Survey of software architecture V0.1

Name of system: VisTrails
Reviewer: Chun Liu
Date: Nov, 19th, 2011
Author of software: Cláudio Silva, Juliana Freire
Author of book chapter: Juliana Freire, David Koop, Emanuele Santos, Carlos Scheidegger, Huy T. Vo
Five star rating of book chapter: 5 stars (good explanations on each component of architecture.)

Purpose of system:

VisTrails is a tool that supports data exploration and data visualization. It includes many useful features of scientific workflow and visualization systems. It helps users easier explore and compare different visual representations of data.

VisTrails has two basic design requirements in mind. First, VisTrails allow users able to specify their own data exploration process, i.e, declaring their process modules in a workflow. The processes should be executable after users define a workflow. Second, VisTrails allow users to view the workflow history, or in other words, the system should capture data provenance. Viewing workflow history helps users remember how the results can be reproduce, and helps users analyze different steps to solve a problem. The second requirement makes VisTrails different from other scientific tools.

VisTrails addresses usability issues. To allow a broader set of users, including some people who do not have programming experiences, VisTrails provides different kinds of data operations and user interfaces that make data exploration process simpler, for example, to query workflows, to suggest workflow completions.

VisTrails is a general tool for data exploration. VisTrails makes simpler for users to integrate external tools and libraries, such as VTK, Web services, and etc. This has been a beneficial to use the system in a wide range of application areas, including astronomy, quantum physics, molecular modeling, and other sciences.

Basic metrics

KLOC:

At least 208,638 LOC python code.

Estimated around 1M LOC. (I can't find code size information on web)

Project start-up:

Initial development in 2004.

First release in September 2007.

Number of major releases:

8 major releases:

1.6.2: April 2011

1.5.1: August 2010

1.4.2: March 2010

1.4: January 2010

1.3: July 2009

1.2: July 2008

1.1: May 2008

1.0: September 2007

Number of developers:

16

Size of user community or number of installations:

Downloads: over 25,000 times

Major stakeholders:

The University of Utah,

The Department of Energy under the SciDAC program (SDM and VACET) and UV-CDAT,

IBM Faculty

Use of concurrency:

Workflow views, Workflow Version Trees, Workflow Executions are separated tasks.

Implementation language:

Initial Development: C++

Late in 2005, VisTrails is mainly implemented in Python/PyQt/Qt.

Supporting software:

VTK, Image Magick, Web Services, pylab

High level architecture

Diagram of software architecture

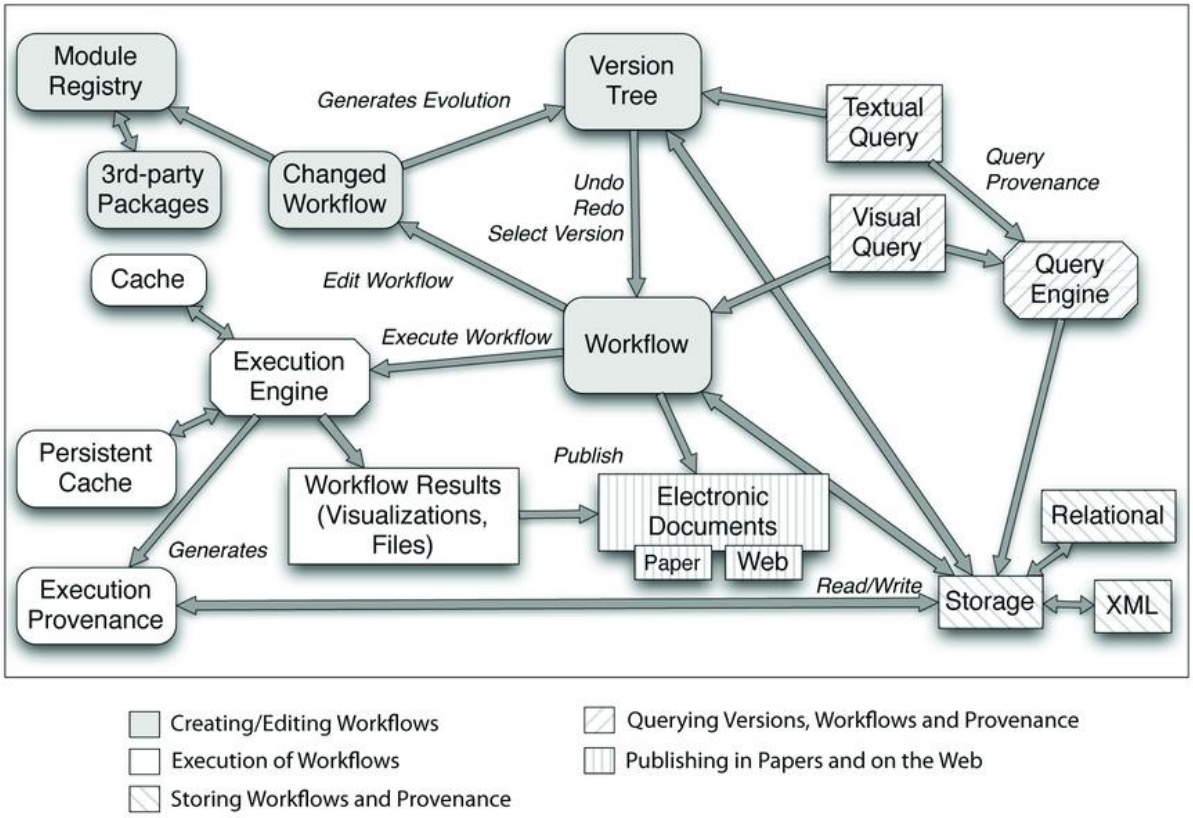

Figure 23.3: VisTrails Architecture

High level scenarios

All data explorations begin with building the workflow graphs. Users can drag modules from the Module Registry and drop them into the Workflow Editor canvas, which creates another version of workflow (i.e. changes the workflow graph) and generates a child in the version tree.

During creating the workflow graph, users can do query via Query Engine, for example, searching keywords in Workflow canvas.

When users are satisfy with the current workflow graph, Execution button will be pressed. Then, the system calls the Execution Engine, and let the Execution Engine to process the data, followed by the steps specified in the workflow graph. Execution Engine will generate and present the results to user via graphs, for example, via VTK GUI. If necessary, Execution Engine will also generate data documentations.

Data structures or algorithms

Data structures:

Bijectivedict

Graph

Point

Queue

Rect

Stack

Tree

And other GUI structures.

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

Workflow pipelines,

Version Trees,

Database Layer,

Spreadsheet-style interfaces.

Architectural style:

Data-Flow style

Major evolutionary changes:

In late 2005, all the implementations have changed from C++/Java to Python.

Performance bottlenecks:

Database Layer.

Real time:

No time restrictions, but still need some reasonable response time.

Notation for architecture:

Module Diagram

Methodology:

No information, since there is no developers' site/forum for VisTrails.

(the methodology is not agile)

Appendix

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

1. **Size:** L	(estimated 1M)
2. **Criticality:** Med	
3. **Age of system:** M	(2004-present)
4. **Rate of change:** Med	(around a release every half year)
5. **Business model:** Commercial & Open source	(plug-in in Maya is Commerial)
6. **Stable architecture:** Lo	
7. **Team distribution:** Med	(managed by University of Utah)
8. **Governance:** Lo	(small team)