

# Curves in ISO19107

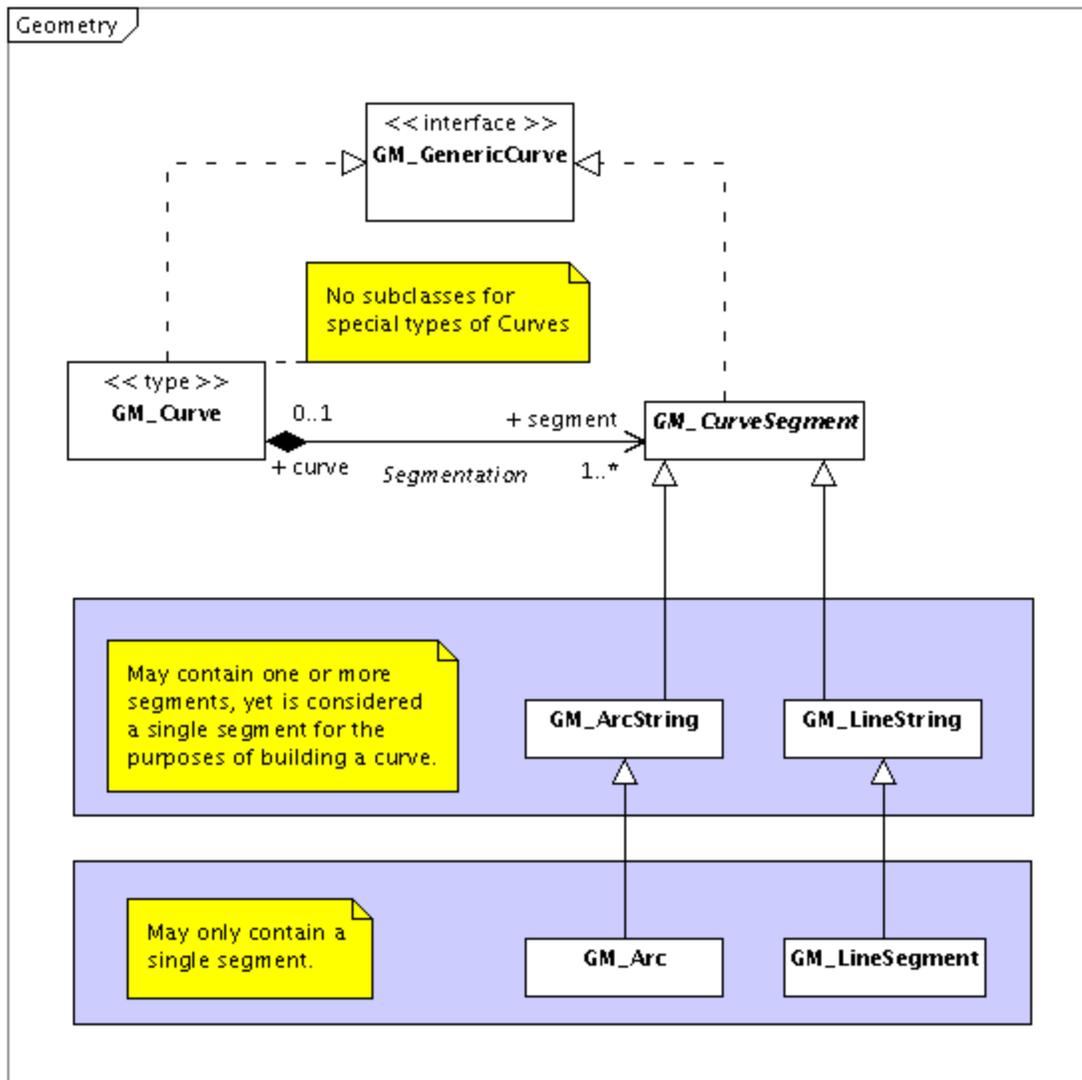
## Introduction

The ISO 19107 geometry specification has some constructs regarding curves which were counterintuitive to me. As I do not really understand anything unless I can explain it, this page represents an attempt to collate and digest some answers received on the GeoAPI development mailing list. This discussion concerns ISO/DIS 19107, which is distributed by the [OGC](#) as [Topic 1: Feature Geometry](#).

As you read this page, keep in mind that much of the discussion may be centered around conceptual differences among the parts of curve geometry. This may have been the starting point, but the UML diagrams and the accompanying text are the content of the standard. These UML diagrams represent an expression of this mental conceptualization in a formalized framework. The expression of concepts may result in artifacts not present in the original conceptualization. Software systems must deal with these artifacts and if these software systems are to be interoperable, then a common treatment of artifacts must be adopted by all implementations. Decisions about how to handle artifacts are best made by those who grasp the concepts. This excludes me presently.

## The Question

To express my question, I require a simplified version of Figure 15 from ISO 19107:



Created with Poseidon for UML Community Edition. Not for Commercial Use.

In summary, my brain viewed the relationship between Arc and ArcString (or between LineSegment and LineString, etc.) as an aggregation. I considered a LineString to be made up of many LineSegments. I was thinking that a LineString is a special type of curve just the same way that a PolyhedralSurface is a special type of surface. As such, I viewed inheritance as a really wierd way to express aggregation.

## Artifacts

The particular UML expression of geometric concepts in 19107 contains artifacts which started me on this quest to figure out what was "really meant". There are three artifacts visible in the above diagram and in the explanatory text associated with the classes in the above figures:

1. A **line string** and a **line segment** are both considered to be curve segments. According to the UML above, both are legal targets of the *Segmentation* association. So **line segments** may be aggregated into a curve (and no longer be a **curve segment**), or they may be aggregated into a **line string** (and remain a **curve segment**, and hence be aggregated *again* into a curve). So what is an aggregation of line segments? Is it a curve or a curve segment? If it can be both, what is the conceptual difference between a curve and a curve segment--especially if a curve segment can have subsegments?
2. This duality exposes another artifact: **curves** and **curve segments** both implement the interface *GM\_GenericCurve* (see above figure). **The meaning of some of the interface properties depends upon the**

**implementation!** Specifically, at the very least, `param()`, `startParam()`, and `endParam()` respond differently if the implementing object is a curve or a curve segment.

3. Finally, **curves** are totally self contained (e.g., a curve always starts at 0 and ends at some terminal length), and **curve segments** must be cognizant of a collection maintained by a *different object* (e.g., a curve segment must start at it's relative position within the collection of curve segments and end at it's terminal distance later.) Additionally, the curve segment must traverse a one-way association in the *wrong direction* in order to properly implement its interface contract.

These artifacts cause a certain level of heartburn from a software design standpoint. Number three, in particular, represents very poor separation of concerns. Regardless of what the abstract notion of a curve may turn out to be, on the software engineering side of things a curve should be responsible for maintaining the relationship between the contained curve segments. Curve segments, when contemplating their `startParam()` should not have to "figure it out for themselves" using a different class's data.

So this page is an attempt to understand how to cope with (at least) these artifacts. Keep this in mind as we delve into the abstractions and history of geometry. Any concepts developed here must make good sense from a software implementation standpoint or else they're not very useful to a software library.

## Response

This question prompted a response from the editor of ISO 19107, Dr. John Herring. The remainder of this article is my attempt to reformulate that response in my own words (with ample quotes from the good doctor) in order to see if I understand it correctly.

## Geometry and Coordinate Representation

### ✓ From Dr. Herring

In ISO 19107, there are two issues driving a lot of the structural decisions: one is the geometry and one is the representation of that geometry in a coordinate space. Curve segments and Surface patches are the coordinate guys, and the Curves, Surfaces, etc, are the more abstract, almost Euclidean guys. Euclid would have understood the curve, but not the curve segment. It would be the early 17th century and Descartes before Euclidean space got coordinates. Makes you wonder how Gerardus Mercator (who died 2 years before Rene was born) came up with the Mercator projection without coordinates - pure geometry I guess.

[...]

The crack about Euclid is that simply put, he knew nothing about the concept of a coordinate. He dealt with geometry in a very abstract, almost object-oriented way. Today we view his stuff through algebra, algorithms (developed by the Arabs), coordinates and calculus (developed by the post-Renaissance Europeans) and topology (late 19th century).

Geometry is a distant memory to me now, so failing to grasp this distinction is probably the root of my problems. It appears that **Geometry** is related to **coordinate representation** by extension.

- A **curve** is an amorphous one dimensional wiggly thing. It is one dimensional because it only has one degree of freedom. However, it can wiggle through one, two, or three dimensional space. A curve is like railroad tracks on the landscape: the tracks can be laid down wherever you please in 2D space. If you have the energy to tunnel or build bridges, then the tracks may be considered to wind through 3D space. However, once laid the tracks provide only one degree of freedom: position along track. A curve has shape, but no way to quantify the shape with respect to any outside coordinate reference system (e.g., can't say tracks turn to heading 255 degrees at 35N 114W).
- A **curve segment** is a way of specifying the path of the tracks through the surrounding space. Curve

segments are capable of translating back and forth between "position along the track" and "external coordinates." This quantitative knowledge unambiguously specifies shape, orientation *and* location. Therefore a **curve** (which doesn't care about location in an external CRS) can be constructed from **curve segments**, but not the other way around. To construct **curve segments** from a **curve** would require extra information, like orientation in the external CRS.

✓ **From Dr. Herring**

- The Curve is defined in ISO 19107 as the continuous image of a 1 dimensional coordinate space (i.e. a line). The concept of continuous is actually from post calculus mathematics, but it has Greek roots in its pre-algebraic approximations of the length of curves. Just to keep things sane, ISO 19107 requires the curve to have a length function, which gets you past the nasty "space filling curves" of Cantor (who died in an insane asylum, a good recommendation not to think too hard about the pathologies of topology).
- Extrapolating on your text, segments are representations of the nice computationally behaved pieces of a curve (we could say between its critical points, but that would be simultaneously wrong in terms of the usual definitions used in the calculus, and too much information). If you were driving a car, you would have to stop doing whatever to the steering wheel and start doing something else [at a segment boundary]. The physics of the situation would require a sudden change in wheel treatment unless the curve is 3-times differentiable.
- Given the historical development of graphics (based mostly on splines, and simpler stuff in 19107) and the associated mathematics for piecewise defined curves that goes with it, ISO 19107 did what was required, and used segment for areas where the definition of the curve was algebraically well behaved, and curve for the whole thing (for those who want to think at Euclidean levels of abstraction).

I gather from this discourse that there are two separable factors which logically distinguish a curve from a curve segment. The first is the notion of abstract geometry (curves) vs. coordinate geometry (curve segments). The second factor is the natural tension between algebraically well-behaved curve segments and the more general notion of a curve which may or may not have a *single* simple defining equation (but which must be composed of algebraically well-behaved segments--else the infinite number of points along a curve couldn't be summarized by a finite number of parameters, ergo: be represented in a finite computing machine.)

⚠ It seems, however, that if the above distinction is correct, the ISO 19107 deviates from this definition at the start. From the above illustration, both curves and curve segments implement GM\_GenericCurve, and this interface embodies the translation between *location-on-the-curve* and position in an external CRS. According to the spec, the main difference between a curve and a curve segment is that curve segments advertise "continuity" information.

## Mathematical Notion of Orientation

✓ **From Dr. Herring**

Orientation is a real mathematical concept that refers to what you learned in calculus about left and right handed coordinate systems. Essentially, for a curve, or a surface, with either 1 or 2 degrees of freedom, the orientation of the object says something about the way it is embedded in the bigger space. For curves, there are two ways, indicating a positive and a negative direction along the preimage line. For a surface, there are two ways, indicating the concept of up and down. It takes a lot of mathematics, but it turns out that this "2 and only 2 orientations" thing is almost universal. The most obvious counterexamples are the Möbius band and the Klein bottle, both of which have only 1 or zero depending on how you generalize the concept. Both of these are surfaces, embedded in 3 and 4 D space respectively. Apparently you need 2D objects to muck up "exactly 2." You might say that these are the canonical pathologies, except that seems to be an oxymoron.

## References

Reference	Comment
<a href="#">Mathworld</a>	Good technical read
<a href="http://www-history.mc.st-andrews.ac.uk/">http://www-history.mc.st-andrews.ac.uk/</a>	Good coverage of historical perspective
<a href="#">Modern Differential Geometry for Physicists by C. J. Isham</a>	
<a href="#">Comprehensive introduction to Differential Geometry. Vol 1</a>	
<a href="#">Comprehensive introduction to Differential Geometry. Vol 2</a>	
<a href="#">Comprehensive introduction to Differential Geometry. Vol 3</a>	
<a href="#">Comprehensive introduction to Differential Geometry. Vol 4</a>	
<a href="#">Comprehensive introduction to Differential Geometry. Vol 5</a>	
<a href="#">Calculus on Manifolds</a>	