Survey of software architecture V0.0

Name of system:

VTK (Visualization Toolkit)

Reviewer:

Claude Richard

Date:

November 14th, 2011

Author of software:

Will Schroeder, Ken Martin and Bill Lorensen

Author of book chapter:

Berk Geveci and Will Schroeder

Five star rating of book chapter:

**** (4)

Purpose of system:

VTK was conceived as a scientific data visualization system, but it is now widely used to visualize various kinds of information. It is intended to be used by the scientific community. It is a toolkit used for rendering various 3D computer graphics.

VTK was initially developed by Will Schroeder, Ken Martin, and Bill Lorensen while they were researchers at GE Corporate R&D. The authors chose to develop VTK under an open-source license because non-open-source licenses would make it difficult to promote their work outside the company, due to intellectual patent concerns.

The primary goal when creating VTK was to have a framework for transforming data from a scientific form (e.g. Vectors) to a form understandable by the human senses (e.g. visual).

Basic metrics

KLOC: 1,500

Project start-up: 1993

Number of major releases: 5

Number of developers: 3 initially

Size of user community or number of installations: 50 download for .NET version on SourceForge, Thousands of researchers and developers use VTK [3].

Major stakeholders: GE Corporate, The scientific community

Use of concurrency: VTK is not thread-safe [2]. You need to be careful with multi-threading.

Implementation language:

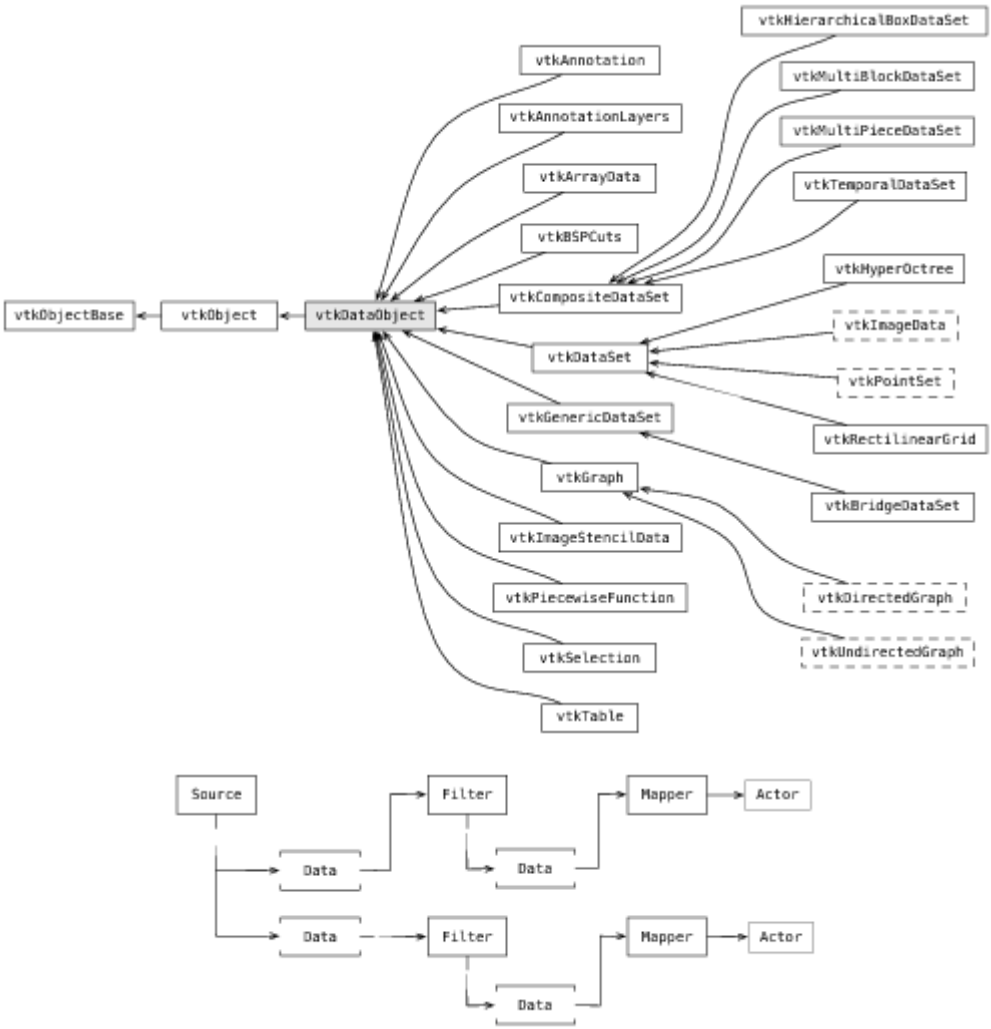
C++. Can automatically generate language bindings to Python, Java, and Tcl.

Supporting software:

BullseyeCoverage, CMake, CDash/CTest, CPack, Doxygen

High level architecture

Diagram of software architecture



http://www.aosabook.org/en/vtk.html

The top diagram is a partial inheritance diagram; boxes denote classes and arrows denote inheritance.

The bottom diagram represents a pipeline at run-time for a very simple scenario. “Filter” boxes denote filters and all other boxes denote pipes. Data flows in the direction of the arrows.

High level scenarios

The following scenario is taken directly from the book. A main program using VTK is written as follows:

```
vtkOBJReader *reader = vtkOBJReader::New();
reader->SetFileName("exampleFile.obj");

vtkTriangleFilter *tri = vtkTriangleFilter::New();
tri->SetInputConnection(reader->GetOutputPort());

vtkQuadricDecimation *deci = vtkQuadricDecimation::New();
deci->SetInputConnection(tri->GetOutputPort());
deci->SetTargetReduction( 0.75 );

vtkPolyDataMapper *mapper = vtkPolyDataMapper::New();
mapper->SetInputConnection(deci->GetOutputPort());

vtkActor *actor = vtkActor::New();
actor->SetMapper(mapper);

vtkRenderer *renderer = vtkRenderer::New();
renderer->AddActor(actor);

vtkRenderWindow *renWin = vtkRenderWindow::New();
renWin->AddRenderer(renderer);

vtkRenderWindowInteractor *interactor = vtkRenderWindowInteractor::New();
interactor->SetRenderWindow(renWin);

renWin->Render();
```

The first 7 lines of this program are setting up a pipeline, going downstream from reader to tri to deci. Then the next two lines set up a mapper to connect the pipeline to the rendering subsystem. The next 6 lines set up the rendering subsystem, including the window that displays the visualization. The last line starts rendering.

Data structures or algorithms

Major object types are: vtkDataObject, vtkRenderer, vtkRederWindow, vtkActor, vtkLight, vtkCamera, vtkRederWindowInteractor, vtkMapper

Representing contiguous data is usually done with various data arrays. This makes the software better able to handle large inputs, serialize and perform IO etc. Another reason why it is implemented this way is that "in scientific computing it is common to interface with systems manipulating arrays (e.g. Fortran)" [1].

For the data filtering pipeline, there is an algorithm for determining whether the data in a pipe is valid or stale. This is necessary since changes in the pipeline can occur (e.g. The user inputs a line in the Python command line). The trigger for making a pipeline start propagating data downstream is a request from downstream for data. The request propagates upstream until it finds valid data (e.g. A pipe containing valid data already executed or a source file). Then the data can be processed and propagated downstream to the requester. VTK includes timestamps as metadata for the data in pipes in order to determine the validity of data. When a pipe or filter changes (e.g. Its parameters change), it invalidates its own data as well as all the data downstream from it. This way when the next request is received the pipes and filters must re-calculate their data, which they do by requesting data upstream.

Control flow and/or data key to the architecture if any

Factory for creating objects: Every object has a public method New(), which returns a new instance of either the class itself or a subclass. Most class instantiation is used with this. This makes it very easy to change the implementation of an object.

VTK does not rely on the Java-style garbage collection. It tries to optimize garbage collection by removing garbage objects as soon as possible. This is because large amounts of data can be used, and it's not a good idea to have gigabytes of garbage still allocated in memory for an unnecessary amount of time. VTK does this with reference counting. For a given object X, every object Y that uses X must register itself with X, thus X's reference count is incremented. This is done inside VTK object code and not by end users. When X is instantiated, its reference count is initialized to one. When X's reference count goes to zero, it automatically self-destructs. The vtkGarbageCollector can be used to look for reference cycles, so that when an isolated group of objects with a reference cycle exists, the entire group can be considered garbage and disposed of.

Architectural style:

Pipe and Filter: Style of the Filtering library. This library is used for the processing of data such as changing objects from one form to another. The downstream end of this pipeline can be used as input to the rendering (e.g. Graphical rendering) subsystem.

Object oriented: VTK has been purposely made very object oriented, with an emphasis on inheritance and making the changing of algorithms/data representations as easy as possible.

Events: "All subclasses of vtkObject maintain a list of observers which register themselves with the object." [1] When registering, the object passes two arguments: the event, and a vtkCommand to execute when the event occurs.

Major evolutionary changes:

"We have struggled with the data execution pipeline, having gone through multiple generations each time making the design better." [1]

VTK is always growing and handling more and more different types of data.

"The data processing pipeline in VTK is still too complex. Methods are under way to simplify and refactor this subsystem." [1]

"The rendering system in VTK has been criticized for being too complex, making it difficult to derive new classes or support new rendering technology." [1]

Performance bottlenecks:

This system can often allocate large amounts of data, so the garbage collection mechanism must deallocate unused data as soon as possible. VTK does this by reference counting. (see Control Flow above)

When an object in the data pipeline requests data upstream, the upstream object must compute its data, requesting data upstream of it if necessary, then return the data downstream. An object caching its data is necessary to avoid excessive recomputation of the same data.

Real time:

Notation for architecture:

Box and arrow

Methodology:

not clear

Appendix:

Kruchten's eight context attributes applied to Brown/Wilson systems

Kruchten attributes described in his slides 17-24 of
https://files.me.com/philippe.kruchten/sbz0ma
Also in: https://files.me.com/philippe.kruchten/1q00nw

1. **Size:** L
2. **Criticality:** Med
3. **Age of system:** XL
4. **Rate of change:** Hi
5. **Business model:** Open-Source
6. **Stable architecture:** Hi
7. **Team distribution:** Hi
8. **Governance:** Med moving to Hi

References:

1. http://www.aosabook.org/en/vtk.html
2. http://www.vtk.org/Wiki/VTK/FAQ#Is_VTK_thread-safe_.3F
3. http://www.vtk.org/