

DOI:10.1145/1897852.1897871

Compose “dream tools” from continuously evolving bundles of software to make sense of complex scientific data sets.

BY KATY BÖRNER

Plug-and-Play Macroscopes

DECISION MAKING IN science, industry, and politics, as well as in daily life, requires that we make sense of data sets representing the structure and dynamics of complex systems. Analysis, navigation, and management of these continuously evolving data sets require a new kind of data-analysis and visualization tool we call a macroscope (from the Greek macros, or “great,” and skopein, or “to observe”) inspired by de Rosnay’s futurist science writings.⁸

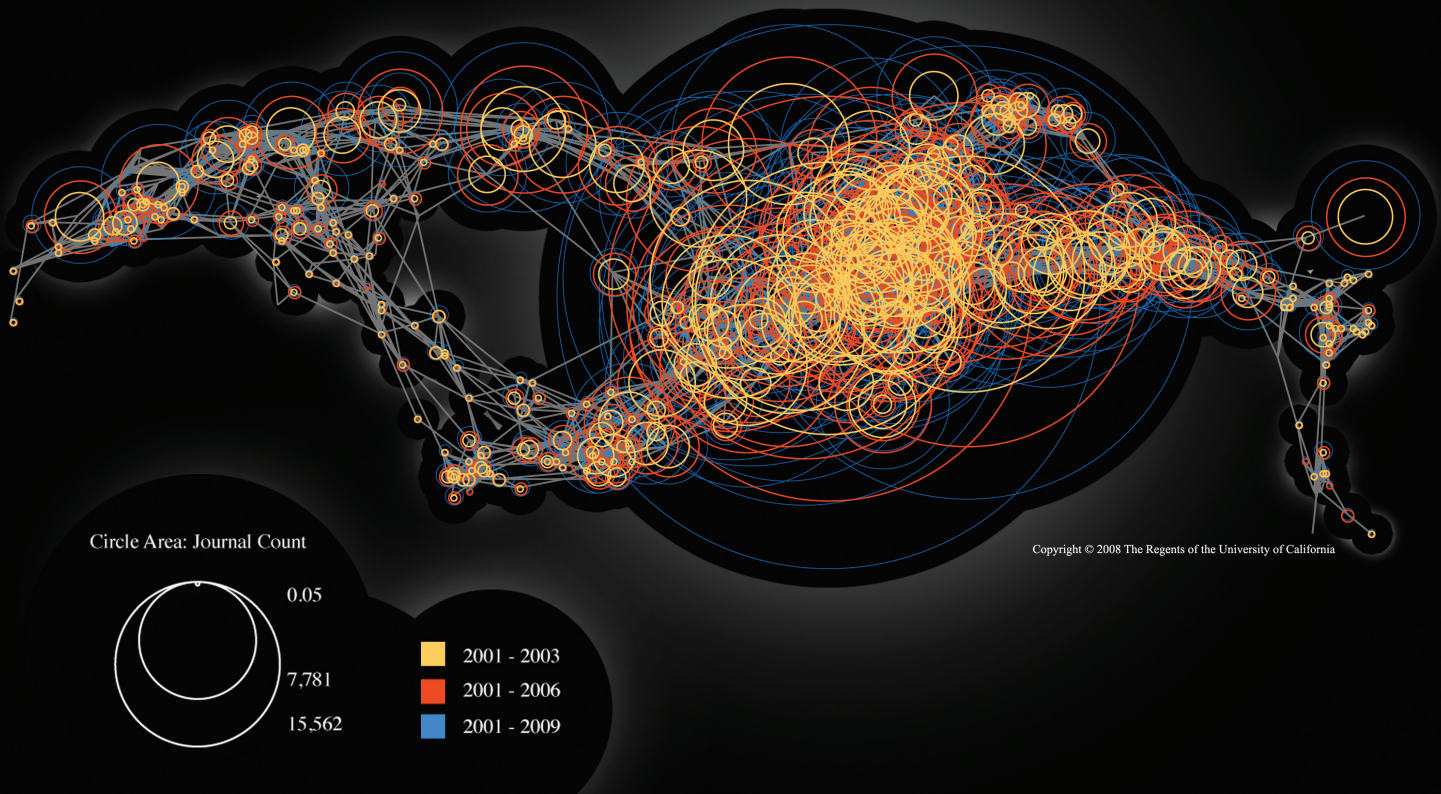
Just as the microscope made it possible for the naked human eye to see cells, microbes, and viruses, thereby advancing biology and medicine, and just as the telescope opened the human mind to the immensity of the cosmos and the conquest of space—the macroscope promises to help make sense of yet another dimension—the infinitely complex. Macroscopes provide a “vision of the whole,” helping us “synthesize” the related elements and detect patterns, trends, and outliers while granting access to myriad details.^{18,19} Rather than make things larger or smaller, macroscopes let us observe what is at once too great, slow, or complex for the human eye and mind to notice and comprehend.

Many of the best micro-, tele-, and macroscopes are designed by scientists keen to observe and comprehend what no one has seen or understood before. Galileo Galilei (1564–1642) recognized the potential of a spyglass for the study of the heavens, ground and polished his own lenses, and used the improved optical instruments to make discoveries like the moons of Jupiter, providing quantitative evidence for the Copernican theory. Today, scientists repurpose, extend, and invent new hardware and software to create macroscopes that may solve both local and global challenges²⁰ (see the sidebar “Changing Scientific Landscape”).

My aim here is to inspire computer scientists to implement software frameworks that empower domain scientists to assemble their own continuously evolving macroscopes, adding and upgrading existing (and removing obsolete) plug-ins to arrive at a set that is truly relevant for their work—with little or no help from computer scientists. Some macroscopes may resemble cyberinfrastructures (CIs),¹ providing user-friendly access to massive amounts of data, services, computing resources, and expert communities. Others may be Web services or stand-alone tools. While microscopes and telescopes are physical instruments, macroscopes resemble continuously changing bundles of software plug-ins. Macroscopes make it easy to select and combine algorithm and tool plug-ins but also interface plug-ins, workflow support, logging, scheduling, and other plug-ins needed for scientifically rigorous work. They make it easy to share

» key insights

- **OSGi/CIShell-powered tools improve decision making in e-science, government, industry, and education.**
- **Non-programmers can use OSGi/CIShell to assemble custom “dream tools.”**
- **New plug-ins are retrieved automatically via OSGi update services or shared via email and added manually; they can be plugged and played dynamically, without restarting the tool.**



UCSD Map of Science with data overlays of MEDLINE publications that acknowledge NIH funding.

plug-ins via email, flash drives, or online. To use new plug-ins, simply copy the files into the plug-in directory, and they appear in the tool menu ready for use. No restart of the tool is necessary. Sharing algorithm components, tools, and novel interfaces becomes as easy as sharing images on Flickr or videos on YouTube. Assembling custom tools is as quick as compiling your custom music collection.

The macroscopes presented here were built using the Open Services Gateway Initiative Framework (OSGi) industry standard and the Cyberinfrastructure Shell (CIShell) that supports integration of new and existing algorithms into simple yet powerful tools. As of January 2011, six different research communities were benefiting from OSGi and/or CIShell powered tools. Several other tool-development efforts consider adoption.

Related Work

Diverse commercial and academic efforts support code sharing; here, I discuss those most relevant for the design and deployment of plug-and-play macroscopes:

Google Code and SourceForge.net provide the means to develop and dis-

tribute software; for example, in August 2009, SourceForge.net hosted more than 230,000 software projects by two million registered users (285,957 in January 2011); also in August 2009 ProgrammableWeb.com hosted 1,366 application programming interfaces and 4,092 mashups (2,699 APIs and 5,493 mashups in January 2011) that combine data or functionality from two or more sources to arrive at a service.

Web services convert any Web browser into a universal canvas for information and service delivery. In addition, there are diverse e-science infrastructures supporting researchers in the composition and execution of analysis and/or visualization pipelines or workflows. Among them are several cyberinfrastructures serving large biomedical communities: the cancer Biomedical Informatics Grid (caBIG) (<http://cabig.nci.nih.gov>); the Biomedical Informatics Research Network (BIRN) (<http://nbirn.net>); and the Informatics for Integrating Biology and the Bedside (i2b2) (<https://www.i2b2.org>). The HUBzero (<http://hubzero.org>) platform for scientific collaboration uses the Rapture toolkit to serve Java applets, employing the TeraGrid, the Open Science Grid, and other national

grid-computing resources for extra cycles. The collaborative environment of myExperiment (<http://myexperiment.org>) (discussed later) supports the sharing of scientific workflows and other research objects.

Missing so far is a common standard for the design of modular, compatible algorithm and tool plug-ins (also called modules or components) easily combined into scientific workflows (also called pipeline and composition). This leads to duplication of work, as even in the same project, different teams might develop several incompatible “plug-ins” that have almost identical functionality yet are incompatible. Plus, adding a new algorithm plug-in to an existing cyberinfrastructure or bundling and deploying a subset of plug-ins as a new tool/service requires extensive programming skills. Consequently, many innovative new algorithms are never integrated into common CIs and tools due to resource limitations.

Web sites like IBM's Many Eyes (<http://manyeyes.alphaworks.ibm.com/manyeyes/visualizations>) and Swivel (<http://swivel.com>) demonstrate the power of community data sharing and visualization. In 2009 alone, Many Eyes

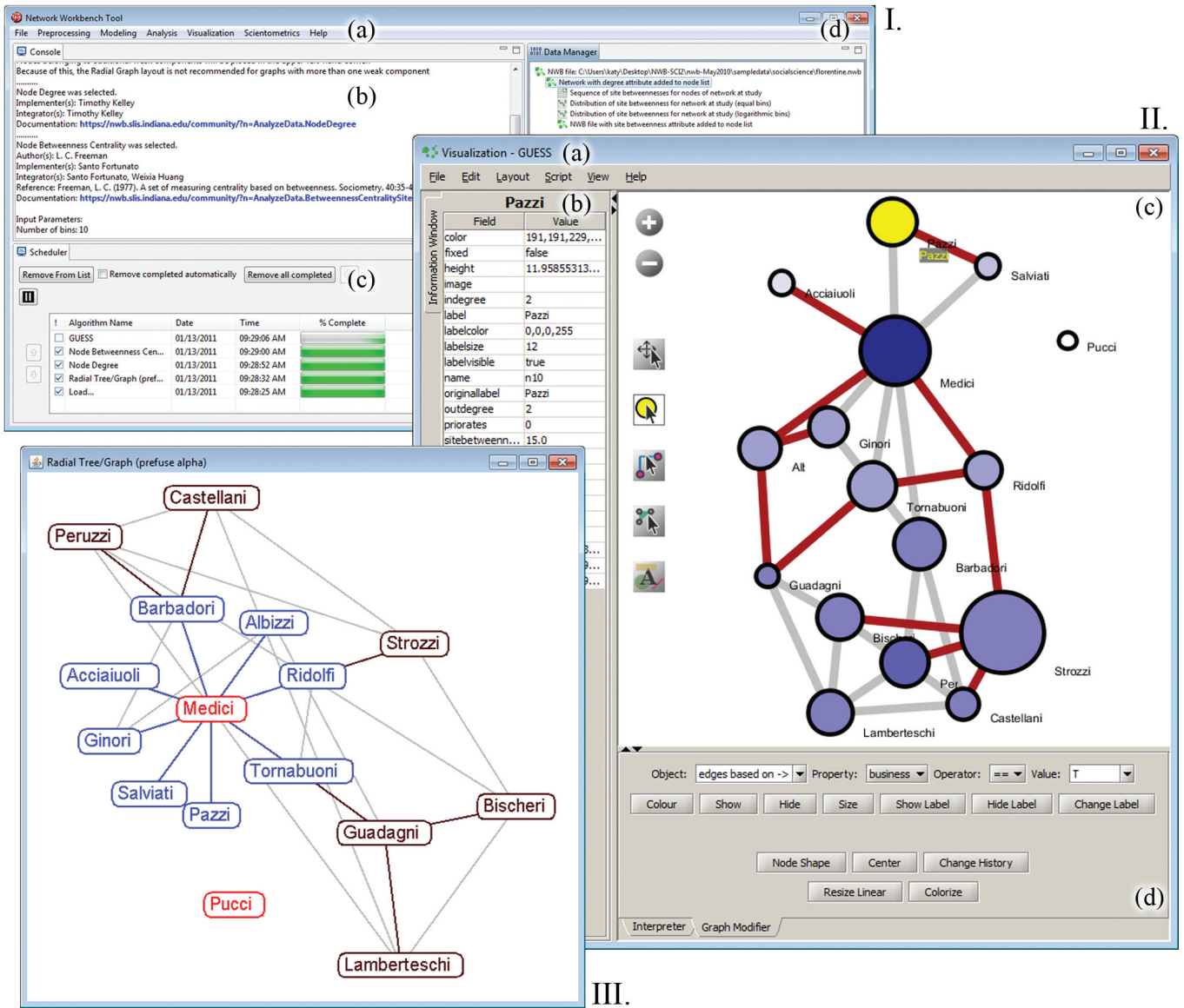


Figure 1. The NWB tool interface (I) with menu (a), Console (b), Scheduler (c), and Data Manager (d). The two visualizations of Renaissance Florentine families used the GUESS tool plug-in (II) and prefuse.org algorithm plug-in (III) Nodes denote families labeled by name; links represent marriage and business relationships among families. In GUESS, nodes are size-coded by wealth and color-coded by degree; marriage relationships are in red using the Graph Modifier (d). The “Pazzi” family in (c) was selected to examine properties in the Information Window (b).

had more than 66,429 data sets and 35,842 visualizations, while Swivel had 14,622 data sets and 1,949,355 graphs contributed and designed by 12,144 users. Both sites let users share data (not algorithms), generate and save different visualization types, and provide community support. In January 2011, the numbers for Many Eyes increased to 165,124 data sets and 79,115 visualizations, while Swivel ceased to exist.

Data analysis and visualization is also supported by commercial tools like Tableau (<http://tableausoftware.com>), Spotfire (<http://spotfire.tibco.com>), and free tools; see Börner et al.⁶ for a review of 20 tools and APIs. While

they offer valuable functionality and are widely used in research, education, and industry, none makes it easy for users to share and bundle their algorithms into custom macrosopes.

Plug-and-Play Software Architectures

When discussing software architectures for plug-and-play macrosopes, it is beneficial to distinguish among: (1) the “core architecture” facilitating the plug-and-play of data sets and algorithms; (2) the “dynamic filling” of this core comprising the actual algorithm, tool, user interface, and other plug-ins; (3) and the bundling of all components

into “custom tools.” To make all three parts work properly, it is important to understand who takes ownership of which ones and what general features are desirable (see the sidebar “Desirable Features and Key Decisions”).

Core architecture. To serve the needs of scientists (see both sidebars) the core architecture must empower non-programmers to plug, play, and share their algorithms and to design custom macrosopes and other tools. The solution proposed here is based on OSGi/CIShell:

Open Services Gateway Initiative. Developed by the OSGi Alliance (<http://osgi.org>), this service platform has

been used since 1999 in industry, including by Deutsche Telekom, Hitachi, IBM, Mitsubishi Electric, NEC, NTT, Oracle, Red Hat, SAP AG, and Siemens Enterprise Communications. It is a dynamic module system for Java, supporting interoperability of applications and services in a mature and comprehensive way with an effective yet efficient API. The platform is interface-based, easing plug-and-play integration of independent components by managing class and dependency issues when combining components. It is also dynamic; that is, new components can be added without stopping the program. It also comes with a built-in mechanism for retrieving new components through the Internet. As service-oriented architecture, OSGi is an easy way to bundle and pipeline algorithms into “algorithm clouds.” A detailed description of the OSGi specification and existing reference implementations is beyond the scope of this article but can be explored through <http://www.osgi.org/Specifications>.

Leveraging OSGi provides access to a large amount of industry-standard code—prebuilt, pretested, continuously updated components—and know-how that would otherwise take years to reinvent/re-implement, thus helping reduce time to market, development, and cost of maintenance. OSGi bundles can be developed and run using a number of frameworks, including the Equinox project from Eclipse (<http://eclipse.org/equinox>), the reference implementation of the OSGi R4 core framework. Eclipse includes extensive add-ons for writing and debugging code, interacting with code repositories, bug tracking, and software profiling that greatly extend the base platform.

Cyberinfrastructure Shell (<http://cishell.org>). This open-source software specification adds “sockets” to OSGi into which data sets, algorithms, and tools can be plugged using a wizard-driven process.¹¹ CIShell serves as a central controller for managing data sets and seamlessly exchanging data and parameters among various implementations of algorithms. It also defines a set of generic data-model APIs and persistence APIs. Extending the data-model APIs makes it possible to implement and integrate various data-model plug-ins. Each data model

requires a “persistor” plug-in to load, view, and save a data set from/to a data file in a specific format. Some data models lack a persistor plug-in, instead converting data to or from some other data format that does have one. CIShell also defines a set of algorithm APIs that allows developers to develop and integrate diverse new or existing algorithms as plug-ins.

Though written in Java, CIShell supports integration of algorithms written in other programming languages, including C, C++, and Fortran. In practice, a pre-compiled algorithm must be wrapped as a plug-in that implements basic interfaces defined in the CIShell Core APIs. Pre-compiled algorithms can be integrated with CIShell by providing metadata about their input and output. Various templates are available for facilitating integration of algorithms into CIShell. A plug-in developer simply fills out a sequence of forms for creating a plug-in and exports it to the installation directory and the new algorithm appears in the CIShell graphical user interface (GUI) menu. This way, any algorithm or tool that can be executed from a command line is easily converted into a CIShell compatible plug-in.

CIShell’s reference implementation also includes a number of basic services, including a default menu-driven interface, work-log-tracking module, a data manager, and a scheduler (see Figure 1, left). Work logs—displayed in a console and saved in log files—comprise all algorithm calls and parameters used, references to original papers and online documentation, data loaded or simulated, and any errors. The algorithm scheduler shows all currently scheduled or running processes, along with their progress. CIShell can be deployed as a standalone tool or made available as either a Web or peer-to-peer service. The CIShell Algorithm Developer’s Guide⁷ details how to develop and integrate Java and non-Java algorithms or third-party libraries.

OSGi/CIShell combined. Software designed using OSGi/CIShell is mainly a set of Java Archive bundles, also called plug-ins. OSGi services, CIShell services, and data set/algorithm services all run in the OSGi container. The CIShell framework API is itself an OSGi bundle that does not register OSGi ser-

vices, providing instead interfaces for data-set and algorithm services, basic services (such as logging and conversion), and application services (such as scheduler and data manager). Each bundle includes a manifest file with a dependency list stating which packages and other bundles it must run; all bundles are prioritized. Upon application start-up, the bundles with highest priority start first, followed by bundles of second, third, fourth,... priority. Bundles can also be started at runtime.

A bundle can create and register an object with the OSGi service registry under one or more interfaces. The services layer connects bundles dynamically by offering a “publish-find-bind” model for Java objects. Each service registration has a set of standard and custom properties. An expressive filter language is available to select relevant services. Services are dynamic; that is, bundles can be installed and uninstalled on the fly, while other bundles adapt, and the service registry accepts any object as a service. However, registering objects under (standard) interfaces (such as OSGi and CIShell) helps ensure reuse. Due to the declarative specification of bundle metadata, a distributed version of CIShell could be built without changing most algorithms.

The result is that domain scientists can mix and match data sets and algorithms, even adding them dynamically to their favorite tool. All plug-ins that agree on the CIShell interfaces can be run in software designed with the OSGi/CIShell core architecture. No common central data format is needed. Plug-ins can be shared in a flexible, decentralized fashion.

Dynamic filling. As of January 2011, the OSGi/CIShell plug-in pool included more than 230 plug-ins, including approximately 60 “core” OSGi/CIShell plug-ins and a “filling” of more than 170 algorithm plug-ins, plus 40 sample data sets, as well as configuration files and sample data files. Nearly 85% of the algorithm plug-ins are implemented in Java, 5% in Fortran, and the other 10% in C, C++, Jython, and OCaml; see <http://cishell.wiki.cns.iu.edu>.

Custom tools. The OSGi/CIShell framework is at the core of six plug-and-play tools that resemble simple microscopes and serve different sci-

entific communities; for example, the Information Visualization Cyberinfrastructure (IVC) was developed for research and education in information visualization; the Network Workbench (NWB) tool was designed for large-scale network analysis, modeling, and visualization; the Science of Science (Sci²) tool is used by science-of-science researchers, as well as by science-policy analysts; the Epidemics (EpiC) tool is being developed for epidemiologists; TEXTrend supports analysis of text; and DynaNets will be used to advance theory on network diffusion processes. Here, NWB and Sci² are covered in detail:

The NWB tool (<http://nwb.cns.iu.edu>) supports the study of static and dynamic networks in biomedicine, physics, social science, and other research areas. It uses 39 OSGi plug-ins and 18 CISHell plug-ins as its core architecture; two of them define the functionality of the simple GUI in Figure 1 (I), top left with the menu (I.a) for users to load data and run algorithms and tools. The *Console* (I.b) logs all data and algorithm operations, listing acknowledgment information on authors, programmers, and documentation URLs for each algorithm. The Data Manager (I.d) displays all currently loaded and available data sets. A Scheduler (I.c) lets users keep track of the progress of running algorithms. Worth noting is that the interface is easily branded or even replaced (such as with a command-line interface).

The NWB tool includes 21 converter plug-ins that help load data into in-memory objects or into formats the algorithms read behind the scenes. Most relevant for users are the algorithm plug-ins that can be divided into algorithms for preprocessing (19), analysis (56), modeling (10), and visualization (19). Three standalone tools—Discrete Network Dynamics (DND), GUESS, and GnuPlot—are available via the NWB menu system. GUESS is an exploratory data-analysis-and-visualization tool for graphs and networks (<http://graphexploration.cond.org>), as shown in Figure 1, II, containing a domain-specific embedded language called Gython (an extension of Python, or more specifically Jython) that supports the customization of graph designs. GnuPlot is a portable, command-line-driven,

interactive plotting utility for data and related functions (<http://gnuplot.info>). NWB uses 15 supporting libraries, including Colt, JUNG, Jython, and Prefuse (see Prefuse layouts in Figure 1, III); detailed listings are provided in the NWB tutorial³ and wiki (<http://nwb.wiki.cns.iu.edu>).

A common network-science workflow includes data loading and/or modeling, preprocessing, analysis, visualization, and export of results (such as tables, plots, and images). More than 10 different algorithms may be run in one workflow, not counting data converters. Common workflows and references to peer-reviewed papers are given in Börner et al.³ Here are six exemplary NWB workflows from different application domains:

- ▶ Error-tolerance and attack-tolerance analysis in physics and computer science requires loading or modeling a network and deleting random nodes (such as by error) or deleting highly connected hub nodes (such as in an attack);

- ▶ Peer-to-peer network analysis in computer science can include simulation of various networks and an analysis of their properties;

- ▶ Temporal text analysis in linguistics, information science, and computer science might apply the burst-detection algorithm to identify a sudden increase in the usage frequency of words, with results visualized;

- ▶ Social-network analysis in social science, sociology, and scientometrics might compare properties of scholarly and friendship networks for the same set of people; the scholarly network can be derived from publications and the friendship network from data acquired via questionnaires;

- ▶ Discrete network dynamics (biology) can be studied through the DND tool, which bundles loading and modeling a multistate discrete network model, to generate the model's state-space graph, analyze the attractors of the state space, and generate a visualization of an attractor basin; and

- ▶ Data conversion across sciences can use multiple converter algorithms to translate among more than 20 data formats.

Most workflows require serial application of algorithms developed in different areas of science and contributed

by different users. Much of the related complexity is hidden; for example, users do not see how many converters are involved in workflow execution. Only those algorithms that can be applied to a currently selected data set can be selected and run, with all others grayed out. Expert-workflow templates and tutorials provide guidance through the vast space of possible algorithm combinations.

The Science of Science tool (<http://sci2.cns.iu.edu>). The Sci² tool supports the study of science itself through scientific methods; science-of-science studies are also known as scientometric, bibliometric, or informetric studies. Research in social science, political science, physics, economics, and other areas further increases our understanding of the structure and dynamics of science.^{2,5,16} The tool supports the study of science at micro (individual), meso (institution, state), and global (all science, international) levels using temporal, geospatial, topical, network-analyses, and visualization techniques (<http://sci2.wiki.cns.iu.edu>).

Algorithms needed for these analyses are developed in diverse areas of science; for example, temporal-analysis algorithms come from statistics and computer science; geospatial-analysis algorithms from geography and cartography; semantic-analysis algorithms from cognitive science, linguistics, and machine learning; and network analysis and modeling from social science, physics, economics, Internet studies, and epidemiology. These areas have highly contrasting preferences for data formats, programming languages, and software licenses, yet the Sci² tool presents them all through a single common interface thanks to its OSGi/CISHell core. Moreover, new algorithms are added easily; in order to read a novel data format, only one new converter must be implemented to convert the new format into an existing format.

Multiple workflows involve more data converters than algorithms, as multiple converters are needed to bridge output and input formats used by consecutive algorithms. Workflows are frequently rerun several times due to imperfect input data, to optimize parameter settings, or to compare different algorithms. Thanks to the Sci² tool, an analysis that once required weeks

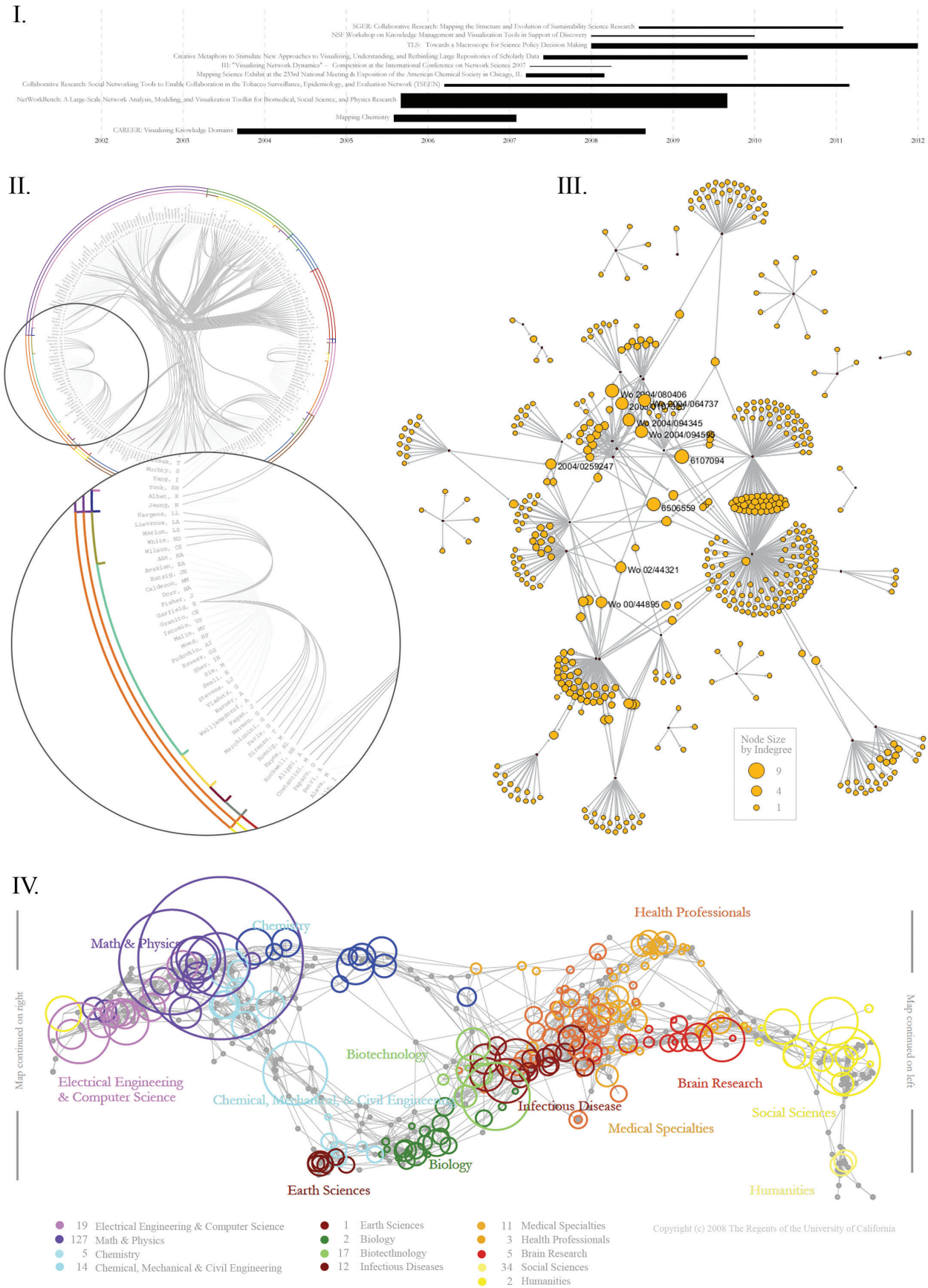


Figure 2. Exemplary Sci2 tool workflows: horizontal-bar-graph visualization of NSF funding for one investigator (I); circular layout of a hierarchically clustered co-author network of network-science researchers, with zoom into Eugene Garfield’s network (II); citation network of U.S. patents on RNAi and patents they cite, with highly cited patents labeled (III); and UCSD science base map with overlay of publications by network-science researchers (IV).

Changing Scientific Landscape

As science becomes increasingly data driven and computational, as well as collaborative and interdisciplinary, there is increased demand for tools that are easy to extend, share, and customize:

- ▶ *Star scientist* → *Research teams*. Traditionally, science was driven by key scientists. Today, science is driven by collaborating co-author teams, often comprising experts from multiple disciplines and geospatial locations^{5,17};
- ▶ *Users* → *Contributors*. Web 2.0 technologies empower users to contribute to Wikipedia and exchange images, videos, and code via Flickr, YouTube, and SourceForge.net. Wikispecies, WikiProfessionals, and WikiProteins combine wiki and semantic technology to support real-time community annotation of scientific data sets¹⁴;
- ▶ *Disciplinary* → *Cross-disciplinary*. The best tools frequently borrow and synergistically combine methods and techniques from different disciplines of science, empowering interdisciplinary and/or international teams of researchers, practitioners, and educators to collectively fine-tune and interpret results;
- ▶ *Single specimen* → *Data streams*. Microscopes and telescopes were originally used to study one specimen at a time. Today, many researchers must make sense of massive data streams of multiple data types and formats and of different dynamics and origin; and
- ▶ *Static instrument* → *Evolving cyberinfrastructure*. The importance of hardware instruments that are static and expensive tends to decrease relative to software tools and services that are highly flexible and evolving to meet the needs of different sciences. Some of the most successful tools and services are decentralized, increasing scalability and fault tolerance.

Good software-development practices make it possible for “a million minds” to design flexible, scalable software that can be used by many:

- ▶ *Modularity*. Software modules with well-defined functionality accept contributions from multiple users reduce costs and increase flexibility in tool development, augmentation, and customization;
- ▶ *Standardization*. Standards accelerate development, as existing code is leveraged, helping pool resources, support interoperability, and ease migration from research code to production code and hence the transfer of research results into industry applications and products; and
- ▶ *Open data and open code*. The practice of making data sets and code freely available allows users to check, improve, and repurpose data and code, easing replication of scientific studies.

or months to set up and run can now be designed and optimized in a few hours. Users can also share, rerun, and improve automatically generated work logs. Workflows designed, validated, and published in peer-reviewed works can be used by science-policy analysts and policymakers alike. As of January 2011, the Sci² tool was being used by the National Science Foundation, the National Institutes of Health, the U.S. Department of Energy, and private foundations adding novel plug-ins and workflows relevant for making decisions involving science policy.

The Sci² tool supports many different analyses and visualizations used to communicate results to a range of stakeholders. Common workflows and references to peer-reviewed papers are given in Börner et al.³ and the Sci² wiki (<http://sci2.wiki.cns.iu.edu>). Four

sample studies are discussed here and included in Figure 2, I–IV:

- ▶ Funding portfolios (such as funding received by investigators and institutions, as well as provided by agencies) can be plotted using a horizontal bar graph (HBG); for example, all funding for one researcher was downloaded from the National Science Foundation Award Search site (<http://nsf.gov/awardsearch>), loaded into Sci², and visualized in HBG, as in Figure 2, I. Each project is represented by a bar starting to the left at a certain state date and ending right at an end date, with bar width representing project duration. Bar-area size encodes a numeric property (here total awarded dollar amount), and equipment grants show as narrow bars of significant height. A label (here project name) is given to the left of the bar. Bars can be color-coded

by award type (such as Small Business Innovation Research and Career);

- ▶ Looking for collaboration patterns among major network-science researchers, the publications of four major researchers were downloaded from the *Web of Science* by Thomson Reuters (<http://wokinfo.com>). The data was then loaded into Sci², the co-author network extracted, the Blondel community-detection algorithm applied to extract hierarchical clusters of the network, and the result laid out using the Circular Hierarchy visualization, with author names plotted in a circle and connecting lines representing co-author links (see Figure 2, II). Two of the researchers share a combined network, while the others are at the centers of unconnected networks. Also shown is a zoom into Eugene Garfield’s network;

- ▶ To understand what patents exist on the topic of RNA interference (RNAi) and how they built on prior work, data was retrieved from the Scholarly Database (<http://sdb.cns.iu.edu>).⁶ Specifically, a query was run over all text in the U.S. patent data set covering 1976–2010. The U.S. Patent and Trademark Office citation table was downloaded, read into the Sci² tool, the patent-citation network extracted, the “indegree” (number of citations within the set) of all patent nodes calculated, and the network displayed in GUESS (see Figure 2, III). The network represents 37 patents (in red) matching the term RNAi and their and the 487 patents they cite (in orange). Nodes are size-coded by indegree (number of times a patent is cited); patents with at least five citations are labeled by their patent number. One of the most highly cited is no. 6506559 on “Genetic Inhibition by Double-Stranded RNA”; and

- ▶ The topical coverage of publication output is revealed using a base map of science (such as the University of California, San Diego map in Figure 2, IV.). The map represents 13 major disciplines of science in a variety of colors, further subdivided into 554 research areas. Papers are matched to research areas via their journal names. Multiple journals are associated with each area, and highly interdisciplinary journals (such as *Nature* and *Science*) are fractionally associated with multiple areas.

Circle size represents number of papers published per research area; the number of publications per discipline given below the map. The knowledge input (such as in terms of read or cited papers) and output (such as in terms of published or funded papers) of an individual, institution, or country can be mapped to indicate core competencies. Most publication output of the four network-science researchers is in physics.


These and many other Sci² analyses and corresponding visualizations are highly scalable; thousands of authors, references, and projects can be viewed simultaneously, and visualizations can be saved in vector format for further manipulation.

Macroscope Synergies


Just as the value of the earliest telephones increased in proportion to the number of people using them, plug-and-play macrosopes gain value relative to the increase in their core functionality; numbers of data-set and algorithm plug-ins; and the researchers, educators, and practitioners using and advancing them.

OSGi/CIShell-compliant plug-ins can be shared among tools and projects; for example, network-analysis algorithms implemented for the NWB tool can be shared as Java Archive files through email or other means, saved in the plug-in directory of another tool, and made available for execution in the menu system of that tool. Text-mining algorithms originally developed in TEXTrend (discussed later) can be plugged into the Sci² tool to support semantic analysis of scholarly texts. Though National Science Foundation funding for the NWB tool formally ended in 2009, NWB's functionality continues to increase, as plug-ins developed for other tools become available. Even if no project or agency were to fund the OSGi/CIShell core for some time, it would remain functional, due to it being lightweight and easy to maintain. Finally, the true value of OSGi/CIShell is due to the continuously evolving algorithm filling and the "custom tools" continuously developed and shared by domain scientists.

Over the past five years, a number of projects have adopted OSGi (and in two cases, CIShell):



As the functionality of OSGi/CIShell-based software frameworks improves, and as the number and diversity of data-set and algorithm plug-ins increases, so too will the capabilities of custom macrosopes.



► *Cytoscape* (<http://cytoscape.org>). Led by Trey Ideker at the University of California, San Diego, this open-source bioinformatics software platform enables visualization of molecular-interaction networks, gene-expression profiles, and other state data.¹⁵ Inspired by a workshop on software infrastructures in July 2007 (<https://nwb.slis.indiana.edu/events/ivsi2007>), Mike Smoot and Bruce W. Herr implemented a proof-of-concept OSGi-based Cytoscape core several months later; OSGi bundles are available at <http://chianti.ucsd.edu/svn/core3>. Once the new Cytoscape 3.0 core is implemented (projected mid-2011), sharing plug-ins between the NWB tool and Cytoscape will be much easier, thereby extending the functionality and utility of both;

► *Taverna Workbench* (<http://taverna.org.uk>). Developed by the myGrid team (<http://mygrid.org.uk>) led by Carol Goble at the University of Manchester, U.K., this suite of free open-source software tools helps design and execute workflows,¹² allowing users to integrate many different software tools, including more than 8,000 Web services from diverse domains, including chemistry, music, and social sciences. The workflows are designed in the Taverna Workbench and can then be run on a Taverna Engine, in the Workbench, on an external server, in a portal, on a computational grid, or on a compute cloud. Raven (a Taverna-specific classloader and registry mechanism) supports an extensible and flexible architecture (with approximately 20 plug-ins) but an implementation using an OSGi framework, with alpha release was scheduled for February 2011. The myExperiment (<http://myexperiment.org>) social Web site supports the finding and sharing of workflows and provides special support for Taverna workflows⁹;

► *MAEviz* (<https://wiki.ncsa.uiuc.edu/display/MAE/Home>). Managed by Shawn Hampton of the National Center for Supercomputing Applications, this open-source, extensible software platform supports seismic risk assessment based on Mid-America Earthquake Center research in the Consequence-Based Risk Management framework.¹⁰ It also uses the Eclipse Rich Client Platform, including Equinox, a com-

Desirable Features and Key Decisions

The socio-technical design of plug-and-play software architectures involves major decisions based on domain requirements to arrive at powerful tools and services:

- ▶ **Division of labor.** The design of a “core architecture” requires extensive computer science expertise and a close collaboration with domain experts. Data-set and algorithm plug-ins—the “filling”—are typically provided by domain experts most invested in the data and knowledgeable about the inner workings and utility of different algorithms. The design of “custom tools” is best performed by domain experts, as only they have the expertise needed to bundle different plug-ins relevant for diverse workflows. Technical manuals on how to use, improve, or extend the “core” need to be compiled by computer scientists, while data-set, algorithm, and tool descriptions are written by domain experts;
- ▶ **Ease of use.** As most plug-in contributions come from domain experts with limited programming skills, non-computer scientists must be able to contribute, share, and use plug-ins without having to write new code. What seems to work well is wizard-driven integration of algorithms and data sets, sharing of plug-ins through email and online sites, deploying plug-ins by adding them to the “plug-in directory,” and running them via a menu-driven user interface, as in word-processing systems and Web browsers;
- ▶ **Core vs. plug-ins.** The “core architecture” and the plug-in filling can be implemented as sets of plug-in bundles. Determining whether the graphical user interface, interface menu, scheduler, and data manager should be part of the core or its filling depends on the types of tools and services to be delivered;
- ▶ **Plug-in content and interfaces.** Should a plug-in be a single algorithm or an entire tool? What about data converters needed to make the output of one algorithm compatible with the input of another algorithm? Should they be part of the algorithm plug-in? Should they be packaged separately? What general interfaces are needed to communicate parameter settings, input, and output data? Answers are domain-specific, depending on existing tools and practices and the problems domain experts aim to solve;
- ▶ **Supported (central) data models.** Some tools (such as Cytoscape) use a central data model to which all algorithms conform. Others (such as NWB and Sci²) support many internal data models and provide an extensive set of data converters. The former often speeds execution and visual rendering, and the latter eases integration of new algorithms. In addition, most tools support an extensive set of input and output formats, since a tool that cannot read or write a desired data format is usually of little use by domain experts; and
- ▶ **Supported platforms.** Many domain experts are used to standalone tools (like MS Office and Adobe products) running on a specific operating system. A different deployment (such as Web services) is necessary if the software is to be used via Web interfaces.

ponent framework based on the OSGi standard (<https://wiki.ncsa.uiuc.edu/display/MAE/OSGi+Plug-ins>).

▶ **TEXTrend** (<http://textrend.org>). Led by George Kampis at Eötvös Loránd University, Budapest, Hungary, this E.U.-funded project is developing a framework for flexible integration, configuration, and extension of plug-in-based components in support of natural-language processing, classification/mining, and graph algorithms for analysis of business and governmental text corpuses with an inherently temporal component.¹³ In 2009, TEXTrend adopted OSGi/CIShell as its core architecture and has since added a number of plug-ins, including: the Unstructured Information Management Architecture ([\[incubator.apache.org/uima\]\(http://incubator.apache.org/uima\)\); the data-mining, machine-learning, classification, visualization toolset WEKA \(<http://cs.waikato.ac.nz/ml/weka>\); Cytoscape; Arff2xgmml converter; R \(<http://r-project.org>\) via iGgraph and scripts \(<http://igraph.sourceforge.net>\); yEd \(<http://yworks.com>\); and the CFinder clique percolation-analysis-and-visualization tool \(<http://cfinder.org>\). In addition, TEXTrend extended CIShell’s workflow support and now offers Web services to researchers.](http://</p>
</div>
<div data-bbox=)

▶ **DynaNets** (<http://www.dynanets.org>). Coordinated by Peter M.A. Sloot at the University of Amsterdam, The Netherlands, DynaNets is an E.U.-funded project for studying and developing a new paradigm of computing that employs complex networks. One related tool

under development is Dyneta, which uses OSGi/CIShell as its core to support the study of dynamically evolving networks. The tool is able to generate networks corresponding to different network models, execute a specific event chain on them, and analyze the interplay of network structure and dynamics at runtime. The tool will be used to develop a theory of spreading in networks (such as HIV infections and transmission of drug resistance). An initial set of plug-ins is available at <http://egg.science.uva.nl/dynanets/nightly/latest>.

As the functionality of OSGi/CIShell-based software frameworks improves and the number and diversity of data-set and algorithm plug-ins increase, so too will the capabilities of custom macroscopes.

Outlook

Instead of working at the Library of Alexandria, the Large Hadron Collider, or any of the world’s largest optical telescopes, many researchers have embraced Web 2.0 technology as a way to access and share images and videos, along with data sets, algorithms, and tools. They are learning to navigate, manage, and utilize the massive amounts of new data (streams), tools, services, results, and expertise that become available every moment of every day. Computer scientists can help make this a productive experience by empowering biologists, physicists, social scientists, and others to share, reuse, combine, and extend existing algorithms and tools across disciplinary and geospatial boundaries in support of scientific discovery, product development, and education. Computer scientists will have succeeded in the design of the “core architecture” if they are not needed for the filling or bundling of components into custom tools.

The Cyberinfrastructure for Network Science Center (<http://cns.iu.edu>) at Indiana University is working on the following OSGi/CIShell core extensions, as well as on more effective means for sharing data sets and algorithms via scholarly markets:

Modularity. The OSGi/CIShell core supports modularity at the algorithm level but not at the visualization level. Like the decomposition of workflows

into algorithm plug-ins, it is algorithmically possible to modularize visualization and interaction design. Future work will focus on developing “visualization layers” supporting selection and combination of reference systems, projections/distortions, graphic designs, clustering/grouping, and interactivity.

Streaming data. The number of data sets that are generated and must be understood in real time is increasing; examples are patient-surveillance data streams and models of epidemics that predict the numbers of susceptible, infected, and recovered individuals in a population over time. EpiC tool development funded by the National Institutes of Health contributes algorithms that read and/or output streams of data tuples, enabling algorithms to emit their results as they run, not only on completion. Data-graph visualizations plot these tuple streams in real time, resizing (shrinking) the temporal axis over time.

Web services. The OSGi/CIShell-based tools discussed here are stand-alone desktop applications supporting offline work on possibly sensitive data, using a GUI familiar to target users. However, some application domains also benefit from online deployment of macroscopes. While the OSGi specification provides basic support for Web services, CIShell must still be extended to make it easy for domain scientists to design their own macro-scope Web services.

Incentive design. Many domain experts have trouble trying to use an evolving set of thousands of possibly relevant data sets compiled for specific studies of inconsistent quality and coverage, saved in diverse formats, and tagged using terminology specific to the original research domains. In addition, thousands of algorithms that support different functionality and diverse input and output formats are written in different languages by students and experts in a range of scientific domains and packaged as algorithms or tools using diverse licenses. More-effective means are needed to help domain experts find the data sets and algorithms most relevant for their work, bundle them into efficient workflows, and relate the results to existing work. Scholarly markets resembling a Web 2.0

version of Craigslist.org can help ease the sharing, navigation, and utilization of scholarly data sets and algorithms, reinforcing reputation mechanisms by, say, providing ways to cite and acknowledge users who share, highlight most downloaded and highest-rated contributions, and offer other means for making data sets, algorithms, workflows, and tutorials part of a valued scholarly record.

Acknowledgments

I would like to thank Micah Linnemeier and Russell J. Duhon for stimulating discussions and extensive comments. Bruce W. Herr II, George Kampis, Gregory J. E. Rawlins, Geoffrey Fox, Shawn Hampton, Carol Goble, Mike Smoot, Yanbo Han, and anonymous reviewers provided valuable input and comments to an earlier draft. I also thank the members of the Cyberinfrastructure for Network Science Center (<http://cns.iu.edu>), the Network Workbench team (<http://nwb.cns.iu.edu>), and Science of Science project team (<http://sci2.cns.iu.edu>) for their contributions toward this work. Software development benefits greatly from the open-source community. Full software credits are distributed with the source, but I especially acknowledge Jython, JUNG, Prefuse, GUESS, GnuPlot, and OSGi, as well as Apache Derby, used in the Sci2 tool. This research is based on work supported by National Science Foundation grants SBE-0738111, IIS-0513650, and IIS-0534909 and National Institutes of Health grants R21DA024259 and 5R01MH079068. Any opinions, findings, and conclusions or recommendations expressed here are those of the author and do not necessarily reflect the views of the National Science Foundation. C

References

1. Atkins, D.E., Drogemeier, K.K., Feldman, S.I., Garcia-Molina, H., Klein, M.L., Messerschmitt, D.G., Messian, P., Ostriker, J.P., and Wright, M.H. *Revolutionizing Science and Engineering Through Cyberinfrastructure. Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure.* National Science Foundation, Arlington, VA, 2003.
2. Börner, Katy. *Atlas of Science: Visualizing What We Know.* MIT Press, Cambridge, MA, 2010; supplemental material at <http://scimaps.org/atlas>
3. Börner, K., Barabási, A.-L., Schnell, S., Vespignani, A., Wasserman, S., Wernert, E.A., Balcan, D., Beiró, M., Biberstine, J., Duhon, R.J., Fortunato, S., Herr II, B.W., Hidalgo, C.A., Huang, W.B., Kelley, T., Linnemeier, M.W., McCranie, A., Markines, B., Phillips, P., Ramawat, M., Sabbineni, R., Tank, C., Terkhorn, F., and Thakre, V.

Network Workbench Tool: User Manual 1.0.0., 2009; <http://nwb.cns.iu.edu/Docs/NWBTool-Manual.pdf>

4. Börner, K., Chen, C., and Boyack, K.W. Visualizing knowledge domains. In *Annual Review of Information Science & Technology*, B. Cronin, Ed. Information Today, Inc./American Society for Information Science and Technology, Medford, NJ, 2003, 179–255.
5. Börner, K., Dall’Asta, L., Ke, W., and Vespignani, A. Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams. *Complexity (Special Issue on Understanding Complex Systems)* 10, 4 (Mar/Apr. 2005), 57–67.
6. Börner, K., Huang, W.B., Linnemeier, M., Duhon, R.J., Phillips, P., Ma, N., Zoss, A., Guo, H., and Price, M.A. Rete-Netzwerk-Red: Analyzing and visualizing scholarly networks using the Network Workbench tool. *Scientometrics* 83, 3 (June 2010), 863–876.
7. Cyberinfrastructure for Network Science Center. *Cyberinfrastructure Shell (CIShell) Algorithm Developer’s Guide*, 2009; <http://cishell.wiki.cns.iu.edu>
8. de Rosnay, J. *Le Macroscopie: Vers une Vision Globale. Editions du Seuil.* Harper & Row Publishers, Inc., New York, 1975.
9. De Roure, D., Goble, C., and Stevens, R. The design and realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows. *Future-Generation Computer Systems* 25 (2009), 561–567.
10. Elnashai, A., Hampton, S., Lee, J.S., McLaren, T., Myers, J. D., Navarro, C., Spencer, B., and Tolbert, N. Architectural overview of MAEviz-HAZTURK. *Journal of Earthquake Engineering* 12, 1 Suppl.2, 01 (2008), 92–99.
11. Herr II, B.W., Huang, W.B., Penumathy, S., and Börner, K. Designing highly flexible and usable cyberinfrastructures for convergence. In *Progress in Convergence: Technologies for Human Wellbeing*, W.S. Bainbridge and M.C. Roco, Eds. Annals of the New York Academy of Sciences, Boston, 2007, 161–179.
12. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., and Oinn, T. Taverna: A tool for building and running workflows of services. *Nucleic Acids Research (Web Server Issue)* 34, Suppl. 2 (July 1, 2006), W729–W732.
13. Kampis, G., Gulyas, L., Szasz, Z., and Szokolci, Z. Dynamic social networks and the TEXTrend/CIShell framework. Presented at the Conference on Applied Social Network Analysis (University of Zürich, Aug. 27–28), ETH Zürich, Zürich, Switzerland, 2009.
14. Mons, B., Ashburner, M., Chicester, C., Van Mulligen, E., Weeber, M., den Dunnen, J., van Ommen, G.-J., Musen, M., Cockerill, M., Hermjakob, H., Mons, A., Packer, A., Pacheco, R., Lewis, S., Berkeley, A., Melton, W., Barris, N., Wales, J., Meijssen, G., Moeller, E., Roes, P.J., Börner, K., and Bairoch, A. Calling on a million minds for community annotation in WikiProteins. *Genome Biology* 9, 5 (2008), R89.
15. 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 integrating models of biomolecular interaction networks. *Genome Research* 13, 11 (2002), 2498–2504.
16. Shiffrin, R. and Börner, K. Mapping knowledge domains. *Proceedings of the National Academy of Sciences* 101, Suppl. 1 (Apr. 2004), 5183–5185.
17. Shneiderman, B. Science 2.0. *Science* 319, 5868 (Mar. 2008), 1349–1350.
18. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages* (Boulder, CO, Sept. 3–6). IEEE Computer Society, Washington, D.C., 1996, 336–343.
19. Thomas, J.J. and Cook, K.A., Eds. *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* National Visualization and Analytics Center, Richland, WA, 2005; <http://nvac.pnl.gov/agenda.stm>
20. World Bank and International Monetary Fund. *Global Monitoring Report 2009: A Development Emergency.* The World Bank, Washington, D.C., 2009.

Katy Börner (katy@indiana.edu) is the Victor H. Yngve Professor of Information Science at the School of Library and Information Science, Adjunct Professor at the School of Informatics and Computing, and Founding Director of the Cyberinfrastructure for Network Science Center (<http://cns.iu.edu>) at Indiana University, Bloomington, IN.