

7. Olle T.W. *The Codasyl Approach to Data Base Management*. Wiley, Chichester, UK, 1978.
8. Rotem D., Stockinger K., and Wu K. Minimizing I/O costs of multi-dimensional queries with bitmap indices. In *Proc. 18th Int. Conf. on Scientific and Statistical Database Management*, 2006, pp. 33–44.
9. Shannon P., Markiel A., Ozier O., Baliga N.S., Wang J.T., Ramage D., Amin N., Schwikowski B., and Ideker T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, 13(11):2498–2504, 2003.
10. Shoshani A., Olken F., and Wong H.K.T. Characteristics of scientific databases. In *Proc. 10th Int. Conf. on Very Large Data Bases*, 1984, pp. 147–160.
11. Shoshani A. and Wong H.K.T. Statistical and scientific database issues. *IEEE Trans. Softw. Eng.*, 11(10):1040–1047, 1985.
12. Zhang X., Kurc T., Saltz J., and Parthasarathy S. Design and analysis of a multi-dimensional data sampling service for large scale data analysis applications. In *Proc. 20th Int. Parallel and Distributed Processing Symp.*, 2006.

---

## Scientific Knowledge Bases

- ▶ Biomedical Scientific Textual Data Types and Processing

---

## Scientific Medicine

- ▶ Evidence Based Medicine

---

## Scientific Query Languages

- ▶ Query Languages for the Life Sciences

---

## Scientific Visualization

RONALD PEIKERT  
ETH Zurich, Zurich, Switzerland

### Definition

Scientific visualization [1] provides graphical representations of numerical data for their qualitative and quantitative analysis. In contrast to a fully automatic analysis (e.g. with statistical methods), the final analytic step is left to the user, thus utilizing the power of the human visual system. Scientific visualization differs

from the related field of information visualization in that it focuses on data that represent samples of continuous functions of space and time, as opposed to data that are inherently discrete.

The challenge in scientific visualization is to cope with massive data, which cannot be presented to the user in an unprocessed way for several reasons:

1. Volumetric data, i.e. data given on a three-dimensional domain, occlude each other. This problem becomes even more severe if data are not scalars, but vectors or even tensors.
2. Visualization should provide a global picture of the spatial and temporal behavior of the data, but also allow for interactive exploration of details.
3. There can be multiple data (different physical quantities, multiple data channels, etc.) at each point in the domain.
4. Visualization of scientific data should also include visualization of their uncertainty.
5. The amount of raw data often exceeds limitations of processor speed, transfer rates, memory size, and display resolution.

Applications of scientific visualization cover a wide spectrum of science and engineering disciplines. Currently, some of the most active fields are medical and biomedical image data, simulation and measurement data from fluid or solid mechanics, molecular data, data from geology and geophysics, astronomy, weather and climate.

### Key Points

Scientific visualization evolved in the 1980s from earlier graphing techniques when 3D computer graphics opened new ways of displaying numerical data. The abstraction from the application domain led to interdisciplinary visualization software systems with a modular dataflow architecture. This approach is still successful [2], since for many visualization tasks, the semantics of the data is less relevant than mathematical properties such as the discretization type or the categorization into scalars, vectors, and tensors.

For volumetric scalar data, important visualization techniques are isosurfaces and direct volume rendering. For vector fields examples are arrow glyphs, integral lines (streamlines, streaklines) and texture advection. Tensor fields are visualized with glyphs (ellipsoids, superquadrics) or tensor lines. In the special case of diffusion tensor MRI data, fiber tracking techniques are applied. Scalar,

vector, and tensor fields all are amenable to topology-based visualization, which provides both the singularities and a segmentation of the domain into regions of “similar data behavior.” More general, feature extraction and feature tracking techniques aim at reducing the data complexity and providing the viewer with only the most salient information. Features (e.g. edges, ridges, flow structures) are typically defined in terms of data and their derivatives. User-defined feature definitions are possible in visualization systems built on the linked views paradigm, where simultaneous views of both physical and data space are available all of which allow for interactions such as data coloring and subsetting. The visualization of very large data requires optimization techniques including multi-resolution, parallel and out-of-core algorithms, as well as view-dependent visualization.

### Cross-references

- ▶ Data Visualization
- ▶ Visualization Pipeline

### Recommended Reading

1. Hansen C.D. and Johnson C.R. (eds.). *Visualization Handbook*. Academic Press, San Diego, CA, 2004.
2. Schroeder W., Martin K., and Lorensen B. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics Kitware*, Inc., New York, 2006.

---

## Scientific Workflows

BERTRAM LUDÄSCHER, SHAWN BOWERS,  
TIMOTHY MCPHILLIPS  
University of California-Davis, Davis, CA, USA

### Synonyms

In silico experiment; Grid workflow

### Definition

A *scientific workflow* is the description of a process for accomplishing a scientific objective, usually expressed in terms of *tasks* and their *dependencies*. Typically, scientific workflow tasks are computational steps for scientific simulations or data analysis steps. Common elements or stages in scientific workflows are acquisition, integration, reduction, visualization, and publication (e.g., in a shared database) of scientific data. The tasks of a scientific workflow are organized (at

design time) and orchestrated (at runtime) according to dataflow and possibly other dependencies as specified by the workflow designer. Workflows can be designed visually, e.g., using block diagrams, or textually using a domain-specific language.

### Historical Background

Workflows have a long history in the database community and in business process modeling, in which case they are sometimes called *business workflows* to distinguish them from scientific workflows. The database community realized early [10] that scientific data management has different characteristics from more traditional business data management. Early work on scientific workflows within the database community took a database-centric view by defining data models and query languages suitable for scientific experiment management systems. The MOOSE data model and FOX query language have their roots in the late 1980’s [5] and early 1990’s [13] and gave rise to the ZOO experiment management environment [6], an early system based on an underlying object-oriented database. Another pioneering work that emphasized the importance of workflow concepts in scientific data management is WASA, a *Workflow-based Architecture for Scientific Applications* [8]; the related publication [12] introduced the term “scientific workflow” and contrasted such workflows with office automation and business workflows. An early benchmark comparing different database architectures for scientific workflow applications is LabFlow-1 [1].

Other roots of scientific workflow systems include *problem solving environments*, which emerged in the nineties in the computational sciences community as intuitive tools to “solve a target class of problems for scientific computing” [4], and *laboratory information management systems* (LIMS) [9], which can be seen as special scientific workflow systems that are used in a laboratory environment for the management of samples, instrument-based measurements, and other functions, including data analysis and workflow automation. Similar to many scientific workflow systems, problem solving environments and LIMS sometimes employ a visual programming paradigm to link together components. An early, if not the first, visual language that allowed simple interfacing with lab instruments was G in LabVIEW1.0, released in 1986 for the Apple Macintosh. Modern incarnations of LIMS can include functions of enterprise resource planning (ERP) systems