Network Specification and Report System for ISEAGE – Phase 2

Final Report
Senior Design Team: May06_10

Client
Information Assurance Center

Faculty Advisor
Dr. Douglas W. Jacobson

Team Members
Eric Anders   CprE
Piyush Patel  CprE
Jonathan Cook CprE
Trent Robertson CprE

DISCLAIMER: This document was developed as a part of the requirements of an electrical and computer engineering course at Iowa State University, Ames, Iowa. This document does not constitute a professional engineering design or a professional land-surveying document. Although the information is intended to be accurate, the associated students, faculty, and Iowa State University make no claims, promises, or guarantees about the accuracy, completeness, quality, or adequacy of the information. The user of this document shall ensure that any such use does not violate any laws with regard to professional licensing and certification requirements. This use includes any work resulting from this student-prepared document that is required to be under the responsible charge of a licensed engineer or surveyor. This document is copyrighted by the students who produced this document and the associated faculty advisors. No part may be reproduced without the written permission of the senior design course coordinator.

May 3, 2006
4.5.2 Anticipated potential risks encountered and success in management......................... 34
4.5.3 Resultant changes in risk management made because of encountered unanticipated risks 34

4.6 PROJECT TEAM INFORMATION ........................................................................................... 34
  4.6.1 Client Information............................................................................................................. 35
  4.6.2 Faculty Advisor ............................................................................................................... 35
  4.6.3 Team Members ............................................................................................................... 35

4.7 CLOSING SUMMARY ............................................................................................................. 36
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standard for Bracket Usage</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Standard for Function Declarations</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Standard for Line Length</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Standard for Loop Usage</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>Configuration GUI</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>Configuration Tool State Diagram</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Configuration Tool Use-Case</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>Visualization GUI</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>Visualization Tool State Diagram</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>Visualization Tool Use-Case</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>ASCII File</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Monkey Testing Example 1</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>Monkey Testing Example 2</td>
<td>22</td>
</tr>
<tr>
<td>15</td>
<td>Inter-Group Testing Procedure</td>
<td>23</td>
</tr>
<tr>
<td>16</td>
<td>Personnel Effort</td>
<td>26</td>
</tr>
<tr>
<td>17</td>
<td>Financial Costs</td>
<td>28</td>
</tr>
<tr>
<td>18</td>
<td>Project Schedules</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>Deliverables Schedules</td>
<td>28</td>
</tr>
</tbody>
</table>
List of Definitions

**CVS (Concurrent Versions Systems)** – an open-source network-transparent program that allows developers to keep track of different development versions of source code.

**GIS (Geographical Information Systems)** – A tool used to gather, transform, manipulate, analyze, and produce information related to the surface of the Earth.

**GUI (Graphical user interface)** – Acronym for graphical user interface, an interface for issuing commands to a computer utilizing a pointing device, such as a mouse, that manipulates and activates graphical images on a monitor, pronounced gooey.

**Information Assurance** – The art of securing information on computer systems.

**IPv4** – The Internet Protocol version Four. IPv4 addresses are 32 bits wide. Its headers are 20 bytes at minimum, and grow in chunks of 4 bytes.


**ISEAGE** – Acronym for Internet Scale Event and Attack Generation Environment, the environment the end product will run on.

**Java Swing a.k.a. Swing** – A set of Java class libraries that support building GUIs and graphics functionality for client applications that will run on multiple platforms.

**MSDNAA (Microsoft Developers Network Academic Alliance)** – An alliance between Microsoft Corporation and Iowa State University allowing students access to Microsoft’s complete line of software. Upon graduation, the student must remove the software from their computer.

**Open source** – Software that is free for use and modification.

**SWT (Standard Widget Toolkit)** - The Eclipse Foundation’s framework for developing graphical user interfaces in Java.

**UML (Unified Modeling Language)** – a general-purpose notational language for specifying and visualizing complex software, especially large, object-oriented projects.

**XML (Extensible Markup Language)** – allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organization
1 Project Approach and Results

This section aims to give the reader a brief look at the entire project without getting extremely technical. Among the various sections are the executive summary aimed at decision-makers who need simply an overall understanding of the project, a problem statement section talking about what the problem this project was trying to solve, and the intended users and uses.

1.1 Executive Summary

This section serves as a summary of the entire report and is broken down into several sections including the motivation behind the project, project activities, results and recommendations for the future.

1.1.1 Need for Project

The overall motivation behind the ISEAGE project, of which the end product is a subset, is to build a system that people can use to design and test out their own security ideas to evaluate their effectiveness on a network. Normally, a network simulation is done entirely through software and doesn’t perform an accurate job of representing the actual Internet environment. In ISEAGE, the actual hardware is used to route traffic to simulate a very realistic image of the Internet. Such a system is especially needed today because of the threat of cyber crimes faced by the U.S. after 9/11.

This project is specifically aimed at letting a user design a virtual network graphically, export the configuration to the ISEAGE system, and run out various scenarios against the virtual network. Also involved was a report system to let the user know the results of the simulation in pseudo-real time.

1.1.2 Actual Project Activities

The project activities are centered on the problem of needing a virtual network configuration tool and a reporting system for ISEAGE. Since this is most easily done graphically, the team decided to build a two separate GUI’s and two separate applications. The network configuration tool has the ability to run at the ISEAGE or on a stand alone computer. The reporting system tool needs to run at the ISEAGE to utilize the packet dump the ISEAGE has to offer. Additional to the two applications, a manual will be provided to aide users in properly utilizing the two applications.

1.1.3 Final Results

The hardest part of the project would be building a diagramming object, within the code, to aide in the graph theory needed to implement
the network configuration tool. Along with the diagramming tool, this project will need some sort of an algorithm to assign IP addresses dynamically within a graph. As stated before, a manual will be provided to aid the users in proper use of these applications. This will be provided in electronic form to be printed out if needed.

1.1.4 Recommendations for Future Work

Since the project was an attempt at creating a working network configuration utility, it doesn't have all the desired, but not required, features. For example, one desirable feature was that the configuration tool be cross-platform. The team did look into this option but decided not to use it since C# handles the development process by being extremely well typed. Real-time reporting is another desirable feature for the software. Currently, the project reports back in pseudo-real time. If users can see the results right as they're evaluated on ISEAGE, it would enhance the simulation process.

1.2 Problem Statement

This section talks generally about what the problem was and what solution the team adopted in order to do the project.

1.2.1 General Problem Statement

The goal of this project will be to create a tool or set of tools that will allow a user to define a network specification that will be used to configure ISEAGE. The tool(s) will also be able to show real-time visualization of the ISEAGE network. The visualization will need to display both a latitude/longitude representation (Iowa Only) and a network topology representation. For example, the user should be able to model the entire state of Iowa network using ISEAGE.

1.2.2 General Solution Approach

The team will design an intuitive GUI that provides graphical symbols representing routers, computers, fiber wires, etc for the user to create a network topology. The user will be able to monitor traffic through an XML document, which will store the network configuration data. To enable users to map physical networks, the team will integrate an Iowa map instead of some GIS-based software.

1.3 Operating Environment

The end product is intended to function under Microsoft Windows Operating System, including Windows 95, 98, ME, 2000, and XP. The application will run from a control PC attached to ISEAGE.

1.4 Intended Uses and Intended Users
This section discusses the intended uses and the intended users of the end product.

1.4.1 Intended Uses

Defining the configuration of ISEAGE network/sub-networks. Creating commonly used tools with various teams working with our client. Making the project expandable for use with different operating systems, and making it user friendly is necessary. Expansion of the project topology from the ISEAGE network to the whole state of Iowa, and beyond is possible.

1.4.2 Intended Users

Individuals who are technically fluent with the ISEAGE research project. Users will have direct contact with our client, along with excellent knowledge of the coding scheme within our project. The visualization team within ISEAGE will be connected with the project directly to aid with necessary tools to complete the overall design of the project.

1.5 Assumptions and Limitations

This section includes an updated list of assumptions and limitations that the team worked with in the execution of the project.

1.5.1 Updated Assumptions List

Following are the updated assumptions made by the team:

- The application will run on Windows systems only.
- The code base will be C#.
- The configuration tool has the ability to run independently of ISEAGE. The output configuration file can be implemented immediately and directly, or it can be stored for later use.

1.5.2 Updated Limitations List

Listed below are client-specified limitations:

- The software shall both configure and report real-time information about ISEAGE.
- The real-time display shall have two modes: physical layout and network topology.
- The software shall use a GUI.
- The software shall run on a computer with at least 256 MHz processor and 128MB Memory that is running Windows Operating System 2000 or XP.
1.6  **Expected End Product and Deliverables**

The end product will be an application or applications that will configure ISEAGE and display a visual, real-time representation of the network. The configuration portion of the application will utilize a graphical, drag-and-drop interface that will allow a user to quickly and easily setup and define a network topology. The visualization portion of the application will have two display modes, network topology and physical. The network topology view will present a clear ordered layout of the network, and each node. The physical view will be a map overlay to show the actual physical locations of each node. Each mode will take input from the ISEAGE network to display traffic density information and the status of every node.

A user manual will be provided with the end product. The manual will document each process and option available to the user during the use of the application.
The approach that will be implemented by the team and the step-by-step process of the design of the software end-product are described in this section. Some of the major project aspects include the various requirements defined by the team for this project’s successful completion and the detailed explanation of the different project execution phases.

2.1 **Approach Used**

The proposed approach section describes some of the constraints the team will work with in order to ensure the successful completion of the project. Included are the functions that the software will or will not provide, the security measures taken to safeguard the project while it is being developed, and the safety impacts of the end product. In addition, criteria have been defined in order to evaluate the success of the project at the end of the two-semester period.

2.1.1 **Functional Requirements**

The following defines exactly what the proposed software should and should not do:

2.1.1.1 The software shall allow the configuration of virtual computer networks whether large or small

2.1.1.2 The network specification shall include routers and may include other network devices.

2.1.1.3 The software shall allow network specification through the use of a GUI which provides graphical symbols representing computers, routers, wires, etc., all of which will be available in a sub-window.

2.1.1.4 The software shall allow the user to specify a network topologically and it shall allow the user to specify a network by physical layout.

2.1.1.5 The software shall allow network specification in an ASCII format compatible with ISEAGE.
2.1.2 Design Constraints

This section outlines the design constraints of the software

2.1.2.1 The software must be Windows-compatible

2.1.2.2 The software may be cross-platform compatible

2.1.2.3 The software shall provide a GUI for network specification

2.1.2.4 The software shall produce an ASCII output.

2.1.2.5 The software must be able to specify IPv4 networks.

2.1.2.6 The software may be customizable to IPv6 networks.

2.1.3 Technology Considerations

This section discusses the various software technologies and the technical approaches that the team researched in the process of deciding which tools and methods to adopt prior to the implementation phase.

2.1.3.1 Considered Integrated Developers Environment (IDE)

Several Integrated Developers Environments were considered for this project. Discussed below are Eclipse and Microsoft Visual Studio .NET.

2.1.3.1.1 Eclipse

The Eclipse platform is an open-source integrated developer environment. It is primarily used with Java therefore it is not operating system dependent. However, C++ can be used as long as the C++ compiler is downloaded and installed on Eclipse thus the only drawback being that with using C++, the application becomes operating system dependent. The attributes that made Eclipse appealing were: the ease in which GUI’s can be made using the SWT library, its ability of producing applications that are operating system dependent, and the fact that it is an open source platform.

2.1.3.1.2 Microsoft Visual Studio .NET

The Microsoft Visual Studio .NET platform contains a whole host of tools that can be used to create applications. Contained in the platform are languages Visual C++, Visual Basic, Visual C#, and Visual J#. The Language of interest to us is Visual C#. Because Visual Studio .NET is a Microsoft product, the applications that are built are operating system dependent. Also, Visual Studio .NET isn’t an open source integrated developer environment. The team does have access to the software through the Iowa State/Microsoft Academic Alliance.
2.1.3.1.3 Selected Integrated Developer Environment (IDE)

The team decided to use Microsoft’s Visual Studio .NET to be the IDE on which the Configuration and Visualization Tool will be developed on.

2.1.3.1.4 Reason for Selection

There are two main reasons why Visual Studio .NET was chosen. First, the deployment of the application is much easier through the wizard provided by Visual Studio .NET. We unsuccessfully created a .jar file in Java using the SWT library. We successfully created a setup program, for our application, using Visual Studio’s deployment wizard. Second, even though Eclipse is open source, we can also get Visual Studio .NET for free through the Iowa State/Microsoft Academic Alliance.

2.1.3.2 Considered Visualization Tools

Two main theories were considered for the Network Visualization part of our project. They are Google Earth and an in-house coded visualization application.

2.1.3.2.1 Google Earth

Google Earth is a visualization tool from the perspective of a satellite. The user types in the name of the geographical location and watches as Google Earth zooms onto that location allowing the user a look at the area from the view of a satellite located miles above the Earth. Google Earth was made in an extensible manner. It allows users to overlay objects onto the satellite image through the use of KML, a variation of XML. Google Earth comes as sold in various packages. The free package is the base model barely giving the user the ability to utilize the KML part of the program. The next package up, which has a subscription based annual fee of $20 and is named Google Earth Plus, has all the functionality of the base package plus drawing tools, a GPS device data import function, and a 100 point maximum spreadsheet data import function. Google Earth Pro, which costs $400, has all the functionality of Google Earth Plus with the addition of being able to create unique polygons amongst other things.

2.1.3.2.2 Selected Visualization Tool

The team decided to develop our own “in-house” visualization application.

2.1.3.2.3 Reason for Selection

The visualization tool requirements state that a network could span the entire state of Iowa for now, and then the networks the visualization tool would have to show would span the United States and then the world. Therefore the program would have to be extensible. Google Earth would work very well with the respect to extensibility. The maps are already in
place and all we have to do us create the KML files for the network to be overlaid onto Google Earth. Google Earth also presented some problems. The visualization tool is going to have show packet information through the various routers. Also, Google Earth couldn’t handle the KML files that the team created for testing reasons.

The “in-house” coded visualization tool will allow the team more freedom. The team can utilize the most efficient layout of the GUI. Packet information will be displayed in a frame. The visualization tool will work closer to real-time than if utilized by Google Earth because Google Earth tends to eat up resources. Also, this allows more, or older, computers that the visualization tool can run on. The extensibility requirement will be easily solved by allowing the user to be able to choose which picture file be used as the underlay. For example, the user will not have a choice of using an underlay other than the state of Iowa for the initial release. In later versions, a user could pick between a picture file of the United States of America or Des Moines, Iowa.

2.1.3.3 Considered Programming Languages
This section discusses the various languages that the team considered in deciding on a primary language to use in the coding phase. Among the ones considered are C, C++, C#, and Java.

2.1.3.3.1 C

C is compiled code, instead of interpreted code, and if all else is equal, compiled code is at least as fast as interpreted code. This makes C preferable to an equivalent interpreted language. It is also a nearly universal language; at this level, almost anyone who would see the program's code will have used C at some point.

ISEAGE creates a simulation of a computer network which has discrete components connected to one another. This makes an object-oriented language preferable to a non-object oriented language. C, however, is non-object-oriented, and its advantages are shared by the similar, object-oriented languages C++ and C#, making C++ and C# preferable to C.

2.1.3.3.2 C++

The primary advantage of C++ is that the entire group has used it before. It is also object-oriented making it well-suited to create many simulation objects with customized behavior since these objects can be represented by instances of virtual objects. All else being equal, being compiled code, C++ will be at least as fast as interpreted languages. Properly constructed C++ code is portable as long as the libraries used are available on the platform to which it is being ported.
The main disadvantage of C++ is that it has few built-in functions compared to C# and Java. Instead, it provides substantial low-level functionality which implies that the team must either locate the appropriate libraries or write the equivalent code from the ground up. A strong example of this is the lack of graphical libraries. However, this can be overcome by the use of software that either provides these libraries or the means to accomplish with ease what the libraries can. Both of these options are discussed in the Considered Graphical Packages section.

2.1.3.3.3 C#

C# is more user-friendly than C++ or C because it has many built-in functions which do not require the manual inclusion of libraries. This makes the building of GUI’s quicker and more efficient. However, C# is not cross-platform. As stated before, this doesn’t matter because most computers associated with ISEAGE have the windows operating system on them. Initially, none of the team members had experience with C#.

2.1.3.3.4 Java

Java, like C#, has a lot of built-in functions that do not need the manual inclusion of libraries. This, again, helps with the building of GUI’s. Also, three of the four team members have experience developing applications in Java. As stated above, our knowledge and experience of Java should aid us in the learning curve of C#.

2.1.3.3.5 Selected Programming Language

The team selected to use C# as the programming language.

2.1.3.3.6 Reason for Selection

We chose C# because of its ability to run faster than Java. The team believed that C#’s similarities to more familiar languages would facilitate the learning process.

2.1.3.4 Considered Technical Approaches

This section discusses various approaches that the team has considered for the coding phase of the project.

2.1.3.4.1 Coding Standards

In order to improve the readability of the code, each member is expected to use the same style of format.

2.1.3.4.2 Brackets

Brackets shall always have a line which only they appear. Opening
brackets shall be at the same tab level as the preceding line, and all
following lines will have one more tab preceding the line (see Figure 1).

```
for (int i = 0; i != 1; i++)
{
    Console.WriteLine(i);
}
```

Figure 1. Standard for Bracket Usage

2.1.3.4.3 Creation of Variables

The creation of variables shall be done at the start of the function and only
at the start of the function. The declaration of type shall be followed by
exactly one type. Variables shall be declared in order of type, with exactly
zero spaces between a variable and the following comma and exactly one
space between a variable and the preceding comma (see Figure 2). Variables of the same type shall be declared on the same line unless this
exceeds 80 characters, in which case a carriage return and tab will replace
the last space on the line that would otherwise exceed 80 characters. The
variables of the same type shall be declared in alphabetical order.
Variables should be initialized at their creation, or as soon as possible
afterwards, but may be initialized to any suitable value; by default, this
value is 0 for any variable storing integer-like data.

```
int a, b, c, d, e, f, g, h, i, j, k, l;
a=b=c=d=e=f=g=h=i=j=k=l=0;
```

Figure 2. Standard for Variable Declarations

2.1.3.4.4 Functional Declaration

The return type of the function shall be followed by exactly one space,
followed by the function name, beginning with a lowercase letter and with
capital letters occurring at the start of any new word (instead of
underscores, and without spaces), followed immediately by an opening
parenthesis (see Figure 3). The type of and the variables sent into the
function shall then follow, in that order, with exactly one space between the
return value and the variable. The order of these variables shall be
alphabetical by type, then alphabetical by variable name. The variables shall
also begin with a lowercase letter, with no spaces, and capital letters
beginning subsequent words. After the last variable, a closing parenthesis
and semicolon shall appear without any spaces.

```
int garbageFunction(char firstVariable, char secondVariable,
                    int fourthVariable);
```
2.1.3.4.5 Line Length

Line length will be kept at or below 80 characters in order to facilitate printing of source code (see Figure 4).

```
12345678911234567892123456789312345678941234567895123456789612345678971234567898
```

Figure 5. Standard for Loop Usage

2.1.3.4.6 Loops

Loops will have zero spaces immediately before and after the parentheses. “for” loops will have exactly zero spaces before the semicolons and exactly one space following them (see Figure 5). If multiple comparisons are done at once, with && or ||, then the individual comparisons shall be enclosed in parenthesis, with exactly one space between the parenthesis and the && or ||.

```
for(i=1; (i<5) && (i%3);
    i++)
```

Figure 5. Standard for Loop Usage

2.1.3.4.7 Operations

Both comparison and assignment will be done with zero spaces before and after the operator (see example below). Other operators shall have exactly one space to either side, unless this would cause the line to exceed 80 characters.

Example: x=2 + 2;

2.1.3.4.8 Semicolons

Semicolons shall never be immediately preceded by white space.

Example: function();

2.1.3.4.9 Testing

Two months have been reserved for testing. Because the program is split into two large modules, there are two large modules of testing to be done:
Graphical and non-graphical.

The graphical code may be imported from another source (see “Considered Graphical Packages”) and can therefore be expected to be relatively bug-free. If graphical testing must be done, the team shall attempt to explore each aspect of the GUI.

The non-graphical code will almost certainly need to be written by the team. Fortunately, automated testing is much easier to do with non-graphical code than graphical code and this is what will be utilized.
2.2 Detailed Design

This section will present an in-depth look at the end-product design. It will cover GUIs, state diagrams, outputs, and program internals.

2.2.1 User Interface

There will be two separate user interfaces, one for the configuration tool and one for the visualization tool.

Figure 6 is a mock-up of the GUI for the configuration tool. This is the default view that will be presented to the user upon startup of the program. The user can immediately begin defining a new network layout. On the left-hand pane there are a few network objects that can be dragged to the workspace in the center of the window. Once an object has been placed in the workspace, its properties can be updated via the right-hand pane.

Switching the view via the workspace tabs, or selecting File->Save will cause the current network configuration to be compiled into an XML document and a specially formatted ASCII file. The ASCII file is the output necessary to configure ISEAGE. The XML document is used internally in the configuration
tool, and to configure the visualization tool.

Figure 7 is a state diagram for the configuration tool. This shows a basic program flow as observed by the end user.
Figure 8. Configuration Tool Use-Case

Figure 11 shows the overall use-case for the configuration tool. This describes each operation that the user can perform with their associated actions.
Figure 9 is a mock-up of the GUI for the configuration tool. This shows an example physical view of a network. The main frame will be updated with real-time network statistics. Selecting an object from the main frame will display statistics specific to that object in the right-hand pane. For larger networks, the user can zoom the view in or out.

The user will also have the option to create and view data replays. The user can select a record option that will cause the system to begin saving all network data to a file. This file can be reopened later and the system will replay the data sequence. This interface will also include a replay slider. The slider will allow the user to seek to a particular time during the replay. The replay controls are not shown in Figure 4.

There are two view types, topology and physical. The topology view is the default view, and will look exactly as it was defined in the configuration tool. The physical view uses a coordinate system (either latitude-longitude or user-defined) to display the network as it is physically laid out. The background for the physical view is user-defined, and can range from a state map to an office layout.
Figure 10 is a state diagram for the visualization tool. It shows the basic flow of the visualization tool from the users perspective.
Figure 11 is a use-case for the visualization tool. It outlines each possible user operation and its associated actions from the users perspective.

### 2.2.2 Configuration Tool

The configuration tool is designed to be extensible and scalable. There is a standard API for creating new network node types. The configuration tool will come pre-loaded with the information necessary for designing a network of routers. Other network devices may be added at a later date.
Users of the configuration tool need to specify some basic information about the router. Optionally, they can also enter information about the ISEAGE board on which the router will reside. When the user saves the network, or chooses to view the ISEAGE configuration file the system will parse the user's network and fill in missing information. Most importantly, the system will run a modified version of the RIP protocol to populate the routing tables of every network device. Performing this automatically will greatly reduce the amount of information that the user must provide.

Network configurations are stored as two separate files. The first file is an binary file that describes properties and spatial data for each object. This file format will be native to the configuration and visualization tools. The second file is an ASCII file that is used to configure ISEAGE. Although both of these files will be viewable from the configuration tool, neither will be editable. The details of both of these formats will be discussed in the File Formats section.

2.2.3 Visualization Tool

The visualization tool can display the network in either topology or physical mode. General network statistics and specific object statistics are updated in real-time. The user begins by opening a network configuration. Initially, the network is displayed in topology mode. The network in this mode will appear exactly as it was laid out in the configuration tool. By clicking on the “Physical View” tab, the user can switch to a view that displays the network as it is physically laid out. Each object in the configuration file has an associated set of coordinates that the visualization tool uses to arrange the objects in physical mode.

Once a configuration is loaded the system will begin collecting statistics from ISEAGE. If the information it is collecting does not match the loaded configuration the user will be presented with an error and asked to open a different configuration. The user can click on any object or connection and the right-hand pane will display detailed information about that object. If no object is selected the right-hand pane will display general information about the network as a whole.

At any point the “Replay Toolbar” can be used to record the visualization. The system will create a time stamped file with the raw statistics gathered from ISEAGE. This file can then be used on any system with the visualization tool, without being connected to ISEAGE, to replay the results of attacks or anomalies. The user will be able to jump to any time and pause the replay.

The visualization tool must communicate with ISEAGE in order to retrieve the network statistics. The details of the communication has not been finalized. The simplest solution would involve opening a port for listening on the user’s computer. ISEAGE itself would take care of aggregating and sending out the relevant data. This capability already exists in ISEAGE.

2.2.4 File Formats
ASCII File

This file is used to configure ISEAGE. This file format was specified by the client.

```
#This is a test config file for a simple one board test case
board=0
router=0
type=outside
name=Outside Router
# hwaddr, interface ip address, ip range, mask
if_out=23:FF:45:12:45:55,192.168.5.254,192.168.5.0,24
if_out=23:FF:45:12:45:55,192.168.6.254,192.168.6.0,24
# interface ip address, [hwaddr]
if=192.168.7.8,23:22:45:12:45:59
# dest ip address, mask, board, id, interface
```

Figure 12. ASCII File

2.3 Implementation Process

At the core of both the visualization and configuration tools is a robust diagramming component. This component would provide the majority of the user interaction for the system. In principle, the team wanted a system that appeared functionally similar to Visio and other commercial diagramming solutions. None of the team members had any experience writing software of this nature and scale. After quite a bit of research we were able to develop a robust system, loosely based on the open-source Netron Graph Library by Francois M. Vanderseypen. This portion of the project took up far more time than we had initially estimated.

The next stage of the project involved creating the Router and Network classes. The Router class needs to store user provided properties and generate a few more on its own. The Network class also only stores a few properties. This section actually turned out to be simpler than expected and helped make up some time that was lost during the creation of the diagramming control.

The final stage of the configuration tool implementation is a class that generates the required output. Upon reaching this stage we realized that the user was being asked to provide a large amount of information. When creating a large network this would make the tool's use infeasible. We decided that the user would only be asked for a minimum of information and the system would generate the rest on its own. To this end, we decided to implement a modified version of the RIP protocol to populate every router's routing table. This decision led to the configuration tool having a much broader scope than originally intended. In order to satisfactorily complete the configuration tool we are recommending that the visualization tool be completed by a phase 3 team.
2.4 **End Product Testing**

The final implementation of the product has not yet been completed. Therefore, we have not begun testing on the end product. This section will describe the process that will be followed when testing is begun.

Testing is separated into two major types: Unit and integration. Unit testing is used to determine that a single component is functioning correctly while integration testing is used to determine that a newly-added component is functioning correctly within the context of the rest of the program.

Within these two types, there are three methods of testing which will be applied: Monkey, pre-selected, and user. Monkey and pre-selected apply primarily to unit tests, while user applies almost exclusively to integration testing.

### 2.4.1 Monkey Testing

Monkey tests are created by generating random or a large set of sequential inputs to independent functions. Ideally, every possible combination of inputs will be sent to a given function. If two implementations of the same function are available, then the two will be compared to each other, with any anomalies reported. In order to accomplish this, a loop equivalent to Figure 13 will be used:

```java
for(i=0; i<0xFFFFFFFF; i++)
{
    if(outOfBounds(function(i))
    {
    }
}
```

*Figure 13. Monkey Testing Example 1*

For functions with multiple arguments, a loop equivalent to Figure 14 will be used:

```java
for(i=0; i<0xFFFFFFFF; i++)
{
    for(j=0; j<0xFFFFFFFF; j++)
    {
        outOfBounds(function(i, j));
    }
}
```
Functions with additional arguments will have additional nested loops, as appropriate. The code segment in Figure 14 above iterates through every possible integer, positive and negative, for all inputs, guaranteeing that any error which can occur from an independent input does happen. This will not successfully test for cases where the output of a function depends on previous inputs to a function. The function outOfBounds() is intended to represent a method of checking for whether a function's return value exceeds its limitations. If it does, the function is implemented incorrectly. Error messages are acceptable for the result of outOfBounds().

In order to test functions with string inputs, or any arbitrary sequence of input, a function must be created where the first sequence of the input begins at the lowest possible value of that data type (in the case of strings, this data type is a single character), then increments to its maximum. When it would otherwise exceed its maximum, it is reset to zero, with a second item in the sequence. This continues to a reasonable point, defined in the limitations; in the case of routers, it would be at least 300. These tests ignore PRE conditions, so it must rely on bounds checking for correct output (or errors) to result.

### 2.4.2 Pre-selected Testing

Pre-selected tests are inputs which will be sent to a given function with anticipated output, created by the black box tester. The resulting output of the function will be compared to the anticipated output. If this is identical, that instance of the test will have passed. If not, the entire test will have failed and returned to the original author for debugging. If the original author can not locate the source of the problem, the code will be forwarded to the white box tester for fixing.

### 2.4.3 User Testing

User testing will be done last and is done by all of the team members by attempting to use the program as a regular user would. If any anomalies are found, then the original author will attempt to locate the source of the problem. If unsuccessful, the code will be forwarded to the white box tester. Ideally, someone unrelated to the project can be located to perform the same type of tests, also identifying user interface problems in addition to technical problems.

### 2.4.4 Testing

Each team member is responsible for being the primary black box tester of a given member's code. Black box testers are to test code without examining the
code itself in order to avoid having any assumptions outside of those specified by the conditions of the code (see Figure 15).

Figure 15. Inter-Group Testing Procedure

A second tester will be the white box tester and primary debugger. This tester will perform tests for a given member's code by examining the code and attempting to find errors, then creating tests, which exploit those errors. In simple cases, the white box tester will fix the code and return it to the primary author for evaluation, noting the changes. The original author will then use his discretion to determine if these changes fix the problem without causing new problems and alter the code further if additional problems do occur.

The fourth member (outside of the original author and the two testers) will act as a secondary debugger. In cases where the original author and the white box tester are unable to identify the source of the problem, the code will be forwarded to the secondary debugger. Only if all three fail to locate the source of a problem should the fourth member (the black box tester) attempt to identify the source of the problem within the code.

To date, Eric and Piyush have been assigned to work on the graphical components while JD and Trent have been assigned to the non-graphical components. For testing, Trent will be the black box tester for Eric's code, Piyush for JD's code, Eric for Trent's code, and JD for Piyush's code. Piyush and Eric will be the white box tester for the other's code, while JD and Trent will also be mutual white box testers. Eric is the secondary debugger for JD, JD for Trent, Trent for Piyush, and Piyush for JD. After a bug appears to have been fixed, the original tester will re-check it.
2.5 **Product End Results**

2.5.1 **Configuration Tool**

The configuration tool is approximately 80% complete. We project that it will be fully completed by the end of the semester. The scope of the configuration tool changed midway through the project and it has pushed back the estimated completion time for this tool.

2.5.2 **Visualization Tool**

Due to the change in scope of the configuration tool, we will not be attempting the visualization tool. We are recommending that it be completed by a subsequent team.
3 Resources and Schedules

This section discusses the cost of executing the project in personnel hours, project-related expenses, and income the team would earn in an industry setting.

3.1 Resource Requirements

This is a pure software project so the costs are limited to what must be produced for documentation, labor costs, licensed software, and borrowed hardware. Iowa State University has site licenses for Microsoft Windows XP and Microsoft Visual Studio .NET 2005.
### 3.1.1 Personnel effort

<table>
<thead>
<tr>
<th></th>
<th>Original Projected Effort</th>
<th>Current Effort</th>
<th>Projected Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Problem Definition</td>
<td>Technology Consideration and Selection</td>
<td>End-Product Design</td>
</tr>
<tr>
<td>Eric Anders</td>
<td>9</td>
<td>15</td>
<td>31</td>
</tr>
<tr>
<td>Jonathan Cook</td>
<td>8</td>
<td>9</td>
<td>25</td>
</tr>
<tr>
<td>Trent Roberston</td>
<td>6</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>Piyush Patel</td>
<td>7</td>
<td>12</td>
<td>30</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>30</strong></td>
<td><strong>46</strong></td>
<td><strong>112</strong></td>
</tr>
</tbody>
</table>

#### Current Effort

<table>
<thead>
<tr>
<th></th>
<th>Eric Anders</th>
<th>Jonathan Cook</th>
<th>Trent Roberston</th>
<th>Piyush Patel</th>
<th><strong>Total</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Definition</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>10</td>
<td>42</td>
</tr>
<tr>
<td>Technology</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>32</td>
</tr>
<tr>
<td>Consideration and Selection</td>
<td>34</td>
<td>27</td>
<td>20</td>
<td>30</td>
<td>111</td>
</tr>
<tr>
<td>End-Product Design</td>
<td>70</td>
<td>53</td>
<td>51</td>
<td>56</td>
<td>230</td>
</tr>
<tr>
<td>Implementation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Testing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Documentation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Demonstration</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>Reporting</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>42</strong></td>
<td><strong>32</strong></td>
<td><strong>111</strong></td>
<td><strong>230</strong></td>
<td><strong>443</strong></td>
</tr>
</tbody>
</table>

#### Projected Total Effort

<table>
<thead>
<tr>
<th></th>
<th>Eric Anders</th>
<th>Jonathan Cook</th>
<th>Trent Roberston</th>
<th>Piyush Patel</th>
<th><strong>Total</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Definition</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>10</td>
<td>42</td>
</tr>
<tr>
<td>Technology</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>32</td>
</tr>
<tr>
<td>Consideration and Selection</td>
<td>34</td>
<td>27</td>
<td>20</td>
<td>30</td>
<td>111</td>
</tr>
<tr>
<td>End-Product Design</td>
<td>79</td>
<td>67</td>
<td>69</td>
<td>68</td>
<td>283</td>
</tr>
<tr>
<td>Implementation</td>
<td>15</td>
<td>18</td>
<td>14</td>
<td>17</td>
<td>64</td>
</tr>
<tr>
<td>Testing</td>
<td>10</td>
<td>14</td>
<td>11</td>
<td>12</td>
<td>47</td>
</tr>
<tr>
<td>Documentation</td>
<td>8</td>
<td>12</td>
<td>7</td>
<td>10</td>
<td>37</td>
</tr>
<tr>
<td>Demonstration</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>Reporting</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>163</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>42</strong></td>
<td><strong>32</strong></td>
<td><strong>111</strong></td>
<td><strong>283</strong></td>
<td><strong>644</strong></td>
</tr>
</tbody>
</table>

**Figure 16. Personnel Effort**

### 3.1.2 Other Resources

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poster</td>
<td>$75</td>
</tr>
<tr>
<td>Printing</td>
<td>$20</td>
</tr>
<tr>
<td>Misc.</td>
<td>$55</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>$150</strong></td>
</tr>
</tbody>
</table>
Most of the project’s design will be software implementation. ISEAGE has all of the computer materials needed to complete the project. The software used for the GUI interface, Microsoft Visual Studio .NET 2005, will be freeware thanks to MSDNAA therefore will not be included in this section.

### 3.1.3 Financial Costs

Total financial costs sum up the personnel expenses and the other resources expenses. Labor is calculated at wage of $10.50/hr.

<table>
<thead>
<tr>
<th>Item</th>
<th>W/O Labor</th>
<th>With Labor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Projected Costs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parts and Materials:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Printing expenses including Poster</td>
<td>$95.00</td>
<td>$95.00</td>
</tr>
<tr>
<td>b. Miscellaneous</td>
<td>$55.00</td>
<td>$55.00</td>
</tr>
<tr>
<td><strong>Subtotal</strong></td>
<td>$150.00</td>
<td>$150.00</td>
</tr>
<tr>
<td>Labor Expenses:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eric Anders</td>
<td></td>
<td>$1848.00</td>
</tr>
<tr>
<td>Jonathan Cook</td>
<td></td>
<td>$1837.50</td>
</tr>
<tr>
<td>Trent Roberston</td>
<td></td>
<td>$1732.50</td>
</tr>
<tr>
<td>Piyush Patel</td>
<td></td>
<td>$1858.50</td>
</tr>
<tr>
<td><strong>Subtotal</strong></td>
<td>$0.00</td>
<td>$7276.50</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$150.00</td>
<td>$7426.50</td>
</tr>
</tbody>
</table>

| **Revised Estimated Costs**         |           |            |
| Parts and Materials:                |           |            |
| a. Printing expenses including Poster | $75.00    | $75.00     |
| b. Miscellaneous                    | $14.80    | $14.80     |
| **Subtotal**                        | $89.80    | $89.80     |
| Labor Expenses:                     |           |            |
| Eric Anders                         |           | $1827.00   |
| Jonathan Cook                       |           | $1743.00   |
| Trent Roberston                     |           | $1480.50   |
| Piyush Patel                        |           | $1711.50   |
| **Subtotal**                        | $0.00     | $6762.00   |
| **Total**                           | $89.80    | $6851.80   |

| **Current Costs**                   |           |            |
| Parts and Materials:                |           |            |
| a. Printing expenses including Poster | $75.00    | $75.00     |
| b. Miscellaneous                    | $14.80    | $14.80     |
| **Subtotal**                        | $89.80    | $89.80     |
| Labor Expenses:                     |           |            |
| Eric Anders                         |           | $1386.00   |
| Jonathan Cook                       |           | $1134.00   |
3.2 Schedules

3.2.1 Project schedules

Figure 18. Project schedules

3.2.2 Deliverable schedules

Figure 19. Deliverables schedules
4 Closure Materials

This part of the report deals with some concluding information. Included are the team's evaluation of itself based on the project, the lessons learned through the project, commercialization issues, and contact information.

4.1 Project Evaluation

This section describes the milestones for the project and the criteria used to evaluate the success of the completed project. For each milestone, the team evaluated the completeness and success of that item and assigned an importance score to it. The team then took the scores for all of the individual milestones and multiplied them by their overall importance metric to get a final score for the project. A final score of greater than 85% is considered a success for the project.

4.1.1 Project Definition

Description: The project had to be adequately defined before we could start designing the project. An accurate definition ensured all team members knew the functionality of the project approaches possible.
Evaluation criteria: Requirement
Overall importance: 15%
% Completed: 100%

4.1.2 Technology Considerations and Selection

Description: Team members had to research available software technologies such as graphics packages, GIS software, programming languages, and database structures. From these, the team selected the technologies that helped produce software that met all functional requirements with efficiency and effectiveness in mind.
Evaluation criteria: The team determined how well all options for technology were considered. The team determined how well the selected technologies met the requirements of the project.
Overall importance: 10%
% Completed: 100%

4.1.3 End-product Design

Description: The end-product design detailed all components of the final product. This expanded the functional requirements to include architectural details of the software and implementation-level design choices. The design provided interface specifications for software modules and classes.
Evaluation criteria: The team determined how well the design accomplished the functional requirements and facilitated ease of implementing the design.
Overall importance: 20%
% Completed: 90%

4.1.4 End-product Implementation

Description: The prototype is the realization of the product design in actual software.
Evaluation criteria: The team determined how well the prototype conforms to the functional requirements and design.
Overall importance: 12%
% Completed: 90%

4.1.5 End-product Testing

Description: The end-product testing ensured that the product is free of bugs and errors.
Evaluation criteria: The team determined the quality, usability, and lack of bugs of the software.
Overall importance: 15%
% Completed: 80%

4.1.6 End-product Documentation

Description: The end-product documentation provides a user manual for the software.
Evaluation criteria: The team determined how clearly the end-product documentation described the usage of the system, as well as end-users.
Overall importance: 8%
% Completed: 80%

4.1.7 End-product Demonstration

Description: The final software was presented to the client. The client was informed of the use of the software.
Evaluation criteria: The team determined how well the end-product solved the intended purpose of the project.
Overall importance: 8%
% Completed: 80%

4.1.8 Project Reporting

Description: The team created several documents as the project progresses. These included the project definition, project plan, product poster, end-product design, and final report. The final reporting for the project described all aspects of the system and included maintenance documentation that details the implementation of the project. The final report will be the longest surviving record of the project implementation and had to account for any changes to functional requirements, design,
and architecture made through the development of the end-product.

**Evaluation criteria:** Requirement

**Overall importance:** 12%

**% Completed:** 80%

### 4.1.9 Final Project Score

The result of the project evaluation has resulted in an overall final project score of 89% which results in a successful project based on the criteria defined at the onset of the project.
4.2 Commercialization

There are no considerations made for commercialization of our project. This project is solely intended for private ISEAGE use.

4.3 Recommendations for Future Work

This section makes recommendations for continued work on the project. This additional work includes items that were passed over in the infancy of the project and items that were passed over during the development of the software as it became clear that time constraints would not allow their completion. It should be noted that the majority of these items were not explicitly defined in the team’s design report, as they were not discovered until implementation of the end-product. However, these are items that would be desirable in a finished product.

4.3.1 Optimizations

The first recommendation for additional work would be to optimize the specification tool. The main concern is to get a more precise drag-n-drop GUI, as well as make the diagrammer more diverse by allowing a larger selection of nodes.

4.3.2 Interactive Software Tutorial and Help Guides

The initial release of the project will include a manual and documentation on the functionality of the project. However, an interactive tutorial would be useful to the user, providing a visual learning tool that would otherwise be absent in a set of hardcopy instructions.

4.3.3 Pseudo-real-time reporting

The team will be unable to start the report tool of the project. The idea would be to adapt means of recording, pause, play, fast-forward and rewind.
4.4 Lessons Learned

This section discusses what the team gained from the experience of executing a two-semester project. Included are things that went well in the project, things that didn't go as well, and what we would have differently.

4.4.1 What Went Well

Initial design of the project went well. We did not change the scope of the project. Most of the original project plan was followed directly.

4.4.2 What Did Not Go Well

Coding complexity was underestimated. The report tool will have to be passed on as a future requirement of the final product. A more precise diagrammer will be needed as well.

4.4.3 What Technical Knowledge Was Gained

Programming in C# will be gained by every group member, as none of us have ever used it before. Much of the industry is making a turn toward C# and away from Java, as most of programming is needed for Windows machines.

4.4.4 What Non-Technical Knowledge Was Gained

Project management skills were gained, by project documentation, meetings and project planning/organizing. This will be greatly used by anyone pursuing future in programming as a professional.

4.4.5 What To Do Differently If Starting Again

None seen at this time.
4.5 Risk and risk management

4.5.1 Anticipated potential risks and planned management

Loss of Team Member
Assessment: All team members will track their own progress and notify the team of any concerns or problems that might arise. If the team does lose a member their remaining work will be divided amongst the remaining members.

Insufficient knowledge or experience
Assessment: The group will be responsible for assigning tasks to members that have the necessary knowledge to complete them. If no team member feels that they have the necessary knowledge to complete the task, a team effort will be made to learn the required information or a new approach to the problem will be considered. There will be time for learning built into the schedule.

Loss of data
Assessment: The team will use a CVS or Subversion server to control the source code for the project. In this way the source will be controlled and tracked in a secure environment.

Failure of approach
Assessment: All technologies will be researched to guarantee interoperability. All plans, procedures, and software progress will be submitted to the client for review.

4.5.2 Anticipated potential risks encountered and success in management

Underestimation of the complexity of the coding was a risk that was not assessed. The team got together and formed a new plan as to what was the most important to complete, or to at least get a good code base on. The team concluded that the best thing is to get the user-friendly drag-n-drop GUI, diagrammer, and network specification tool completed and leave the report tool as a future consideration.

4.5.3 Resultant changes in risk management made because of encountered unanticipated risks

No changes were made in risk management due to encountered

4.6 Project Team Information
This section includes contact information for the client, the faculty adviser, and the team members. Team positions and major(s) of study are also listed.

4.6.1 Client Information

**ISU Information Assurance Center**
Dr. Douglas W. Jacobson  
2419 Coover Hall  
Ames, IA 50011  
515-294-8307  
dougj@iastate.edu

4.6.2 Faculty Advisor

**Dr. Douglas W. Jacobson**
2419 Coover Hall  
Ames, IA 50011  
515-294-8307  
dougj@iastate.edu

4.6.3 Team Members

**Eric Anders**  
Computer Engineering  
4912 Mortensen #222  
Ames, IA 50014  
(641) 750-2033  
eanders@iastate.edu

**Jonathan Cook**  
Computer Engineering  
203 Campus Ave. #8  
Ames, IA 50014  
(515) 268-0886  
jdcook@iastate.edu

**Piyush Patel**  
Computer Engineering  
2503 Ferndale Ave. #2  
Ames, IA 50010  
(515) 663-9556  
papatel@iastate.edu

**Trent Robertson**  
Electrical Engineering  
Computer Engineering  
1225 Delaware #9  
Ames, IA 50014
4.7 Closing summary

With the integration of this component into ISEAGE, users will be able to create and evaluate custom networks configured on virtual hardware, defend against simulated attacks, and assess network stability leading to a more reliable configuration of network infrastructure. The network specification tool is used with the intent to construct and monitor a virtual network. The report system will be a real-time system that will overlay a map and allow the topology of a network to be monitored. An extremely easy, user-friendly interface dynamically configures the created virtual network with the corresponding physical one. Creating and configuring custom networks has never been made so easy!