Huang, W., Herr, B., Penumarthy, S., Markines, B., Börner, K. (2006)
CIShell: A Plug-in Based Software Architecture and Its Usage to Design an Easy to Use, Easy to Extend Cyberinfrastructure for Network Scientists.
Proceedings of the International Conference and Workshop on Network Science (NetSci2006), Bloomington, Indiana.

CIShell -- A Plug-in Based Software Architecture and Its Usage to Design an Easy to Use, Easy to Extend Cyberinfrastructure for Network Scientists

Weixia Huang*, Bruce Herr*, Shashikant Penumarthy*, Ben Markines+, Katy Börner*

*School of Library and Information Science
Indiana University

{huangb | bherr | sprao | katy}@indiana.edu

+Department of Computer Science

Indiana University

bmarkine@cs.indiana.edu

ABSTRACT

Today, research communities comprise scholars and practitioners from multiple disciplines. An example is network science that aims at the study of small, medium, and large-scale network datasets collected in social and behavioral science, physics, biology, and other disciplines.

At an increasing rate, the scientists that invent and implement new algorithms are not computer scientists. Yet, the algorithms they invent and implement are useful within and outside of their own disciplines. The question then becomes: How can those hundreds and thousands of individual algorithms that are programmed in different languages for different purposes and data formats most effectively be made available to non-programmer users spread out across multiple disciplines?

This paper introduces CIShell – a plug-in based software architecture that supports the plug and play of algorithms and the handling of diverse data formats via data model transformation. We describe the CIShell architecture and its key features and exemplify its usage in the recently funded Network Workbench project. We conclude with an outlook of planned work.

Keywords

Cyberinfrastructure Design, Plug-in, Usability, Extensibility, Data Models, Analysis, Network Science.

1. INTRODUCTION

Fifty years back in time, scientists did not use any computer to conduct their research. Today, science without computation is unthinkable in almost all areas of science. Many scientists use commercial packages such as MS Excel, Matlab, UCINet[1], Pajek[2] to analyze, model, or visualize their data. However, there are a growing number of grass roots efforts that aim to make the newest, best algorithms developed by the researchers themselves available to other scientists. Examples are R[3], StOCNet[4], Jung[5], Prefuse[6], and Processing[7]. However, many, if not all, of the mentioned packages come as APIs or require scripting of code. None of them supports the easy, wizard based integration of new algorithms and datasets by developers or provides a menu driven, easy to use interface for the general user. Yet, many computer scientists and programmers who have been developing new algorithm implementations are searching for possible utilities to quickly disseminate their work. On the other side, many researchers and educators today are in need of making sense of small, medium and large scale digital datasets yet are not equipped with mathematical sophistication and programming knowledge.

CIShell was designed and implemented to bridge between algorithms developers and algorithm users. It is a plug-in based cyberinfrastructure (CI) including a set of APIs and templates that support the easy integration and dissemination for existing and new algorithms, deliver a mechanism to share the multitude of data with different formats, as well as provide an extensible menu-driven user interface.

2. SYSTEM ARCHITECTURE of CIShell

CIShell was designed to serve as the core for developing data-code-computing CIs for different scientific communities. Some basic requirements of CIShell include (1) integrating various algorithms, (2) handling different data formats, (3) accessing and processing large-scale datasets efficiently, and (4) extending and managing the menu-driven user interface. Based on the above requests, Figure 1 shows the system architecture. Individual components are briefly described in the following subsections.

2.1 A Plug-in Based Architecture

CIShell can be conceptualized as an empty shell that allows various algorithms and datasets to be integrated and run as plug-ins. CIShell also serves as a central controller to manage datasets, to seamlessly exchange the data and parameters among various implementations of algorithms, and to provide the basic services such as the menu driven interface, work log tracking, scheduling, etc. It contains several key components:

- A **menu-driven interface** supports file load, view, conversion, and save as well as the selection of diverse preprocessing, analysis, modeling, and visualization algorithms on the loaded data.
- A **work log tracking** module records the sequence of steps performed by a user such as what file is loaded or saved, what algorithm is run, or what preferences are changed. The log is displayed in the console and is also saved as a record in a log file. Error logs are saved in a separate file.

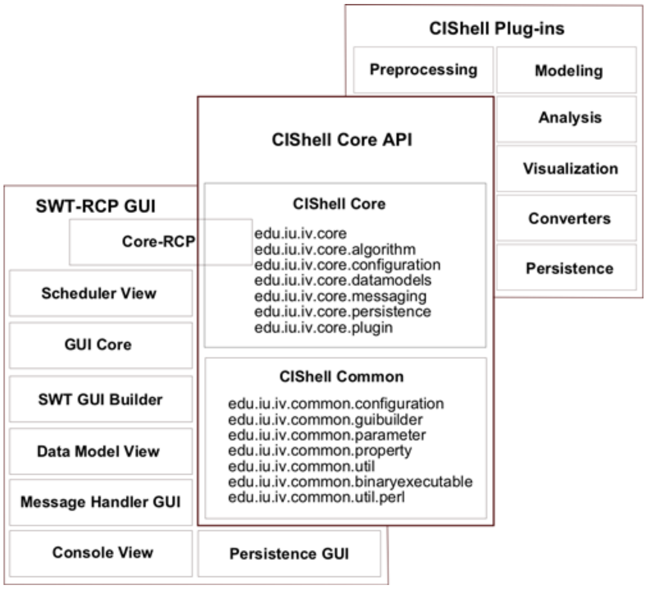

Figure 1. CIShell System Architecture

- **Preferences** can be set to customize the behavior of any plug-in, including the interface, scheduler, data persisters, or algorithms.
- A **scheduler** lets a user run algorithms at a particular date and time and in a specified sequence, which is particularly valuable for computationally demanding jobs. The number and type of algorithms that runs in series or in parallel is only restricted by the amount of memory and processing power available. At any point in time users can see all currently scheduled or running processes, monitor their progresses, and also change the sequence of algorithms scheduled for execution.
- A set of **core APIs** are defined as the extension points for developing algorithm plug-ins, data model plug-ins and persistence plug-ins.

2.2 Data Models

CIShell defines a set of generic data model APIs and persistence APIs. By extending the data model APIs, various data model plug-ins can be implemented and integrated. Each data model requires developing a persister plug-in to load, view, and save a dataset from/to a data file in a certain format. CIShell provides several templates that can guide plug-in developers step by step in the creation of a plug-in. Existing templates can be used to develop data model plug-ins that are highly scalable, e.g., that can represent a graph with millions of nodes without loading the whole graph into the memory. CIShell can support an unlimited number of data models and data formats. Data model converter plug-ins are used to convert one data model to another. Several converters have been developed to conduct the transformation between diverse data models, which facilitates the easy setup of pipelines of data modeling, analysis and visualization algorithms that expect different types of data models. For example, a converter plug-in that transforms the ‘Matrix’ data model to the ‘JUNG Graph’ data model has been developed so that users can use the ‘Spring Layout’ and ‘Kamada-Kawai Layout’ algorithms provided by the JUNG library to visualize the network dataset originally stored in the ‘Matrix’ format. With those converters, the framework can potentially apply network data in any standard format to various algorithms.

2.3 Algorithm Plug-ins

CIShell also defines a set of algorithms APIs that allows developers to easily develop and integrate diverse new or existing algorithms as plug-ins. While CIShell itself is written in JAVA it supports the integration of algorithms written in other programming languages, e.g., in C++ or Fortran. In practice, a pre-compiled algorithm needs to be wrapped as a plug-in that implements basic interfaces defined in the CIShell Core APIs. Different templates are available to facilitate the integration of diverse algorithms into the CIShell. In most cases, no programming is required to integrate an algorithm as a new plug-in. A plug-in developer simply needs to fill out a sequence of forms for creating a plug-in, export the plug-in to the installation directory, and then users are ready to use the new algorithm via the CIShell interface menu.

In the Network Workbench (NWB) tool prototype of CIShell, we have successfully developed and integrated several classic network modeling algorithms, such as ‘Barabasi-Albert’ model, ‘Watts-Strogatz small world’ model and ‘Erdös Random Graph’ model. Several network analysis algorithms have also been integrated into the NWB tool, including directed degree distribution, total degree distribution, directed/undirected k-nearest neighbor, one point correlations, clustering, and page rank algorithm. Since those algorithms are implemented in Fortran, they deliver great performance on network modeling and analysis. A large-scale network, such as a data model with millions of nodes, can be generated and measured in several seconds using the NWB tool.

3. Planned Work

We are in the process of extending the functionality of CIShell to better support the integration of network analysis, modeling and visualization algorithms. For instance, we work on automatically generating plug-ins from code using drag and drop. A generic way to build bi-directed converters as bridges between well-defined data models is in discussion. Eventually, we expect to make an automatic data model transformation module in which a dataset can be transparently converted to a format requested by a specific algorithm. The other direction is to replace the existing SWT-RCP GUI solution (shown in Figure 1) with the web-based GUI solution so that CIShell can serve as a core piece for both web-based applications and standalone Java applications.

4. ACKNOWLEDGMENTS

The project is supported in part by the 21st Century Fund and the National Science Foundation under Grant No. DUE-0333623, IIS-0238261 and IIS-0513650.

5. REFERENCES

[1] Borgatti, S.P., Everett, M.G. and Freeman, L.C. 2002. UCINET for Windows: Software for Social Network Analysis. Harvard: Analytic Technologies.

[2] Batagelj, V., Mrvar A., (1998) Pajek - Program for Large Network Analysis. Connections, 21(1998)2, 47-57.

[3] http://www.r-project.org/

[4] http://stat.gamma.rug.nl/stocnet/

[5] http://jung.sourceforge.net/

[6] http://prefuse.sourceforge.net/

[7] http://processing.org/