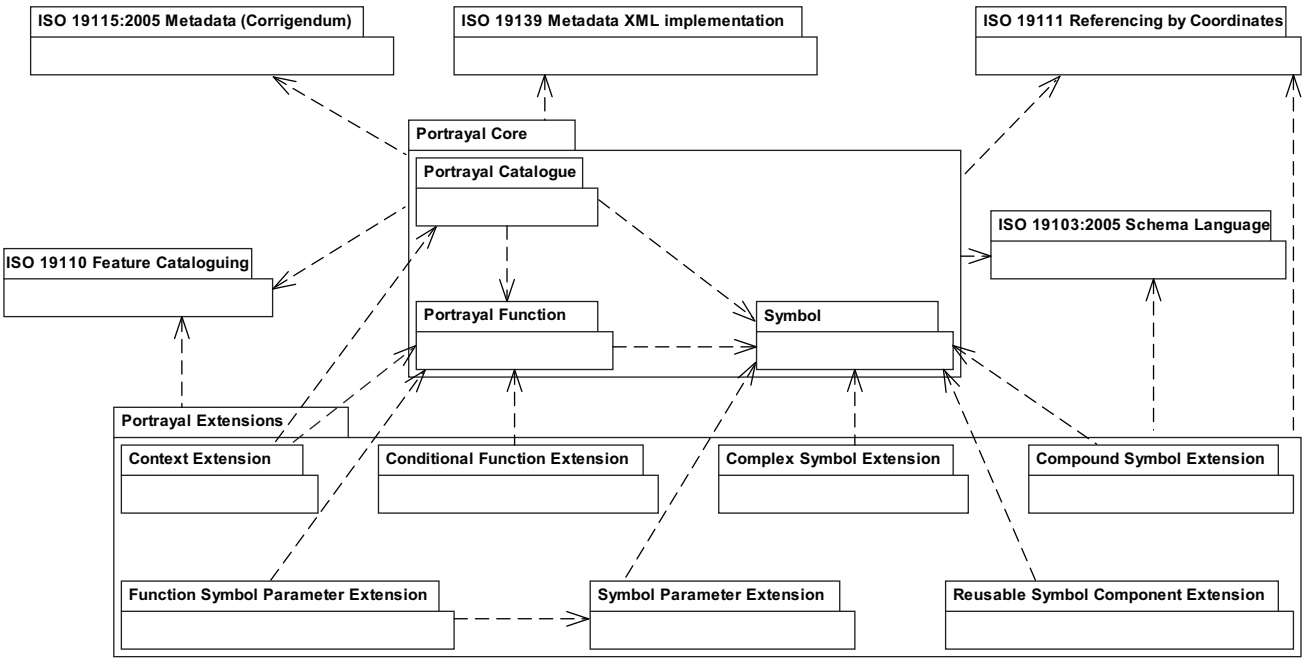

Figure 6 — Package structure with dependencies

7.2 Symbol structure

7.2.1 Introduction

Symbols can be almost infinite in variety and are defined in several different ways. Symbols can be composed of several graphic elements or geometries.

7.2.2 Simple symbols

Simple symbols are included in the portrayal core. Point symbols typically have a single graphic icon, located at one geographic point, with a symbol origin that defines the relationship between the icon and the geographic position of the feature that the icon represents. An example of a simple point symbol might be a black square that represents a building.

Figure 7 — Example symbol: black square representing a building

The point symbol can also be composed of text. The point text is based on a single geographic coordinate. An example might be a label indicating numerous lakes.

Numerous
lakes

Figure 8 — Example symbol: text label indicating numerous lakes

Line symbols are typically represented by a line that follows a geometric curve. An example is using a line to symbolize a river.

Figure 9 — Example symbol: solid line symbolizing a river

Line symbols can sometimes have point components associated with it, on one or both ends. An example of this kind of symbol might be a line symbol for a footbridge with wing ticks on both ends.

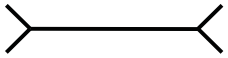

Figure 10 — Example symbol: solid line with wing ticks on both ends symbolizing a footbridge

Line symbols may also be composed of text. An example might be a name label for a region on a map that is spread out along a line.

Figure 11 — Example symbol: text flowing along curve labelling a region

Area symbols are typically symbolized by an area fill or colour that fills up the extent of an areal feature. An example might be a forest on a topographic map.

Figure 12 — Example symbol: area fill colour symbolizing a forest

Sometimes the area symbol has a boundary and a fill. An example might be a symbol for a lake, in which both the water body and shoreline are symbolized.

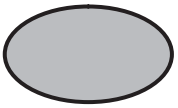

Figure 13 — Example symbol: area with boundary and fill symbolizing a water body and shoreline

Sometimes a point icon graphic is placed inside the area symbol to further illustrate the symbol. An area symbol may have area, line and point components, for example a symbol for a dangerous wreck.

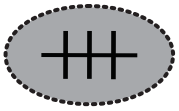

Figure 14 — Example symbol: area symbol with area, line and point components symbolizing a dangerous wreck

7.2.3 Compound symbols

To enable the reuse of graphic components and to build more complex symbols, some symbols may be composed of more than one graphic component of the same type. An example of a compound point symbol might be a school symbol, shown by a black square with a flag on top.

Figure 15 — Example symbol: point symbol composed of black square and flag symbolizing a school

A common example of a compound line symbol is a cased road, in which a narrow line is superimposed over a wider line of a different colour.

Figure 16 — Example symbol: cased line symbolizing a road

A compound area symbol uses complex symbols and is described later.

Each of these compound symbols may also show text components.

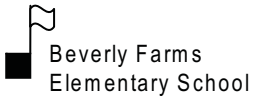

Figure 17 — Example symbol: compound point symbol with text component symbolizing a school

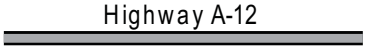

Figure 18 — Example symbol: compound line symbol with text component symbolizing a road

7.2.4 Complex symbols

Complex symbols are made up from various types of component symbols. These include line symbols with repeated point symbols, pattern filled area symbols, and cross-hatched area symbols.

An example of a line symbol with repeated points might be a boundary to an anchorage area, where the anchor point icons are repeated at intervals along a boundary line.

Figure 19 — Example symbol: complex line symbol with anchor point icons repeated along a line symbolizing a boundary to an anchorage area

An example of a pattern fill might be a swamp symbol, in which the grass pattern is tiled and repeated over the area of the swamp.

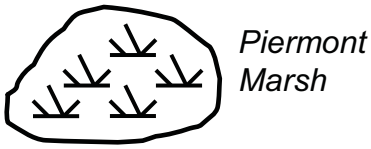

Figure 20 — Example symbol: complex area symbol with grass pattern is tile symbolizing a swamp

An example of a cross-hatch area fill might be a symbol for a filtration bed.

Figure 21 — Example symbol: complex area symbol with line symbol cross-hatching symbolizing a filtration bed

As with compound symbols, complex symbols may also have a text component.

7.2.5 Symbol composition

Symbols are initially without meaning and without location. When they become part of a portrayal they gain both meaning and location both with respect to the portrayal and with respect to the universe they represent. The definition of a symbol has its own engineering coordinate reference system. A coordinate reference system transformation places a point symbol into a portrayal (Figure 22). Similarly, a coordinate reference system transformation places a subsymbol into a parent point symbol.

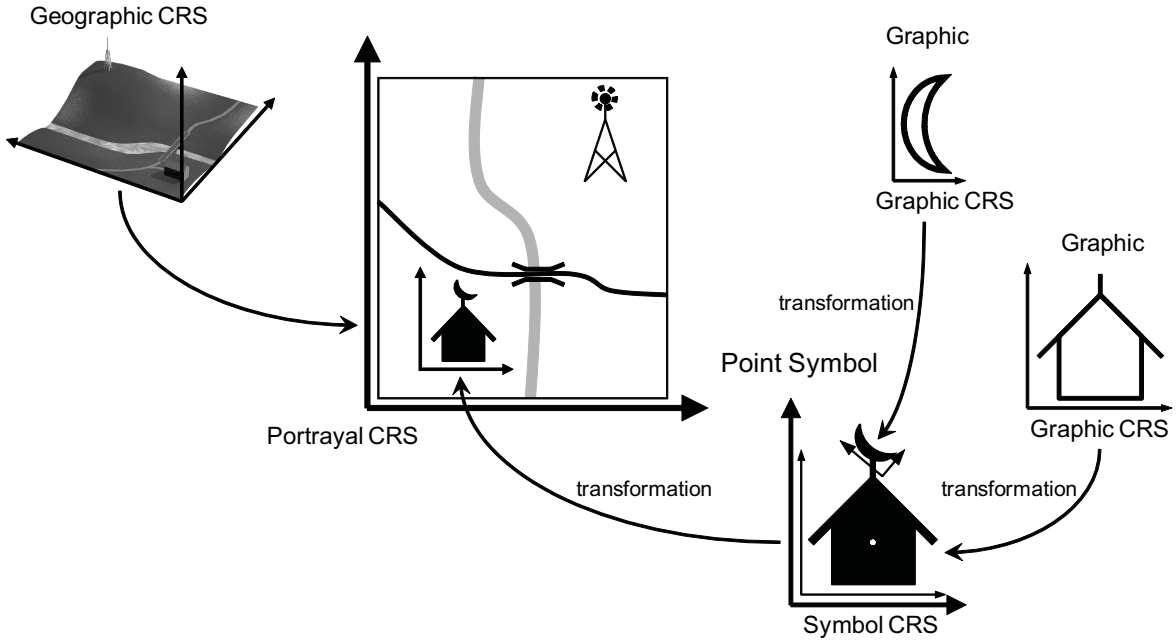

Figure 22 — Portrayal and Symbol CRSs

Line symbols and patterns have two kinds of coordinate reference systems. A line coordinate reference system is used to define the pen or line style. This coordinate reference system is perpendicular to the geometry of the curve and allows for the specification of line widths and offsets. The second kind of coordinate reference system is a local coordinate reference system which is defined for every location along a curve. This coordinate reference system has an x-axis that is tangential to the curve and a y-axis perpendicular to the x-axis (Figure 23).

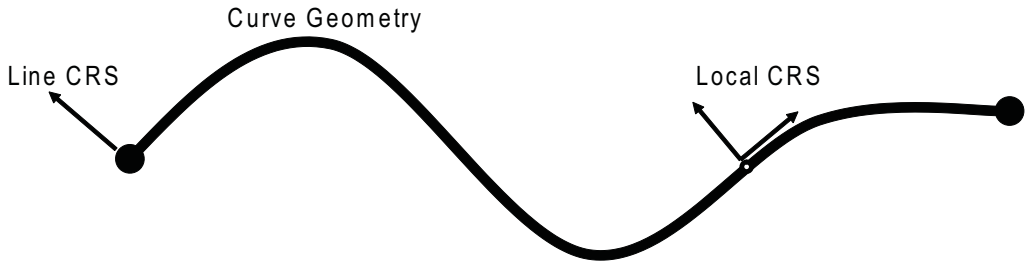

Figure 23 — Line and Local CRSs

An area symbol defines coordinate reference systems for its boundary and for its interior. The boundary coordinate reference systems are those defined for line symbols. The interior of the area symbol has its own coordinate reference systems (Figure 24).

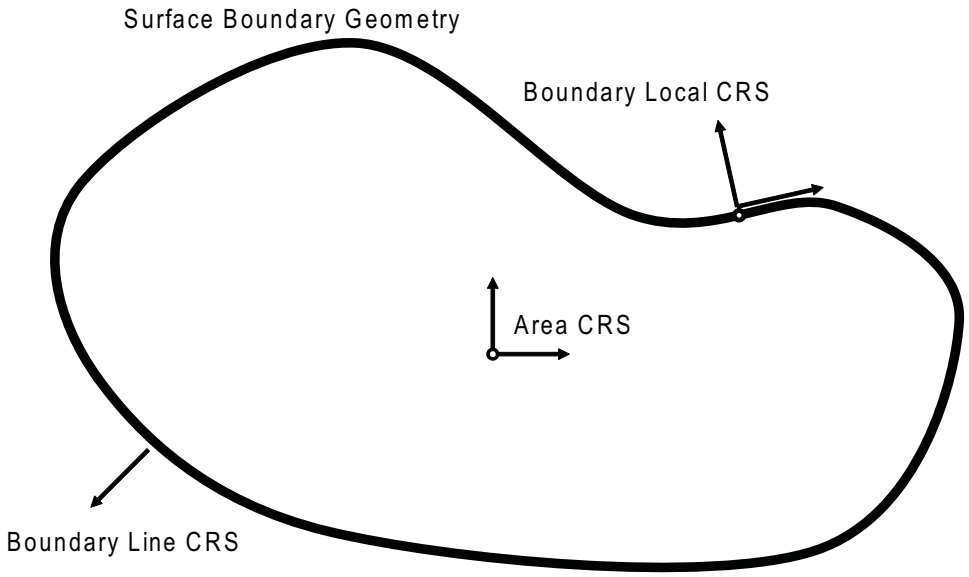

Figure 24 — Area and Boundary CRSs

Should the area symbol include a tiled pattern area fill then there is a tile coordinate reference system as well (Figure 25).

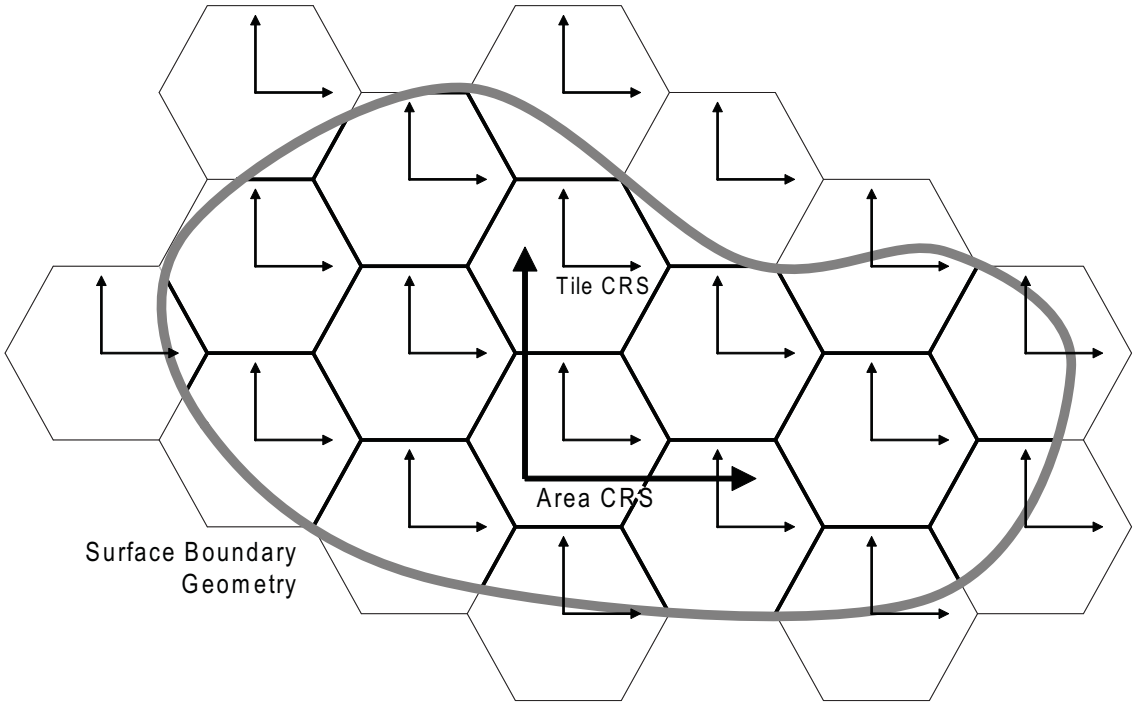

Figure 25 — Tile CRS

8 Package – Portrayal Core

8.1 Package semantics

The Portrayal Core package supplies the core facilities for defining functions for mapping geographic features to symbols, for defining symbols for portrayal, and for packaging functions and symbols in catalogues for interchange by transfer. The package is divided into three subpackages: the Portrayal Function package is used to define mapping functions, the Symbol package is used to define symbol, and the Portrayal Catalogue package is used to define portrayal catalogues.

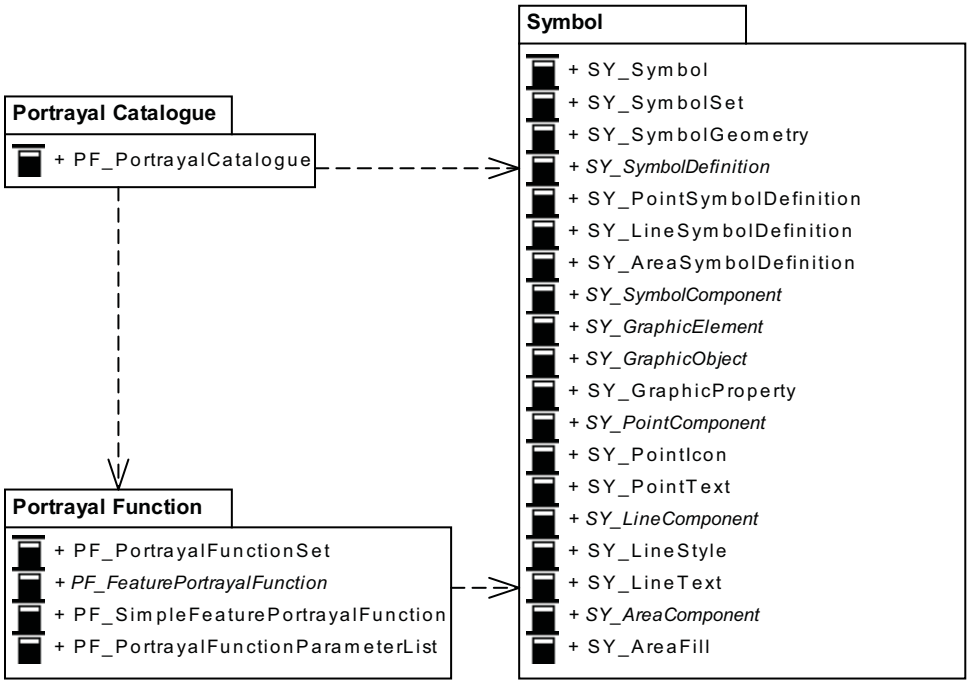

Figure 26 — Core package structure with dependencies

8.2 Package – Portrayal Function

8.2.1 Package semantics

The Portrayal Function package defines functions that map feature types to symbols.

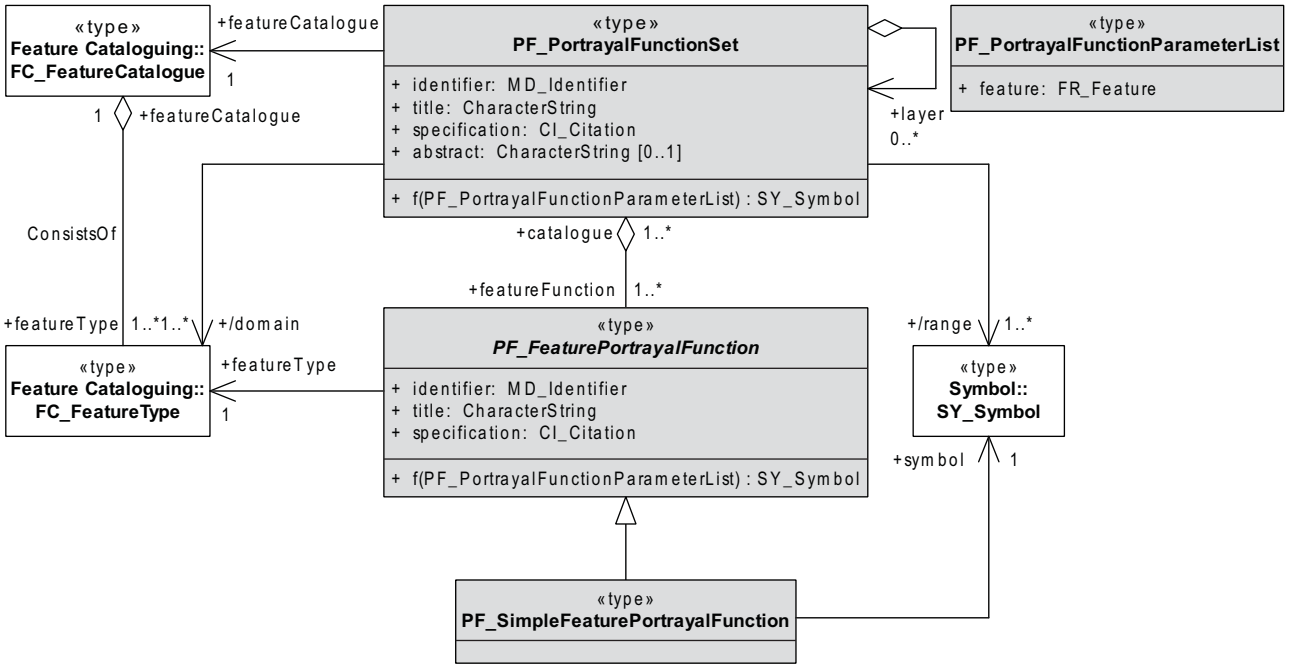

Figure 27 — Portrayal Function

8.2.2 Type – PF_PortrayalFunctionSet

8.2.2.1 Class semantics

PF_PortrayalFunctionSet describes a function, which maps the feature types of a feature catalogue to symbols. A *PF_PortrayalFunctionSet* collects feature portrayal functions, which in turn map an individual feature type to a symbol. *PF_PortrayalFunctionSet* represents the concept of the portrayal function set Φ as illustrated in Figure 4.

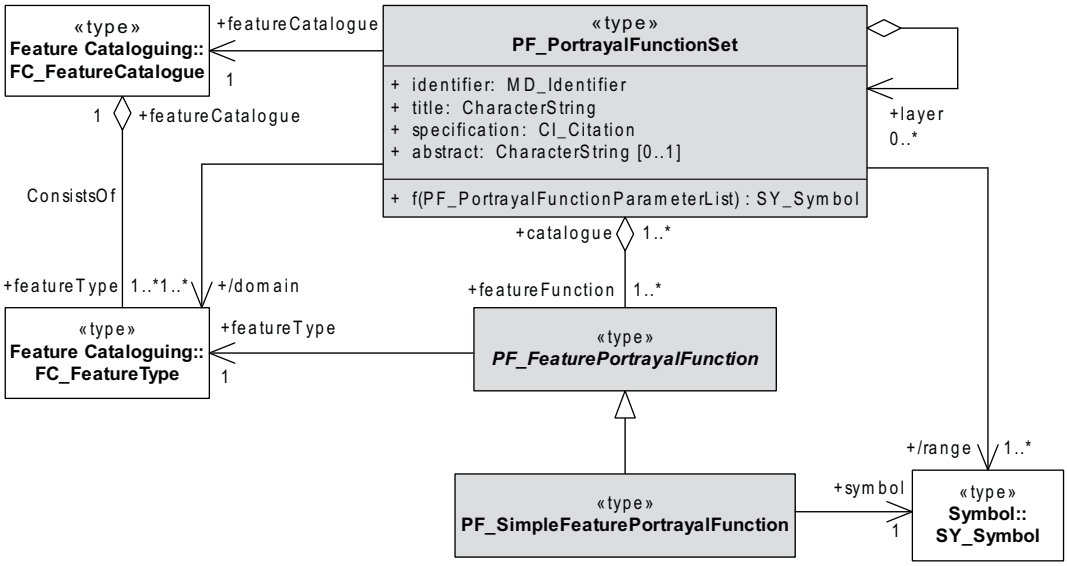

Figure 28 — PF_PortrayalFunctionSet context diagram

8.2.2.2 Association role – featureCatalogue

The association role *featureCatalogue* defines the feature catalogue containing the feature type domain of the feature to symbol mapping.

```
PF_PortrayalFunctionSet::featureCatalogue[1] : FC_FeatureCatalogue
```

8.2.2.3 Association role – domain

The association role *domain* defines the set of feature types that are mapped to symbols.

```
PF_PortrayalFunctionSet::domain[1..*] : FC_FeatureType
```

8.2.2.4 Association role – range

The association role *range* defines the set of symbols to which feature types are mapped.

```
PF_PortrayalFunctionSet::range[1..*] : SY_Symbol
```

8.2.2.5 Aggregation role – featureFunction

The aggregation role *featureFunction* collects the feature portrayal functions that make up the individual feature mappings of the portrayal function set.

```
PF_PortrayalFunctionSet::featureFunction[1..*] : PF_FeaturePortrayalFunction
```

8.2.2.6 Aggregation role – layer

The aggregation role *layer* associates a portrayal function set with a subordinate portrayal function set. The subordinate catalogue defines a layer in the overall portrayal.

```
PF_PortrayalFunctionSet::layer[0..*] : PF_PortrayalFunctionSet
```

8.2.2.7 Attribute – identifier

The attribute *identifier* is a machine readable name for the portrayal function set. The identifier shall be unique.

```
PF_PortrayalFunctionSet::identifier : MD_Identifier
```

8.2.2.8 Attribute – title

The attribute *title* is a multi-lingual human readable name for the portrayal function set.

```
PF_PortrayalFunctionSet::title : CharacterString
```

8.2.2.9 Attribute – specification

The attribute *specification* is a reference to the full details of the portrayal function set.

```
PF_PortrayalFunctionSet::specification : CI_Citation
```

8.2.2.10 Attribute – abstract

The optional attribute *abstract* contains a brief summary of the portrayal function set.

```
PF_PortrayalFunctionSet::abstract : CharacterString[0..1]
```

8.2.2.11 Operation – f

The operation *f* maps the feature in the parameter list to a symbol using the associated feature portrayal functions.

```
PF_PortrayalFunctionSet::f(
     parameterList : PF_PortrayalFunctionParameterList
  ) : SY_Symbol
```

8.2.3 Type – PF_FeaturePortrayalFunction

8.2.3.1 Class semantics

PF_FeaturePortrayalFunction is the abstract root type for types that define feature portrayal functions, which map features to symbols. *PF_FeaturePortrayalFunctions* are collected in a *PF_PortrayalFunctionSet*, mapping sets of features in a feature catalogue (domain) to sets of symbols (range). *PF_FeaturePortrayalFunction* represents the concept of the feature portrayal function Φ_1 as illustrated in Figure 4.

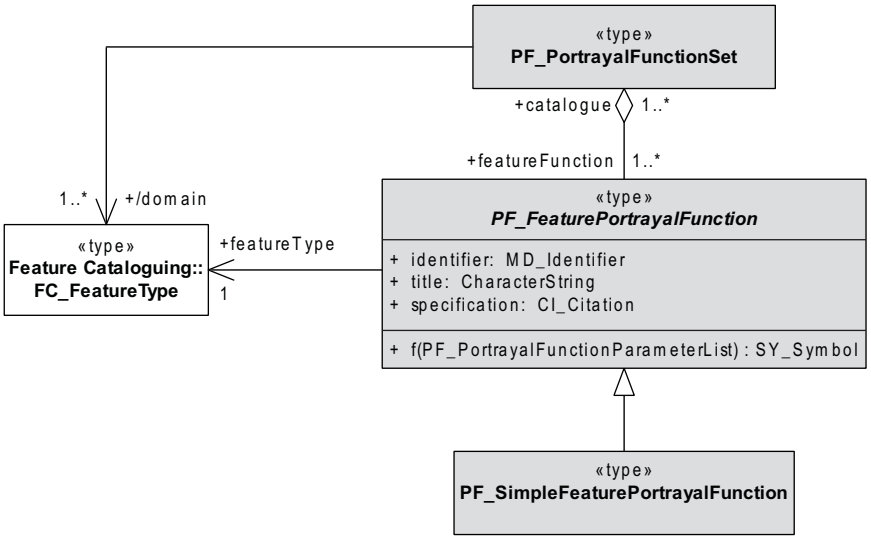

Figure 29 — PF_FeaturePortrayalFunction context diagram

8.2.3.2 Association role – catalogue

The association role *catalogue* associates a feature portrayal function with its containing portrayal function sets.

```
PF_FeaturePortrayalFunction::catalogue[1..*] : PF_PortrayalFunctionSet
```

8.2.3.3 Association role – featureType

The association role *featureType* defines the feature type that is mapped to a symbol.

```
PF_FeaturePortrayalFunction::featureType[1] : FC_FeatureType
```

8.2.3.4 Attribute – identifier

The attribute *identifier* is a machine readable name for the feature portrayal function. The identifier shall be unique.

```
PF_FeaturePortrayalFunction::identifier : MD_Identifier
```

8.2.3.5 Attribute – title

The attribute *title* is a multi-lingual human readable name for the feature portrayal function.

```
PF_FeaturePortrayalFunction::title : CharacterString
```

8.2.3.6 Attribute – specification

The attribute *specification* is a reference to the full details of the feature portrayal function.

```
PF_FeaturePortrayalFunction::specification : CI_Citation
```

8.2.3.7 Operation – f

The operation *f* maps the feature in the parameter list to a symbol. The feature parameter is of the associated feature type and the symbol is an instance of the associated symbol.

```
PF_FeaturePortrayalFunction::f(
     parameterList : PF_PortrayalFunctionParameterList
  ) : SY_Symbol
```

8.2.4 Type – PF_SimpleFeaturePortrayalFunction

8.2.4.1 Class semantics

PF_SimpleFeaturePortrayalFunction specializes the abstract root class *PF_FeaturePortrayalFunction* as a basic feature portrayal function that maps a feature to a symbol with no conditions.

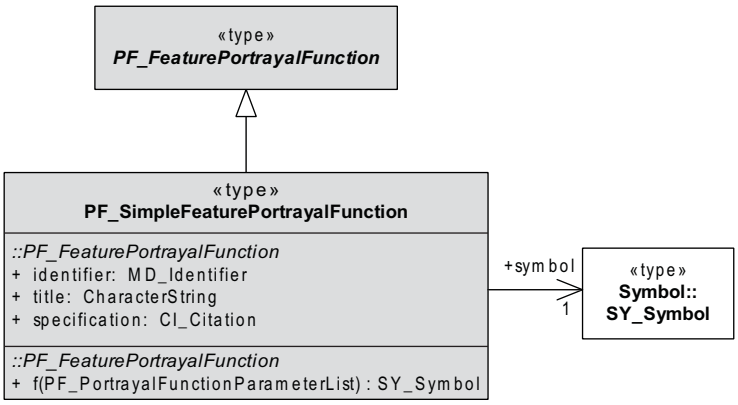

Figure 30 — PF_SimpleFeaturePortrayalFunction context diagram

8.2.4.2 Generalization – PF_FeaturePortrayalFunction

PF_SimpleFeaturePortrayalFunction specializes the abstract root class *PF_FeaturePortrayalFunction* as a basic feature portrayal function. This type of function maps features to symbols directly, there are no attribute or context dependent conditions.

8.2.4.3 Association role – symbol

The association role *symbol* defines the symbol to which a feature type is mapped.

```
PF_SimpleFeaturePortrayalFunction::symbol[1] : SY_Symbol
```

8.2.5 Type – PF_PortrayalFunctionParameterList

8.2.5.1 Class semantics

PF_PortrayalFunctionParameterList contains the feature that is mapped to a symbol in the portrayal function.

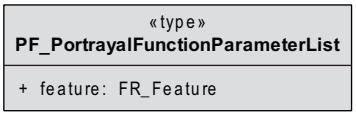

Figure 31 — PF_PortrayalFunctionParameterList context diagram

8.2.5.2 Attribute – feature

The attribute *feature* defines the feature instance that is mapped to a symbol in the portrayal function.

```
PF_PortrayalFunctionParameterList::feature : FR_Feature
```

8.3 Package – Symbol

8.3.1 Package semantics

The Symbol package defines facilities for defining symbols.

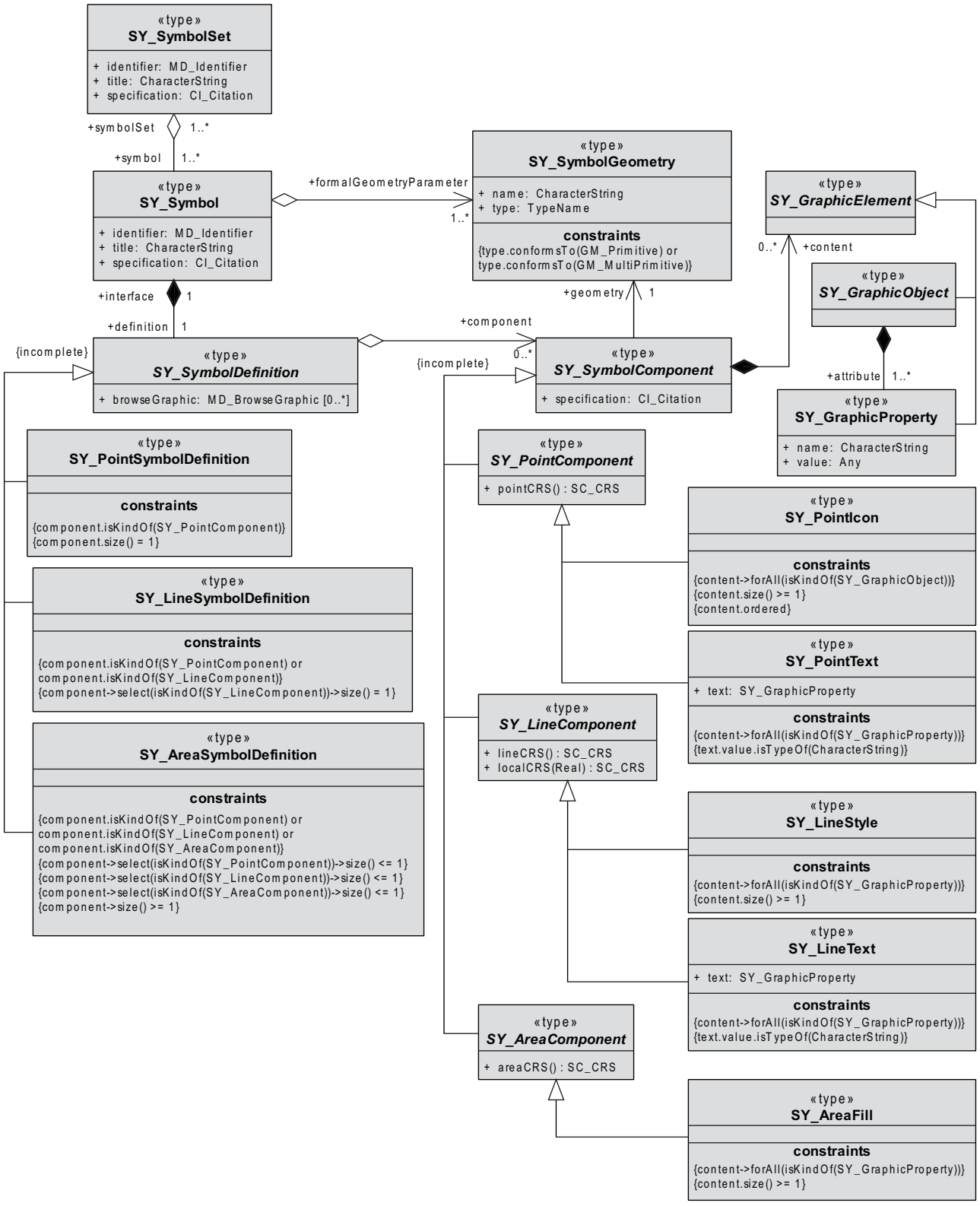

Figure 32 — Symbol

8.3.2 Type – SY_Symbol

8.3.2.1 Class semantics

SY_Symbol is the type used to define symbol classes. Symbols are collected into symbol sets.

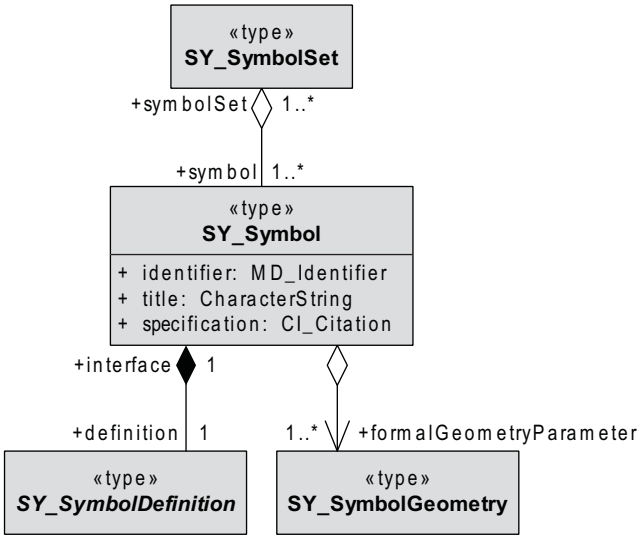

Figure 33 — SY_Symbol context diagram

8.3.2.2 Aggregation role – formalGeometryParameter

The aggregation role *formalGeometryParameter* references the specification of the symbolized geometry of the feature in the coordinate reference system of the portrayal.

```
SY_Symbol::formalGeometryParameter [1..*] : SY_SymbolGeometry
```

8.3.2.3 Composition role – definition

The composition role *definition* references the definition of the symbol that is applied to the feature geometry.

```
SY_Symbol::definition[1] : SY_SymbolDefinition
```

8.3.2.4 Aggregation role – symbolSet

The aggregation role *symbolSet* specifies the symbol sets in which a symbol is contained.

```
SY_Symbol::symbolSet[1..*] : SY_SymbolSet
```

8.3.2.5 Attribute – identifier

The attribute *identifier* is a machine readable name for the symbol. The identifier shall be unique.

```
SY_Symbol::identifier : MD_Identifier
```

8.3.2.6 Attribute – title

The attribute *title* is multi-lingual human readable name for the symbol.

```
SY_Symbol::title : CharacterString
```

8.3.2.7 Attribute – specification

The attribute *specification* is a reference to the full details of the portrayal symbol.

```
SY_Symbol::specification : CI_Citation
```

8.3.3 Type – SY_SymbolSet

8.3.3.1 Class semantics

SY_SymbolSet collects symbols into sets of symbols that are used together. Symbols can be shared among symbol sets. A symbol set can be equated with the legend of a map.

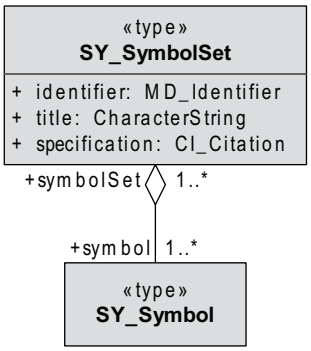

Figure 34 — SY_SymbolSet context diagram

8.3.3.2 Aggregation role – symbol

The aggregation role *symbol* associates the symbol set with the symbols in the set.

```
SY_SymbolSet::symbol[1..*] : SY_Symbol
```

8.3.3.3 Attribute – identifier

The attribute *identifier* is a machine readable name for the symbol set. The identifier shall be unique.

```
SY_SymbolSet::identifier : MD_Identifier
```

8.3.3.4 Attribute – title

The attribute *title* is multi-lingual human readable name for the symbol set.

```
SY_SymbolSet::title : CharacterString
```

8.3.3.5 Attribute – specification

The attribute *specification* is a reference to the full details of the portrayal symbol set.

```
SY_SymbolSet::specification : CI_Citation
```

8.3.4 Type – SY_SymbolGeometry

8.3.4.1 Class semantics

SY_SymbolGeometry is the type which specifies the geometry attributes of a symbol. A symbol geometry has an identifier and a type, which is either a subtype of *GM_Primitive* or a subtype of *GM_MultiPrimitive*.

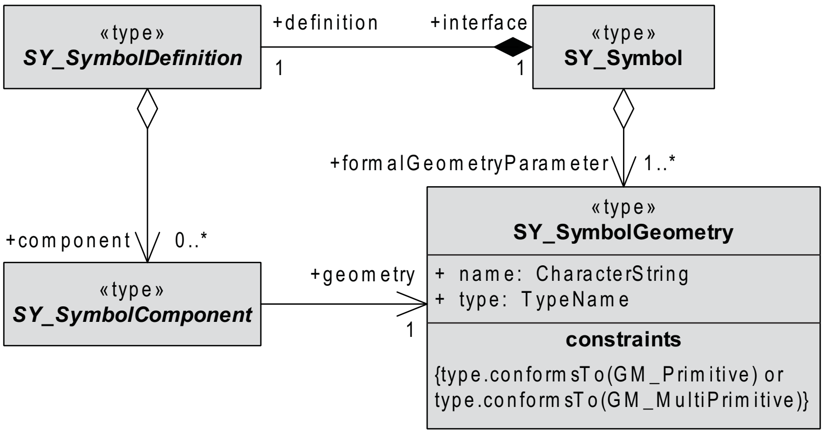

Figure 35 — SY_SymbolGeometry context diagram

8.3.4.2 Attribute – name

The attribute *name* is the name of the geometry attribute. The name shall be unique within the symbol definition.

```
SY_SymbolGeometry::name : CharacterString
```

8.3.4.3 Attribute – type

The attribute *type* specifies the type of the geometry attribute. The type of the geometry attribute shall be either as subtype of *GM_Primitive* or a subtype of *GM_MultiPrimitive*. The geometry types are most commonly *GM_Point*, *GM_Curve* and *GM_Surface*.

```
SY_SymbolGeometry::type : TypeName
```

8.3.5 Type – SY_SymbolDefinition

8.3.5.1 Class semantics

SY_SymbolDefinition is the abstract root type for types that define the composition of symbols. A symbol definition is comprised of a collection of symbol components, which contain the graphic elements and attributes used to define a symbol. Most common specializations are point, line and area symbol definitions which correspond to the common point, curve and surface geometries.

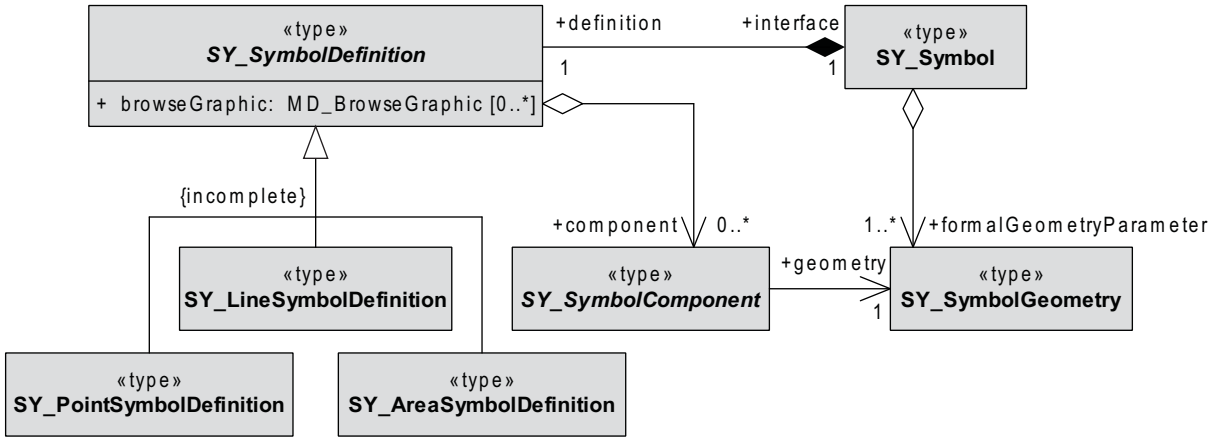

Figure 36 — SY_SymbolDefinition context diagram

8.3.5.2 Aggregation role – component

The optional aggregation role *component* collects the graphic component which makes up the symbol definition. A symbol definition with no components portrays nothing for a given feature (6.3).

```
SY_SymbolDefinition::component[0..*] : SY_SymbolComponent
```

8.3.5.3 Composition role – interface

The composition role *interface* associates a symbol definition with its symbol interface.

```
SY_SymbolDefinition::interface[1] : SY_Symbol
```

8.3.5.4 Attribute – browseGraphic

The optional attribute *browseGraphic* specifies graphics that may be used as metadata for the symbol and used to give a sample of the appearance of the symbol.

```
SY_SymbolDefinition::browseGraphic : MD_BrowseGraphic[0..*]
```

8.3.6 Type – SY_PointSymbolDefinition

8.3.6.1 Class semantics

SY_PointSymbolDefinition specializes the abstract root class *SY_SymbolDefinition* as a symbol definition for point geometry and is composed of a single point component.

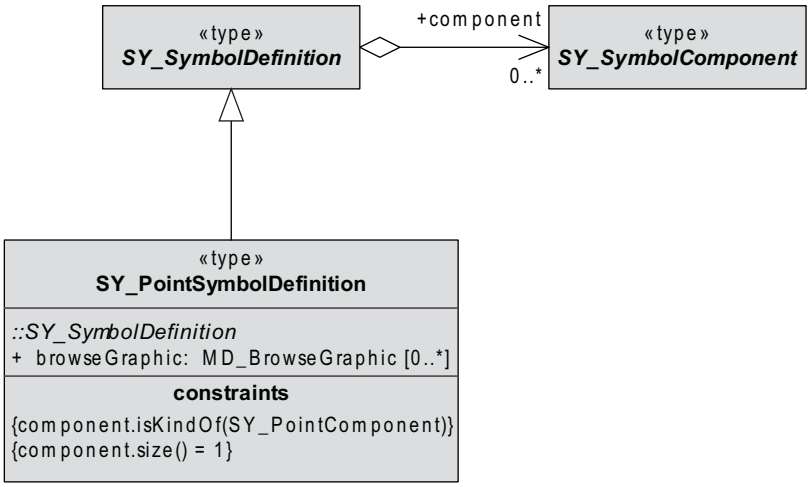

Figure 37 — SY_PointSymbolDefinition context diagram

8.3.6.2 Generalization – SY_SymbolDefinition

SY_PointSymbolDefinition specializes the abstract root class *SY_SymbolDefinition* as a symbol definition for point geometry and shall implement all inherited attributes, operations and associations.

8.3.7 Type – SY_LineSymbolDefinition

8.3.7.1 Class semantics

SY_LineSymbolDefinition specializes the abstract root class *SY_SymbolDefinition* as a symbol definition for curve geometry and is composed of a single line component and possibly multiple point components. Arrow heads at the ends of a line symbol would be an example of such point components.

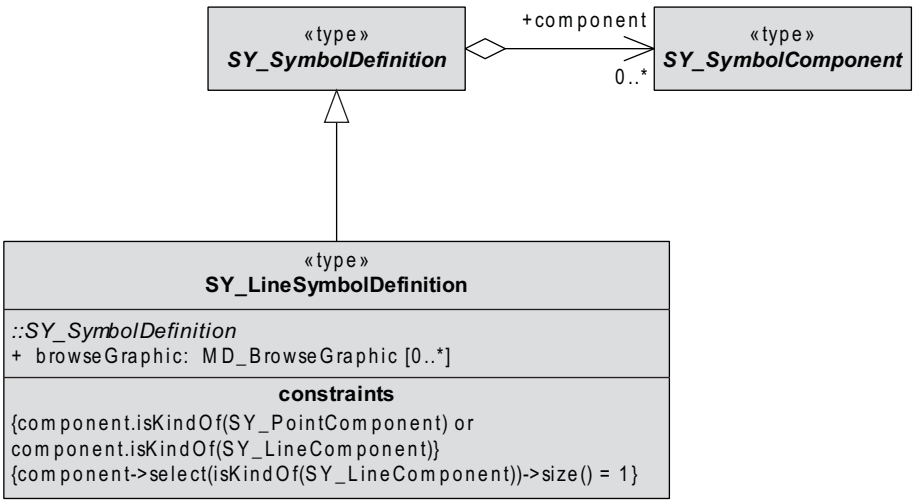

Figure 38 — SY_LineSymbolDefinition context diagram

8.3.7.2 Generalization – SY_SymbolDefinition

SY_LineSymbolDefinition specializes the abstract root class *SY_SymbolDefinition* as a symbol definition for curve geometry and shall implement all inherited attributes, operations and associations.

8.3.8 Type – SY_AreaSymbolDefinition

8.3.8.1 Class semantics

SY_AreaSymbolDefinition specializes the abstract root class *SY_SymbolDefinition* as a symbol definition for surface geometry and is composed of an optional area, line and point components. The area component is used to fill the surface area. The line component defines the boundary line style. The point component is used as a symbol icon on the interior of the surface.

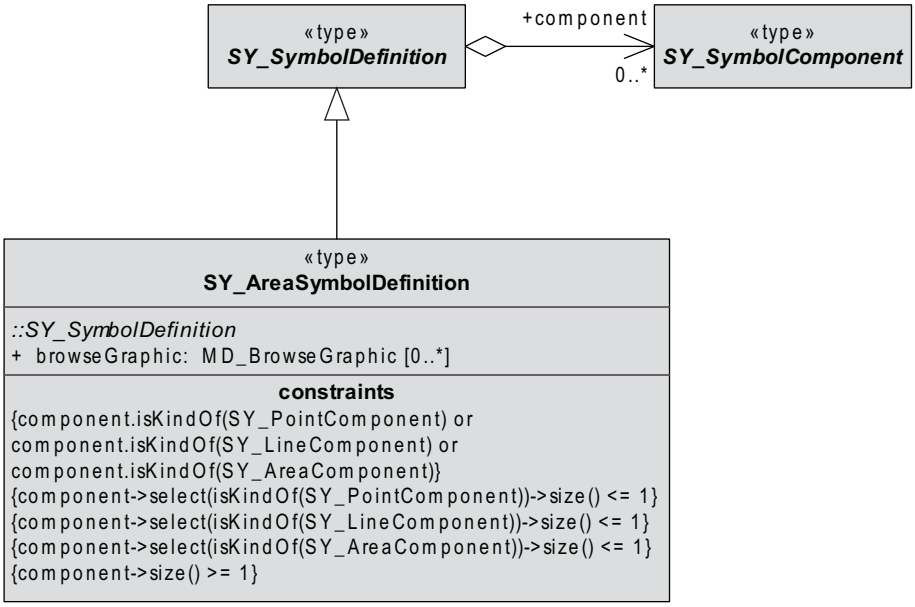

Figure 39 — SY_AreaSymbolDefinition context diagram

8.3.8.2 Generalization – SY_SymbolDefinition

SY_AreaSymbolDefinition specializes the abstract root class *SY_SymbolDefinition* as a symbol definition for surface geometry and shall implement all inherited attributes, operations and associations.

8.3.9 Type – SY_SymbolComponent

8.3.9.1 Class semantics

SY_SymbolComponent is the abstract root type for types that defined the graphic representation of symbols. A symbol component is comprised of a collection of graphic elements, which are the graphic objects and attributes used to define a symbol component. A symbol component references the geometry for which it is defined. *SY_SymbolComponent* has point, line and area specializations corresponding to point, curve and surface geometries.

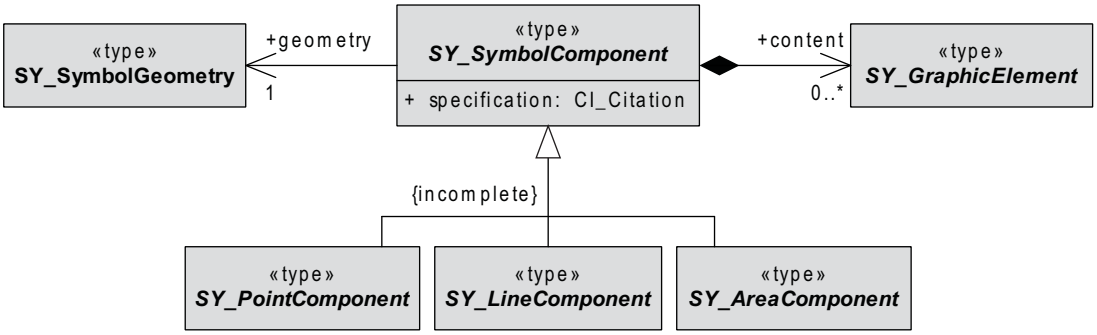

Figure 40 — SY_SymbolComponent context diagram

8.3.9.2 Association role – geometry

The association role *geometry* associates a symbol component with the symbol geometry definition to which the component is applied.

```
SY_SymbolComponent::geometry[1] : SY_SymbolGeometry
```

8.3.9.3 Composition role – content

The composition role *content* specifies the graphic elements which make up the symbol component.

```
SY_SymbolComponent::content[0..*] : SY_GraphicElement
```

8.3.9.4 Attribute – specification

The attribute *specification* cites the specification standard for the graphics definition language used to define the symbol component.

```
SY_SymbolComponent::specification : CI_Citation
```

8.3.10 Type – SY_GraphicElement

SY_GraphicElement is the abstracts root for the graphic elements, as defined in a graphic specification language, that defines a symbol. The graphic elements may be graphic objects, such as ovals, rectangles, or paths, or properties such as colour or line width.

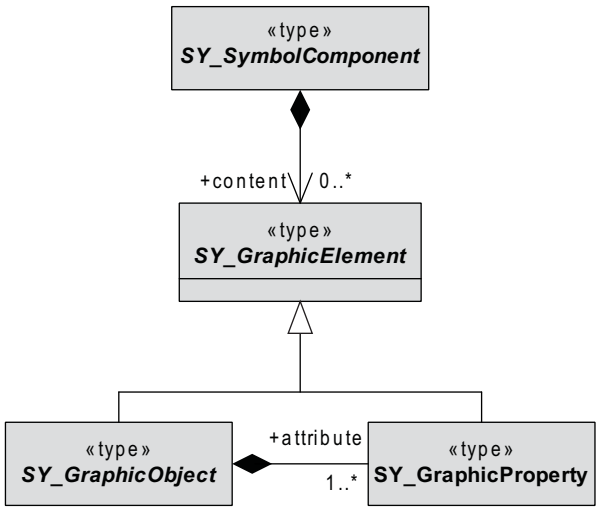

Figure 41 — SY_GraphicElement context diagram

8.3.11 Type – SY_GraphicObject

8.3.11.1 Class semantics

SY_GraphicObject is an abstract specialization of *SY_GraphicElement* as a graphic object defined in a graphic specification language. Graphic objects are graphic elements, such as ovals, rectangles, or paths. A graphic object in turn has properties, such as location attributes, size attributes, colour attributes, etc.

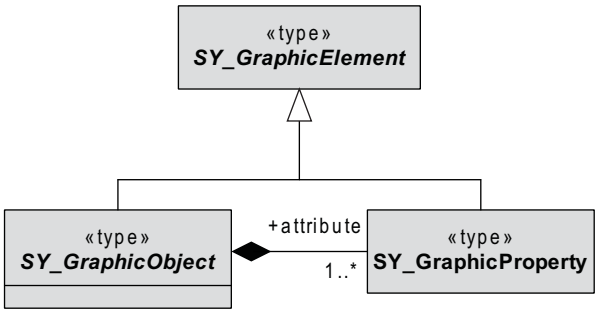

Figure 42 — SY_GraphicObject context diagram

8.3.11.2 Generalization – SY_GraphicElement

SY_GraphicObject is an abstract specialization of *SY_GraphicElement* as a graphic object, such as an oval, rectangle, or path. As such it shall implement all inherited attributes, operations and associations.

8.3.11.3 Composition role – attribute

The composition role *attribute* specifies the attributes of a graphic object, such as location attributes, size attributes, colour attributes, etc.

```
SY_GraphicObject::attribute[1..*] : SY_GraphicProperty
```

8.3.12 Type – SY_GraphicProperty

8.3.12.1 Class semantics

The *SY_GraphicProperty* class template defines graphic properties of symbol components. It can be used to define properties, such as location attributes, size attributes, colour attributes, etc. A graphic property has an identifier and a value of the template type.

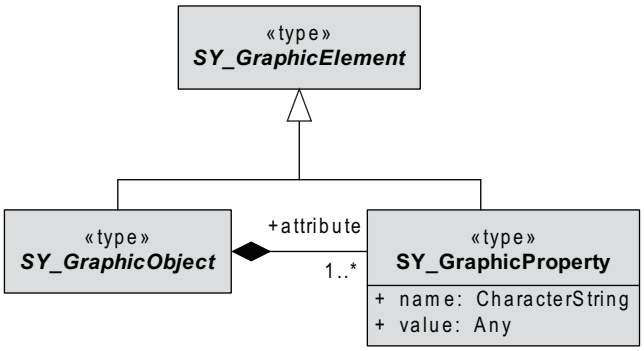

Figure 43 — SY_GraphicProperty context diagram

8.3.12.2 Generalization – SY_GraphicElement

SY_GraphicProperty is a templatized specialization of *SY_GraphicElement* used to define graphic properties, such as location attributes, size attributes, colour attributes, etc. As such it shall implement all inherited attributes, operations and associations.

8.3.12.3 Attribute – name

The attribute *name* names the graphic property.

```
SY_GraphicProperty::name : CharacterString
```

8.3.12.4 Attribute – value

The attribute *value* holds the value of the graphic property.

```
SY_GraphicProperty::value : Any
```

8.3.13 Type – SY_PointComponent

8.3.13.1 Class semantics

SY_PointComponent specializes *SY_SymbolComponent* for point symbol geometries. A point component has a coordinate reference system in which graphic objects are placed.

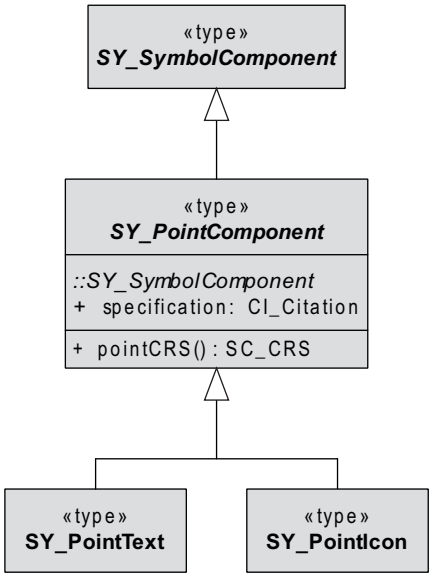

Figure 44 — SY_PointComponent context diagram

8.3.13.2 Generalization – SY_SymbolComponent

SY_PointComponent specializes the abstract root class *SY_SymbolComponent* as a symbol component for point geometry and shall implement all inherited attributes, operations and associations.

8.3.13.3 Operation – pointCRS

The operation *pointCRS* returns the coordinate reference system of the point component.

```
SY_PointComponent::pointCRS(
   ) : SC_CRS
```

8.3.14 Type – SY_PointIcon

8.3.14.1 Class semantics

SY_PointIcon specializes *SY_PointComponent* as a point component of a symbol with an icon graphic representation as opposed to a text graphic representation. A point icon is composed of graphic objects such as ovals, rectangles, and paths placed in the coordinate reference system of the point component.

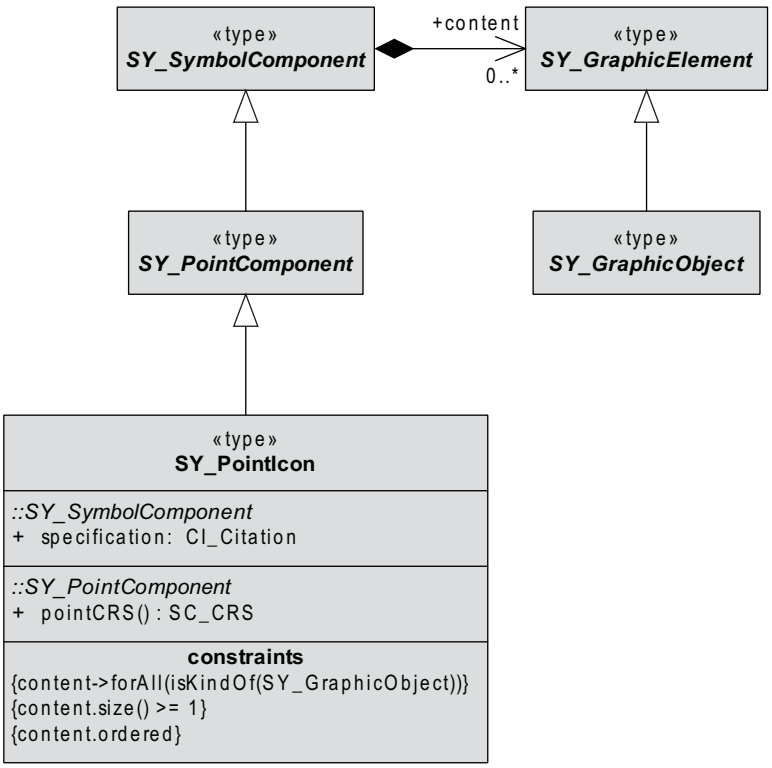

Figure 45 — SY_PointIcon context diagram

8.3.14.2 Generalization – SY_PointComponent

SY_PointIcon specializes the abstract class *SY_PointComponent* as an icon graphic for point geometry and shall implement all inherited attributes, operations and associations.

8.3.15 Type – SY_PointText

8.3.15.1 Class semantics

SY_PointText specializes *SY_PointComponent* as a text attached to a point geometry as opposed to an icon graphic representation. The text is placed in the coordinate reference system of the point component. A point text has graphic properties such as font, colour, and font size.

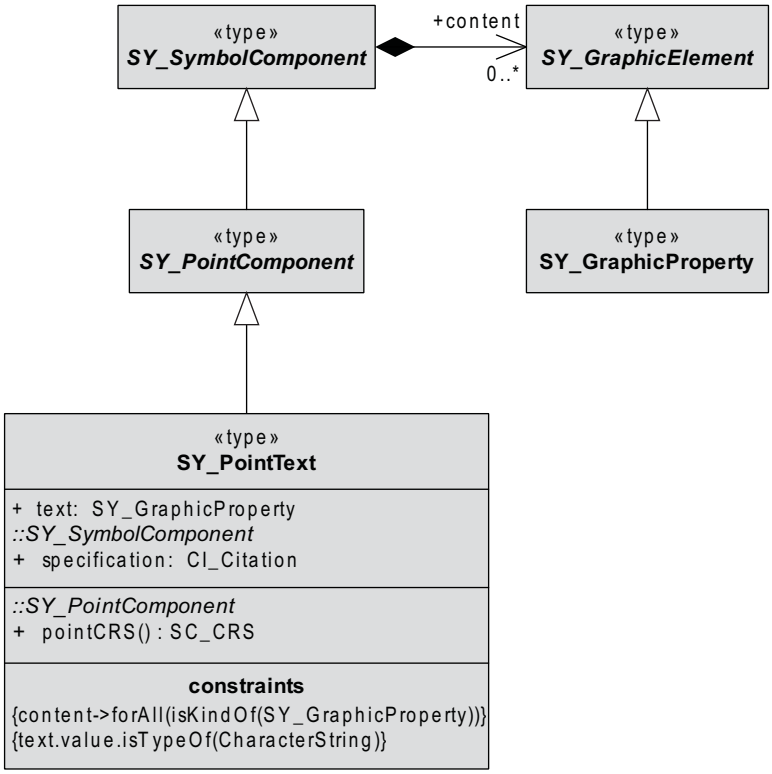

Figure 46 — SY_PointText context diagram

8.3.15.2 Generalization – SY_PointComponent

SY_PointText specializes the abstract class *SY_PointComponent* as a text for point geometry and shall implement all inherited attributes, operations and associations.

8.3.15.3 Attribute – text

The attribute *text* specifies the text of the point text component.

```
SY_PointText::text : SY_GraphicProperty
```

8.3.16 Type – SY_LineComponent

8.3.16.1 Class semantics

SY_LineComponent specializes *SY_SymbolComponent* for line symbol geometries. The line component has a coordinate reference system defined at every point along its path, which has an x-axis that is tangential to the path and a y-axis that is perpendicular to the path. The line style also has a one dimensional coordinate reference system that is defined along the path of the line style.

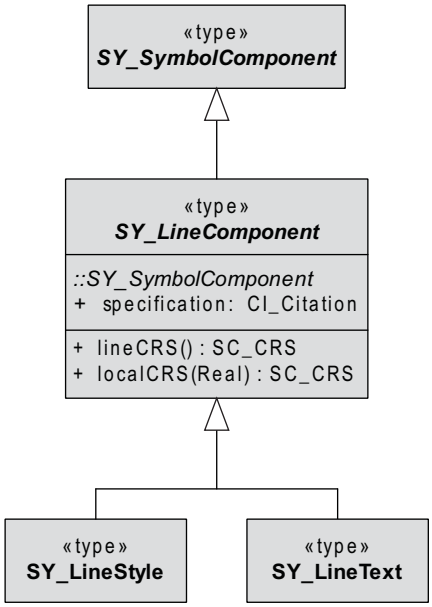

Figure 47 — SY_LineComponent context diagram

8.3.16.2 Generalization – SY_SymbolComponent

SY_LineComponent specializes the abstract root class *SY_SymbolComponent* as a symbol component for line geometry and shall implement all inherited attributes, operations and associations.

8.3.16.3 Operation – lineCRS

The operation *lineCRS* returns a one dimensional coordinate reference system that is defined along the path of the line component.

```
SY_LineComponent::lineCRS(
  ) : SC_CRS
```

8.3.16.4 Operation – localCRS

The operation *localCRS* accepts a real value as input and returns a coordinate reference system. The parameter measure specifies the measure along the path of the line component specifying the origin of the coordinate reference system. The x-axis is tangential to the path at that point.

```
SY_LineComponent::localCRS(
    measure : Real
  ) : SC_CRS
```

8.3.17 Type – SY_LineStyle

8.3.17.1 Class semantics

SY_LineStyle specializes *SY_LineComponent* as a line style component of a line symbol as opposed to a text component. A line style is composed of graphic properties such as colour, width, and dash length. Some of these properties are meaningful within the coordinate reference system of the line component.

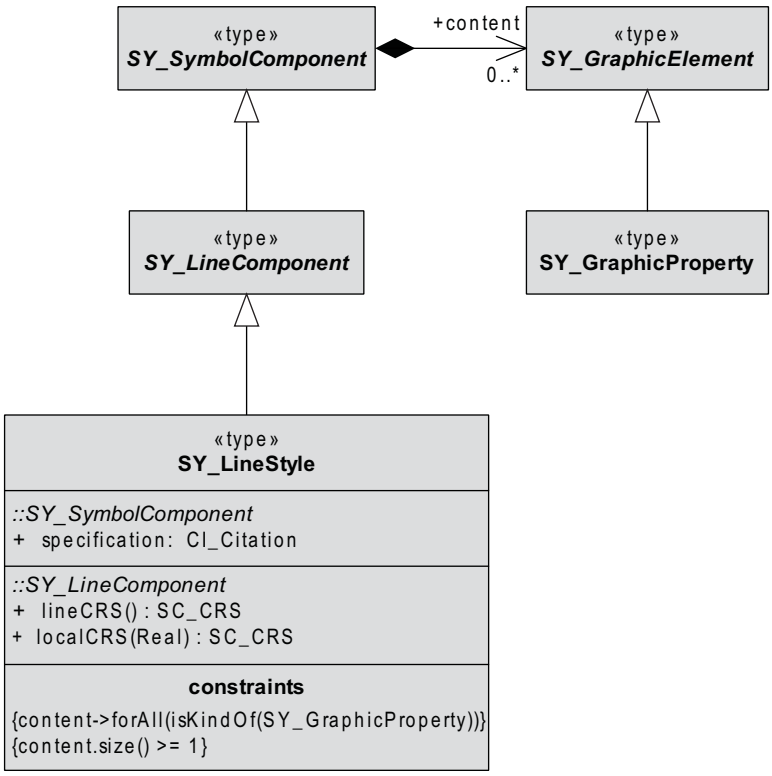

Figure 48 — SY_LineStyle context diagram

8.3.17.2 Generalization – SY_LineComponent

SY_LineStyle specializes the abstract type *SY_LineComponent* as a line style component for line geometry and shall implement all inherited attributes, operations and associations.

8.3.18 Type – SY_LineText

8.3.18.1 Class semantics

SY_LineText specializes *SY_LineComponent* as a text along a line geometry as opposed to a line graphic representation. The text is placed in the coordinate reference system of the line component.

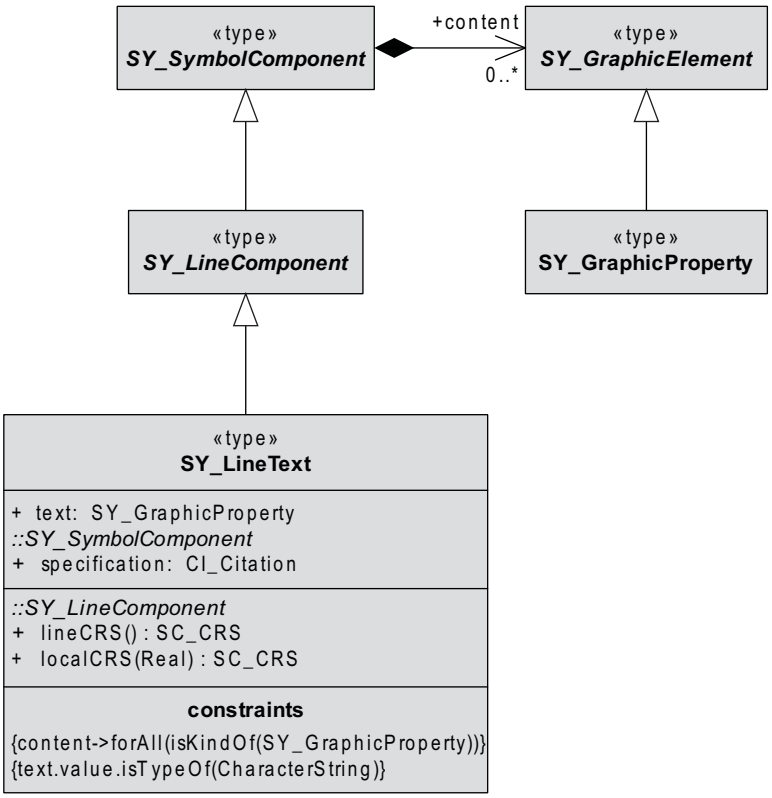

Figure 49 — SY_LineText context diagram

8.3.18.2 Generalization – SY_LineComponent

SY_LineText specializes the abstract class *SY_LineComponent* as a text along a line geometry and shall implement all inherited attributes, operations and associations. A point text has graphic properties such as font, colour, and font size.

8.3.18.3 Attribute – text

The attribute *text* specifies the text of the line text component.

```
SY_LineText::text : SY_GraphicProperty
```

8.3.19 Type – SY_AreaComponent

8.3.19.1 Class semantics

SY_AreaComponent specializes *SY_SymbolComponent* for area symbol geometries. The area component has a coordinate reference system defined for the area.

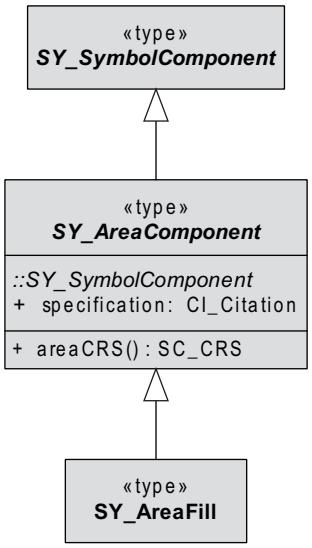

Figure 50 — SY_AreaComponent context diagram

8.3.19.2 Generalization – SY_SymbolComponent

SY_AreaComponent specializes the abstract root class *SY_SymbolComponent* as a symbol component for area geometry and shall implement all inherited attributes, operations and associations.

8.3.19.3 Operation – areaCRS

The operation *areaCRS* returns the two dimensional coordinate reference system of the area component.

```
SY_AreaComponent::areaCRS(
  ) : SC_CRS
```

8.3.20 Type – SY_AreaFill

8.3.20.1 Class semantics

SY_AreaFill specializes *SY_AreaComponent* as an area fill component of a symbol. An area fill is composed of graphic properties such as colour and fill pattern.

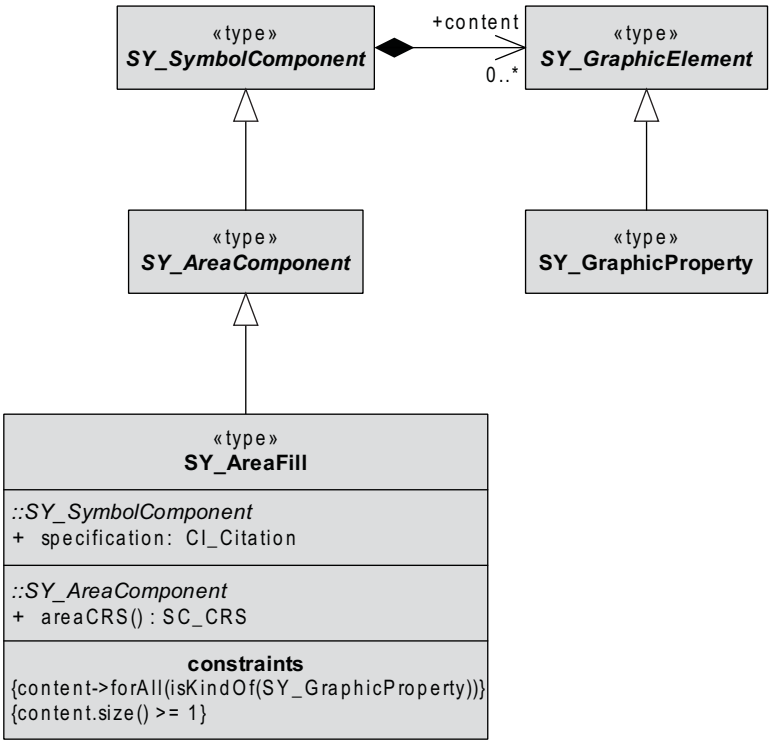

Figure 51 — SY_AreaFill context diagram

8.3.20.2 Generalization – SY_AreaComponent

SY_AreaFill specializes the abstract type *SY_AreaComponent* as a symbol component for area geometry and shall implement all inherited attributes, operations and associations.

8.4 Package – Portrayal Catalogue

8.4.1 Package semantics

The Portrayal Catalogue package defines a catalogue for transmitting portrayal information.

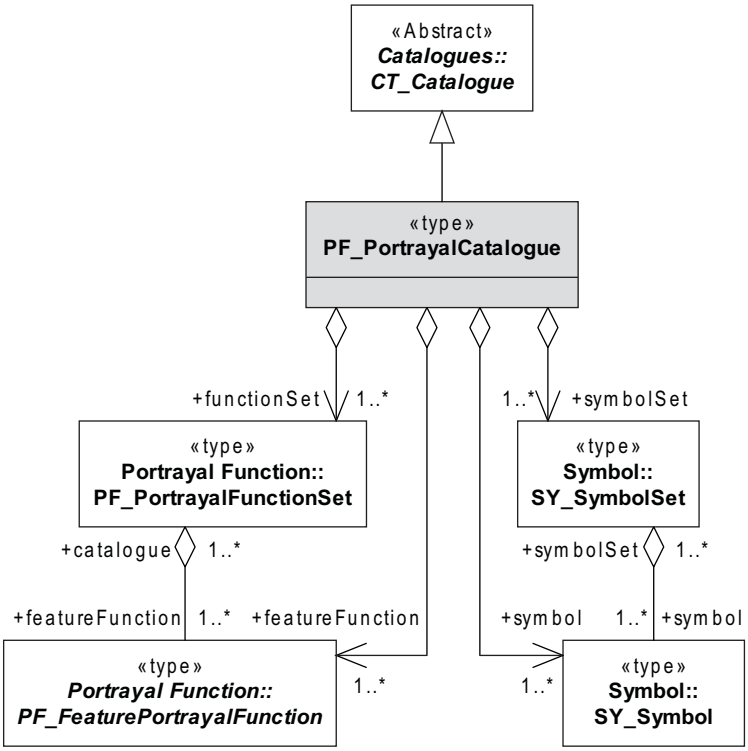

Figure 52 — Portrayal Catalogue

8.4.2 Type – PF_PortrayalCatalogue

8.4.2.1 Class semantics

PF_PortrayalCatalogue specializes *CT_Catalogue* for portrayal and as such contains the elements necessary for a portrayal. A portrayal catalogue contains portrayal function sets and their component feature portrayal functions as well as symbol sets and their component symbols.

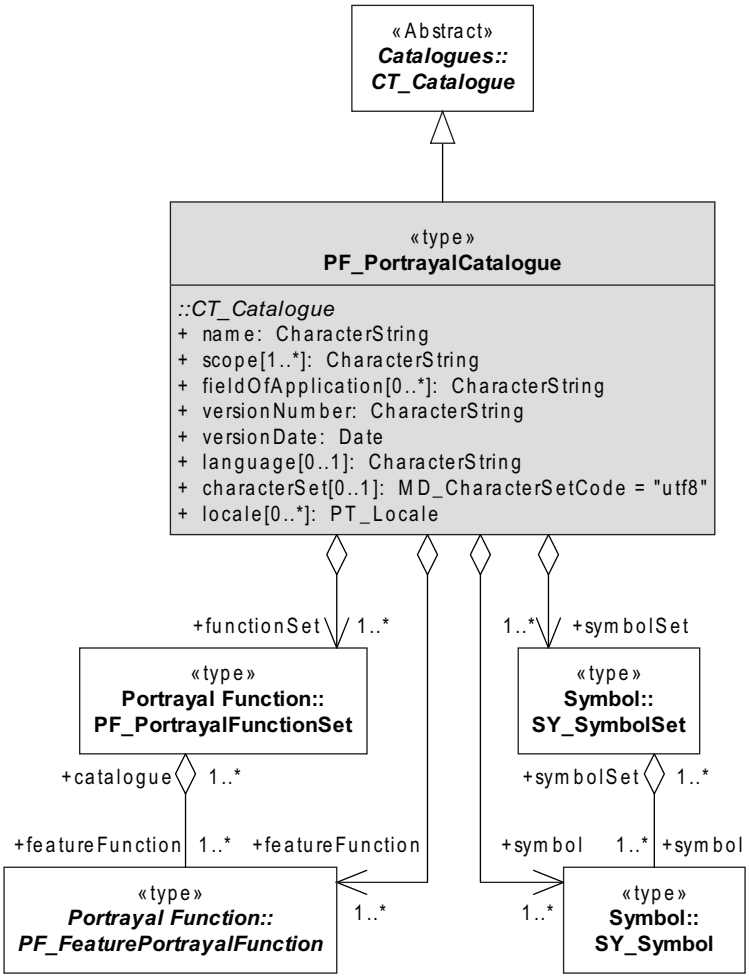

Figure 53 — PF_PortrayalCatalogue context diagram

8.4.2.2 Generalization – CT_Catalogue

PF_PortrayalCatalogue specializes the abstract type *CT_Catalogue* for portrayal and shall implement all inherited attributes, operations and associations.

8.4.2.3 Aggregation role – functionSet

The aggregation role *functionSet* specifies the portrayal function set components of a portrayal catalogue.

```
PF_PortrayalCatalogue::functionSet[1..*] : PF_PortrayalFunctionSet
```

8.4.2.4 Aggregation role – featureFunction

The aggregation role *featureFunction* specifies the feature portrayal function components of a portrayal catalogue.

```
PF_PortrayalCatalogue::featureFunction[1..*] : PF_FeaturePortrayalFunction
```

8.4.2.5 Aggregation role – symbolSet

The aggregation role *symbolSet* specifies the symbol set components of a portrayal catalogue.

```
PF_PortrayalCatalogue::symbolSet[1..*] : SY_SymbolSet
```

8.4.2.6 Aggregation role – symbol

The aggregation role *symbol* specifies the symbol components of a portrayal catalogue.

```
PF_PortrayalCatalogue::symbol[1..*] : SY_Symbol
```

9 Package – Portrayal Extensions

9.1 Package semantics

The Portrayal Extensions package supplies extension facilities to extend the Portrayal Core package. The package is divided into seven subpackages that may, in large part, be used independently of each other. Three packages extend the Portrayal Function core package: the Conditional Function Extension package, the Context Extension package, and the Function Symbol Parameter Extension package. The Context Extension package also extends the Portrayal Catalogue core package. Four packages extend the Symbol core package: the Compound Symbol Extension package, the Complex Symbol Extension package, the Reusable Symbol Component Extension package, and the Symbol Parameter Extension package.

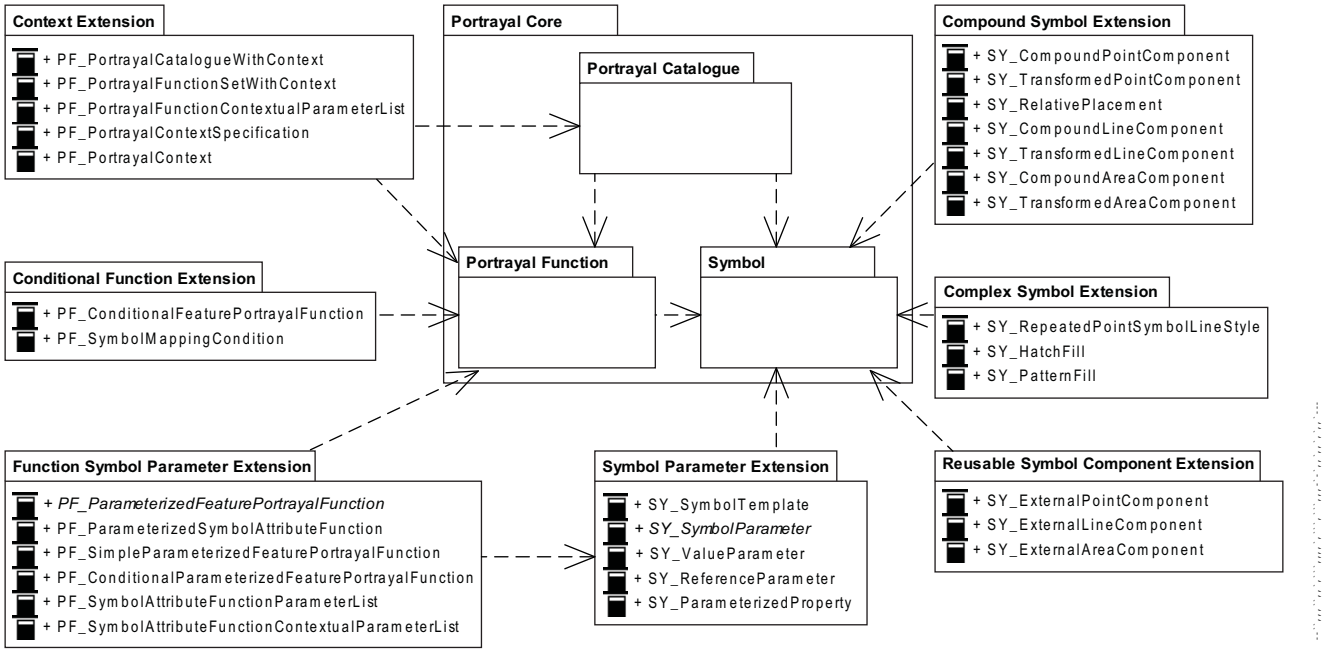

Figure 54 — Extension package structure with dependencies

9.2 Package – Conditional Function Extension

9.2.1 Package semantics

The Conditional Function Extension package adds the capability to define portrayal functions which map features to different symbols depending on the properties of the features.

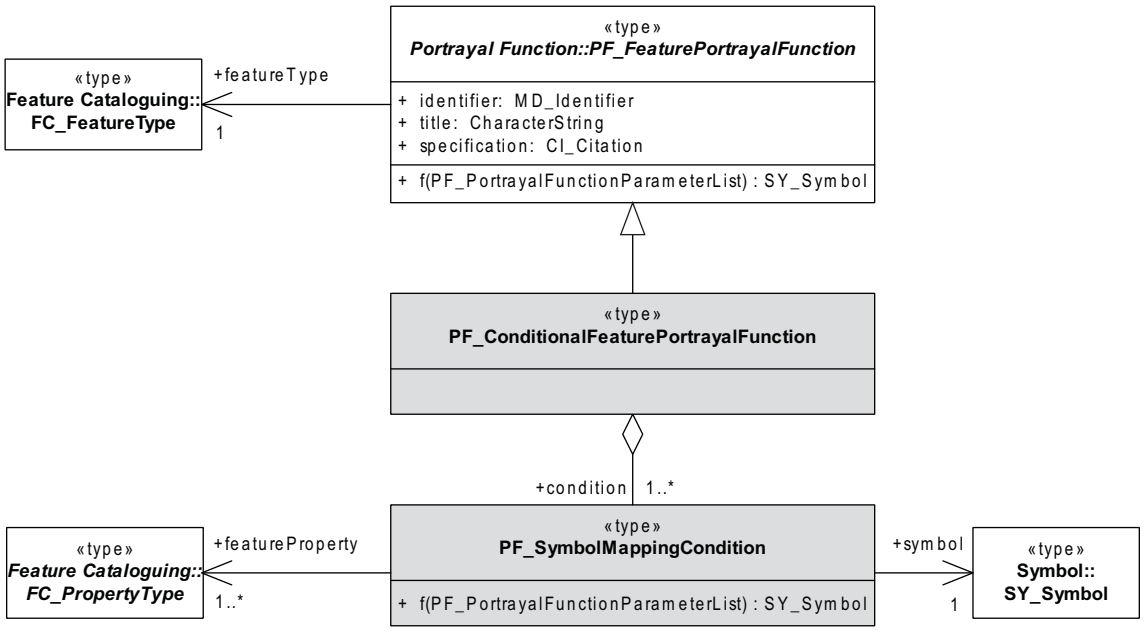

Figure 55 — Portrayal Function – Conditional

9.2.2 Type – PF_ConditionalFeaturePortrayalFunction

9.2.2.1 Class semantics

PF_ConditionalFeaturePortrayalFunction specializes the root type *PF_FeaturePortrayalFunction* as a feature portrayal function which maps to different symbols depending on associated symbol mapping conditions.

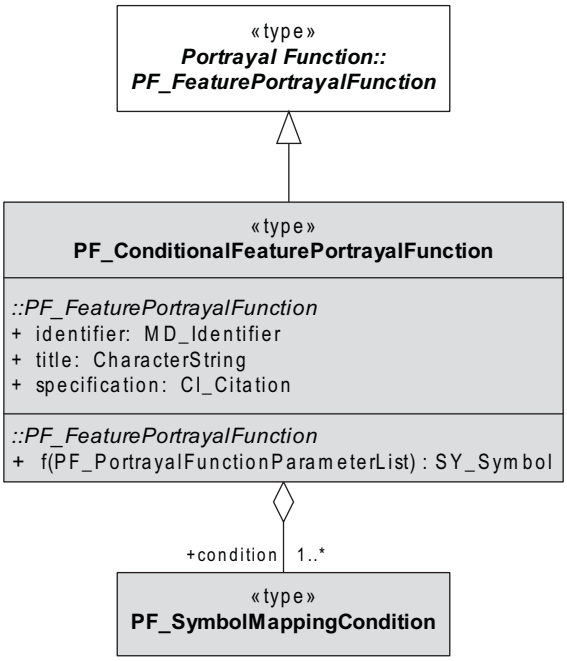

Figure 56 — PF_ConditionalFeaturePortrayalFunction context diagram

9.2.2.2 Generalization – PF_FeaturePortrayalFunction

PF_ConditionalFeaturePortrayalFunction specializes the root class *PF_FeaturePortrayalFunction* as a feature portrayal function which maps to different symbols depending on associated symbol mapping conditions. As such it shall implement all inherited attributes, operations and associations.

9.2.2.3 Aggregation role – condition

The aggregation role *condition* aggregates the symbol mapping conditions of a conditional feature portrayal function.

```
PF_ConditionalFeaturePortrayalFunction::condition[1..*] :
     PF_SymbolMappingCondition
```

9.2.3 Type – PF_SymbolMappingCondition

9.2.3.1 Class semantics

PF_SymbolMappingCondition maps the feature of the associated conditional feature portrayal function to a symbol depending on the properties of a given feature.

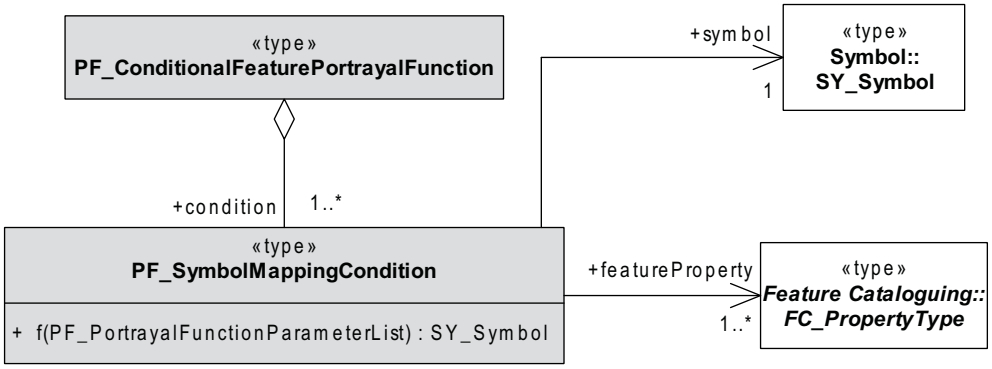

Figure 57 — PF_SymbolMappingCondition context diagram

9.2.3.2 Association role – symbol

The association role *symbol* defines the symbol to which the symbol mapping condition maps.

```
PF_SymbolMappingCondition::symbol[1] : SY_Symbol
```

9.2.3.3 Association role – featureProperty

The association role *featureProperty* defines properties on which the mapping depends.

```
PF_SymbolMappingCondition::featureProperty[1..*] : FC_PropertyType
```

9.2.3.4 Operation – f

The operation *f* maps the feature in the parameter list to a symbol. The symbol is an instance of the associated symbol.

```
PF_SymbolMappingCondition::f(
```

```
    parameterList : PF_PortrayalFunctionParameterList
) : SY_Symbol
```

9.3 Package – Context Extension

9.3.1 Package semantics

The Context Extension package adds context information to the Portrayal Catalogue and Portrayal Function packages. Context information is information that is general to the portrayal, not feature specific, such as lighting conditions, type of user, or type of media.

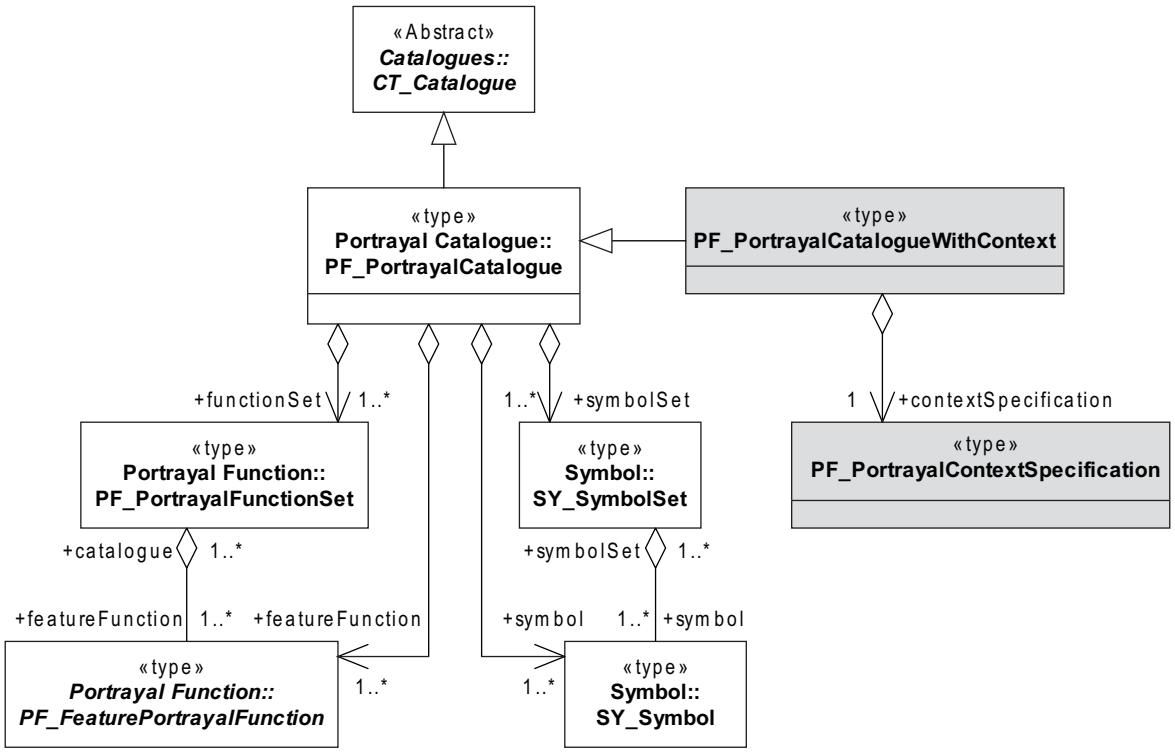

Figure 58 — Portrayal Catalogue - Context

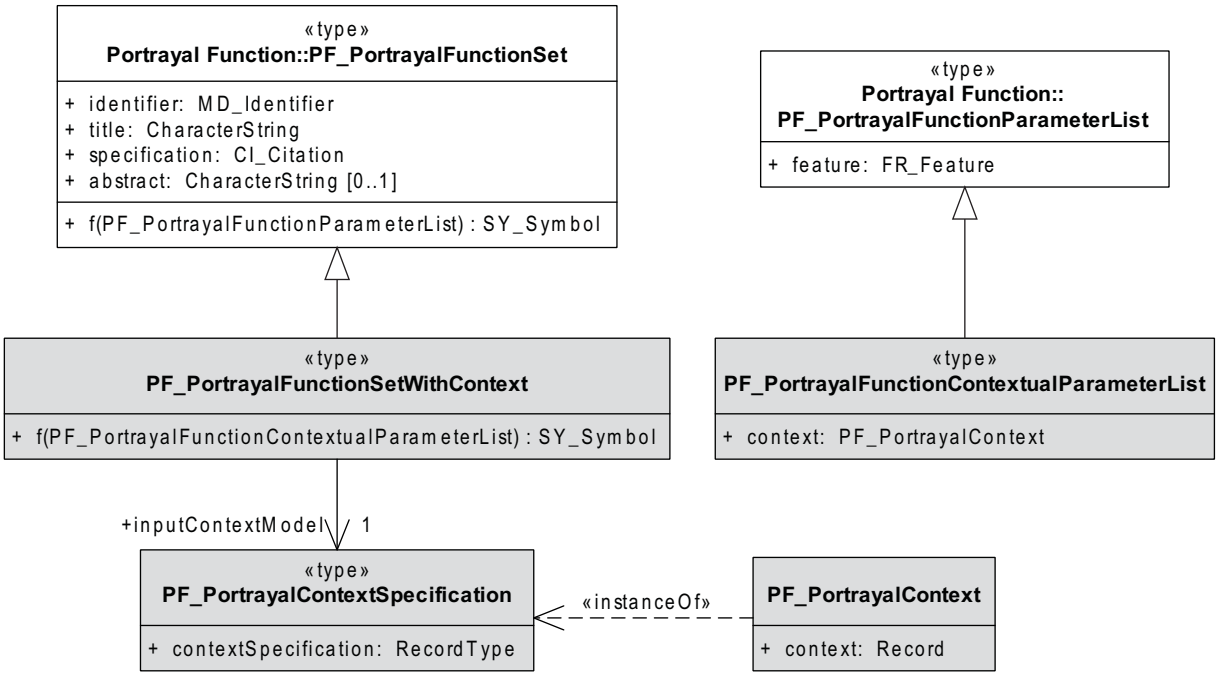

Figure 59 — Portrayal Function – Context

9.3.2 Type – PF_PortrayalCatalogueWithContext

9.3.2.1 Class semantics

PF_PortrayalCatalogueWithContext specializes *PF_PortrayalCatalogue* as a portrayal catalogue that includes a context specification in addition to portrayal function sets, feature portrayal functions, symbol sets, and symbols.

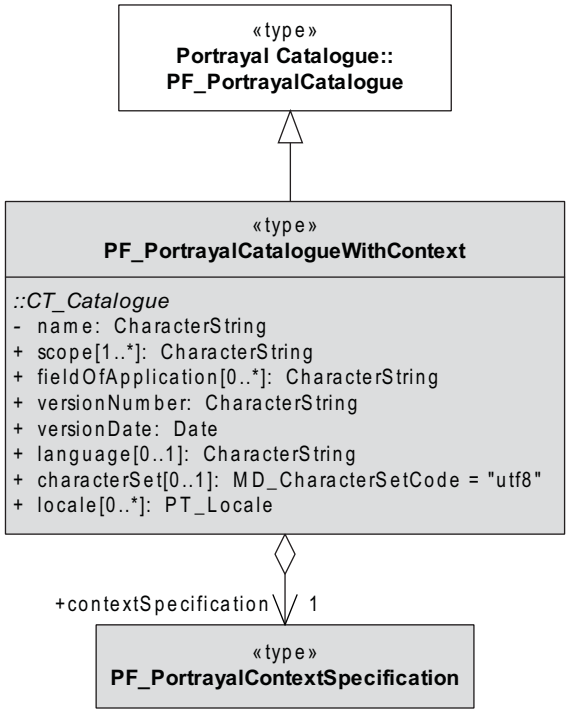

Figure 60 — PF_PortrayalCatalogueWithContext context diagram

9.3.2.2 Generalization – PF_PortrayalCatalogue

PF_PortrayalCatalogueWithContext specializes *PF_PortrayalCatalogue* as a portrayal catalogue with a context specification and shall implement all inherited attributes, operations and associations.

9.3.2.3 Aggregation role – contextSpecification

The association role *contextSpecification* defines the portrayal context specification component of a portrayal catalogue.

```
PF_PortrayalCatalogueWithContext::contextSpecification[1] :
     PF_PortrayalContextSpecification
```

9.3.3 Type – PF_PortrayalFunctionSetWithContext

9.3.3.1 Class semantics

PF_PortrayalFunctionSetWithContext specializes the root class *PF_PortrayalFunctionSet* as a portrayal function set which uses context information in addition to feature information to accomplish the mapping to symbols. The portrayal function set is associated with the portrayal context specification which specifies the structure of the context that is part of the parameter list of the mapping function f.

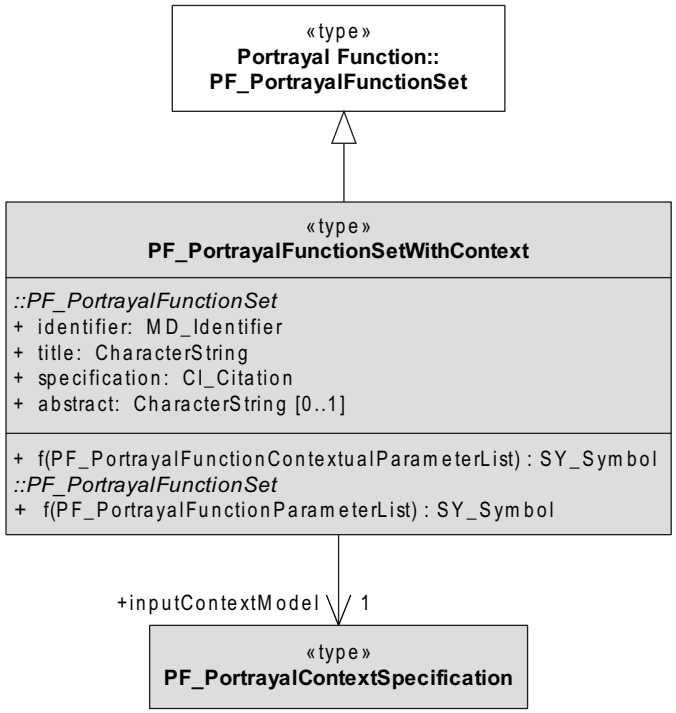

Figure 61 — PF_PortrayalFunctionSetWithContext context diagram

9.3.3.2 Generalization – PF_PortrayalFunctionSet

PF_PortrayalFunctionSetWithContext specializes the root class *PF_PortrayalFunctionSet* as a portrayal function set which uses context information in addition to feature information to accomplish the mapping to symbols. As such it shall implement all inherited attributes, operations and associations.

9.3.3.3 Association role – inputContextModel

The association role *inputContextModel* specifies the portrayal context specification which defines the structure of the contexts record for a portrayal function set. This may be shared by multiple portrayal function sets.

```
PF_PortrayalFunctionSetWithContext::inputContextModel[1] :
     PF_PortrayalContextSpecification
```

9.3.3.4 Operation – f

The operation *f* specializes *PF_PortrayalFunctionSet::f* (8.2.2.11) accepting a parameter list which includes a context parameter in addition to a feature parameter.

```
PF_PortrayalFunctionSetWithContext::f(
     parameterList : PF_PortrayalFunctionContextualParameterList
  ) : SY_Symbol
```

9.3.4 Type – PF_PortrayalFunctionContextualParameterList

9.3.4.1 Class semantics

PF_PortrayalFunctionContextualParameterList specializes the root class *PF_PortrayalFunctionParameterList* as a portrayal function parameter list which contains context information in addition to feature information.

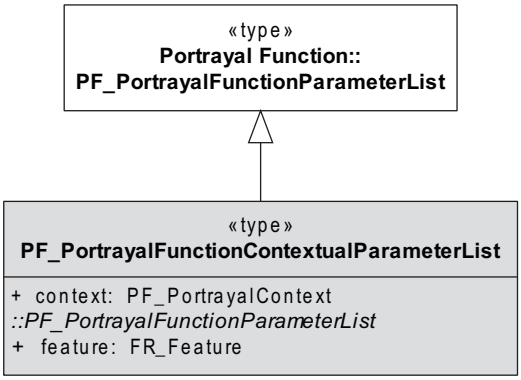

Figure 62 — PF_PortrayalFunctionContextualParameterList context diagram

9.3.4.2 Generalization – PF_PortrayalFunctionParameterList

PF_PortrayalFunctionContextualParameterList specializes the root class *PF_PortrayalFunctionParameterList* to include context information. As such it shall implement all inherited attributes, operations and associations.

9.3.4.3 Attribute – context

The attribute *context* contains the context information of the portrayal function parameter list.

```
PF_PortrayalFunctionContextualParameterList::context : PF_PortrayalContext
```

9.3.5 Type – PF_PortrayalContextSpecification

9.3.5.1 Class semantics

PF_PortrayalContextSpecification specifies the structure of a context record.

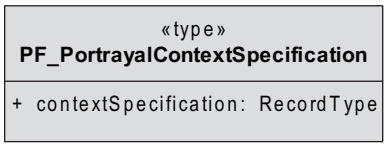

Figure 63 — PF_PortrayalContextSpecification context diagram

9.3.5.2 Attribute – contextSpecification

The attribute *contextSpecification* specifies the record structure of a context record.

```
PF_PortrayalContextSpecification::contextSpecification : RecordType
```

9.3.6 Type – PF_PortrayalContext

9.3.6.1 Class semantics

PF_PortrayalContext is used to represent an implementation of the type *PF_PortrayalContextSpecification*. It contains a context record structured as specified in the PF_PortrayalContextSpecification::contextSpecification attribute.

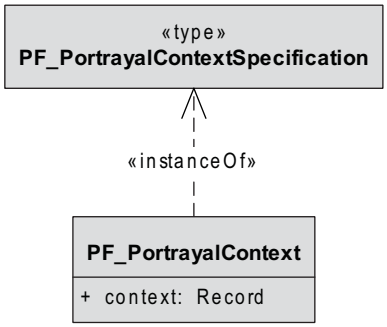

Figure 64 — PF_PortrayalContext context diagram

9.3.6.2 Attribute – context

The attribute *context* is the context record with the context information of a portrayal.

```
PF_PortrayalContext::context : Record
```

9.4 Package – Compound Symbol Extension

9.4.1 Package semantics

The Compound Symbol Extension adds the ability to compose symbols from symbol components.

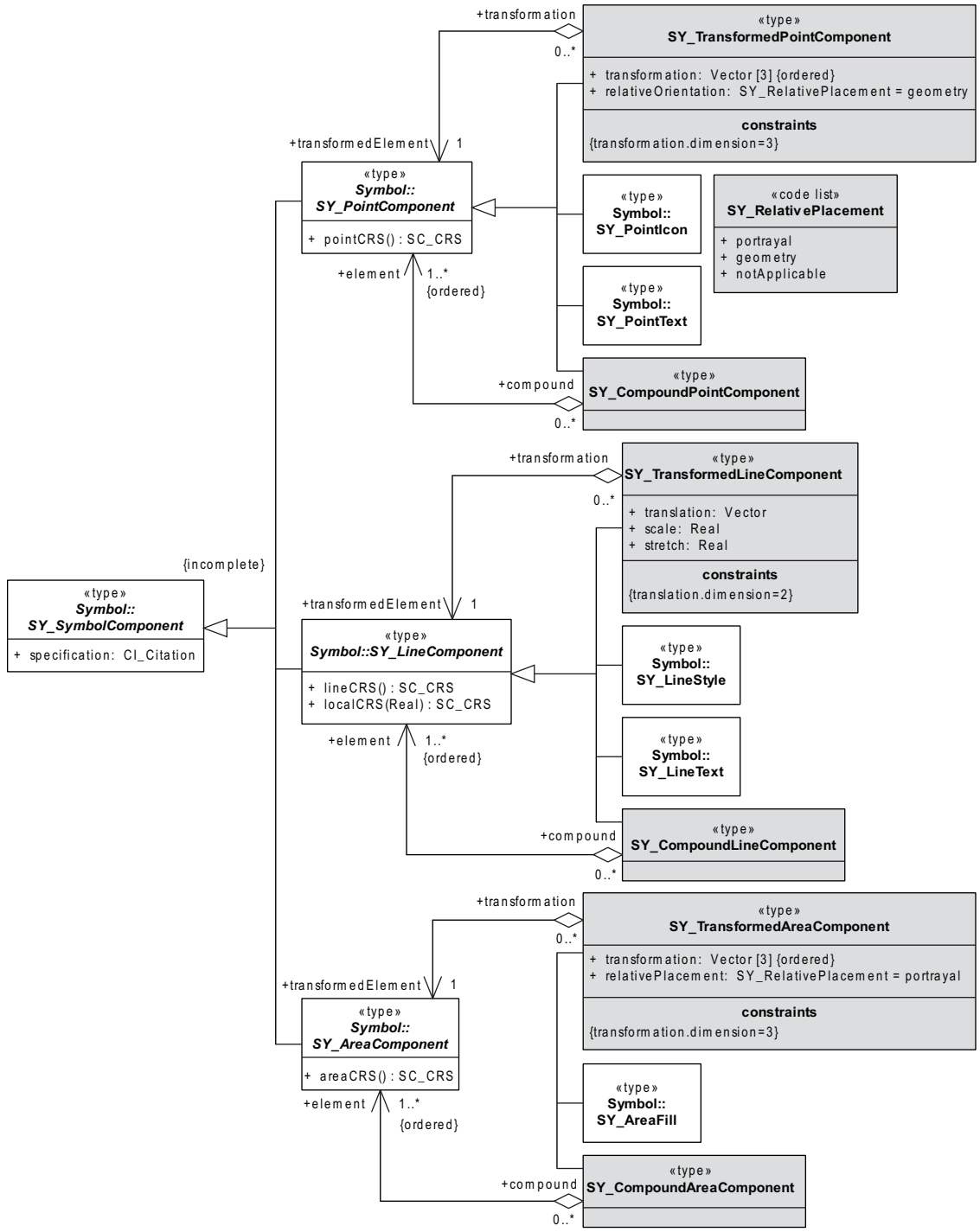

Figure 65 — Symbol – Compound

Figure 66 shows a simple example of symbol composition. The point symbol is composed of a point icon, in this case a filled circle, and a text. Figure 66 shows this in an ad hoc textual representation of the symbol definition, lower left side, while Figure 67 is a graphical depiction. The lower right item (Beeches:StartPtSym) in Figure 66 is a textual representation of a symbol instance.

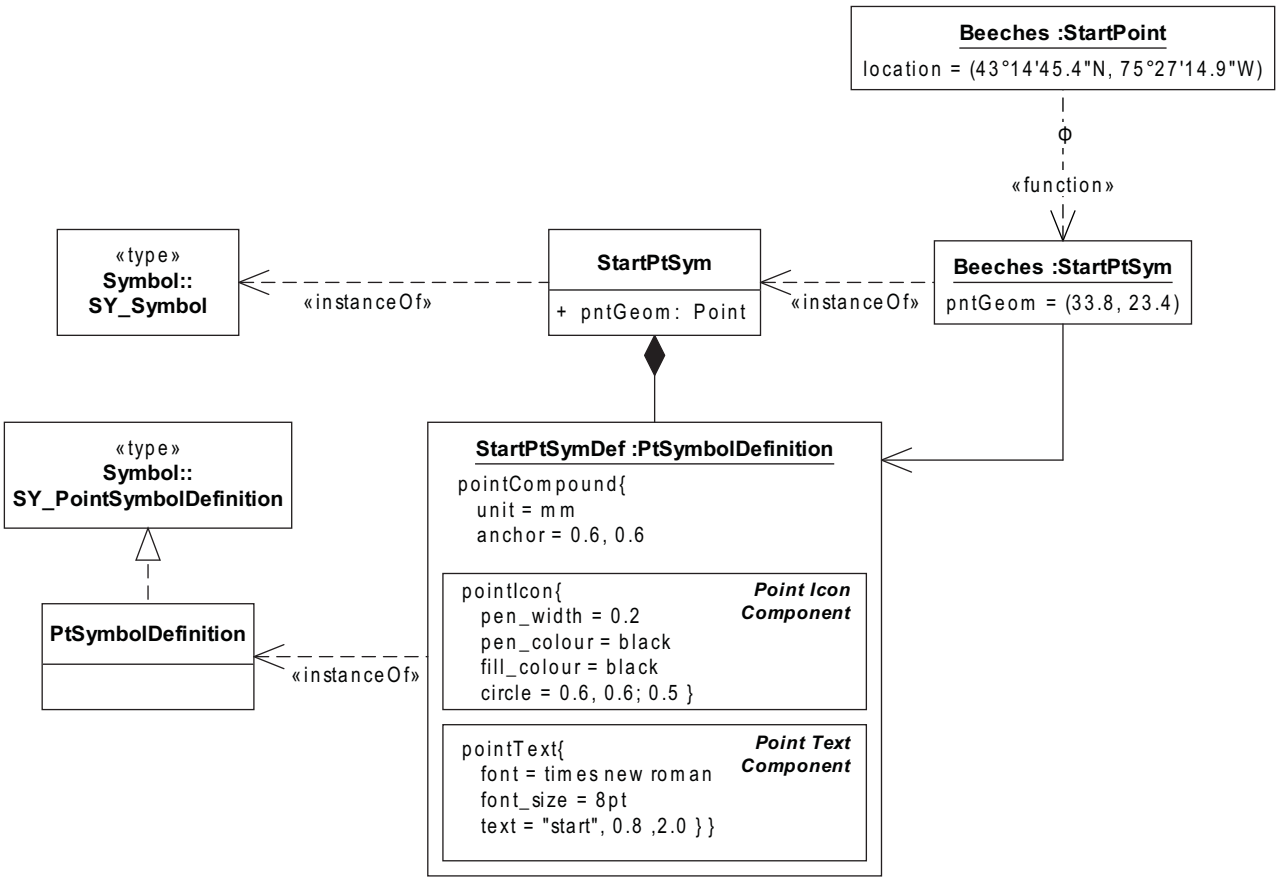

Figure 66 — Compound symbol example

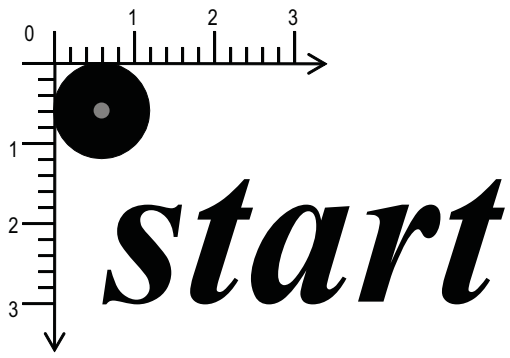

Figure 67 — Compound symbol definition

9.4.2 Type – SY_CompoundPointComponent

9.4.2.1 Class semantics

SY_CompoundPointComponent specializes *SY_PointComponent* as a collection of point subcomponents. The elements of the compound are ordered for display.

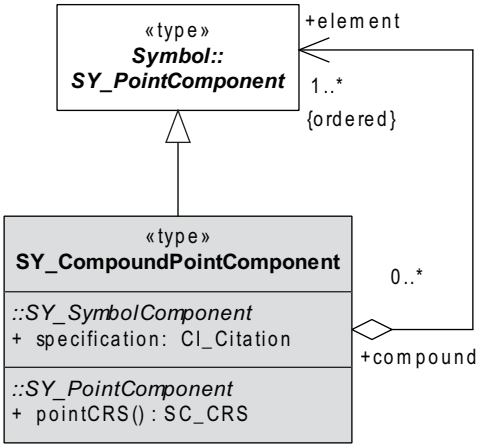

Figure 68 — SY_CompoundPointComponent context diagram

9.4.2.2 Generalization – SY_PointComponent

SY_CompoundPointComponent specializes *SY_PointComponent* as a collection of point subcomponents and shall implement all inherited attributes, operations and associations.

9.4.2.3 Aggregation role – element

The aggregation role *element* collects the point subcomponents of the compound point component.

```
SY_CompoundPointComponent::element[1..*] : SY_PointComponent
```

9.4.3 Type – SY_TransformedPointComponent

9.4.3.1 Class semantics

SY_TransformedPointComponent specializes *SY_PointComponent* as a geometric transformation of a point component. The transformed point component is deformed, rotated, and translated with a transformation matrix. The orientation of the point component can be specified either relative to the containing point component or relative to the overall portrayal.

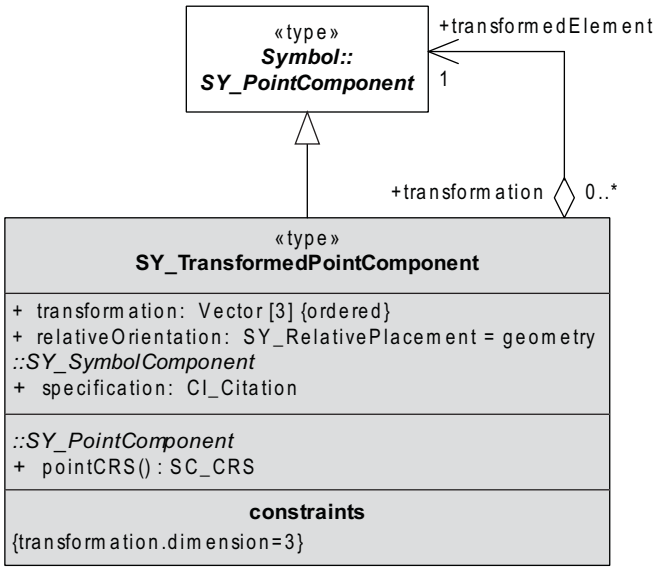

Figure 69 — SY_TransformedPointComponent context diagram

9.4.3.2 Generalization – SY_PointComponent

SY_TransformedPointComponent specializes *SY_PointComponent* as a geometric transformation of a point component and shall implement all inherited attributes, operations and associations.

9.4.3.3 Aggregation role – transformedElement

The aggregation role *transformedElement* defines the point component that is transformed.

```
SY_TransformedPointComponent::transformedElement[1] : SY_PointComponent
```

9.4.3.4 Attribute – transformation

The attribute *transformation* uses homogeneous coordinates to define a geometric transformation of a point component.

```
SY_TransformedPointComponent::transformation : Vector[3]{ordered}
```

9.4.3.5 Attribute – relativeOrientation

The attribute *relativeOrientation* specifies whether a transformation is relative to the containing symbol component, relative to the portrayal coordinate reference system, or whether the attribute is not applicable. The default value of this attribute is *geometry*.

```
SY_TransformedPointComponent::relativeOrientation : SY_RelativePlacement =
    geometry
```

9.4.4 Code List – SY_RelativePlacement

9.4.4.1 Class semantics

The code list *SY_RelativePlacement* names the possible relative placements of a symbol component, in particular point symbols. A placement can either be relative to the coordinate reference system of the portrayal or relative to the coordinate reference system of the containing symbol component within which the current component is being placed.

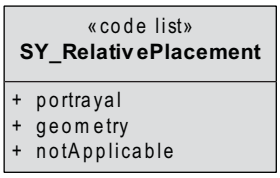

Figure 70 — SY_RelativePlacement context diagram

9.4.4.2 Attribute – portrayal

The enumeration value *portrayal* indicates that a symbol component is placed relative to the coordinate reference system of the portrayal.

```
SY_RelativePlacement::portrayal
```

9.4.4.3 Attribute – geometry

The enumeration value *geometry* indicates that a symbol component is placed relative to the coordinate reference system of the containing symbol component within which the current component is being placed.

```
SY_RelativePlacement::geometry
```

9.4.4.4 Attribute – notApplicable

The enumeration value *notApplicable* indicates that a symbol component's orientation cannot be defined relative to either the coordinate reference system of the portrayal or the coordinate reference system of the containing symbol component.

```
SY_RelativePlacement::notApplicable
```

9.4.5 Type – SY_CompoundLineComponent

9.4.5.1 Class semantics

SY_CompoundLineComponent specializes *SY_LineComponent* as a collection of line subcomponents. The elements of the compound are ordered for display.

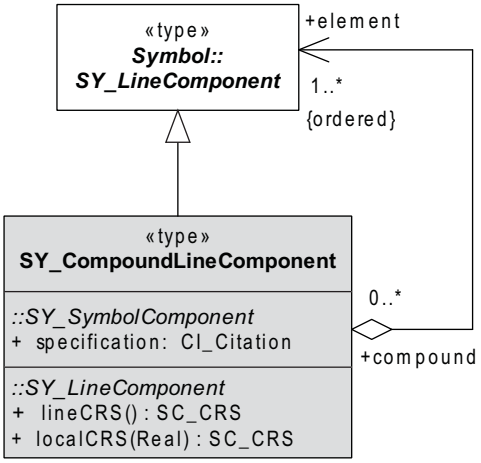

Figure 71 — SY_CompoundLineComponent context diagram

9.4.5.2 Generalization – SY_LineComponent

SY_CompoundLineComponent specializes *SY_LineComponent* as a collection of line subcomponents and shall implement all inherited attributes, operations and associations.

9.4.5.3 Aggregation role – element

The aggregation role *element* collects the line subcomponents of the compound line component.

```
SY_CompoundLineComponent::element[1..*] : SY_LineComponent
```

9.4.6 Type – SY_TransformedLineComponent

9.4.6.1 Class semantics

SY_TransformedLineComponent specializes *SY_LineComponent* as a geometric transformation of a line component. The transformed line component is translated along and perpendicular to the curve (Figure 73), scaled perpendicular to the curve (Figure 74), and stretched along the curve (Figure 75).

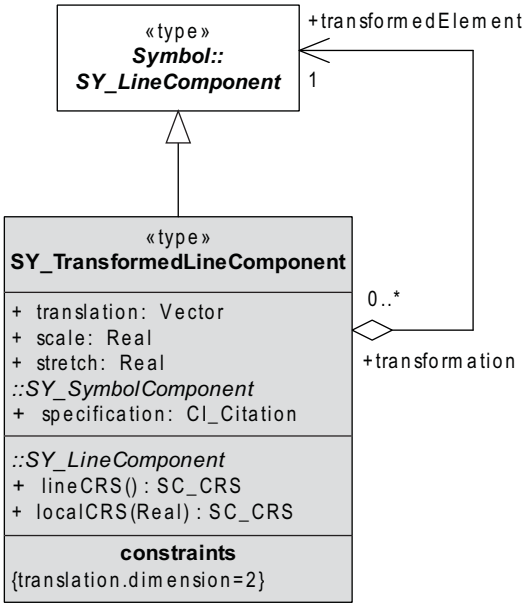

Figure 72 — SY_TransformedLineComponent context diagram

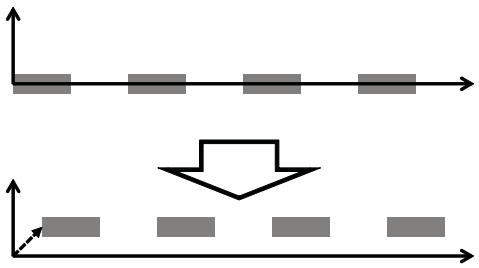

Figure 73 — Translation of a line component

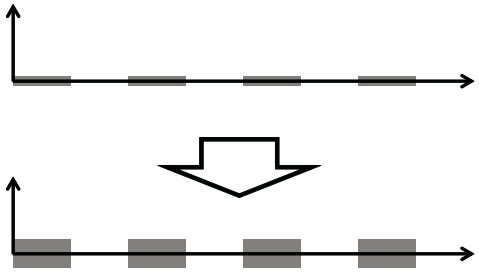

Figure 74 — Scaling of a line component

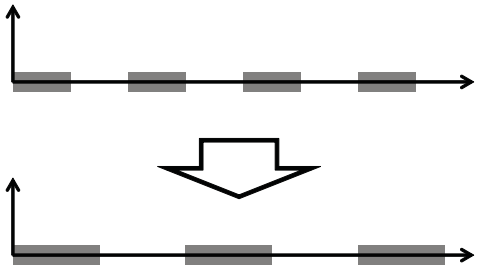

Figure 75 — Stretching of a line component

9.4.6.2 Generalization – SY_LineComponent

SY_TransformedLineComponent specializes *SY_LineComponent* as a geometric transformation of a line component and shall implement all inherited attributes, operations and associations.

9.4.6.3 Aggregation role – transformedElement

The aggregation role *transformedElement* defines the line component that is transformed.

```
SY_TransformedLineComponent::transformedElement[1] : SY_LineComponent
```

9.4.6.4 Attribute – translation

The attribute *translation* specifies the translation of the line component relative to the containing line component in two dimensions: along the path of the curve, and perpendicular to the curve.

```
SY_TransformedLineComponent::translation : Vector
```

9.4.6.5 Attribute – scale

The attribute *scale* specifies the scaling of the line component perpendicular to the curve.

```
SY_TransformedLineComponent::scale : Real
```

9.4.6.6 Attribute – stretch

The attribute *stretch* specifies the stretching of the line component along the curve.

```
SY_TransformedLineComponent::stretch : Real
```

9.4.7 Type – SY_CompoundAreaComponent

9.4.7.1 Class semantics

SY_CompoundAreaComponent specializes *SY_AreaComponent* as a collection of area subcomponents. The elements of the compound are ordered for display.

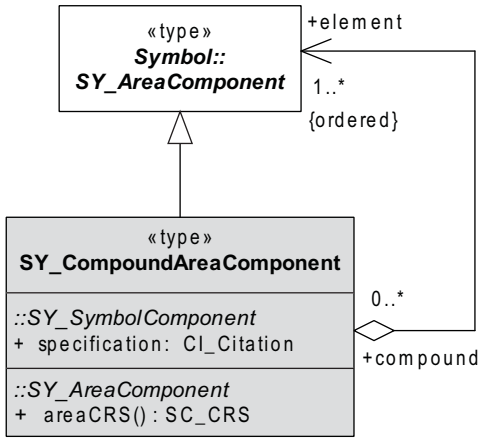

Figure 76 — SY_CompoundAreaComponent context diagram

9.4.7.2 Generalization – SY_AreaComponent

SY_CompoundAreaComponent specializes *SY_AreaComponent* as a collection of area subcomponents and shall implement all inherited attributes, operations and associations.

9.4.7.3 Aggregation role – element

The aggregation role *element* collects the subfills of the compound area component.

```
SY_CompoundAreaComponent::element[1..*] : SY_AreaComponent
```

9.4.8 Type – SY_TransformedAreaComponent

9.4.8.1 Class semantics

SY_TransformedAreaComponent specializes *SY_AreaComponent* as a geometric transformation of an area component. The transformed area component is deformed, rotated, and translated with a transformation matrix. The orientation of the area component can be specified either relative to the containing area component or relative to the overall portrayal.

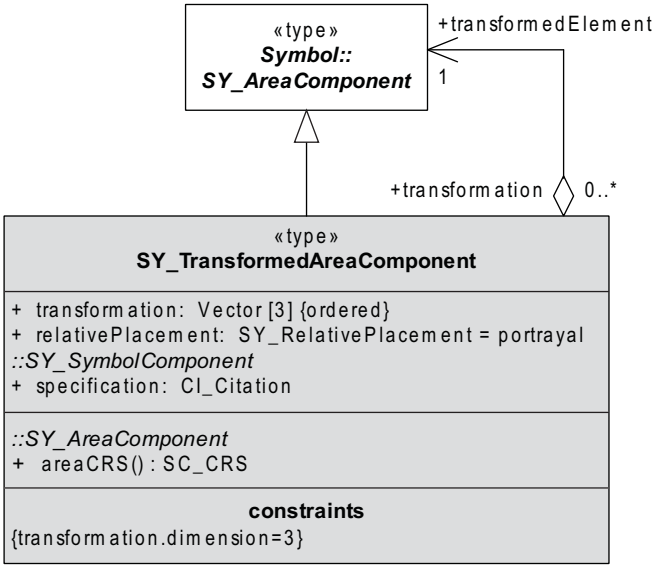

Figure 77 — SY_TransformedAreaComponent context diagram

9.4.8.2 Generalization – SY_AreaComponent

SY_TransformedAreaComponent specializes *SY_AreaComponent* as a geometric transformation of an area component and shall implement all inherited attributes, operations and associations.

9.4.8.3 Aggregation role – transformedElement

The aggregation role *transformedElement* defines the area component that is transformed.

```
SY_TransformedAreaComponent::transformedElement[1] : SY_AreaComponent
```

9.4.8.4 Attribute – transformation

The attribute *transformation* uses homogeneous coordinates to define a geometric transformation of an area component.

```
SY_TransformedAreaComponent::transformation : Vector[3]{ordered}
```

9.4.8.5 Attribute – relativePlacement

The attribute *relativePlacement* specifies whether a transformation is relative to the containing symbol component, relative to the portrayal coordinate reference system, or whether the attribute is not applicable. The default value of this attribute is *portrayal*.

```
SY_TransformedAreaComponent::relativePlacement : SY_RelativePlacement =
     portrayal
```

9.5 Package – Complex Symbol Extension

9.5.1 Package semantics

The Complex Symbol Extension adds the capability to create more complex symbols by using various types of components when composing symbols.

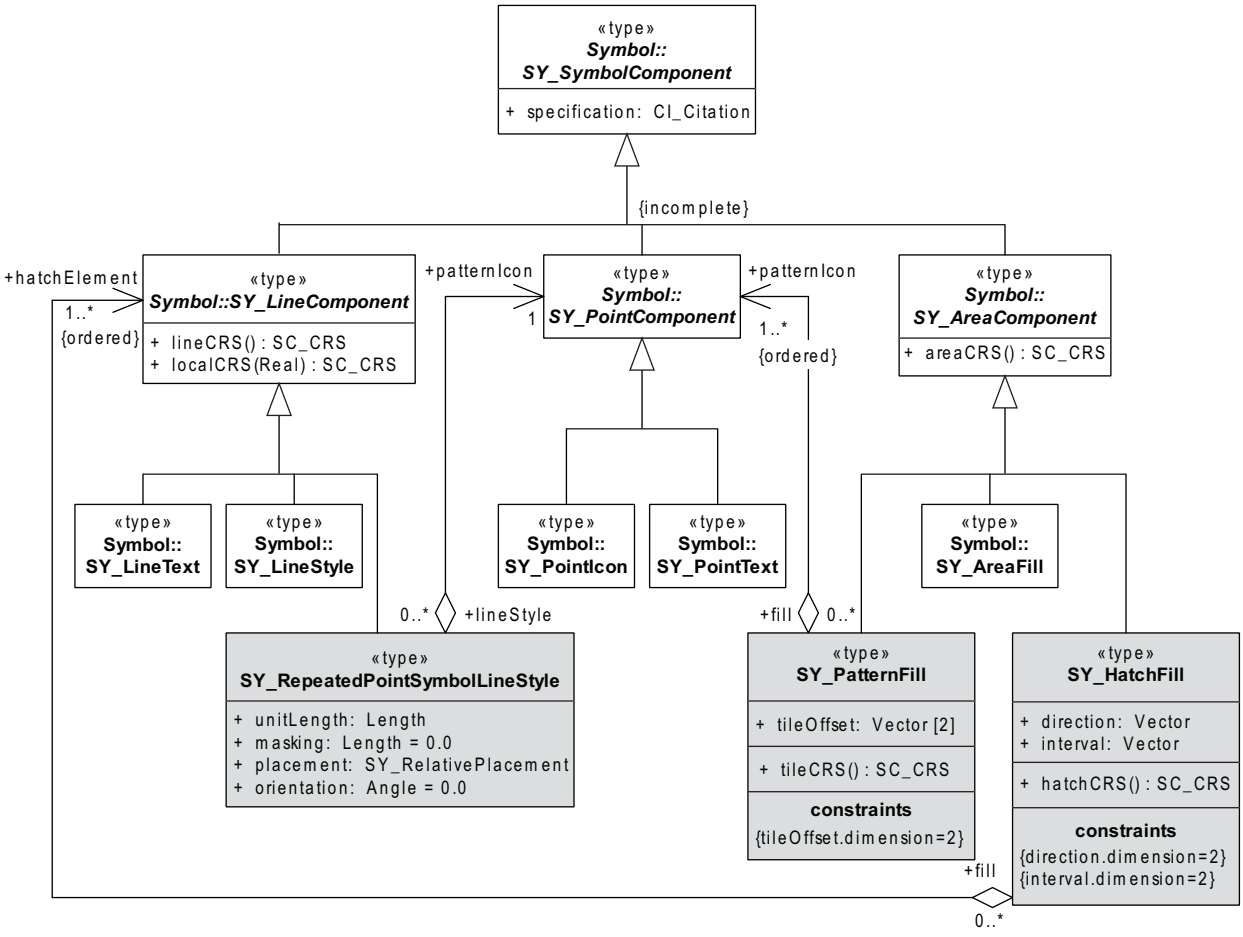

Figure 78 — Symbol – Complex [Component]

9.5.2 Type – SY_RepeatedPointSymbolLineStyle

9.5.2.1 Class semantics

SY_RepeatedPointSymbolLineStyle specializes *SY_LineComponent* as a pattern of repeated point components along the path of the line component. The point component is repeated at regular intervals and may have a masking, which masks out the underlying line components around the point component. For the point component not to be rendered at the unit length intervals but offset, a line component transformation may be used.

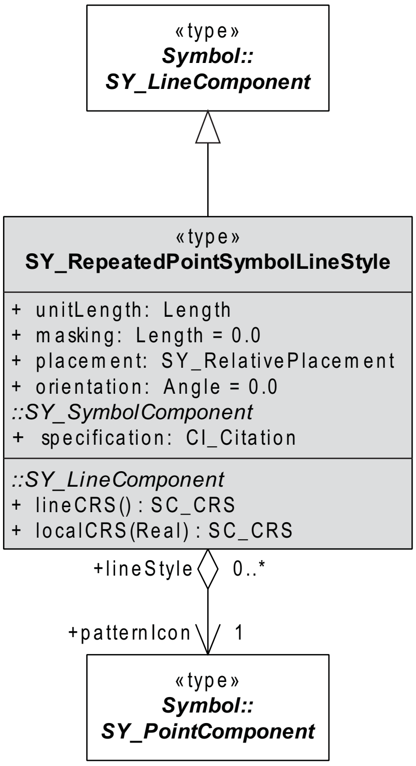

Figure 79 — SY_RepeatedPointSymbolLineStyle context diagram

9.5.2.2 Generalization – SY_LineComponent

SY_RepeatedPointSymbolLineStyle specializes *SY_LineComponent* as a pattern of repeated point components along the path of the containing line component and shall implement all inherited attributes, operations and associations.

9.5.2.3 Aggregation role – patternIcon

The aggregation role *patternIcon* defines the repeated point component of the containing line component.

```
SY_RepeatedPointSymbolLineStyle::patternIcon[1] : SY_PointComponent
```

9.5.2.4 Attribute – unitLength

The attribute *unitLength* specifies the length of the repetition interval for the point components along the path of the line component. The interval is given in the coordinate reference system of the line component (operation lineCRS).

```
SY_RepeatedPointSymbolLineStyle::unitLength : Length
```

9.5.2.5 Attribute – masking

The attribute *masking* specifies the extent of the masking border, which masks out the underlying line components around the point component. The default value of this attribute is 0.0.

```
SY_RepeatedPointSymbolLineStyle::masking : Length = 0.0
```

9.5.2.6 Attribute – placement

The attribute *placement* specifies whether the orientation is relative to the local coordinate reference system at the current location along the curve of the containing line component, relative to the portrayal coordinate reference system, or whether the attribute is not applicable.

```
SY_RepeatedPointSymbolLineStyle::placement : SY_RelativePlacement
```

9.5.2.7 Attribute – orientation

The attribute *orientation* specifies the orientation of the point component which may be relative to the curve of the containing line component, or relative to the portrayal, depending on the placement attribute. The default value of this attribute is 0.0.

```
SY_RepeatedPointSymbolLineStyle::orientation : Length = 0.0
```

9.5.3 Type – SY_HatchFill

9.5.3.1 Class semantics

SY_HatchFill specializes *SY_AreaComponent* as a hatch filled area component created by filling an area with repeated application of a line style, applied in a specified direction, spaced evenly over a surface, spaced with a specified interval.

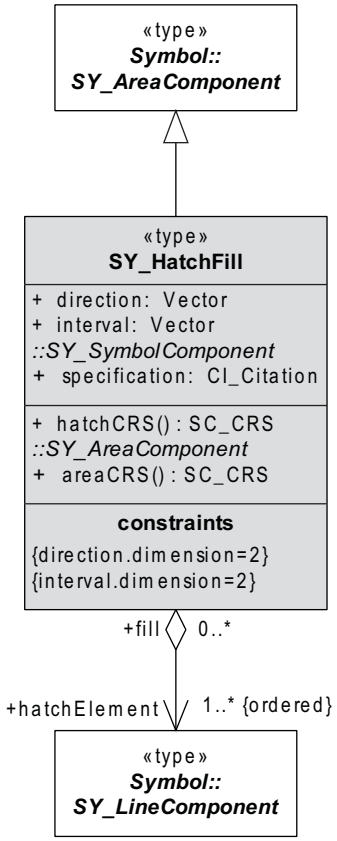

Figure 80 — SY_HatchFill context diagram

9.5.3.2 Generalization – SY_AreaComponent

SY_HatchFill specializes *SY_AreaComponent* as a hatch filled area component and shall implement all inherited attributes, operations and associations.

9.5.3.3 Aggregation role – hatchElement

The aggregation role *hatchElement* defines the line component used in the containing area component.

```
SY_HatchFill::hatchElement[1..*] : SY_LineComponent
```

9.5.3.4 Attribute – direction

The attribute *direction* specifies the direction of the hatch lines.

```
SY_HatchFill::direction : Vector
```

9.5.3.5 Attribute – interval

The attribute *interval* specifies the interval between the hatch lines.

```
SY_HatchFill::interval : Vector
```

9.5.3.6 Operation – hatchCRS

The operation *hatchCRS* returns a one dimensional coordinate reference system that is defined along the path of a hatch line.

```
SY_HatchFill::hatchCRS(
  ) : SC_CRS
```

9.5.4 Type – SY_PatternFill

9.5.4.1 Class semantics

SY_PatternFill specializes *SY_AreaComponent* as an area fill component created by filling an area with repeated application of point component tiles. The tiles are applied regularly at intervals defined by two vectors.

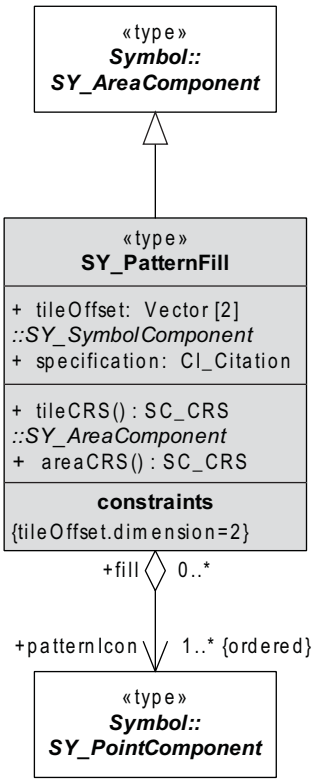

Figure 81 — SY_PatternFill context diagram

9.5.4.2 Generalization – SY_AreaComponent

SY_PatternFill specializes *SY_AreaComponent* as an area fill component and shall implement all inherited attributes, operations and associations.

9.5.4.3 Aggregation role – patternIcon

The aggregation role *patternIcon* defines the point components used in the tiles of a containing area component. The point components are ordered for display. and placed within the tile coordinate reference system.

```
SY_PatternFill::patternIcon[1..*] : SY_PointComponent
```

9.5.4.4 Attribute – tileOffset

The attribute *tileOffset* specifies two vectors that describe the offsets of the tiles in two directions.

```
SY_PatternFill::tileOffset : Vector[2]
```

9.5.4.5 Operation – tileCRS

The operation *tileCRS* returns the coordinate reference system for a tile of the area component.

```
SY_PatternFill::tileCRS(
  ) : SC_CRS
```

9.6 Package – Reusable Symbol Component Extension

9.6.1 Package semantics

The Reusable Symbol Component Extension adds the capability to use reusable symbol components when composing a symbol.

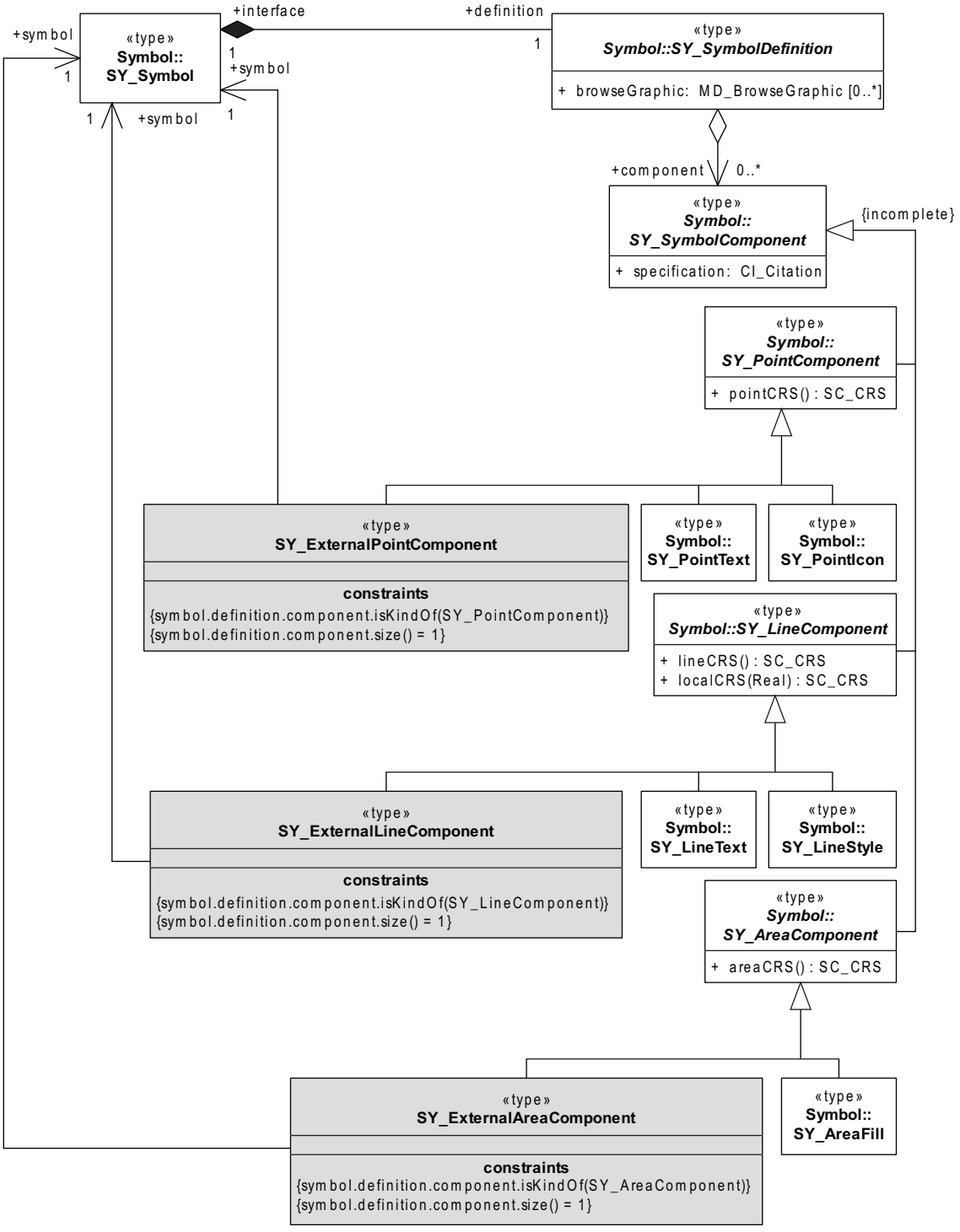

Figure 82 — Symbol – Reusable

9.6.2 Type – SY_ExternalPointComponent

9.6.2.1 Class semantics

SY_ExternalPointComponent specializes *SY_PointComponent* as a point symbol used as a component of another symbol. This allows symbols to be assembled from shared reusable point component symbols.

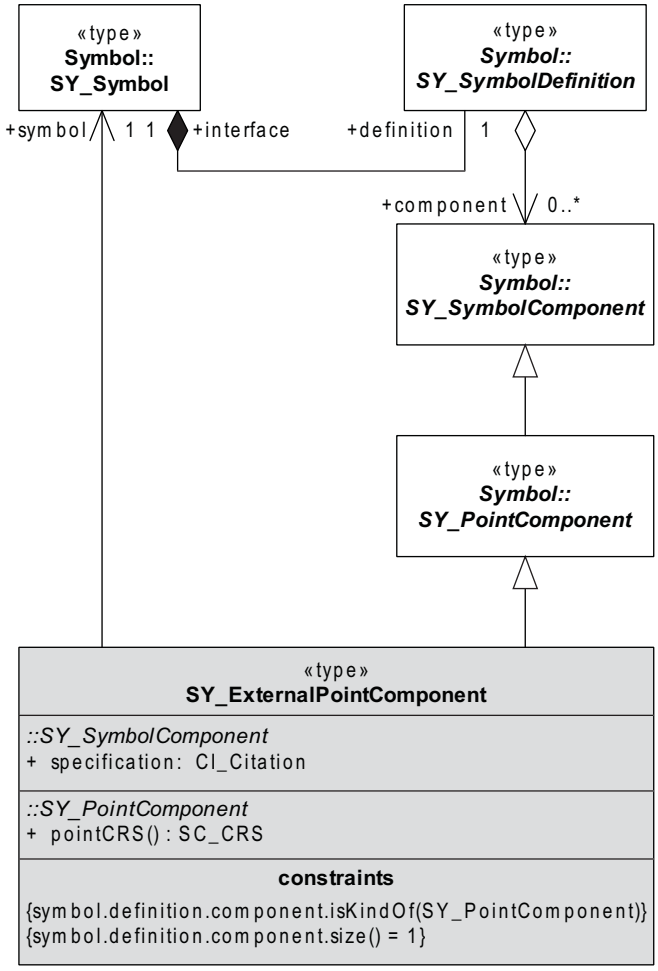

Figure 83 — SY_ExternalPointComponent context diagram

9.6.2.2 Generalization – SY_PointComponent

SY_ExternalPointComponent specializes *SY_PointComponent* as a reusable point component symbol and shall implement all inherited attributes, operations and associations.

9.6.2.3 Association role – symbol

The association role *symbol* associates the point component with the point symbol that functions as a point component.

```
SY_ExternalPointComponent::symbol[1] : SY_Symbol
```

9.6.3 Type – SY_ExternalLineComponent

9.6.3.1 Class semantics

SY_ExternalLineComponent specializes *SY_LineComponent* as a line symbol used as a component of another symbol. This allows symbols to be assembled from shared reusable line component symbols.

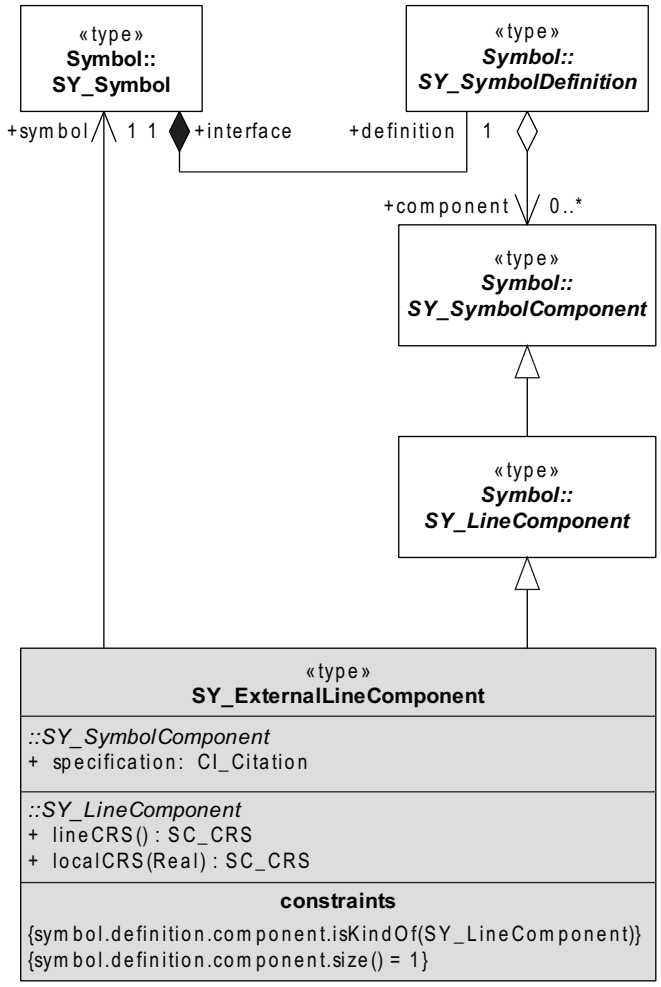

Figure 84 — SY_ExternalLineComponent context diagram

9.6.3.2 Generalization – SY_LineComponent

SY_ExternalLineComponent specializes *SY_LineComponent* as a reusable line component symbol and shall implement all inherited attributes, operations and associations.

9.6.3.3 Association role – symbol

The association role *symbol* associates the line component with the line symbol that functions as a line component.

```
SY_ExternalLineComponent::symbol[1] : SY_Symbol
```

9.6.4 Type – SY_ExternalAreaComponent

9.6.4.1 Class semantics

SY_ExternalAreaComponent specializes *SY_AreaComponent* as an area symbol used as a component of another symbol. This allows symbols to be assembled from shared reusable area component symbols.

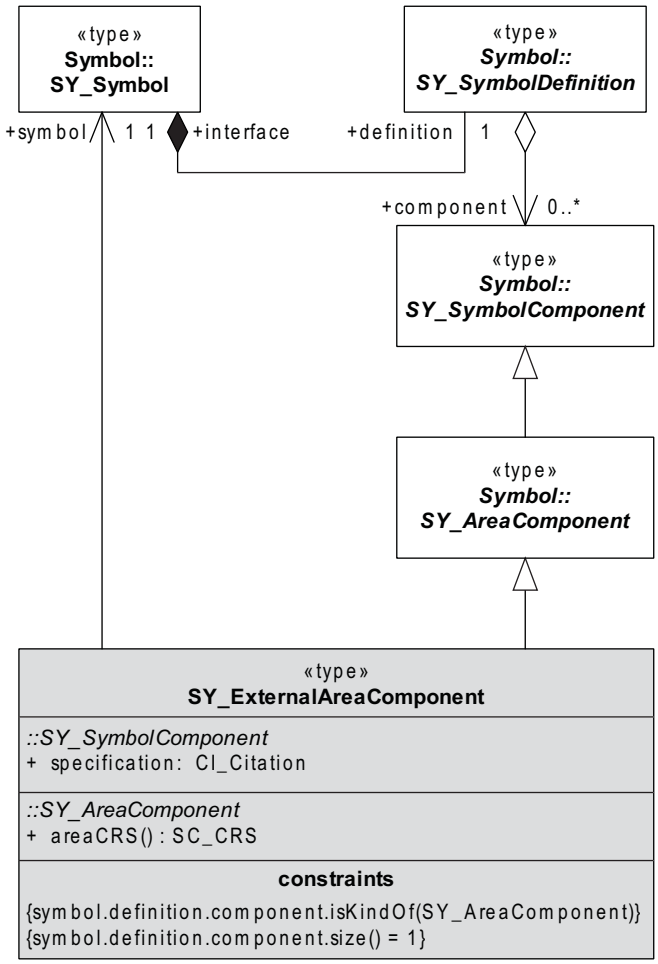

Figure 85 — SY_ExternalAreaComponent context diagram

9.6.4.2 Generalization – SY_AreaComponent

SY_ExternalAreaComponent specializes *SY_AreaComponent* as a reusable area component symbol and shall implement all inherited attributes, operations and associations.

9.6.4.3 Association role – symbol

The association role *symbol* associates the area component with the area symbol that functions as an area component.

```
SY_ExternalAreaComponent::symbol[1] : SY_Symbol
```

9.7 Package – Symbol Parameter Extension

9.7.1 Package semantics

The Symbol Parameter Extension package adds the capability to parameterize symbols, augmenting the reusability of symbol components. Parameters can be values that are meaningful within the graphic definition of a symbol such as colour or size. A parameter can be used to define the colour of part or all of a symbol instance, or used to define the size of a symbol instance. Parameters can also be symbols. This can be used to compose a symbol from parts depending on the attribution of the feature.

Figure 86 shows a simple example of a parameterized compound symbol. The point symbol is composed of a point icon, in this case a filled circle, and a parameterized text. Figure 86 shows this in an ad hoc textual representation of the symbol definition, lower left, while Figure 87 is a graphical depiction. The lower right item in Figure 86 is a textual representation of a symbol instance and Figure 88 shows this in context.

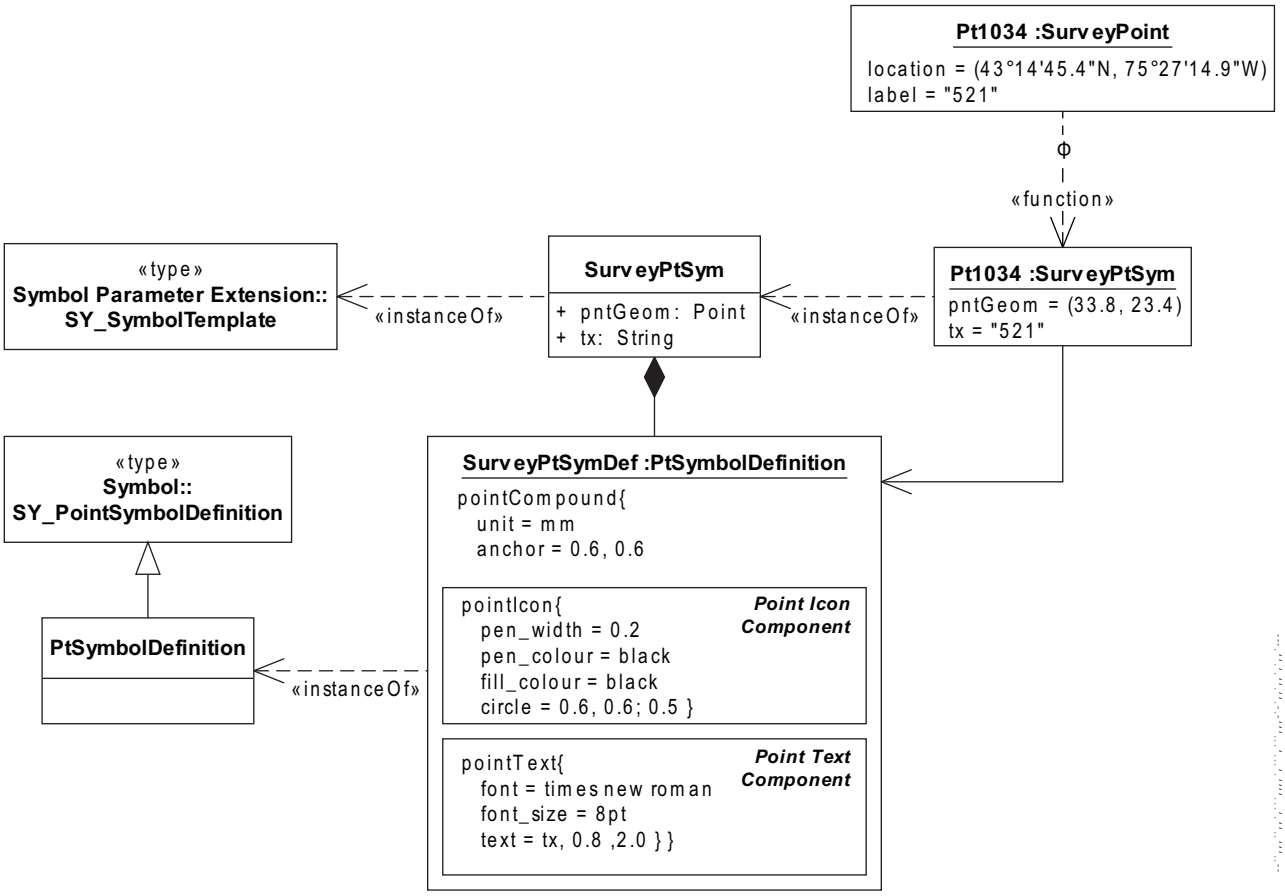

Figure 86 — Example of a parameterized symbol

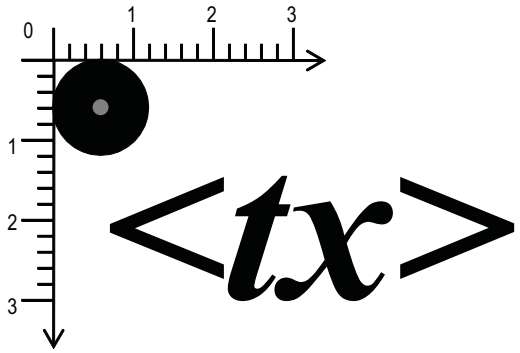

Figure 87 — Parameterized symbol definition

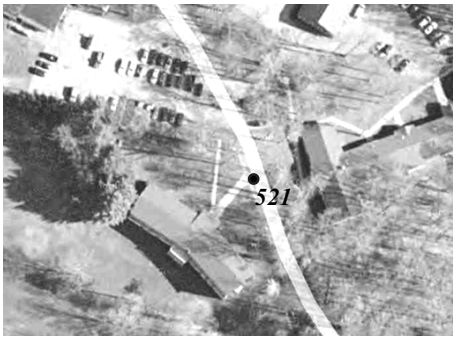

Figure 88 — Parameterized symbol instance

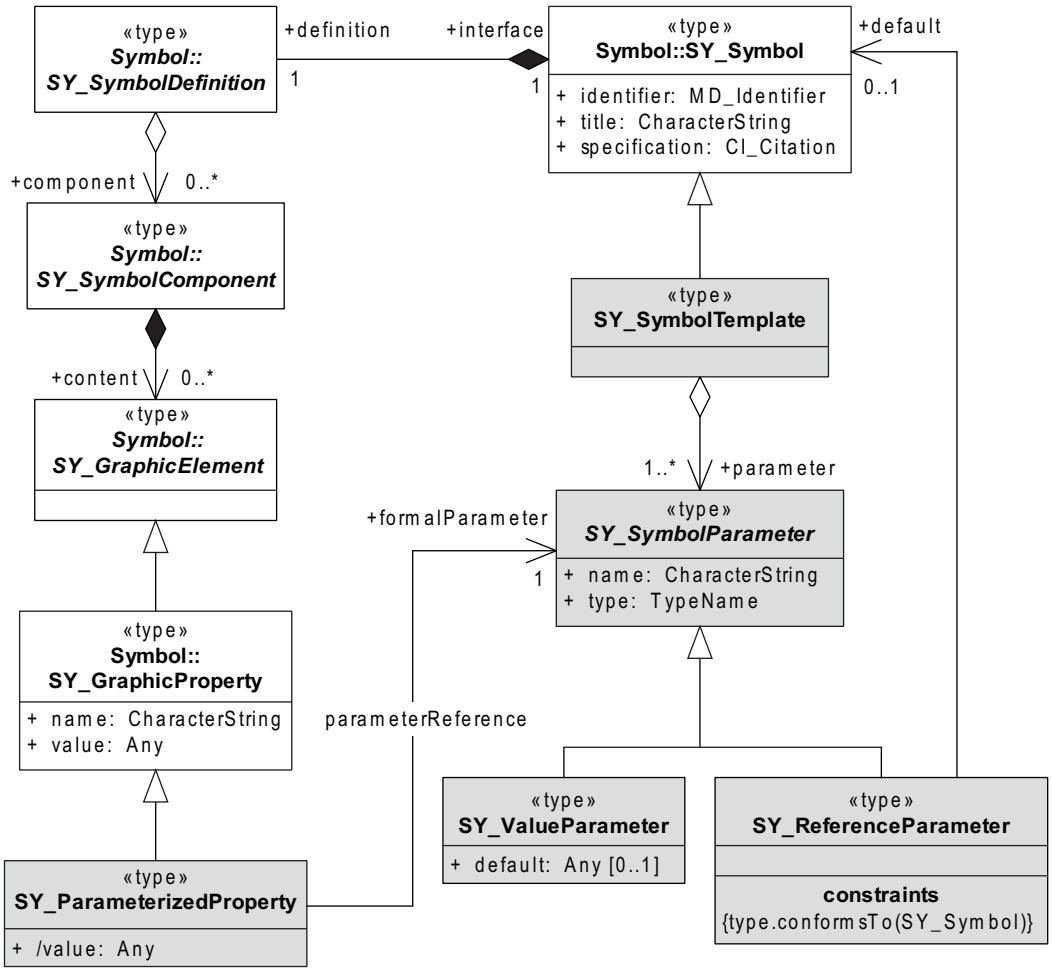

Figure 89 — Symbol – Parameter

9.7.2 Type – SY_SymbolTemplate

9.7.2.1 Class semantics

SY_SymbolTemplate specializes *SY_Symbol* as a type used to define parameterized symbols. The attributes and association of symbol classes defined using *SY_ParameterizedSymbol* are parameters to the associated symbol definitions. Geometry and text attributes are special cases of symbol parameters: geometry, because it is required for any portrayal, and text, because after geometry it is so essential to any labelling applied to a portrayal.

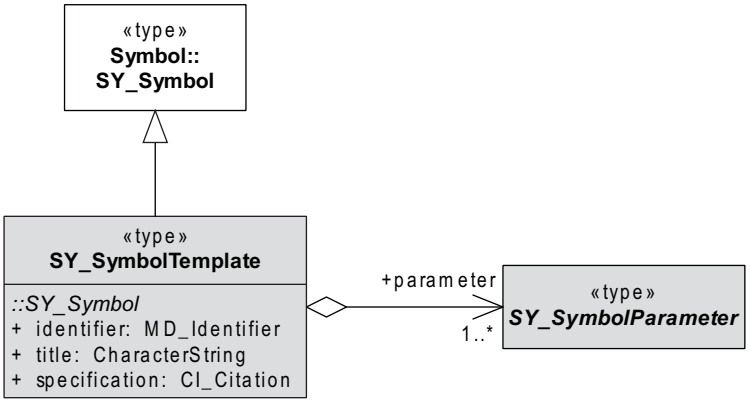

Figure 90 — SY_SymbolTemplate context diagram

9.7.2.2 Generalization – SY_Symbol

SY_SymbolTemplate specializes *SY_Symbol* as a type used to define parameterized symbols and shall implement all inherited attributes, operations and associations.

9.7.2.3 Aggregation role – parameter

The aggregation role *parameter* specifies the parameters of a symbol.

```
SY_SymbolTemplate::parameter[1..*] : SY_SymbolParameter
```

9.7.3 Type – SY_SymbolParameter

9.7.3.1 Class semantics

SY_SymbolParameter is the abstract root type for defining properties, both attributes and associations, of symbols. A symbol property has an identifier.

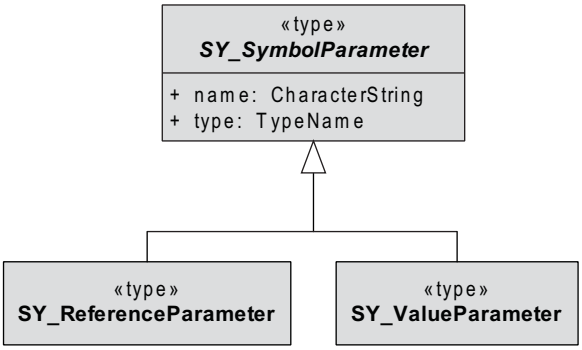

Figure 91 — SY_SymbolParameter context diagram

9.7.3.2 Attribute – name

The attribute *name* defines the name of the symbol parameter.

```
SY_SymbolParameter::name : CharacterString
```

9.7.3.3 Attribute – type

The attribute *type* specifies the type of the symbol parameter.

```
SY_SymbolParameter::type : TypeName
```

9.7.4 Type – SY_ValueParameter

9.7.4.1 Class semantics

SY_ValueParameter specializes the abstract root type *SY_SymbolParameter* as a value parameter. A symbol attribute has, in addition to an inherited identifier, a type and a default value.

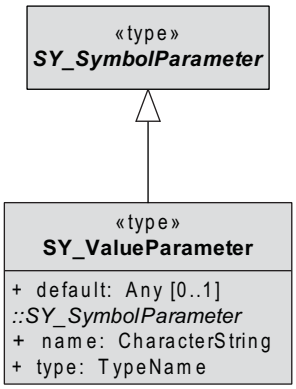

Figure 92 — SY_ValueParameter context diagram

9.7.4.2 Generalization – SY_SymbolParameter

SY_ValueParameter specializes the abstract root type *SY_SymbolParameter* as a value parameter and shall implement all inherited attributes, operations and associations.

9.7.4.3 Attribute – default

The optional attribute *default* specifies a default value for a symbol attribute.

```
SY_ValueParameter::default : Any[0..1]
```

9.7.5 Type – SY_ReferenceParameter

9.7.5.1 Class semantics

SY_ReferenceParameter specializes the abstract root type *SY_SymbolParameter* as an association role of a symbol. A symbol association role has, in addition to an inherited identifier, an association to a default symbol.

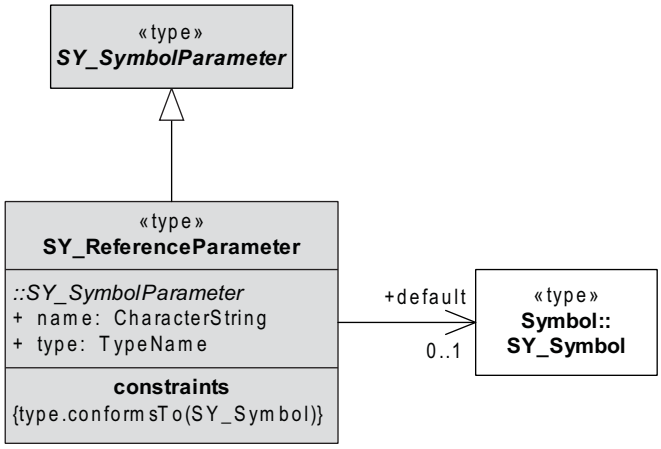

Figure 93 — SY_ReferenceParameter context diagram

9.7.5.2 Generalization – SY_SymbolParameter

SY_ReferenceParameter specializes the abstract root type *SY_SymbolParameter* as an association role of a symbol and shall implement all inherited attributes, operations and associations. The type of the reference parameter is always a symbol.

9.7.5.3 Association role – default

The optional association role *default* associates the symbol reference parameter with a default symbol value.

```
SY_ReferenceParameter::default[0..1] : SY_Symbol
```

9.7.6 Type – SY_ParameterizedProperty

9.7.6.1 Class semantics

The *SY_ParameterizedProperty* class template specializes the *SY_GraphicProperty* class template as a graphic property whose value is derived from the property value of a symbol instance. The parameterized property derives its reference to the property value from a reference to the symbol property element of a symbol. A parameterized property allows, for example, text strings to be passed to symbol instances for labelling with variable texts.

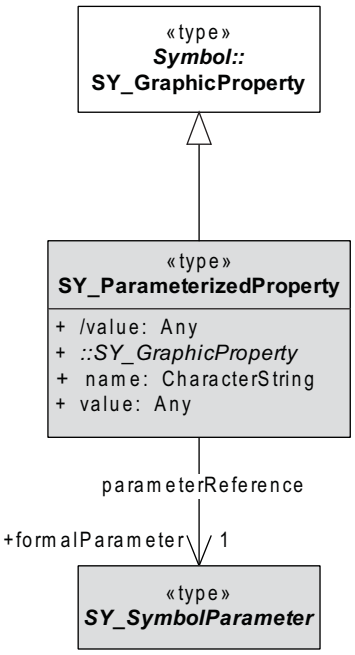

Figure 94 — SY_ParameterizedProperty context diagram

9.7.6.2 Generalization – SY_GraphicProperty

The *SY_ParameterizedProperty* class template specializes the *SY_GraphicProperty* class template as a graphic property whose value is derived from the property value of a symbol instance. As such it shall implement all inherited attributes, operations and associations.

9.7.6.3 Association role – formalParameter

The association role *formalParameter* associates the parameterized property with its formal definition.

```
SY_ParameterizedProperty::formalParameter[1] : SY_SymbolParameter
```

9.7.6.4 Attribute – value

The derived attribute *value* is derived from the actual value of the property of the symbol instance.

```
SY_ParameterizedProperty::value : Any
```

9.8 Package – Function Symbol Parameter Extension

9.8.1 Package semantics

The Function Symbol Parameter Extension package adds the ability to reference parameterized symbols from portrayal functions.

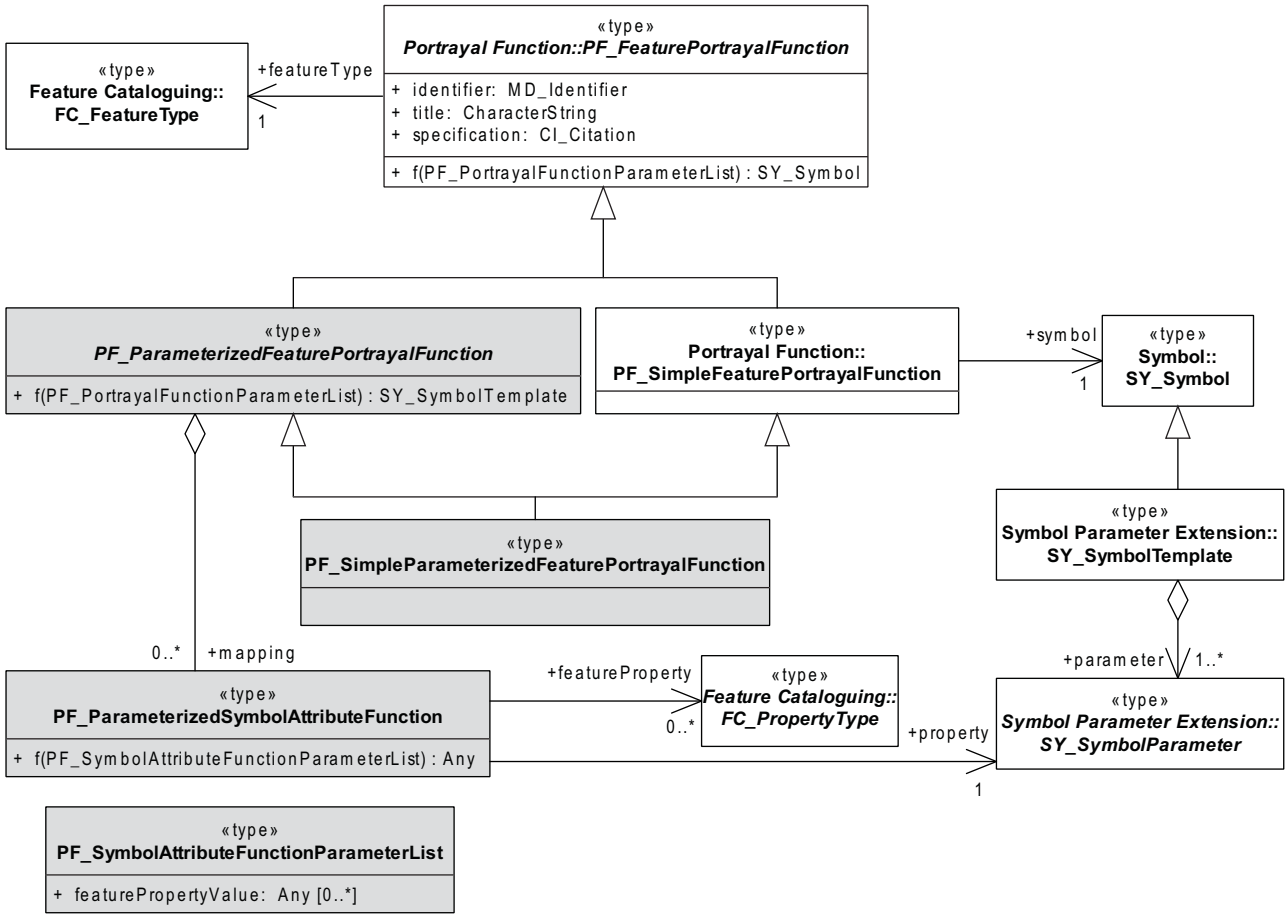

Figure 95 — Portrayal Function – Parameter

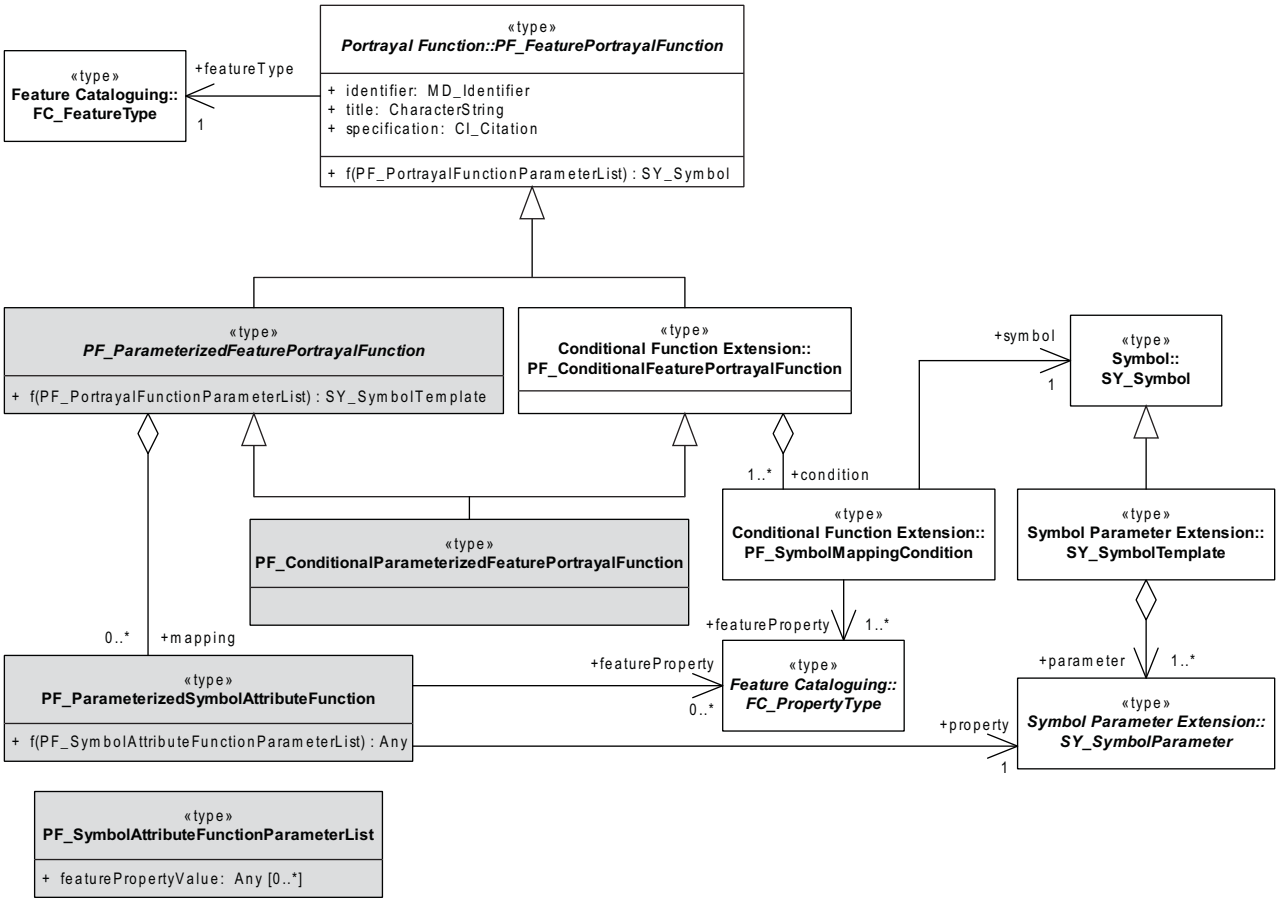

Figure 96 — Portrayal Function – Parameter Conditional

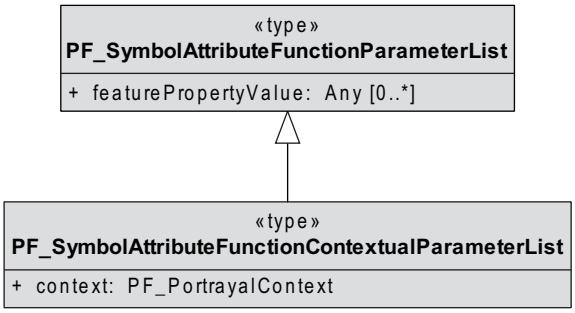

Figure 97 — Portrayal Function – Parameter Context

9.8.2 Type – PF_ParameterizedFeaturePortrayalFunction

9.8.2.1 Class semantics

PF_ParameterizedFeaturePortrayalFunction specializes *PF_FeaturePortrayalFunction* as an abstract feature portrayal function whose mapping function evaluates to a parameterized symbol. The individual parameter values are derived from the feature attributes through parameterized symbol attribute functions.

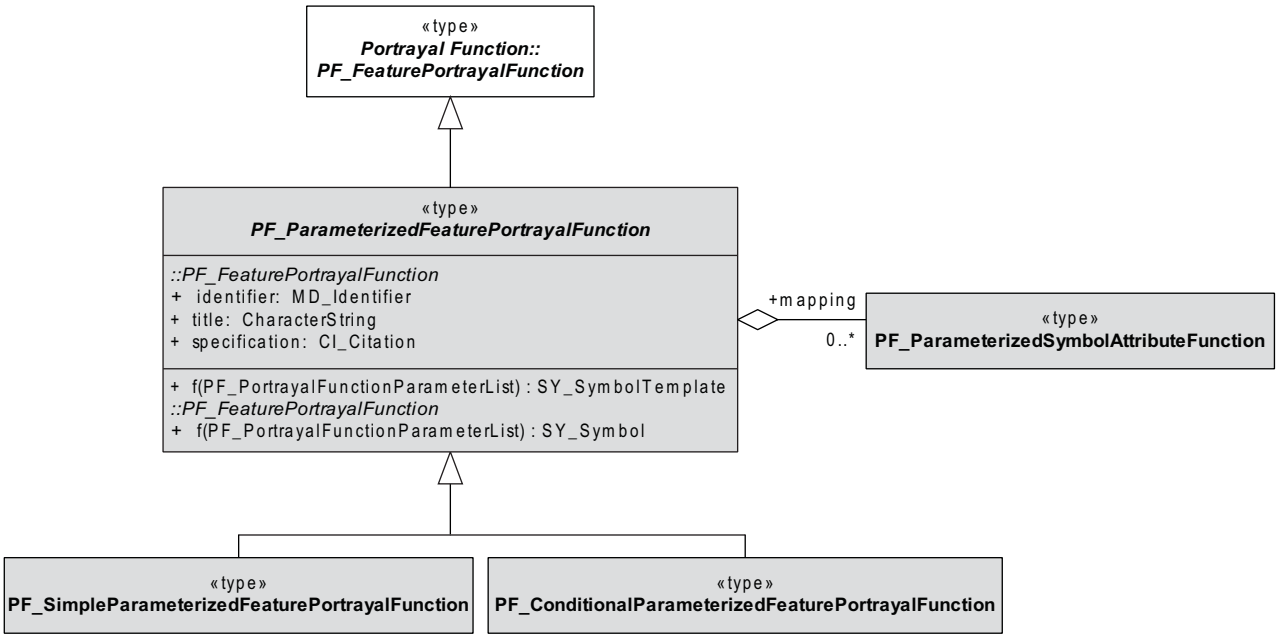

Figure 98 — PF_ParameterizedFeaturePortrayalFunction context diagram

9.8.2.2 Generalization – PF_FeaturePortrayalFunction

PF_ParameterizedFeaturePortrayalFunction specializes *PF_FeaturePortrayalFunction* as a feature portrayal function whose mapping function evaluates to a parameterized symbol. As such it shall implement all inherited attributes, operations and associations.

9.8.2.3 Aggregation role – mapping

The aggregation role *mapping* collects the parameterized symbol attribute functions that make up the individual attribute mappings of the feature portrayal function.

```
PF_ParameterizedFeaturePortrayalFunction::mapping[0..*] :
     PF_ParameterizedSymbolAttributeFunction
```

9.8.2.4 Operation – f

The operation *f* maps the feature in the parameter list to a symbol template. The feature parameter is of the associated feature type and the symbol template is an instance of the associated symbol template type.

```
PF_ParameterizedFeaturePortrayalFunction::f(
     parameterList : PF_PortrayalFunctionParameterList
  ) : SY_SymbolTemplate
```

9.8.3 Type – PF_ParameterizedSymbolAttributeFunction

9.8.3.1 Class semantics

PF_ParameterizedSymbolAttributeFunction describes a function, which maps properties of a feature to a value. Parameterized symbol attribute functions are collected in a parameterized feature portrayal function and are used to populate the actual parameter values of a symbol reference.

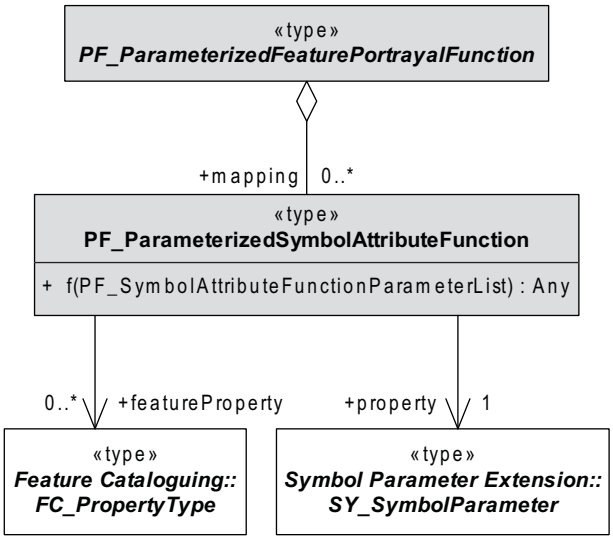

Figure 99 — PF_ParameterizedSymbolAttributeFunction context diagram

9.8.3.2 Association role – featureProperty

The association role *featureProperty* specifies the feature properties whose values are inputs to the symbol attribute function.

```
PF_ParameterizedSymbolAttributeFunction::featureProperty[0..*] :
     FC_PropertyType
```

9.8.3.3 Association role – property

The association role *property* specifies the property of a parameterized symbol that the function populates.

```
PF_ParameterizedSymbolAttributeFunction::property[1] : SY_SymbolParameter
```

9.8.3.4 Operation – f

The operation *f* maps the feature properties in the parameter list to an instance of any type. This includes symbols.

```
PF_ParameterizedSymbolAttributeFunction::f(
     parameterList : PF_SymbolAttributeFunctionParameterList
  ) : Any
```

9.8.4 Type – PF_SimpleParameterizedFeaturePortrayalFunction

9.8.4.1 Class semantics

PF_SimpleParameterizedFeaturePortrayalFunction specializes both *PF_SimpleFeaturePortrayalFunction* and *PF_ParameterizedFeaturePortrayalFunction*, combining the parameterization functionality of *PF_ParameterizedFeaturePortrayalFunction* and the simple function functionality of *PF_SimpleFeaturePortrayalFunction*.

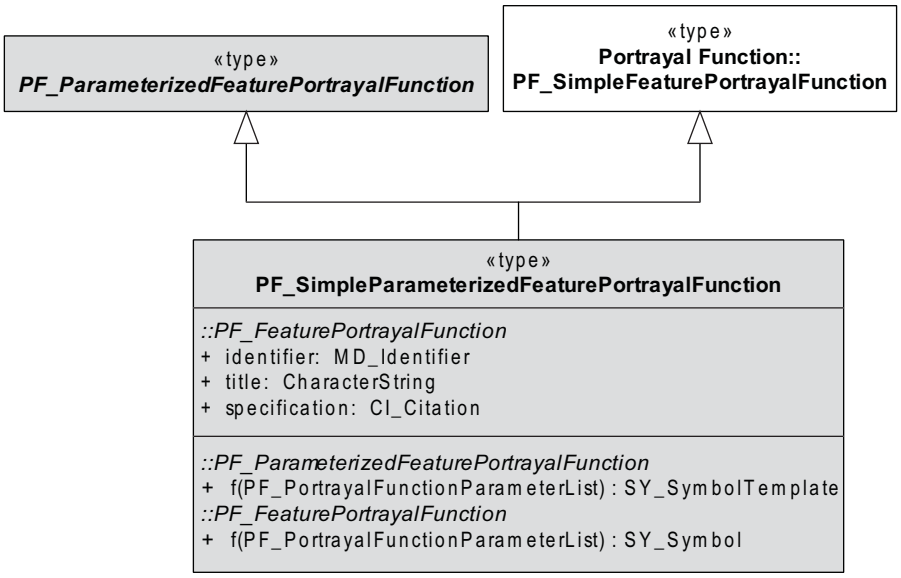

Figure 100 — PF_SimpleParameterizedFeaturePortrayalFunction context diagram

9.8.4.2 Generalization – PF_SimpleFeaturePortrayalFunction

PF_SimpleParameterizedFeaturePortrayalFunction specializes *PF_SimpleFeaturePortrayalFunction* as a feature portrayal function whose mapping function evaluates to a parameterized symbol. As such it shall implement all inherited attributes, operations and associations.

9.8.4.3 Generalization – PF_ParameterizedFeaturePortrayalFunction

PF_SimpleParameterizedFeaturePortrayalFunction specializes *PF_ParameterizedFeaturePortrayalFunction* as a basic parameterized feature portrayal function, which maps features to symbols directly. As such it shall implement all inherited attributes, operations and associations.

9.8.5 Type – PF_ConditionalParameterizedFeaturePortrayalFunction

9.8.5.1 Class semantics

PF_ConditionalParameterizedFeaturePortrayalFunction specializes both *PF_ConditionalFeaturePortrayalFunction* and *PF_ParameterizedFeaturePortrayalFunction*, combining the parameterization functionality of *PF_ParameterizedFeaturePortrayalFunction* and the conditional function functionality of *PF_ParameterizedFeaturePortrayalFunction*.

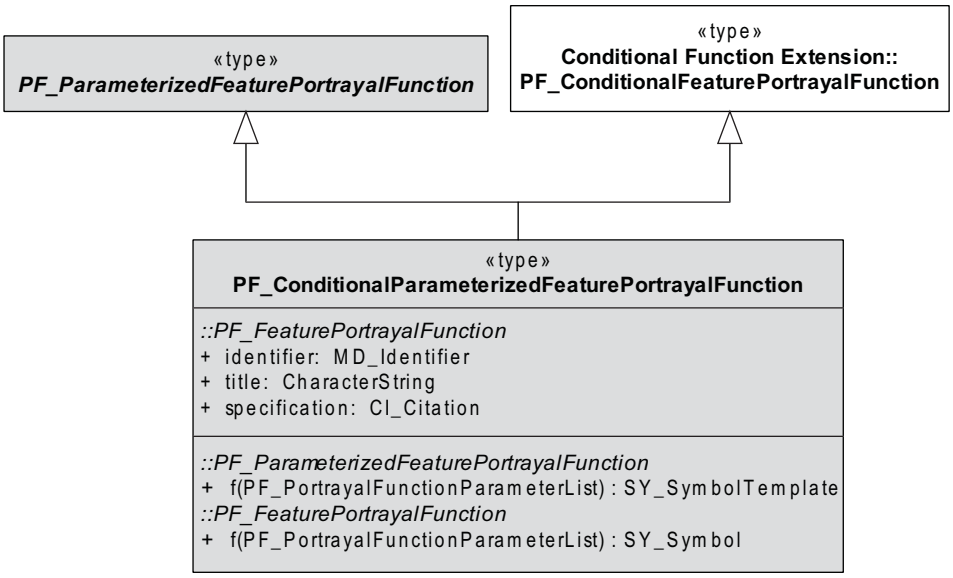

Figure 101 — PF_ConditionalParameterizedFeaturePortrayalFunction context diagram

9.8.5.2 Generalization – PF_ConditionalFeaturePortrayalFunction

PF_ConditionalParameterizedFeaturePortrayalFunction specializes *PF_ConditionalFeaturePortrayalFunction* as conditional feature portrayal function whose mapping function evaluates to a parameterized symbol. As such it shall implement all inherited attributes, operations and associations.

9.8.5.3 Generalization – PF_ParameterizedFeaturePortrayalFunction

PF_ConditionalParameterizedFeaturePortrayalFunction specializes *PF_ParameterizedFeaturePortrayalFunction* as a parameterized feature portrayal function which maps to different parameterized symbols depending on associated symbol mapping conditions. As such it shall implement all inherited attributes, operations and associations.

9.8.6 Type – PF_SymbolAttributeFunctionParameterList

9.8.6.1 Class semantics

PF_SymbolAttributeFunctionParameterList collects the feature property values that are inputs to the symbol attribute function.

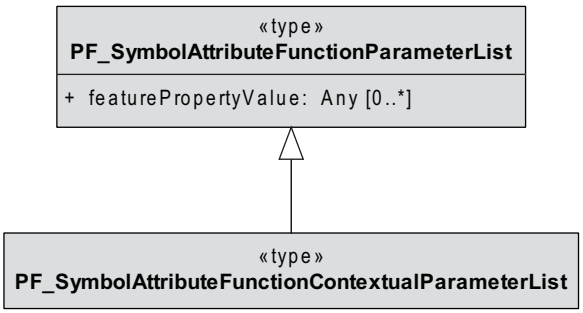

Figure 102 — PF_SymbolAttributeFunctionParameterList context diagram

9.8.6.2 Attribute – featurePropertyValue

The attribute *featurePropertyValue* defines the collection of feature property values that are inputs to the symbol attribute function.

```
PF_SymbolAttributeFunctionParameterList::featurePropertyValue : Any[0..*]
```

9.8.7 Type – PF_SymbolAttributeFunctionContextualParameterList

9.8.7.1 Class semantics

PF_SymbolAttributeFunctionContextualParameterList specializes *PF_SymbolAttributeFunctionParameterList* as an attribute function parameter list that include the portrayal context.

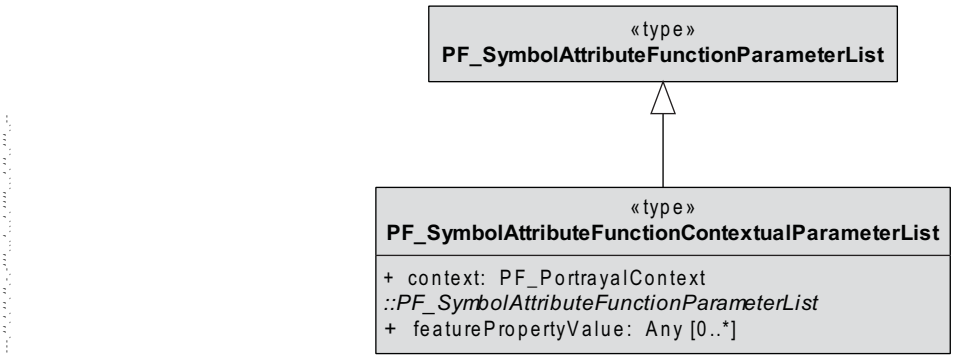

Figure 103 — PF_SymbolAttributeFunctionContextualParameterList context diagram

9.8.7.2 Generalization – PF_SymbolAttributeFunctionParameterList

PF_SymbolAttributeFunctionContextualParameterList specializes *PF_SymbolAttributeFunctionParameterList* as an attribute function parameter list that include the portrayal context. As such it shall implement all inherited attributes, operations and associations.

9.8.7.3 Attribute – context

The attribute *context* contains the context information of the attribute function parameter list.

```
PF_SymbolAttributeFunctionContextualParameterList::context :
     PF_PortrayalContext
```

10 Basic implementation package

10.1 Package – Feature Data Model

10.1.1 Package semantics

In addition to the internal and external class and package dependencies shown in Figure 6, a generic feature class is required as a parameter to a portrayal function (PF_PortrayalFunctionParameterList). For this purpose the abstract type FR_Feature is defined in this package. This class mimics the FD_Feature class from the Feature Data Model package of ISO 19133 and the GFI_Feature class from the General Feature Instance package of ISO 19156.

Figure 104 — Feature Data Model

10.1.2 Type – FR_Feature

The class FR_Feature is an instance of the «metaclass» GF_FeatureType (ISO 19109:2005). It is the abstract root off all feature type instances of GF_FeatureType.

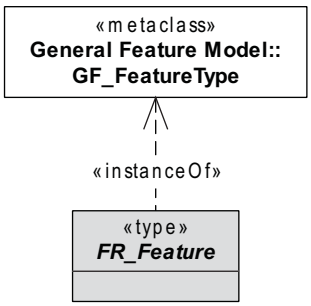

Figure 105 — FR_Feature context diagram

Annex A
(normative)

Abstract test suite

A.1 Portrayal core

A.1.1 Portrayal core – Separation of data from portrayal (general)

a) Test purpose: to verify that symbols and portrayal functions are not a part of the dataset.

b) Test method: Examine portrayal data to ensure it is separate from the geospatial dataset.

c) Reference: 6.1

d) Test type: Basic test

A.1.2 Portrayal core – portrayal function

A.1.2.1 Application schema

a) Test purpose: To verify that information to be portrayed is defined in an application schema.

b) Test method: Verify that the geographic information to be portrayed has been defined in the application schema for the dataset, and that the portrayal function set selected to portray the data has been designed in accordance with that application schema.

c) Reference: 6.5

d) Test type: Basic test.

A.1.2.2 Portrayal function design

a) Test purpose: To verify that all portrayal function elements are included.

b) Test method: Verify that all mandatory portrayal function elements are present. Verify that all conditional portrayal function elements are present if the conditions apply.

c) Reference: Clause 8

d) Test type: Capabilities test.

A.1.2.3 Implementation of portrayal functions

a) Test purpose: To verify the application being tested implements the portrayal function capability.

b) Test method: Verify that the application implements a mapping functionality of Portrayal function set such that elements of a feature catalogue are mapped to symbols. Verify that the feature portrayal functions are specified for the feature classes they will be applied on. For a given feature, verify that the application implements a mapping functionality of Feature Portrayal Function such that a feature is mapped to a symbol.

c) Reference: 6.1

d) Test type: Capabilities test.

A.1.2.4 Portray nothing

a) Test purpose: To verify that features that are not intended to be portrayed are properly accounted for.

b) Test method: Verify that features that are not intended to be portrayed are mapped by feature portrayal functions to a symbol that has no components.

c) Reference: 6.3 and 8.3.5.2.

d) Test type: Capabilities test.

A.1.3 Portrayal core – symbol

A.1.3.1 Symbol design

a) Test purpose: To verify that all symbol elements are included.

b) Test method: Verify that all mandatory symbol elements are present. Verify that all conditional symbol elements are present if the conditions apply.

c) Reference: Clause 8.

d) Test type: Capabilities test.

A.1.3.2 Default symbol

a) Test purpose: To verify that a default symbol is present.

b) Test method: Verify that

- the default symbol is applied when no feature portrayal function is applicable for a feature instance;
- the default symbol is applied according to at least one of the spatial attributes of the feature;
- the provider of the portrayal function has specified the value;
- portrayal contextual information has not been used in the default symbol;
- if the application fails to portray the data, the failure is handled by the application.

c) Reference: 6.4.

d) Test type: Capabilities test.

A.1.4 Portrayal core – portrayal catalogue

A.1.4.1 Availability of portrayal information

a) Test purpose: To verify that portrayal information is present.

b) Test method: Verify that when portraying a geographic dataset, a portrayal catalogue, containing feature portrayal functions, and applicable symbols is present, or referenced from appropriate metadata.

c) Reference: 6.1

d) Test type: Basic test.

A.1.4.2 Portrayal catalogue

a) Test purpose: Verify that the portrayal functions and symbols are able to be interchanged by transfer in a portrayal catalogue.

b) Test method: Test that portrayal information provider can send and portrayal service provider can receive a portrayal catalogue. Verify that the portrayal catalogue contains the necessary portrayal function set and symbols.

c) Reference: Clause 6.

d) Test type: Capabilities test

A.2 Portrayal function extensions

A.2.1 Conditional function extension

a) Test purpose: To verify that when conditional feature portrayal functions are used, they map feature instances to symbols based on the properties of the feature.

b) Test method: Verify that all of the symbol mapping conditions adhere to the properties in the feature catalogue. Verify that each symbol mapping condition assigns a symbol.

c) Reference: 9.2

d) Test type: Capabilities test

A.2.2 Context extension

a) Test purpose: To verify that when context is used it is included in the portrayal catalogue and includes portrayal context specifications that describe the context.

b) Test method: Verify that the portrayal catalogue includes the contexts used, and each context has a description in the portrayal context specification. Verify that the Portrayal function set with Context includes the Portrayal Function Context Parameter List.

c) Reference: 9.3

d) Test type: Capabilities test.

A.2.3 Function symbol parameter extension

a) Test purpose: To verify that when parameterized symbols are used the feature portrayal functions provide the information necessary to pass to the parameterized symbol.

b) Test method: Verify that parameterized feature portrayal functions contain symbol attribute function parameter lists, and symbol parameter values. Verify that when conditional feature portrayal functions are used they also contain parameterized conditional feature portrayal functions.

c) Reference: 9.8

d) Test type: Capabilities test.

A.3 Symbol extensions

A.3.1 Compound symbol extension

a) Test purpose: To verify that when compound symbols are used the component graphics are defined and their spatial relationships to other components are defined.

b) Test method: Verify that each point component has a point CRS and transformed point symbol. Verify that each line component has a line CRS and local CRS, and transformed line style. Verify that each area component has an area CRS and Transformed area fill.

c) Reference: 9.4

d) Test type: Capabilities test.

A.3.2 Complex symbol extension

a) Test purpose: To verify that when complex symbols are used they have the applicable coordinate reference system and placement parameters present.

b) Test method: Verify that complex symbol line component has line CRS and local CRS. Verify that symbol area component has area CRS. Verify that repeated point symbol line style has required attribute values. Verify that pattern fill has tile CRS and tile offset attributes. Verify that hatch fill has hatch CRS and direction and interval attributes.

c) Reference: 9.5

d) Test type: Capabilities test.

A.3.3 Reusable symbol component extension

a) Test purpose: To verify that symbols that identify reusable components by reference have the component reference as part of their symbol definition.

b) Test method: Verify that point symbol components that are reusable have a point component reference. Verify that line symbol components that are reusable have a line component reference. Verify that area symbol components that are reusable have an area fill reference.

c) Reference: 9.6

d) Test type: Capabilities test.

A.3.4 Symbol parameter extension

a) Test purpose: To verify that when parameterized symbols are used they contain the symbol properties, attributes and roles in accordance with the parameterized symbol specification.

b) Test method: Verify that each parameterized symbol has one or more symbol properties, and that these symbol properties have symbol attributes. Verify that the actual symbol specification parameters are included as symbol properties. Verify that symbol association roles are defined.

c) Reference: 9.7

d) Test type: Capabilities test.

A.4 Portrayal catalogue extensions

None

Annex B
(informative)

Rules-based portrayal functions

B.1 Portrayal rules

This annex defines a feature-centred rule-based portrayal mechanism. Instances of features are portrayed based on rules, which make use of geometry and attribute information. The portrayal information is handled as symbol definitions applied according to specific portrayal rules.

The portrayal mechanism is illustrated by Figure B.1.

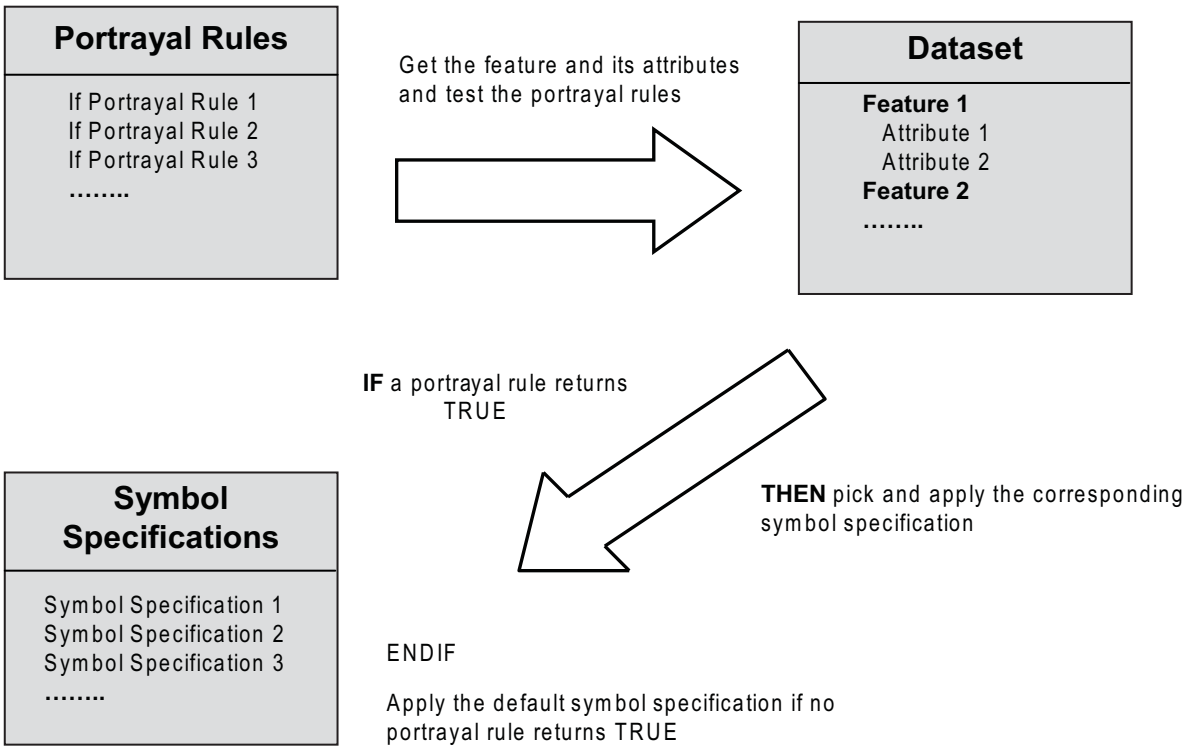

Figure B.1 — Portrayal mechanism without priority attributes

Portrayal rules are a specific type of portrayal function that is expressed in some declarative language (rule language with if/else, switch case, etc.) The portrayal rule mechanism may be used to handle portrayal issues that have to be solved as they happen, such as how to automatically place text on maps, and special representations of the feature instances according to, for example, time of day or scale. The value of context information, such as time of day or scale, may be included in the portrayal rules. The portrayal rules in the portrayal catalogue shall be tested on the attributes of the feature instances in the dataset. The portrayal rule shall be applied as a query statement that returns TRUE or FALSE. The symbol specification associated with that particular portrayal rule shall then be applied. If no portrayal rule returns TRUE then the default symbol specification shall be used. A portrayal service is used to portray a feature instance or instances. The portrayal service applies operations using the parameters defined in a symbol specification

B.2 Priority attribute

An optional priority attribute may be added to the portrayal rules. Portrayal rules can be processed in several different ways.

— All of the rules in a rule set can be evaluated, and if more than one TRUE rule is encountered, the priority attribute can be used to determine which symbol specification to apply. The attribute gives an integer value deciding in which order portrayal rules shall be applied if more than one returns TRUE for one feature instance. A portrayal rule with a high priority number (1 being the highest) takes precedence over one with a lower number. If two portrayal rules returning TRUE have the same priority value, then the application shall decide which one takes precedence. If priority attributes are used, all the portrayal rules shall have a priority attribute.

— Only one portrayal rule can be TRUE for a feature instance. When the first TRUE rule is encountered, the corresponding symbol specification is applied, and no additional rules need to be evaluated. No priority attribute is required.

— Multiple TRUE rule statements are allowed and apply multiple symbol specifications to the same feature instance to create a composite symbol. The symbol specifications shall define the positional and overprinting relationships between the component parts of the composite symbol. No priority attribute is required.

B.3 Examples

EXAMPLE 1 A dataset contains instances of the feature class Road. The feature class Road contains two attributes: classification and segment. The classification attribute is of string data type, and can have the value “country road” or “town road”. The segment attribute is of GM_Curve type and contains the spatial description of the road. The symbol specification used is called N50_specification. The two portrayal rules in this example look like this (“quotes” are used to show the contents of a string):

IF (Road.classification EQ “country road”) THEN drawCurve (“N50_specification.Solid_red_line”, Road.segment)

IF (Road.classification EQ “town road”) THEN drawCurve (“N50_specification.Solid_yellow_line”, Road.segment)

In this example, the THEN separates the query and action statements. The drawCurve is an action statement drawing an actual curve using geometry from Road.segment and colour, line width, etc. information from N50_specification.Solid_red_line and N50_specification.Solid_yellow_line.

EXAMPLE 2 If the portrayal varies with the scale, contextual information is needed as part of the query statement. One of the portrayal rules then can look like this (“quotes” are used to show the contents of a string):

IF (Road.classification EQ “country road” AND Scale (<=20000)) THEN drawCurve (“N50_specification.Solid_thin_red_line”, Road.segment)

Here Scale is context that gets the display scale from the display device. The portrayal rule refers to the appropriate attributes, functions and relationships defined in an application schema. The portrayal catalogue also lists the contexts used, including the parameters and returned values.

EXAMPLE 3 In the following cases, external functions are necessary.

— The electronic map in a car navigation system has to be displayed so that the up-direction of the map is always in the direction the car is moving. To be able to specify the rotation of the map, the current position of the car must be retrieved continuously from an external position device using context.

— For electronic chart displays onboard a vessel, some of the symbols are only valid for certain scale-intervals. To be able to turn the symbols on and off, the system must be told what scale the map is displayed in by the display part of the chart system. A danger zone is defined spatially as a surface. Below a certain scale the danger zone is better displayed by a point symbol. An external function can be used to compute the centroid of the area and the coordinates of the centroid used to position the point symbol.

— An external function can be used to avoid visual conflicts between text and symbols placed on a map, or to handle the placement of text along curves.

Annex C
(informative)

Enterprise view of portrayal

C.1 Introduction

The enterprise viewpoint in a Reference Model of Open Distributed Processing (RM-ODP [ISO/IEC 10746-3]) focuses on the purpose, scope, and policies of the system. The purpose is provided as the objective of the geographic portrayal community. The scope is defined through a high-level scenario described in section 1. The enterprise viewpoint provides context for the development of standards in other viewpoints.

C.2 Geographic portrayal community objective

The objective of the geographic portrayal community is to develop collections of symbols and portrayal functions that can be used to portray geographic datasets in a variety of contexts, and in support of a variety of applications, independent of the specific content of those datasets.

C.3 Geographic portrayal scope

The geographic portrayal process starts with geographic feature data, and uses portrayal functions to associate symbols with feature instances in a geographic feature dataset, and renders the symbolized features on a display medium. A symbol is most commonly graphic in nature, but other media (audio, tactile, etc.) can also be used to portray geographic data. The geographic features and their associated symbol definitions are used to render a map display.

The portrayal community consists of the collection of actors, operations, data, and metadata required by the geographic portrayal process. Definition of datasets and data production, shown in grey in Figure C.1, are not part of the portrayal process but data is an essential input to the portrayal process.

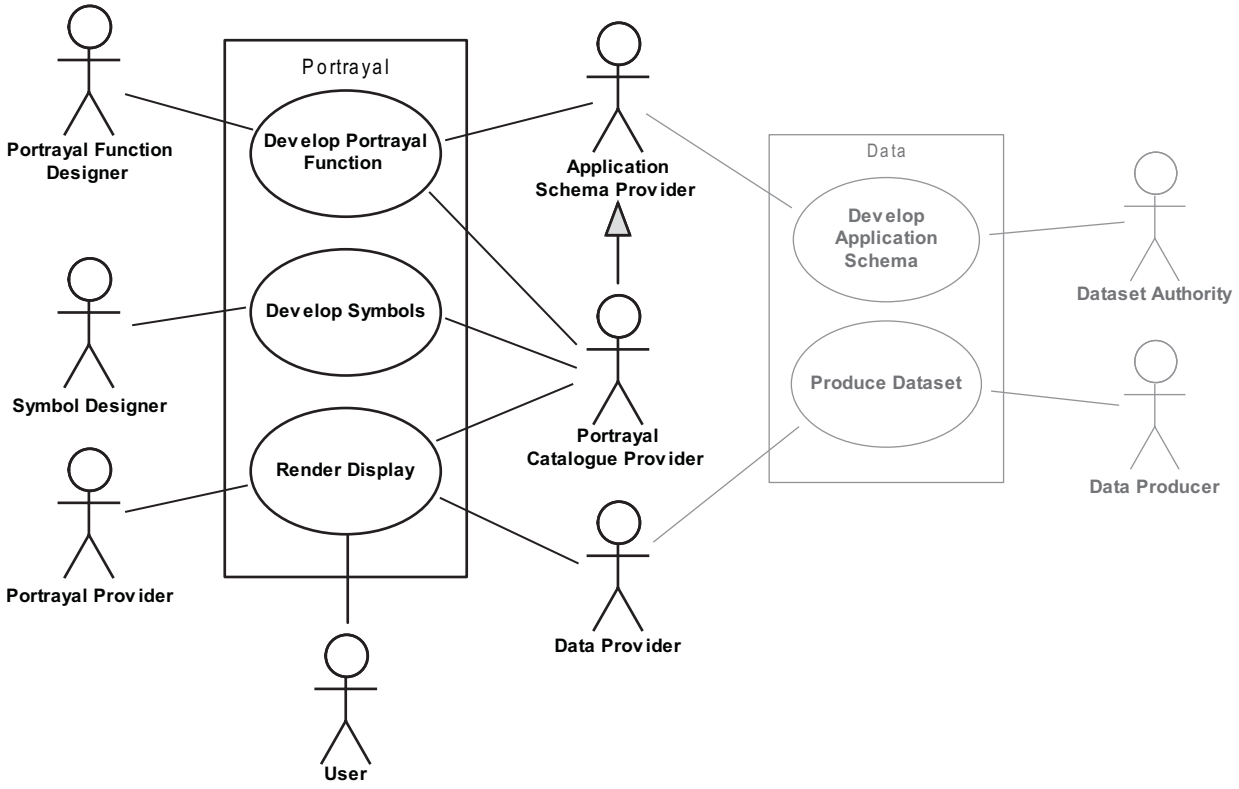

Figure C.1 — Enterprise View Scenario

The elements in Figure C.1 are defined below.

Actors

— Application schema provider – a person or organization that provides the application schema of the dataset being portrayed.

— Data producer [external to portrayal process] – a person or organization that provides data to be portrayed. Note that the data producer may be different from the dataset authority.

— Data provider – a person or organization that provides data being portrayed.

— Dataset authority [external to portrayal process] – the organization or person who defines the geospatial dataset being portrayed. Note that the dataset authority may not be the actual data producer.

— Portrayal provider – an organization, individual, or service that renders a geographic portrayal.

— Portrayal catalogue provider – an organization, individual, or service that provides a portrayal catalogue (portrayal functions and symbols) used to render a map display.

— Symbol designer – an organization or individual that creates symbols. Symbols can be visual, aural, tactile, etc.

— Portrayal function designer – an organization or individual that creates portrayal functions that provide mappings between the data product application schema and the symbols.

— User – the person receiving the geospatial information being portrayed.

Use Cases

— Develop application schema [external to portrayal process] – develop the application schema of the data product.

— Develop portrayal functions – develop the mapping between the source (geographic data) application schema and the symbol definitions.

— Develop symbols – design and define symbols that are called for by the portrayal functions.

— Produce dataset [external to portrayal process] – produce a geographic dataset in accordance with a data product specification (data application schema).

— Render display – generate a display from geospatial data, portrayal functions and symbols.

C.4 Geographic portrayal policy

The portrayal mechanism defined in this International Standard aims to separate the geographic data from the portrayal of that data, associate geographic data with the presentation of the data by use of a portrayal function, and allow a geographic dataset to be portrayed in various ways to satisfy the requirements and needs of the intended audience.

C.5 The portrayal process

The portrayal process is illustrated in Figure C.2.

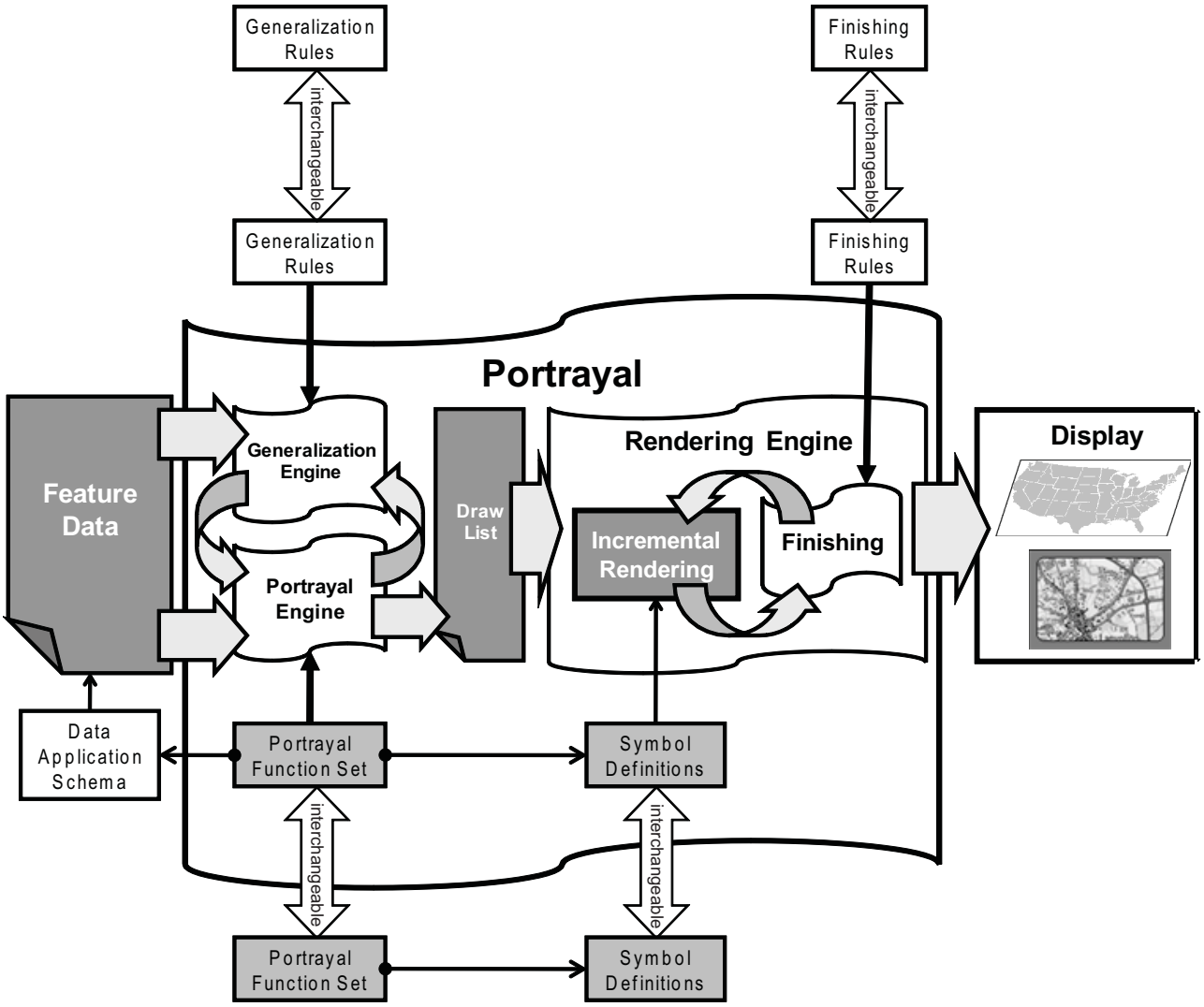

Figure C.2 — Portrayal process

The geographic portrayal process has one input, the geospatial feature data being portrayed, and one output that is a rendered display of the data. Portrayal functions, symbol definitions, and finishing rules serve as controls to the process.

The first step in the portrayal process is to map or associate each feature instance in a geographic dataset to a symbol definition. The portrayal function is dependent on the application schema and feature catalogue of the geographic dataset. A portrayal function set defines this mapping for each feature type in a feature catalogue. The geographic data is input to a portrayal engine. The portrayal function is a control measure to the portrayal engine. The portrayal function set assigns a symbol to each feature type in a feature catalogue, based on the portrayal function. The portrayal function specifies which symbol to use to portray the feature, and references a symbol definition for that symbol. The output of the portrayal engine is a set of resolved symbols or drawing instructions, which include a geometry, a symbol definition for that geometry, and if parameterized symbols are being used, assigned input values, for each feature instance to be symbolized.

One feature type in the geographic dataset may be mapped to one symbol, or it may be mapped to one of several different symbols, based on attribute values of the feature instance. An example of a geographic feature might be a bridge feature type, with a feature attribute of bridge opening type. Bridge symbols might include a symbol for bridges that do not open (fixed span), and separate symbols for lift bridges, swing bridges, drawbridges, and ”other” opening bridges. A many-to-one mapping is also possible, for example where a snag feature type and a stump feature type in a geographic dataset are both mapped to a combined ”miscellaneous

underwater feature" symbol. A feature instance in the geographic dataset will be mapped to a single symbol instance.

The portrayal process has two mandatory controls, portrayal functions, and symbol definitions. Finishing rules are an optional control. Portrayal functions define the mapping from geographic features to symbols, and are separate from the geographic data. A feature portrayal function will reference a symbol definition. Any number of portrayal function sets may exist that will portray a particular geographic dataset, and portrayal function sets that are based on the same feature catalogue are interchangeable. Since portrayal functions reference symbol definitions, the symbol definitions are also interchangeable. The outcome of the portrayal process will depend on which portrayal function set is selected for use. Two examples are:

The portrayal of a geographic dataset to a group of subject matter experts may be different from the portrayal of that same data to the general public, as illustrated in Figure C.3.

Figure C.3 — Symbols for thunder storms

The portrayal of geographic data may have to vary depending on the environmental factors in which it is presented, for example normal display and display in low light conditions, or display at large or small scale, as illustrated by Figures C.4 and C.5.

Figure C.4 — Symbols for different lighting conditions

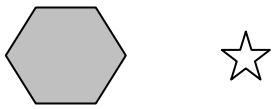

Figure C.5 — Symbols for different display scales

In each of these cases a selected portrayal function set will map the features in the dataset to different symbol definitions than another portrayal function set will.

The rendering process takes the geospatial features, including their geometry, and transforms them into a symbolized map display based on the symbol definitions, and symbol parameters. The initial rendering is an automatic process. The characteristics of this display will vary depending on the medium of the portrayal. If the portrayal is for a digital display, the rendering engine may generate a rasterized (pixel) image that will be displayed on the screen. For a hardcopy map the rendering engine may generate a digital plot file that will be sent to a plotter or a digital-to-plate lithographic press that will print the maps. Non-graphic portrayals will have their own unique characteristics.

Another aspect of generating the final display is to refine the portrayal based on finishing rules. While portrayal functions will define the relationship between geographic data and their symbols, finishing rules generally define relationships between symbols and other symbols, although in some situations such as generalization,

it may be necessary to actually modify the geospatial data being input into the portrayal process to achieve the desired result. Finishing rules take the rendered portrayal and refine it to resolve issues such as overprinting and congestion to make the portrayal easier to understand by the recipient. This aspect of finishing is an optional process. For many uses, a rendered but unfinished portrayal may be perfectly acceptable, or the only possible action under a time constraint. Other portrayals may require higher quality and are not as limited by time constraints. The finishing process may be repeated multiple times, and may require human interactive editing before a portrayal of acceptable quality is achieved. Finishing rule sets are also interchangeable. Different finishing rule sets may produce a higher or lower quality map, take more or less time to execute, or be more or less expensive to create.

Bibliography

[1] ISO 19101:2002, *Geographic information — Reference model*

[2] ISO 19128:2005, *Geographic information — Web map server interface*

[3] ISO 19133: *Geographic information — Location-based services — Tracking and navigation*

[4] ISO/IEC 10746-3:2009, *Information technology — Open Distributed Processing — Reference model: Architecture*

[5] GILL, A.: *Applied Algebra for the Computer Sciences*. Series in Automatic Computation, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976

[6] ISO 19156, *Geographic information — Observations and measurements*

ICS 35.240.70

Price based on 95 pages