

Analysis Visualization Framework

- [Overview](#)
- [Results Data Model](#)
 - [Spatial Field Primitives and Elements](#)
 - [Domain](#)
 - [Result Values](#)
 - [Management and Persistence](#)
- [Presentation Methods](#)
 - [Presentation Settings](#)
 - [Implementation](#)
- [Visualization details](#)
 - [Z-fighting Prevention](#)
 - [Colored Surface GRep](#)
 - [Markers and captions](#)
- [TODO \(Gaps and Open Questions\)](#)

Overview

The Analysis Visualization Framework (AVF) is designed to display external data coming from analysis, simulation, or actual measurement, - in context of Revit model. It is mainly intended to support addins performing various kinds of analysis and simulation - structural, mechanical, energy, lighting, circulation, CFD, etc. Its goal is to let the addin focus on computing the data and let the end user to select one of several common visualization methods preprogrammed in AVF to display these data in Revit view (including printing on sheets). This way AVF decouples semantics of the analysis/simulation from the user interaction with the results of the analysis.

Note that AVF is designed for cases when the user is interested in detailed analysis results that are distributed in space (and sometimes in time) and should be looked at in context of model geometry. AVF is not useful for the cases when the result of the analysis is just one number or a few numbers (e.g. total energy consumption or total embedded carbon for the building).

Useful references:

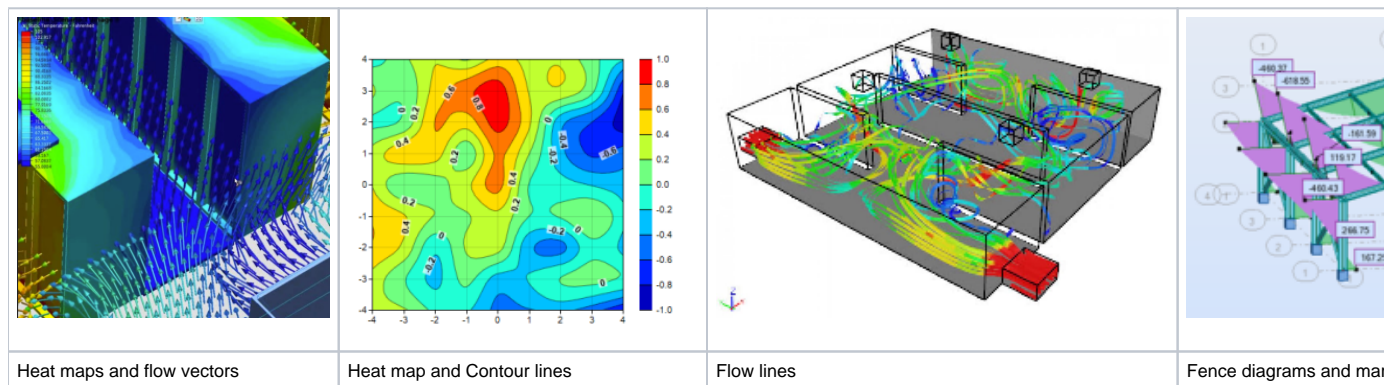
- [Revit API Developers Guide on AVF](#)
- [AVF presentation at the AEC SWA seminar, 2023](#)
- [AVF presentation at the 2012 Technical Summit](#)
- [Articles about AVF applications in Jeremy Tammik's blog](#)

Analysis result from AVF point of view is a function defined on some subset of the model space. For example, solar analysis may compute amount of solar radiation received distributed along the roof and exterior walls of the building. MEP analysis may compute pressure distribution along the pipes on a plant. Structural analysis may compute stresses distributed along beams, columns and floor plates. We call such function a ***spatial field***. In fact, a spatial field can contain multiple such functions, that we call *measurements*. For example, CFD analysis may compute pressure, density, temperature and velocity of air at every point - three scalar and one vector measurement.

The subset of space where the spatial field is defined is its *domain*. The domain can be a 3d volume, e.g. interior of a room or of the entire building. In other cases the domain can be a combination of one or more surfaces or one or more curves (e.g. beams and columns are commonly represented by lines or curves in the structural analytical model). The values (range) of the spatial field are usually scalars (e.g. temperature, pressure, illumination, density, etc.) or 3d vectors (e.g. wind velocity, heat flow, beam deflection, loads, etc.). Other kinds of values (e.g. 3x3 matrices for stress and strain in solids) are not currently supported by AVF, but could be added. It is quite common for a simulation to produce a spatial field that changes in time; these are not explicitly supported in AVF, but could be implemented through a workaround (explicit support could be added to AVF).

Of course analyses discretize the domain and compute results at specific points in this domain.

Common methods of visualizing spatial fields are applied across different engineering disciplines. Some of them are shown here (note the use of multiple methods at once in some cases):



Most methods are applicable only to certain types of spatial fields, e.g. specific dimension of the domain and either scalar or vector value range. Often several alternative methods can be used to visualize the same spatial field; e.g. pressure distribution on a surface can be represented as a heat map or with contour lines.

To facilitate user experience with the simulation results AVF takes care of:

- visualization of the results in context of the model, in 3d views;
- including the results in the printable documentation (drawings), also in context of the model;
- UI to choose the presentation method and UI to control over details of such presentation (e.g. choice of colors, markers, etc.).

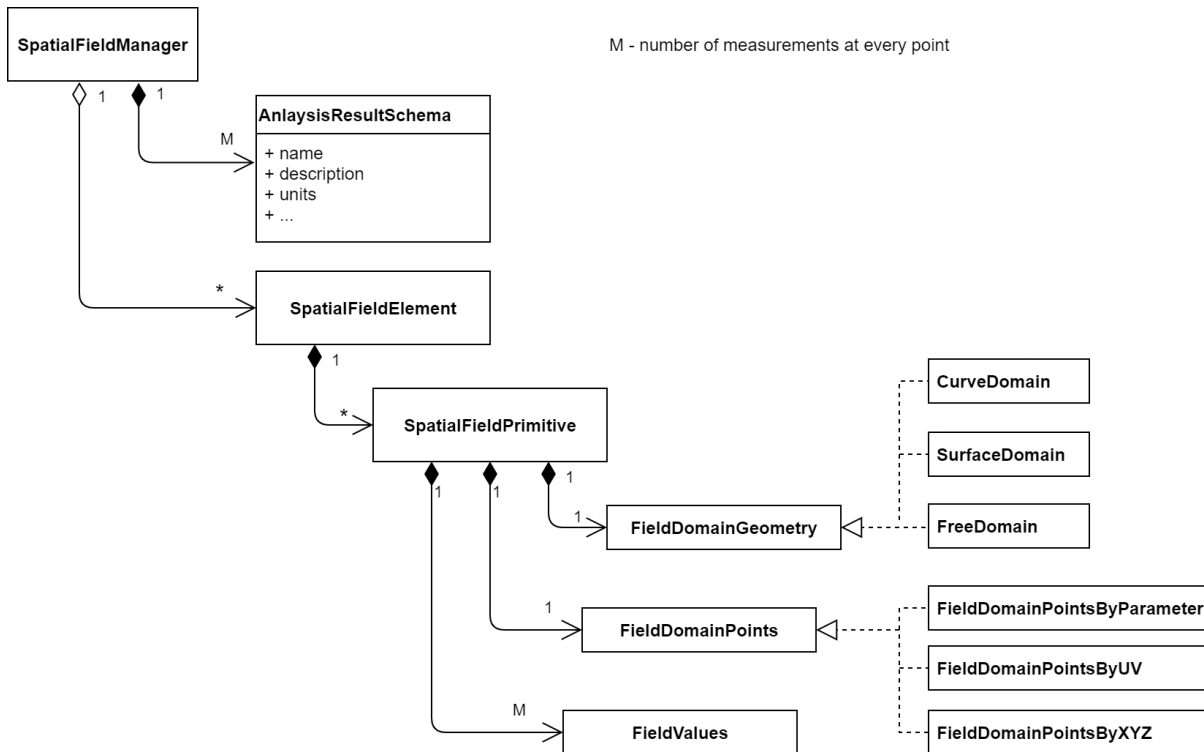
Results Data Model

The *SpatialFieldManager* class is the top-level element class containing one set of analysis results (one spatial field). It is used to create, delete, and modify the "containers" in which subsets of the analysis results are stored. Note that one spatial field can contain multiple *measurements*, like pressure, density, temperature and velocity of air at every point as in CFD case.

The *AnalysisResultSchema* class contains metadata describing one measurement, such as a description and the names and multipliers of all units for result visualization. Multiple *AnalysisResultSchemas* can be registered with the *SpatialFieldManager*, their number should be equal to the number of measurements. *AnalysisResultSchema* is the only place describing the *meaning* (semantics) of the data by containing user-facing name, description and units in which the values are provided.

The supported data model assumes that analysis results consist of one or more spatial fields. Since spatial field can have a very complex domain, e.g. consisting of all floor faces in a building, we define a simpler construct, *SpatialFieldPrimitive*. This is a spatial field defined on a simple domain - one of several enumerated domain types. *SpatialFieldPrimitive* objects are owned by *SpatialFieldElements*. A *SpatialFieldElement* contains one or more *SpatialFieldPrimitive*.

The core data structure of the analysis results is represented in this diagram (many attributes and other details are not shown):



Spatial Field Primitives and Elements

SpatialFieldPrimitive contains:

- The definition of the spatial domain (a curve, a face, a 3D volume) - *FieldDomainGeometry*
- Set of *domain points* distributed in that domain - *FieldDomainPoints*
- Set of *measurements* corresponding to the domain points - array of *FieldValues*

Note that both *SpatialFieldElement* and *SpatialFieldPrimitive* classes are not exposed in the API; *SpatialFieldElements* are created by *AddSpatialFieldPrimitive(...)* methods of the *SpatialFieldManager*. The *SpatialFieldElements* are created to "match" model elements, i.e. all primitives whose domains are based on picks within one model element are put into the same *SpatialFieldElement*. This is irrelevant from the API user standpoint and is done to simplify visualization and user interaction (selection). Revit graphics in general does not like "large" elements (elements with bounding box comparable with the size of the whole building). It especially does not like GREps that are both geometrically large and internally very complex - because such GREps fail quick rejection mechanisms and spend a lot of time in drawing code even if nothing gets drawn at the end.

Domain

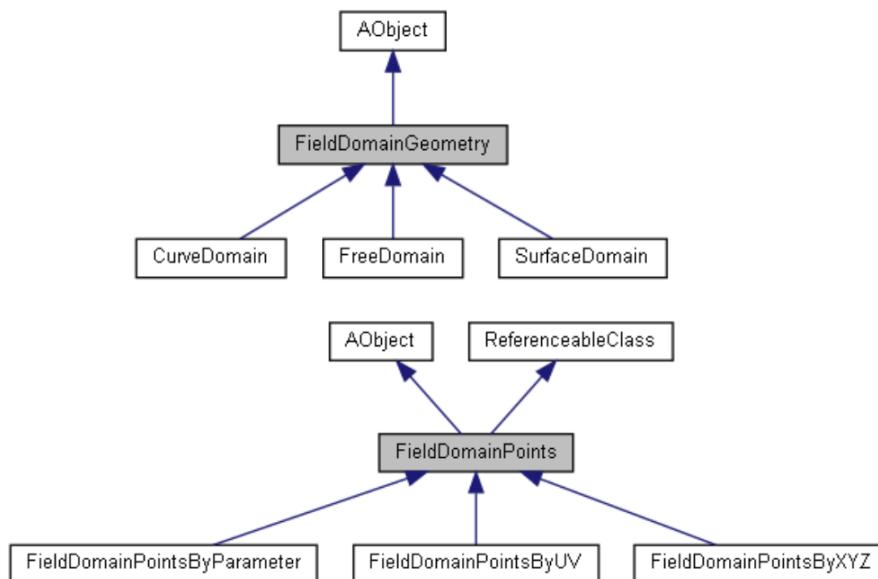
Field domain for one SpatialFieldPrimitive includes geometry primitive (by value or referenced by a Pick) and a set of points on this geometry. When domain geometry is a curve or a surface the points on this domain are defined by values of curve parameter or by UV coordinates.

Specifically, three supported kinds of spatial domains are implemented as subclasses of the *FieldDomainGeometry* class:

- CurveDomain - one-dimensional domain represented either by a GCurve owned by the CurveDomain object or by a reference (Pick) to a model curve;
- SurfaceDomain - two-dimensional domain represented either by a GFace owned by the SurfaceDomain (technically it owns a Geometry containing a single face) or by a reference to a model face;
- FreeDomain - three-dimensional domain consisting of arbitrary XYZ points.

Corresponding set of domain points (at which the results or measurements are defined) are represented by three corresponding subclasses of the *FieldDomainPoints* class:

- FieldDomainPointsByParameter encapsulates an array of doubles - curve parameter values defining points on the curve;
- FieldDomainPointsByUV encapsulates an array of UV points on a surface;
- FieldDomainPointsByXYZ encapsulates an array of XYZ points.



The 1:1 relationship between *FieldDomainGeometry* and *FieldDomainPoints* classes is implicit in the current implementation; it would be better perhaps to enforce it by the class design.

Result Values

The results of the simulation are referred as *measurements*. A measurement is a set of scalars or 3d vectors, corresponding to the *FieldDomainPoints* - one value for each domain point. This set of values is represented by the *FieldValues* object. It can be constructed by taking an input array of *ValueAtPoint* or *VectorAtPoint* objects. The size of these arrays must be equal to the number of points in the corresponding *FieldDomainPoints* object.

SpatialFieldPrimitive contains not one *FieldValues* object, but an array *FieldValues[number_of_measurements]*. The intent of multiple measurements is to support several common scenarios:

- The simulation computes more than one physical quantity at every point. E.g. structural analysis might compute stress and deformation at every point of the simulation domain.
- The simulation may compute one physical quantity under several different (enumerated) conditions, e.g. structural analysis repeated for the wind load coming from different directions, or thermal analysis can compute minimal and maximal temperature during the day.
- This feature can also be used to represent time dimension by using separate measurement for each point in time. This is a bit hacky and it might be better to extend AVF to explicitly support the notion of time.

The number of measurements is assumed to be the same across all *SpatialFieldElements* and primitives within *SpatialFieldManager*.

The implementation assumes that *all* measurements have the same result type - either all results are scalars or they are all vectors. This assumption seems to be superfluous; it is quite conceivable that a simulation computes some scalar fields (e.g. temperature and pressure) and some vector fields (e.g. air flow).

Management and Persistence

The SpatialFieldManager is a view-specific element and the implementation allows to associate only one such element with a DBView. The SpatialFieldManager elements (and all the results data) are transient. These decisions (view-specific and transient) apparently were taken to avoid dealing with more complex user experience; they are not fundamental to AVF and can be reconsidered if needs to. For example, it makes sense to be able to see the same analysis results in multiple views, or to save them in the model so that another user could print the sheets with analysis results without redoing the analysis. However, such features are not supported at the moment.

Presentation Methods
















AVF currently supports five presentation methods:

- colored surfaces (aka "heat maps")
- markers with text (captions)
- deformations
- diagrams (aka "fence diagrams")
- vectors (aka "arrows")

Several more methods could be added in the same spirit, such as contour lines, flow lines, flying particles and other animations.

These methods can be combined, i.e. one data set can be simultaneously displayed using e.g. colored surfaces and markers with text, giving more information to the user.

Although one set of results can be visualized in multiple different ways not all combinations make sense, see the table below. For example, scalar field cannot be visualized with arrows (vectors) or interpreted as deformations. All combinations that do make sense logically are supported. When the presentation method requires a scalar field (e.g. to represent as a "heat map") while the data is a vector field AVF consistently applies the presentation method to the norms (lengths) of the given vectors.

Presentation method	Domain geometry			Restrictions & interpretations
	Surface	Curve	"Free" 3d	
Colored Surfaces				If FieldValues are vectors, then the lengths of these vectors are used
Markers with Text				If FieldValues are vectors, then the lengths of these vectors are used
Deformations				FieldValues must be 3d vectors
Diagrams				If FieldValues are vectors, then the lengths of these vectors are used
Vectors				FieldValues must be 3d vectors

Presentation Settings

The AnalysisDisplayStyle class is used to control the presentation of the results in a view. AnalysisDisplayStyle can be created/modified from an addin, but end users can also create/modify it through the Revit UI. So while adding new kind of analysis requires addin code to create SpatialFieldManager and populate it with data it does not require any work at all in the addin to get these data visualized per user requirements.

Implementation

Each SpatialFieldElement generates view-specific GRep according to the presentation method(s) selected for the DBView. The code is located in the methods of SpatialFieldPrimitive:

- generateColoredSurfaceGRep()
- generateMarkersAndTextGRep()
- generateDeformedShapeGRep()
- generateDiagramGRep()
- generateVectorGRep()

Display pipeline does not have any special awareness of the analysis results (where do we do tessellation refinement for the colored surfaces?)

Visualization details

A naïve approach to visualization of the simulation results is to let simulation add to "draw on top" of the canvas that already contains model graphics, e.g. by adding pixels to z-buffer with certain depth. There are some challenges, however, that make this a lot more difficult than it sounds. Solving these problems, along with extensive UI to control presentation look and feel (e.g. the color palette), constitute the bulk of AVF value.

Z-fighting Prevention

If visualization draws on the existing model face or curve - we skip drawing the original. Otherwise, since tessellations of the original model surface and the results surface don't exactly match (they use different tessellation points) we would get unpredictable results - known as [z-fighting](#).

Colored Surface GRep

We represent colored surface as a triangular mesh with colors specified at every vertex. See `SpatialFieldPrimitive::createMeshAndGeometry_()`.

The mesh is computed by tessellating the original analytical Face specified in the SurfaceDomain. The tessellation is done using same method (by R. Shewchuk) as the one used for visualization of solids. However, we add the UV points on which the spatial field is defined (FieldDomainPointsByUV) as additional constraints to the tessellation routine to make sure these points become some of the vertices of the resulting mesh.

To compute colors at the mesh vertices we interpolate the spatial field. For any vertex we calculate the value at an arbitrary point (u,v) as the average of values in the vertices where results are specified, weighted by the inverse square of the Euclidian distances to those vertices. See implementation in `valueFromOtherPoints()` method in `SpatialFieldPrimitive.cpp`. The values are then mapped to colors according to the AnalysisDisplayStyle settings.

Additional complexity is introduced when AnalysisDisplayStyle requires discrete (rather than continuous) colors. To achieve discrete colors we further tweak the tessellation by subdividing the triangles through which the boundary between colors passes.

Markers and captions

Marker with text is a pretty complex creature. It is attached to a model point. If the model point is obscured by something else in the model, then both the marker and the caption should not be drawn, but if the point is visible, then the marker and the caption should be fully drawn (not partially obscured). These checks should be very fast to allow e.g. interactive moving the camera. All the while the captions should stay readable in terms of font size and text orientation. Some reasonable behavior is expected when the camera zooms out so that markers and captions get too crowded.

TODO (Gaps and Open Questions)

1. How much AVF is used (ask PMs)?
 - a. Can we find out via ADP?
2. What do we hear from AVF API users (partners)? - Ask PMs and check Revit Ideas.
3. Why Revit does not persist analysis results (what UX problems we'd need to solve) + indicate that serialization itself is not a problem, can be driven by the same schema (DD)
 - a. Do we hear customer requests for persistence (ask PM)?
 - b. The problems with "staleness" (less acute for Forma/AECDM)
 - c. Automatic detection of "relevance"
4. Can we visualize more than one SpatialFieldsManager at the same time in the same view? What are the obstacles? How this could be done?
 - a. We could combine domain points from multiple analysis into one tessellation.
5. Explain that visualization is done by inserting graphical objects (GReps) corresponding to SpatialFieldElements in the view *scene graph* (could be done on USD).
6. Do we allow SpatialFields on meshes, e.g. toposurfaces? Why not?
 - a. Did we test AVF with Toposolids? - ask Palladio
7. Do people face visualization problems (z-fighting) when they create "artificial" face coincident with some of the "native" faces?
8. How do we support additional calculations on top of the raw data from one or several analyses? E.g. "code checking" addin can use results of noise analysis to calculate whether every unit has at least one wall with low noise level.
 - a. The problem is finding whether right raw data are already computed and present in the model. The issue of identifying the result set.