Development and Evaluation of AGNI:

A Distributed Netgraph Data Collection and Analysis Tool

by

Ian J. C. Duggan

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Compuer Science

at the Massachusetts Institute of Technology

August 24, 1998

Author .....
Department of Electrical Engineering and Computer Science
August 24, 1998

Certified by .....
Thomas J. Allen
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Eng

Development and Evaluation of AGNI:
A Distributed Netgraph Data Collection and Analysis Tool

by

Ian J. C. Duggan

Submitted to the
Department of Electrical Engineering and Computer Science

August 24, 1998

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

AGNI is a tool developed to facilitate the analysis of communication networks between people in large organizations. Determining patterns of communication within organizations is critical to the analysis of the effectiveness of their structure. Until recently, large organizations presented a special problem. The copious amounts of data that had to be analyzed made the process slow and tedious. AGNI makes the job easier by automating many of the tasks involved in this process. It provides a user friendly graphical environment in which data can be collected and analyzed. It also provides an easy way to display the data in a graphical format so that it is easy to visualize.

Thesis Supervisor: Thomas J. Allen
Title: Professor of Management, MIT Sloan School of Management

Dedication

To my loving parents, Lynn and Jim, and to my brother Jordan for their loving support and encouragement throughout the years.

Acknowledgments

I would like to thank Professor Thomas J. Allen for supervising this thesis and for being friendly and understanding.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

This paper serves several purposes. It is primarily a Masters thesis and thus exists to document work performed on AGNI over the past year. It also serves as a reference for future programmers. It can be used to understand the new architecture as well as how future parts of the project can be constructed. It is hoped that a thorough understanding of AGNI via this document will prevent or at least allay the frustrations associated with trying to absorb several thousand lines of working code, an experience which occurred this past year. Finally, this document can be used by those wishing to familiarize themselves with the AGNI project in order to develop an understanding of how the program can be used effectively.

The first section of this paper provides the reader with some background on the study of organizational communications. A short summary of the reasons that such study is useful is provided in the background chapter. The background chapter discusses how the movement of production from rural communities into the cities spurned large corporations and thus the development of complex organizational structures.

The next topic covered is a history of the work that has been done to date in the field of organizational science, with particular attention placed on the methods used to visualize human networks. A short overview of the techniques developed during the past century is provided. The principal methods covered are sociograms, adjacency matrices, and netgraphs.

The following chapters go on to discuss the most recent work done at the Sloan School of Management at MIT. The discussion is focused on the AGNI project which has been developed over the past 15 years. A brief introduction to the AGNI project is provided followed by a description of the state of the project just before work on this

thesis began. An evaluation of the methods used in by AGNI for data collection, analysis, and visualization is provided along with observations about drawbacks to the system.

The remaining chapters discuss the changes to the system that were proposed for this thesis and the work done on the project to implement those changes. A discussion of the relative merits of each of the changes is provided. The architecture of the new system is provided first as an overview and then as a low level description of the processes involved in the system.

AGNI existed solely as a stand-alone Unix application when work on it for this paper first began. The contribution of this thesis was essentially to modernize the entire AGNI system. The interface was converted to an object-oriented architecture utilizing a library that can run natively on both Unix and Windows NT. The data collection methods were updated to take advantage of the widespread popularity of the world wide web. This was done by incorporating a webserver into the AGNI system. Lastly, the new architecture designed for AGNI utilizes a database as the central hub of the system. Data collected via the web is entered directly into the database and a client interface is used to visualize data retrieved from the database. Enhanced reusability resulting from the redesign of the system should allow AGNI to be extended and enhanced more successfully.

Concluding thoughts are provided at the end as to the success of the work and the value of the final system. Ideas for future work are provided along with reason for why they would be beneficial to the project.

Chapter 2: Background

Large organizations are confronted with the problem of how to ensure efficient communication among their members. Tracking and analyzing communication patterns within them has always been difficult. Managers have long sought surefire methods for fine tuning their organizations for maximum productivity. Unfortunately, these managers have been forced to rely mostly on common sense and vague reasoning when allocating resources. It was not until the latter half of this century that researchers began formal studies of the patterns of communication within large organizations.

The Industrial Revolution moved the primary centers of production from rural communities and brought them into the cities. These new groupings of workers, companies and corporations brought with them new organizational challenges. As they grew larger and divisions of labor were created, they were split up into manufacturing, accounting, marketing, and managerial roles. It used to be that the rules of engagement in organizations were fairly clear. Innovations were slow to move into the marketplace, and a large cumbersome organization could anticipate changes to the market environment long before they were affected by them. Managerial efficiency was not a problem.

Enter the modern era of competition. Since the beginning of this century, the economies of the world have been moving forward with innovations and advances at an ever increasing rate. The pace of the business environment has made it so that slow moving companies are unable to compete with those that can adapt readily to changes in the marketplace. Managers have realized that the internal organization of a company ultimately affects its overall productivity and profitability. Knowledge of organizational efficiency is now a financial boon, so many companies have invested in its study.

Over the years, a number of techniques have been used to tackle the problem of organizational management. The earliest attempts were made by statisticians who applied known techniques for working with data sets to these organizations. The problem with statistics is that it gives one lot of information about a data set, but it does not necessarily aid in identifying the patterns within the data. The data associated with the internal communications of a group is heavily interdependent and fraught with complicated dynamics, making it difficult to visualize. Attempts at the study of organizational data tried to overcome the visualization problem by developing techniques to view the data. The AGNI project, about which this paper is concerned, is one of the tools developed to help identify patterns in this kind of data.

Chapter 3: History of the Problem

The formal study of communication structures within large organizations lay largely untouched until German sociologists in the 1930's began to study human group behavior (Krebs, 1996) It was about the same time that anthropologists began making networks out of the social patterns of interaction found in primitive societies as well. These researchers quickly discovered that the data being studied was of a complex nature. If the groups under study grew too large, the amount of data involved became unwieldy

Fortunately for the researchers, the computer was coming onto the scene at the same time. It provided a means to access and manipulate much larger quantities of data than was previously possible. As algorithms for analyzing human networks were developed and implemented on computers, researchers were slowly able to form meaningful conclusions about the networks. They did this by creating the tools needed to work with the data generated by the study of complex human interactions.

Recent studies done in regards to understanding the nature of communication networks have been aimed at improving productivity among workers. Research and development organizations are a prime targets for this type of research because they are dependent on the talents and creativity of many workers at the same time. For workers to stay in top form, they must constantly be exposed to new ideas and ways of doing things. Otherwise, their knowledge grows stale and their effectiveness declines. This is especially the case in industries such as software development where product life cycles are on the order of a few years. If a worker does not keep abreast of emerging technologies, his or her skills become obsolete. Employees are assets, and companies need to maintain the effectiveness of their workers.

Many factors have come together in the last twenty years that have pushed the study of human organization. The introduction and widespread use of the computer has made it possible for researchers to handle the large amounts of data involved. The quick pace of the marketplace has made the understanding of organizational communication patterns a competitive advantage, motivating companies to sponsor research.

Early researchers were forging into new territory and did not have the vast array of tools now available for data analysis. Their only real tools were statistical analysis methods, so while their studies were largely empirical in nature, they lacked the ability to critically explain the data. No doubt, these early researchers also found the task of analyzing the wealth of information produced by human interactions to be prohibitive. The following is an overview of the techniques developed by past researchers.

Sociometry and Network Representation

Some of the first real innovations in the study of human networks were made by Moreno who chose to represent networks of humans in a way familiar to anyone who might have studied graph theory in school (Levi, 1990). Moreno represented individuals as circles or nodes in a graph and social relationships by placing arrows or links between the nodes or individuals. These directed graphs, or digraphs, can also be used to represent relationships among individuals in an organization. An example of a sociogram is shown in Figure 3.1. A relation between people is shown by a singular arrow. If node A has an arrow that points to node B, but node B doesn't have a reciprocating arrow, then there is an asymmetrical relationship between A and B. A symmetrical relationship would be represented by either a pair of reciprocating arrows, one from A to B and one from B to A, or by an arrow with a head on each end. The relative strengths of connections between nodes can be represented by varying the thickness or the color of the links. Allen (1967) adapted this method to represent communications networks as mathematically formulated

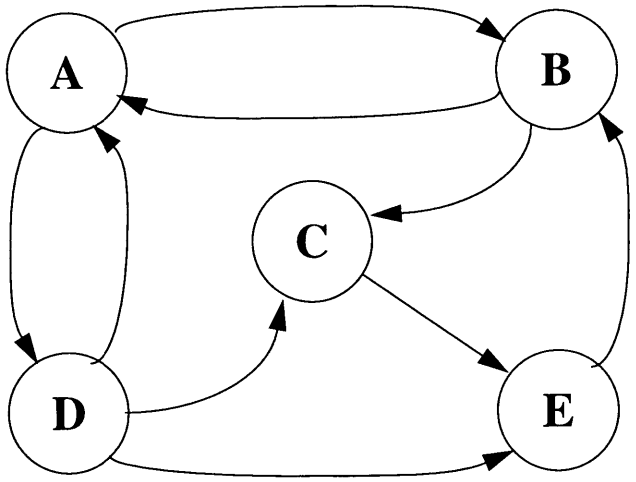


Figure 3.1: Example of a Sociogram

graphs. In this way, one is immediately able to reason about the network by utilizing any of the numerous algorithms developed for analyzing the properties of directed graphs. Algorithms exist for calculating the centrality of nodes, their hierarchy, the mean distance between nodes, cliques or groupings within the graph as well as many others. The primary benefit of sociograms was the visual aspect they added to analysis. By constructing the sociogram on paper, one could easily see many of the relationships that are not so obvious from the raw numerical data. For example, one can easily identify a group of individuals in a graph by noticing that there are several nodes with redundant links to each other. It is immediately recognizable that this group operates as a unit and is in close communication with itself. It is not so easy to garnish this information from the raw data. Consider Figure 3.2 and Table 3.1. Try to identify the same group in each. It is clearly easier to identify the group in the sociogram since we have the aid of our visual senses.

Sociograms, though a great step beyond raw data representations and statistical analysis, have their drawbacks. Though they allow one to easily visualize many properties within a graph, they can be difficult to create and maintain when the number of nodes and links grows large. Consider Figure 3.3 which is an actual network constructed by Allen

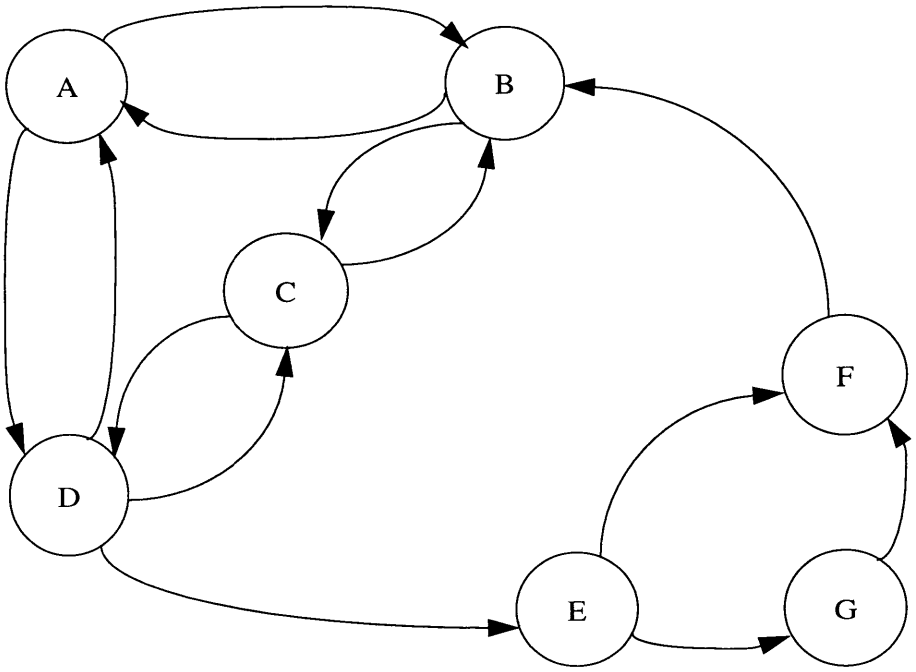


Figure 3.2: Sociogram revealing group structure.

Person	Talks To
A	B
A	C
A	D
B	A
B	C
C	A
C	B
C	D
D	A
D	C
D	E
E	F
E	G
F	B
F	G

Table 3.1: Table indicating communications in a network.

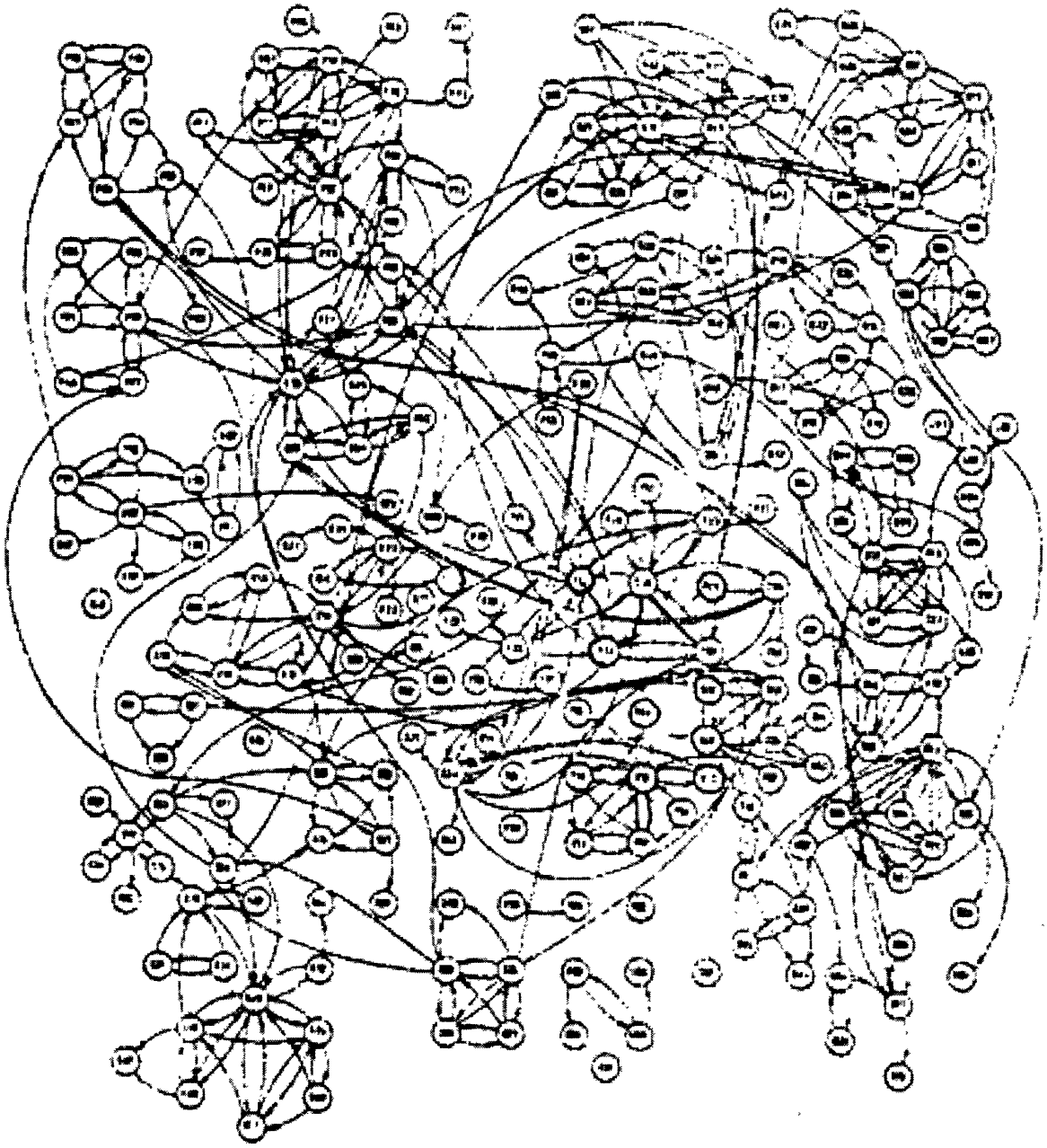


Figure 3.3: Sociogram showing communication in a complex organization (Allen, 1986)

(1986). Creating networks of this size requires a lot of patience as it is often necessary to redo the diagram many times until the nodes are arranged in a way that makes the groupings obvious. Another drawback is that no two people are likely to build the same appearance of a sociogram from the same data set. Sociograms and networks are excellent for visualizing relationships, but they do not do much to organize the data in an easily reproducible format. It became clear to researchers that better tools were needed to manipulate these graphs.

Adjacency Matrices

Computers are wonderful tools when it comes to processing large amounts of numerical data, but it is sometimes problematic to represent the data in a format that can be understood by a computer. It is not easy to make a computer understand the spatial relationships of sociograms, but it is possible to have them manipulate matrices of numbers. Researchers developed a representation for directed graphs known as an adjacency matrix. They noticed that it is possible to represent a graph of N nodes and M links as an N x N matrix of numbers. The matrix is formed by listing one row and one column for each individual to be represented. If two cells in a network are adjacent to each other, meaning that there is a link between them, then the matrix cell corresponding to that link is filled in with the weight of the link.

Adjacency matrices were an innovation because they allowed researchers to numerically specify the structure of human networks and manipulate them via computer. With the aid of the computer, large complex analyses could be carried out on the data. In addition to the computational benefits that come from storing graph data in adjacency matrix form is the added bonus of the organization created by the matrix format. Any particular adjacency matrix becomes uniquely specified when one defines a permutation

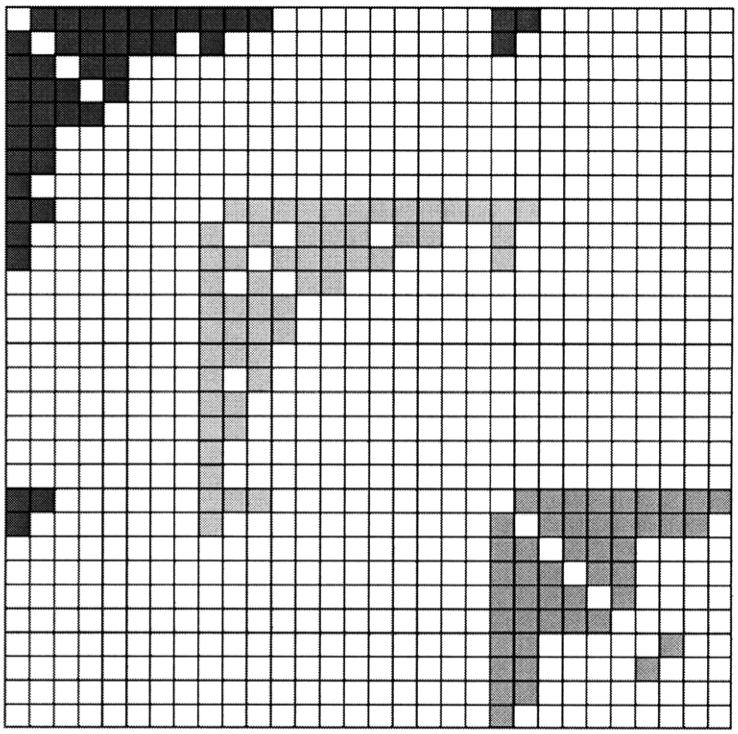


Figure 3.4: Example of a netgraph with 40 individuals represented.

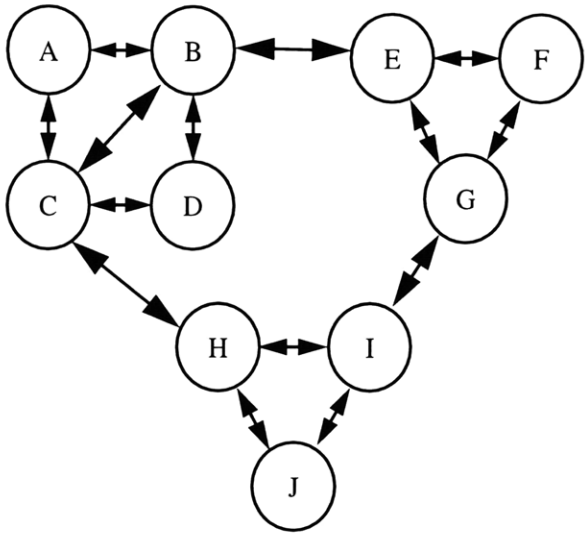
for its rows and columns. The same set of data could now be used to produce identical adjacency matrices, something which was difficult to do with sociograms.

Netgraphs

Adjacency matrices provided researchers with a means of manipulating data via a computer, but they sacrificed the appealing visual quality of the sociograms. To compensate for this, a middle ground between the visual features of sociograms and the computational efficiency of adjacency matrices was developed in the form of a netgraph. It is created by taking an adjacency matrix and coloring in the cells that represent links. An example of a netgraph is shown in Figure 3.4. To get an idea of how the different representations compare to each other, see Figure 3.5 which shows the same set of data represented in raw format, in a sociogram, in an adjacency matrix, and in the form of a

Person	Person Talked To	Person	Person Talked To
A	B	E	G
A	C	F	E
A	D	F	G
B	A	G	E
B	D	G	F
B	E	G	I
C	A	H	C
C	D	H	I
C	H	H	J
D	A	I	G
D	B	I	H
D	C	I	J
E	B	J	H
E	F	J	I

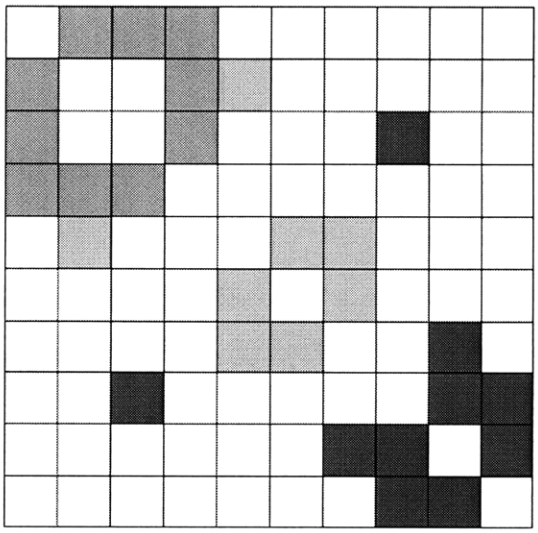
Data in Raw Format



Sociogram

0	1	1	1	0	0	0	0	0	0
1	0	0	1	2	0	0	0	0	0
1	0	0	1	0	0	0	3	0	0
1	1	1	0	0	0	0	0	0	0
0	2	0	0	0	2	2	0	0	0
0	0	0	0	2	0	2	0	0	0
0	0	0	0	2	2	0	0	3	0
0	0	3	0	0	0	0	0	3	3
0	0	0	0	0	0	3	3	0	3
0	0	0	0	0	0	0	3	3	0

Adjacency matrix with nonzero values encoding groups.



Netgraph with colors encoding groups.

Figure 3.5: Four different ways to represent communication network data: Raw data, sociograms, adjacency matrices, and netgraphs.

netgraph. Each format has its advantages and disadvantages, but for large sets of data netgraphs are clearly the most efficient and useful representation.

Netgraphs have other benefits that make them useful as visualization tools. The basic underlying structure of the netgraph is still an adjacency matrix, allowing the use of algorithms previously developed for matrices. Netgraphs can also be encoded with more information than adjacency matrices. By using multiple colors in a netgraph, it is possible to show how different variables, such as group membership, age and salary considerations locate themselves within the netgraph. Additionally, if the rows and columns are sorted by group, for example, lines can be inserted into the netgraph to help distinguish where divisions take place. See Figure 3.6 for an example. This makes it easier for viewers to draw conclusions from netgraphs. The netgraphs that have been discussed so far have all been two dimensional. It is possible to encode yet another variable into the third dimension, creating a three dimensional structure. An example of a three dimensional netgraph is shown in Figure 3.7. Netgraphs have a tremendous amount of versatility and are excellent for use in the visualization of complex human networks. The key benefits include the underlying adjacency structure, the ability to uniquely specify the netgraph by permuting the rows and columns, the ability to represent different data features by using multiple colors in the netgraph and the ability to encode a third variable by using a three dimensional netgraph.

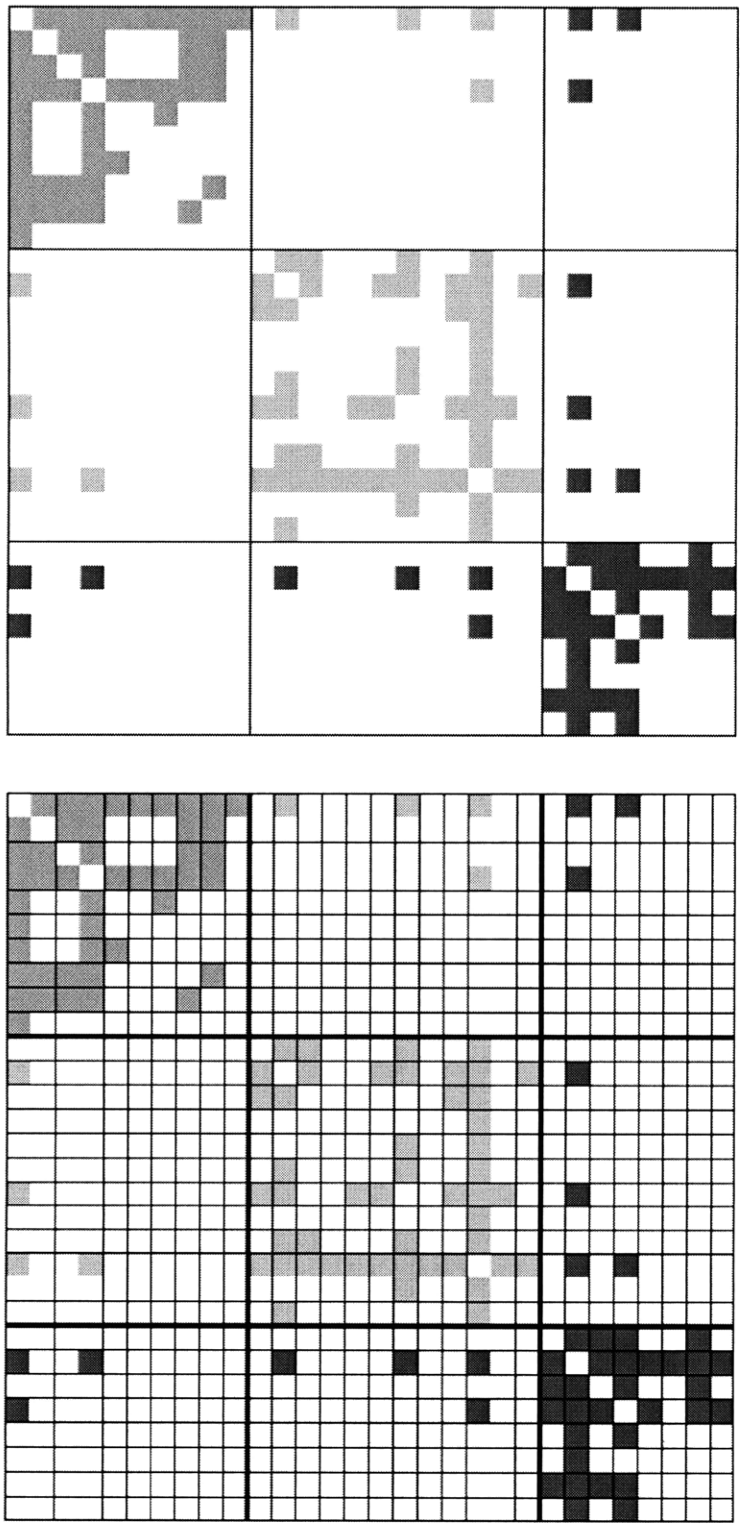


Figure 3.6: Netgraphs sorted by group with divisions indicated by lines. Two different levels of division are shown here.

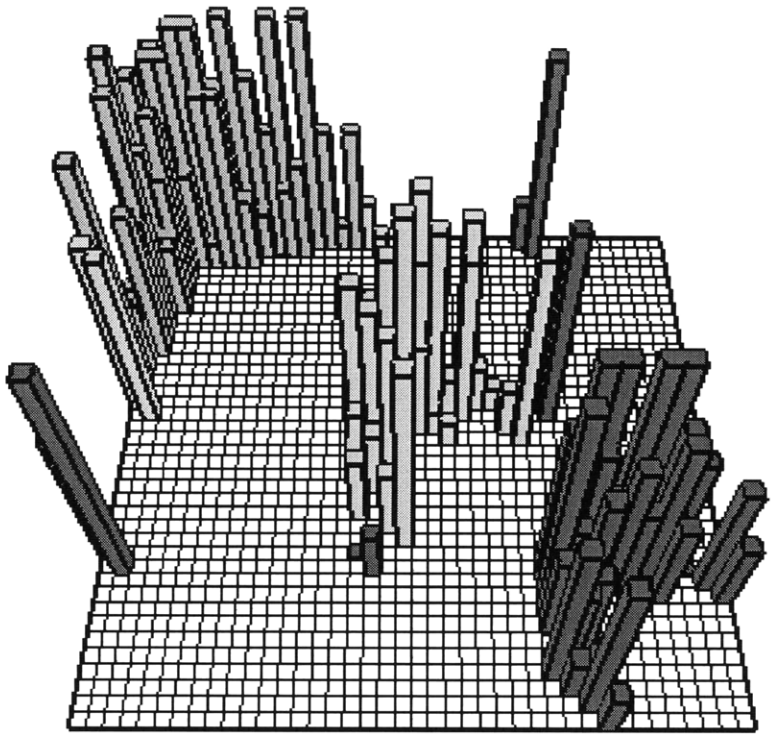


Figure 3.7: Three dimensional netgraph produced by AGNI.

Chapter 4: The AGNI Project

Background

AGNI is a software tool that allows users to manipulate the data associated with communications networks, providing easy access to many analytical algorithms as well as tools for visualizing both two and three dimensional netgraphs. The program can manipulate data about these networks, applying a number of algorithms and displaying netgraphs of the data. Netgraphs are specialized adjacency matrices that reveal various properties and patterns within the data. The program was written for X Windows on Unix and was ported to Windows NT during the past year.

The AGNI project has existed at MIT for several years. It was initially designed as a set of programs that ran on the mainframe at the Sloan School of Management in the mid 1980s. Over the years, attempts have been made to convert the system to run on Macs, PCs and Unix. There was even an attempt to integrate the AGNI system with a database as was done with the newest version of AGNI. For one reason or another, however, most of these past works have fallen by the wayside.

Reasons for Change

The processes used to collect and analyze data had to be performed manually for each study. The initial step was the collection of data from the company of interest. While several different methods of data collection are possible, the one used by AGNI has been a manual survey of companies. Employees at a firm are surveyed on randomly chosen days, about once a week for several weeks. They are asked to report who in their organization they had conversations with on that day, how the conversations occurred, and what the content of the conversation was, be it technical or non-technical. After this data was collected, it had to be averaged out over the time of study and normalized via statistical methods. The end result of this data reduction is an adjacency matrix that represents the

structure of the human interactions within the company. The data from this graph had to be placed in two specially formatted files. The characteristic file contained all the information associated with an individual: group, age, salary, etc. The communications file contained the list of links that should be present in the graph. These two files had to be created by hand with relatively few tools to help verify their formats. Overall, this process was tedious and slow.

The drawbacks of this system seem obvious in hindsight. The amount of manual intervention required to collate and enter the data from studies was tremendous. Any process which requires human labor is also prone to errors. In addition to the lack of automation, the AGNI program was written as a stand alone application. While this is fine if the data is going to be analyzed by one individual, it does not lend itself to team efforts in data analysis. Finally, the code for the program was rather large and complex, making it buggy and difficult to maintain. It was decided that the interface was due for a major overhaul, and that it should be redesigned from the ground up for both NT and Unix.

Existing Technical Structure

The original system was completely Unix based, having been initially designed to operate on Sun and Dec Unix machines. The architecture of the software system was relatively simple, consisting of two basic parts, a kernel and an interface. The kernel was written in C on Unix as a library that could be statically linked to by the interface. All of the functions that manipulated the netgraph data were located in the kernel.

Since the system was designed to run on Unix, the interface was written for X Windows. X Windows is a windowing specification that has become the standard for graphical applications on Unix. It provides a system independent set of functions that allow programs to create and manipulate windows on the user's desktop. The combination of a portable kernel written in C and an interface written for X Windows made it easy to

port AGNI between various Unix variants, but the next step in AGNI was a move to NT. NT uses it's own windowing standard, so the interface code could not easily be moved over to it. An attempt was made to port the interface to the Exceed Windows NT X Server, but the program proved to be too buggy. Fortunately, the kernel was straight C code and was easily ported to NT.

A major drawback of the statically linked kernel library was that it failed to provide fault tolerance between the kernel and the interface. The kernel code includes several complicated algorithms and was prone to segmentation faults, causing the whole program to crash. If more separation was provided, it would be possible to isolate system errors in one part of the program and prevent them from damaging the rest of the system. Due to the many bugs in the existing AGNI program, it was felt that a rewrite of the interface and its connection to the kernel would be beneficial to the project.

Chapter 5: Evaluation of AGNI

An initial survey of the state of the AGNI code and general process used to analyze the data revealed several weaknesses. The AGNI program had not taken advantage of any of the most recent advances in computer science. To rectify this, a plan of development was created and implemented successfully. AGNI is now up to modern coding standards, and the groundwork has been established for future improvements.

The work done on the AGNI project is interesting because it represents an exercise in software engineering. The redesign and redeployment of this antiquated computer program converted AGNI from a single monolithic, functionally structured program to a distributed, object-oriented program that makes use of both a database for its data manipulation needs and a webserver for its data collection needs. During this process, changes to the system were made in an incremental fashion so that the system could still be used at any stage of the process. This is interesting because it has parallels in the business world where it is sometimes necessary to revamp and modernize a system without completely changing the way it interacts with the outside world. This very thing was accomplished with AGNI through the careful application of modern programming disciplines.

Design Methodology

The interface was being completely rewritten, so a new set of design methodologies was developed. Time constraints demanded that developers take advantage of techniques that allowed for rapid development and maximum code reuse. This was accomplished by employing good object-oriented design and using code libraries available on the internet. In choosing these libraries, the fact that AGNI was targeted for both Unix and NT was a prime consideration.

Analysis of AGNI

The AGNI data analysis procedure has four main components. They are data collection, management, manipulation, and visualization. Each of these stages was considered in the development of the new architecture. The goal was to automate the analysis process, making the execution of these four stages possible with a minimal amount of training and manual intervention.

Data Collection

The conventional method of data collection for AGNI was the manual distribution of surveys. Someone had to make sure that surveys went to the correct people at the correct times. It was also necessary to carefully collect and tabulate the surveys to ensure the accuracy and integrity of the data. Data collection by hand is reasonable when the number of subjects under study is relatively small, but when one is studying companies with research and development groups consisting of hundreds of employees, manual collection is a daunting task.

The world wide web offered a chance to automate much of the data collection process. Since everyone fills out a generic form, it seemed possible to provide access to that form via a web page. This would relieve the researcher of the burden of having to track down and deliver surveys to large numbers of people. There is still the task of notifying when people should fill out the surveys, but that too can be automated with an email notification system for companies that have email. Data collection over the web would be a boon to communication network researchers.

Data Management

In the previous version of AGNI, data had to be manually converted into the characteristic and communications files used by the program. This involved a two step process of first entering the data into a spreadsheet program and then converting the

spreadsheet data into the two data files. The format of these files was relatively specific, and it took awhile to get the files to the point they could be successfully processed by the program.

A database allows one to store large amounts of data in tables and retrieve it at will. The data being represented in the characteristic and communications files is tabular in nature. Using a database as the focus of the data management system would be a large improvement for several reasons. First of all, it is much easier to organize and access data stored in a database. Second, it is easier to maintain a single database than it is to keep track of scores of characteristic and communications files. Last, using a database for record storage allows collected data to be entered directly into the database. There is no need for any intermediate formats.

Data Manipulation

The data manipulation stage of AGNI was performed by the algorithms stored in the statically linked kernel library. Since the kernel was written in portable C code, no work was needed for the NT version of the kernel. Data manipulation in the new version of AGNI is largely unchanged except for the fact that the kernel no longer has to be statically linked to the interface. A special interface kernel class was created to abstract the kernel library calls from the interface code. This allowed the interface to first be developed statically linked to the kernel, but later set up to connect to the kernel via a network connection. This was done through good object-oriented design which hid the actual kernel function calls from the interface code.

Data Visualization

Data visualization is the actual process of taking the information collected from the survey and using the AGNI program to create netgraphs from it. The previous version of AGNI accomplished this phase with the use of a graphical interface which ran on X

Windows and was limited primarily to Unix environments. A port to NT of the interface was attempted to see if it could be made to run on an NT X Server, but the interface proved to contain too many bugs to be useful.

Additionally, since a native NT interface for AGNI was going to be created, it did not make sense to try to support two separate code bases. A little bit of hunting turned up an interface library called Qt which allows for the simultaneous development of Windows and X Windows interfaces from the same source code. Rewriting the interface using this library allowed AGNI to maintain support of Unix while creating a native NT version of the interface.

Chapter 6: Design of the New AGNI System

A diagram of the complete AGNI system for the collection and analysis of communications network data is shown in Figure 6.1. The system design is broken up into its key components. This allows for abstraction between the different parts of the system as well as the independent development of each. Another benefit of the distributed approach is that the core of the functionality of the AGNI system resides on a single server and can be updated and maintained by a single individual. This minimizes the number of computer knowledgeable personnel needed to work with the system.

The central strategy of the new AGNI design is the use of a database as the foundation for the AGNI system. A database opens up tremendous possibilities in the use of the system. All of the data collected in surveys can be centrally stored and maintained, allowing one to access it from anywhere on the network. By storing the data generically in database tables one can dynamically access and sort the data for study. The characteristic and communications files are no longer needed because the data contained in them can be extracted from the database.

The data collection process is highly automated in the new AGNI design by incorporating a webserver into the system. Data can be collected via a web browser and entered directly into the database using a CGI program, which is a program designed to be activated by a webserver, as the interface between the webserver and the database. The webserver/database link is not restricted to a CGI program, however. It should be possible to use a Java servlet or a natively programmed server extension to accomplish the same task.

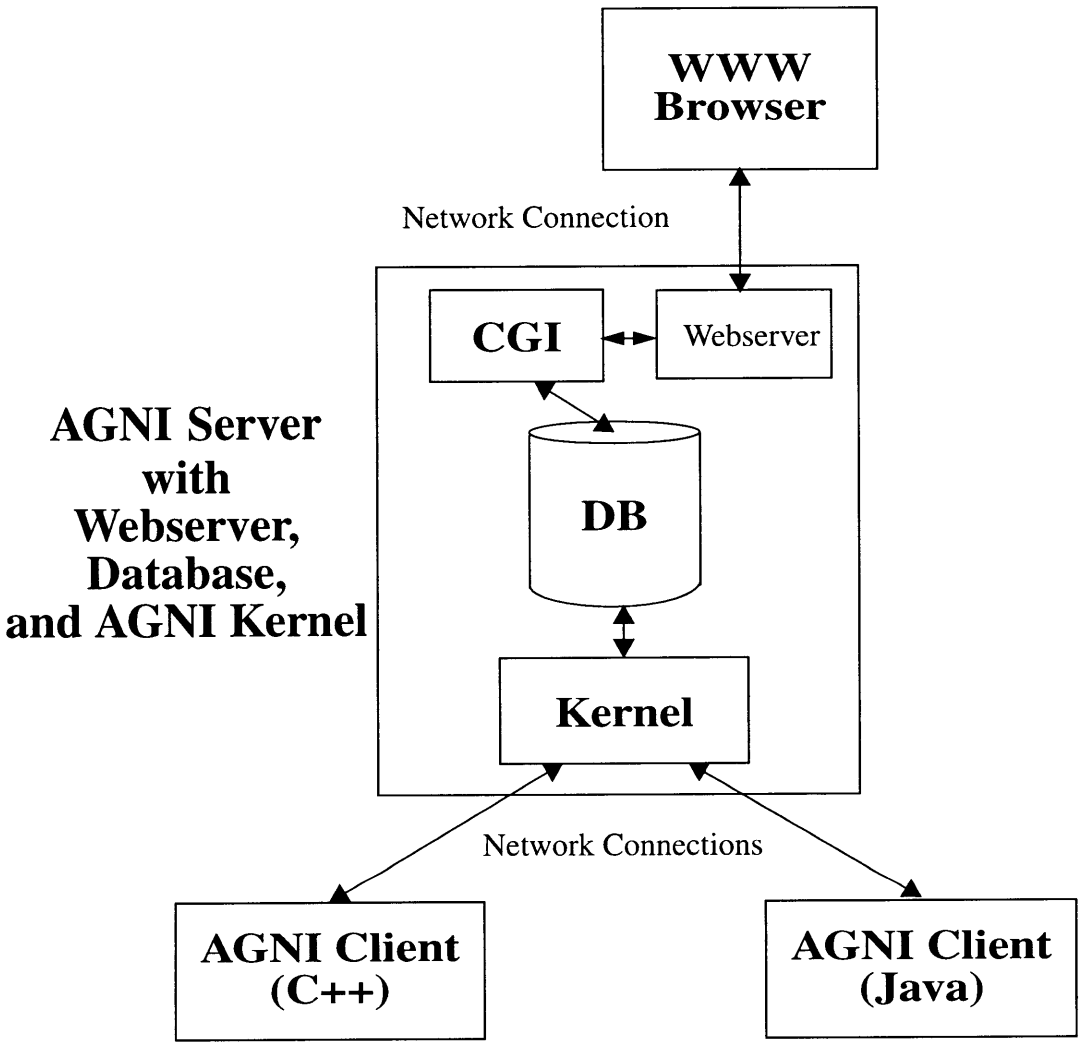


Figure 6.1: General architecture of the new AGNI system.

Data management is supported in the new design by the kernel which now resides on the server. The kernel contains all of the methods used by the system to manipulate data. The goal is to have the kernel working directly with the database rather than characteristic and communications files. This will require a reworking of some of the kernel code, so for the moment a utility can be provided to generate characteristic and communications files from data stored in the database. An added benefit of storing the data in a database is that programs other than AGNI can be used to access and manipulate the data. This opens up the data analysis stage to a score of programs that could not access the previous characteristic and communication files due to their proprietary formats.

In the new design, data manipulation and visualization are accomplished through the use of an AGNI client which connects to the kernel over the network. This architecture allows the same kernel to be used for several different types of clients. The current client is written in C++, but it shouldn't be too difficult to develop a Java version of the client as well. The java version could run as a stand alone application or as an applet served via the webserver. It is also conceivable that an html interface could be developed as well, though this might be more effort than it is worth.

In addition to allowing several different types of client to access the kernel, it should also be possible to build support for concurrent access to the data in the kernel. Currently the system only supports one client at a time since the consequences of multiple clients have not yet been considered. The kernel seems to have support for multiple sessions as it does refer to all queries to the same set of data using a kernel context set during initialization. This particular feature will have to be investigated more carefully to determine its feasibility.

The benefits of this new design lie in its distributed architecture. This design is more fault tolerant than the old design. If the kernel should happen to undergo a fatal error, the interfaces are protected from crashing since they are connected over the network and

are no longer running in the same process. It is also easier to extend the functionality of the system since the different subsystems are isolated from each other. As long as the interfaces between the elements are maintained, one should be able to add functionality to each of the systems incrementally, allowing for development and use to occur side by side. Lastly, the addition of the webserver and the database to the system greatly increases the usability of the system since many of the processes that were once tedious are now automated.

Chapter 7: AGNI Client Interface

The AGNI client interface received the majority of the development efforts for this past year. It initially existed as a stand-alone X Windows application with calls to the statically linked kernel functions interspersed among the various X Windows function calls. The program sponsors wanted AGNI to run on Windows NT which has its own windowing API distinctly different from X Windows. An NT version of the software had to be written from scratch. This was an opportunity to evaluate and restructure the AGNI program to take advantage of modern programming techniques.

In general, supporting multiple platforms with the same code base is not trivial. Different operating systems have different implementations that must be considered when writing code targeted for multiple platforms. The initial plan was to develop an interface layer that would provide a set of classes that could be used independent of the windowing system to produce the same application on multiple platforms. It was felt that the majority of the development on the project would be in building this window interface abstraction layer.

The Qt Toolkit

The problem of supporting NT and X Windows at the same time did not seem like an AGNI specific problem, so a survey of the internet was performed to see if someone else had already produced a version of the interface abstraction layer. The Qt library, a product of Troll Tech of Oslo, Norway, was discovered (Troll Tech, 1998). Qt was found to be a library that provided the abstraction layer we needed to support NT and X Windows simultaneously. In addition to multiple platform support, Qt was blessed with several other features that made it irresistible to the AGNI project.

The particular features of Qt that were attractive to the AGNI project were the ability to compile on Unix and Windows, the OpenGL extension which provides a GL enabled widget for three dimensional graphics, the high-level support of printing routines which allow the printer to be drawn on as if it was a widget and the support for component programming via a Signal/Slot interface. Signals and slots allow objects to be created as reusable components because they can operate without any knowledge of what they are connected to. This mechanism gives the programmer the power of callbacks with the added bonus of type safety. It eliminates the messy callback core dumps that plague typical applications.

QtArchitect

The client interface uses twelve different dialogs, so QtArchitect, a dialog editor for Qt, was used to streamline their creation. The dialog editor allowed the dialogs to be created and updated via a graphical interface. QtArchitect produces source code for the dialog created by the user. It works by producing two class definitions for each dialog, a data class inherited from one of the Qt classes and a user class which inherits the data class. QtArchitect places geometry and configuration information into the data class and any changes made to it propagate through the inheritance hierarchy to the user class. The programmer can then fill in the member functions of the user class to round out the implementation.

Object Hierarchy

The object hierarchy for the entire AGNI interface is shown in Figure 7.1 and Figure 7.2. The Qt classes are shown shaded, and the user classes derived from them are in plain boxes. The Qt library was designed to support extensibility, so as much attempt as possible was made to use the Qt widgets in the design of AGNI. The aspects of the hierarchy of particular note are the classes derived from the QCollection class, the kernel

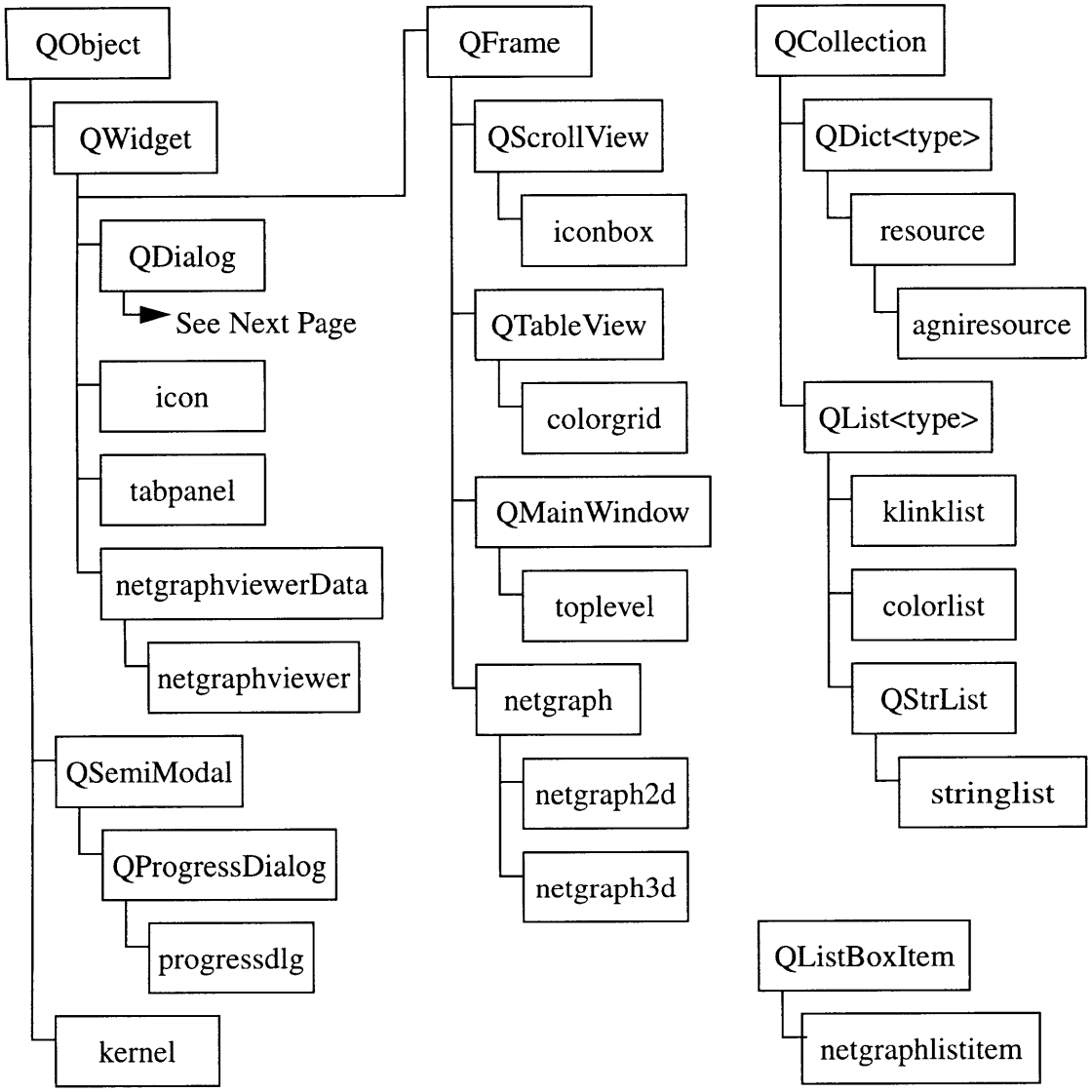


Figure 7.1: Module dependency diagram for AGNI. (Part 1)

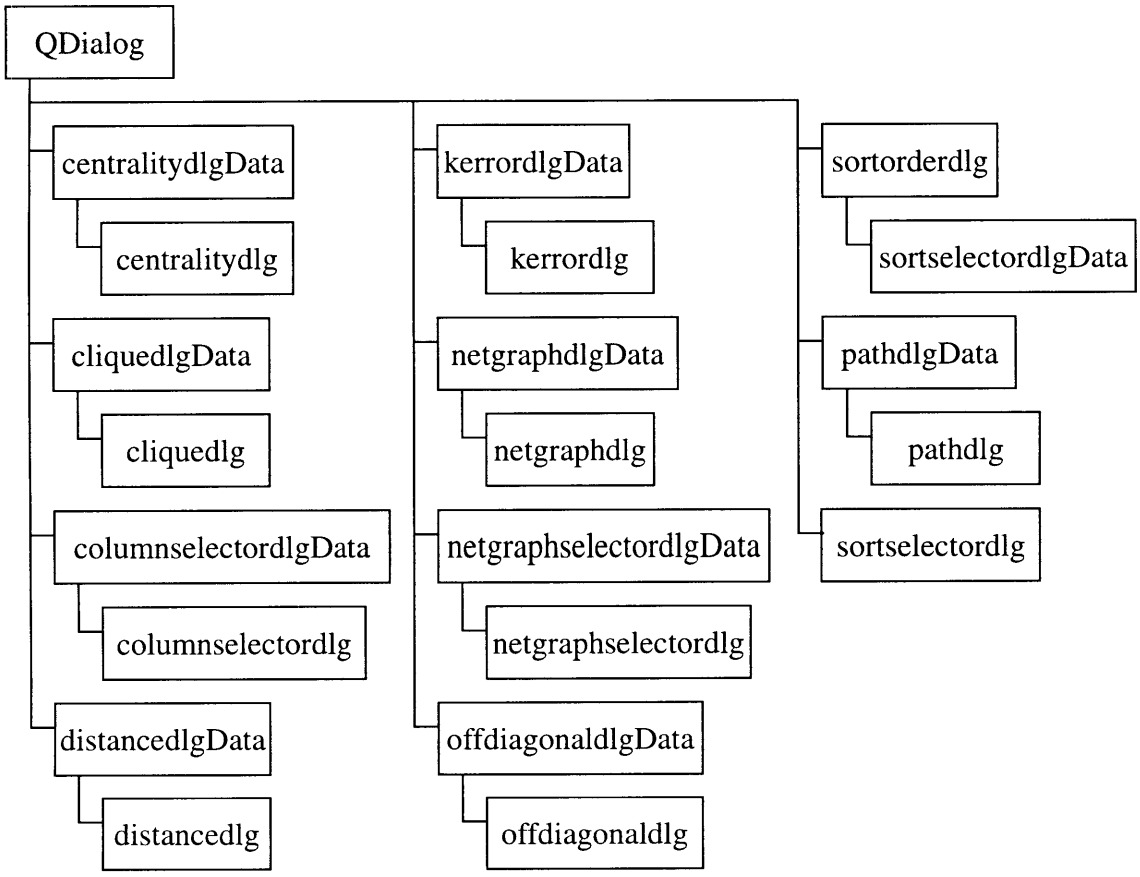


Figure 7.2: Module dependency diagram for AGNI. (Part 2)

class derived from QObject, and the netgraph classes derived from QFrame, as these are the classes that aided the most in the smooth transition to the new version of AGNI.

Resource and Agniresource

The first classes to notice are the resource class and the agniresource class derived from the template class QDict<type>. The QDict class defines a hash lookup structure that allows the referencing of generic variables of any type via string keys. The original AGNI version used a configuration file to handle the foreign language requirements of the program. An alternate language was specified by providing translations for the text in AGNI in an X Resource file. Since no such device existed for Qt, the resource class was created to take its place. The resource class takes a filename and parses it for key/value pairs, storing them in its QDict structure. Agniresource is an extension of the resource class that inserts a set of default values into the QDict structure before parsing the file. This allows the program to work with default values in case the configuration file, agni.ini, is not present. The use of this file allowed the users of AGNI to specify a foreign language in a fashion that was already familiar to them. This made the transition to the new AGNI version smoother.

Stringlist and KLinklist

The largest obstacle to a smooth transition between the versions of AGNI was the interface to the already existing kernel routines. The klinklist and stringlist collection classes derived from QList<type> are used to ease communication with the kernel library code. Several functions in the kernel accept or return arrays of strings. The stringlist class makes it easy to pass these arrays of strings around the program without having to worry about pointers. It provides methods for converting to and from the kernel representation as an array of pointers to char, and the new interface's representation as a QList of QStrings. The klinklist class serves a similar purpose, as some of the kernel functions use a

particular linked list structure in their parameters. Klinklist encapsulates this structure into a class for ease of use, providing methods to convert to and from the kernel format of the linked list as well.

Kernel

The kernel class was the single most important class to the smooth transition between versions. It was created to encapsulate all communications with the kernel functions into a single class. This provides an abstraction layer between the interface and the implementation of the kernel. This was done because it was expected that the kernel calls would eventually be made over a network connection rather than having them statically linked to the interface

This abstraction layer was created by writing the kernel class as a wrapper around the kernel library calls. The interface created an instance of the kernel object and used it to make calls to the kernel. The class functions by taking the calls and translating them into the necessary kernel library calls. The kernel class also made it easier to deal with the fact that the interface was written in C++ while the kernel library was in C. When the class is constructed, it obtains a kernel context from the kernel library which it stores and uses in subsequent kernel calls. This latter implementation is common to C style coding, but the kernel class provides an abstraction to it so that the kernel appears to operate as an object from the standpoint of the interface.

The extension of the AGNI interface to connect to a server over the network is made simpler by the kernel class abstraction. None of the interface needs to be changed. Only the member functions of the kernel class need to be changed to support the transfer of information over the network. The kernel class successfully isolates the majority of the interface from the details of network operations. By insulating the interface from the

kernel, the migration from a stand-alone application to a networked application is much easier.

Netgraph, Netgraph2d and Netgraph3d

The netgraph, netgraph2d and netgraph3d classes are used by the interface to produce netgraphs. The underlying data needed to visualize netgraphs is virtually the same for both two and three dimensional netgraphs, but the methods of visualization are quite different. The common aspects between the two formats are encapsulated into the netgraph class. The netgraph2d class inherits the netgraph class and makes use of the standard two dimensional drawing routines to produce netgraphs. OpenGL graphics were used for the three dimensional netgraph, however. This presented a stumbling block since both QFrame and QGLWidget were descendents of QObject. This multiple inheritance of two QObjects caused the Signal/Slot mechanism of Qt to break. To overcome this, a netgraph3dWindow class was created as a friend class to netgraph3d. Netgraph3d inherited QFrame as normal, and netgraph3dWindow inherited QGLWidget so it could use the OpenGL drawing functionality. The netgraph3d class contained a netgraph3dWindow object as a child so that it could use the Signal/Slot mechanism while still appearing to use OpenGL drawing. Despite the need to use a friend class, the netgraph implementation turned out to be rather elegant.

Chapter 8: Design of a Networked Kernel

The final goal of the AGNI project is the fully functional distributed architecture shown in Figure 6.1. The requirements for the functionality of the kernel portion of the design are varied enough to warrant close attention. Several technologies are available for supporting network communication between programs. Remote procedure call is one of these technologies that seems promising. It is hoped, however, that several different client types will one day be available for AGNI. If this is to be the case, then AGNI's designers must avoid the use of anything that may limit future opportunities. Remote procedure call libraries are typically aimed at functional calls between the same language, so it would probably be difficult to support the networked kernel using RPC. The architecture being suggested is simply a text based network protocol since any language will be able to parse text messages.

The network protocol for AGNI must be chosen carefully since it is likely to evolve through many incarnations over time. It needs to be easily extensible so that new features can be added to the AGNI system. Additionally, the data that is currently sent over the kernel varies greatly in type and length. This set of parameters eliminates the possibility of using fixed length packets. A viable protocol would allow versioning to support the extending the protocol and named parameters to support both extensibility and variable length. This can be accomplished by defining a protocol format similar to HTTPD and HTML, for example.

A sample network packet might look something like this:

```
AGNI/1.0
sessionid=12;
version=1.1;
opcode=netgraph2d;
data=<netgraph><row><cell color=#ff0000><cell color=#00ff00></row>
<row><cell color=#ff0000><cell color=#00ff00></row>
</netgraph>
```

This is of course just one possible format. All of the client/server interactions should be considered. The benefits of using something similar to HTML is that there are libraries available for parsing text blocks of this format. It should be possible to avoid having to hand code a parser.

Chapter 9: AGNI Server Components

The server components of the AGNI system consist mainly of the database and the webserver. Since the target system for the server is Windows NT, the Microsoft IIS webserver and SQL Server database would make a nice integrated solution. The webserver portion of the server for data collection is relatively trivial. One just needs to provide a form submission page with the correct questions on it. The form activates a CGI program on the server that inserts the surveyed information into the database. It is likely that some control will need to be placed in the CGI to prevent multiple submissions of the same survey, etc.

The database tables used to represent the system data will need to be designed to support an arbitrary number of simultaneous surveys. This can be done by having a registry table which serves as an index of the surveys currently held in the database. Each survey will have a separate set of questions, so solutions for dealing with that need to be developed. This can most likely be done by adding columns to the index table that indicate the column types present in the characteristic and communications tables for a survey. A sample database schema might look something like this:

reference	char_table	comm_table	columns
1	survey1_char	survey1_comm	userid;targetid;comm_type;timestamp;
2	survey2_char	survey1_comm	userid;targeidt;comm_type;dept;timestamp

Table 9.1: Index Table

userid	name	group	age	salary
1	Jim	Manufacturing	30	30000
2	Sally	Manufacturing	28	32000
3	Greg	Sales	35	38000
4	Susan	Marketing	31	55000

Table 9.2: survey1_char table

userid	targetid	comm_type	timestamp
1	2	telephone	1:00
1	3	email	2:00
2	3	face	2:15
3	2	face	2:15
3	4	phone	1:30
3	1	email	1:00
4	2	telephone	1:05
4	2	face	1:20

Table 9.3: survey1_comm table

The comm tables will grow to be large since they contain an entry for each communication reported on a survey. The kernel will need to be modified to provide sorting and collation of the raw table data into the person to person network data format used by the kernel.

Chapter 10: Conclusions and Future Work

This paper has described the work done on the AGNI project over the past year. The work was successful in taking the ungainly mass of X Windows code that ran on Unix and reengineering it as an object-oriented architecture that ran on both Windows NT and Unix. Design work was also done on the AGNI system in general. A plan was formulated for a distributed architecture that would allow for a much higher degree of automation in the use of the AGNI data analysis package.

Future work on the AGNI project will involve finalizing and bringing to completion the server components of the newly designed AGNI system. The networked kernel needs to be integrated with the database. This will involve modifying some of the kernel code so that it can access the data stored in the database. The data collection features of the webserver have to be set up manually at present time. It would be nice if some tools were developed that allowed for the easy definition of new surveys so that the operation of the system could occur with minimal intervention the part of a programmer. Last but not least, other types of AGNI clients for use with the kernel should be considered.

References

Allen, Thomas J. (1967). Communications in the Research and Development Laboratory. Technology Review. Massachusetts Institute of Technology, Cambridge.

Allen, Thomas J. (1986) Managing the Flow of Technology. Cambridge, Massachusetts: The MIT Press.

Everett, Rogers M. & Kincaid, D. Lawrence. (1981). Communication Networks Toward a New Paradigm for Research. New York: The Free Press, A Division of Macmillan Publishing Co., Inc.

Forsythe, Elaine, & Katz, Leo. (1960). A Matrix Approach to the Analysis of Sociometric Data: Preliminary Report. In J. L. Moreno, The Sociometry Reader. The Free Press of Glencoe, Illinois.

George, Varghese P., & Allen, Thomas J. (1989). Netgraphs: A Graphic Representation of Adjacency Matrices as a Tool for Network Analysis. MIT Sloan School of Management Working Paper, WP1-89, Massachusetts Institute of Technology, Cambridge.

George, Varghese P. (1990). Optimizing Organizational Processes and Tailoring Networks. Unpublished master's thesis, Massachusetts Institute of Technology, Cambridge.

George, Varghese P., & Allen, Thomas J. (1993). Relational Data in Organizational Settings: An Introductory Note for Using AGNI and Netgraphs to Analyze Nodes, Relationships, Partitions and Boundaries. MIT Sloan School of Management Working Paper, WP85-93, Massachusetts Institute of Technology, Cambridge.

Krebs, Valdis. (1996). Visualizing Human Networks. *Release 1.0: Esther Dyson's Monthly Report.* New York, New York.

Lee, Jack K. (1991). The Design of a Data Pre-Processor in d-Base III Plus for Analysis of Organizational Communication Networks. Unpublished bachelor's thesis, Massachusetts Institute of Technology, Cambridge

Levi, Lorenzo. (1990). *An X Window Graphic Tool For Analysis Of Organizational Communication Networks.* Unpublished bachelor's thesis, Massachusetts Institute of Technology, Cambridge.

QtArchitect (1998). http://www.primenet.com/~jtharris/qtarch.

Rice, Ronald E. & Richards, William D. Jr. (1985). An Overview of Network Analysis Methods and Programs. In B Dervin, & M. J. Voight, Progress in Communication Sciences, Volume VI. Norwood, New Jersey: Ablex Publishing Corporation.

Stouffer, Samuel A., Suchman, Edward A., DeVinney, Leland C., Star, Shirley A., Williams, Robin M., Jr. (1949). The American Soldier: Adjustment During Army Life, Volume I. Manhattan, Kansas: MA/AH Publishing.

Toll Tech (1998). http://www.troll.no.

Thirumalaisamy, Pillan K. (1990). Creating Netgraphs Using the X Window System To Study Organizational Communication. Unpublished bachelor's thesis, Massachusetts Institute of Technology, Cambridge.

Yang, Theodore M. (1997). The Development of Business Applications: The World Wide Web Paradigm. Unpublished master's thesis, Massachusetts Institute of Technology, Cambridge.