Fall 12-1-2005

Software Project Management Plan for System Management Tool (SMT)

Diana J. Marrs
Dakota State University

SOFTWARE PROJECT MANAGEMENT PLAN FOR SYSTEM MANAGEMENT TOOL (SMT)

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

December, 2005

By

Diana J. Marrs

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Project Committee

Faculty supervisor: [illegible] Date: 12/9/05

Committee member: [illegible] Date: 12/9/05

Committee member: [illegible] Date: 12/9/05

ACKNOWLEDGMENT

I'd like to thank my project sponsor, IT Director Joe Lorino, for giving me this opportunity. He has been supportive and active in the project throughout the development process. I'd also like to thank Adam Sailer for his programming expertise and participation and dedication to this project.

ABSTRACT

The University of Kansas Edwards Campus expanded from one building with three computer labs to two buildings with six computer labs, doubling the number of computers that must be managed, yet there is no funding for additional IT staff. The campus will add a third building in the next five years. The current method for maintaining computers, service packs, updates, and software is to manually check them. There is no systematic method for keeping track of license counts and maintaining virus protection and updates. A more efficient method of managing computers was needed. We examined commercial products and determined that they did more than was needed and yearly client and server licenses would be cost prohibitive.

The IT director, as project sponsor, decided to develop a management tool in-house. Diana Marrs, project manager and secondary programmer, and Adam Sailer, primary programmer created an object-oriented development plan. The project scope included obtaining and displaying system information, hardware and software per machine in the network. Three functional areas were developed – data acquisition and storage, network communication, and a graphical user interface. Because there was no need for historical data and each data object was stored on the server and read directly from the GUI, there was no need for a database to be developed so this item was removed from the original plan. Additions to the scope included a systray icon, a print function, an install program, a delete function, and a system details section. Weekly meetings were held between the project sponsor, the network administrator, and the design team for progress updates and revision requests.

The project is near completion. The computer management tool successfully broadcasts a request for data, the client machines gather their data and return it to the server where the data objects are stored, and the server provides a GUI for display. This was progressively tested on 1-69 machines to analyze network load and speed of results. The client will be installed on all 213 computers within the next two weeks and final testing for response and network load will be conducted. A second install will occur when the additional items have been completed. All code up to this point has been open source or newly developed by the team. The amount of data obtained goes far beyond the original request and will identify machines that need updating and help the institution remain within licensing limits.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

[illegible]

<Student name>

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Background of the Problem

The University of Kansas Edwards Campus (KUEC) began as a satellite campus at its current location in 1993. At that time, it consisted of 1 building offering 10 graduate degree programs. In the past twelve years, it has expanded to offering 26 degree programs – 3 of them undergraduate programs, the newest area of program expansion. In addition, in 2004, the campus doubled in size, adding a second building. This added three computer labs, three new office suites and twenty-two new classrooms. In its twelve years of growth, there has never been an increase in IT staff. There are four core IT administrators and a revolving group of student lab assistants. Computers maintenance includes current virus protection, software and operating system updates, standard software installs on all machines and specialized installs on some machines. Up to now this has been done manually.

When the second building opened, computer support for classroom, office, and lab computers increased from 110 PCs to 213 machines. In addition to a computer, every classroom in both buildings is equipped with a projector, sound system, and OHP all supported by the IT department. There are also specialized classrooms for streaming, video conferencing, and interactive TV all of which must be supported by the minimal IT staff. The future expansion plan for the campus includes two more buildings added in the next 10 years. The mission of the campus is to meet the needs of the workforce, economic and community development by offering high quality academic programs, providing the resources and tools needed to support working professionals. In line with the campus goal is the IT department goal:

> The Information Technology Services group at the KU Edwards Campus supports the teaching, learning and research pursuits of its faculty and students, and the professional activities of the campus administration and community groups by the thoughtful and innovative application of technology. It also embraces experimental and leading-edge technology initiative and strives for the highest level of infrastructure in both virtual and physical learning environments.

The campus goal is to grow and offer more quality programs while maintaining the same level of customer service and student support, ultimately requiring more IT infrastructure, without growing its operational costs or increasing staff.

Statement of the problem

In light of the recent campus expansion, limited budget, and inability to hire additional staff, a remote management tool would enable the IT staff to more efficiently and effectively maintain hardware, quickly identify where special software is installed and track licensing.

The Edwards Campus IT department already has the ability to remotely connect to computers on campus. This is used to for rapid software installs and technical support issues. What it doesn't offer is an efficient way to tell which computers need updates and service packs, the system information for each computer, and a list of software installed for license management. This information has to be collected machine by machine manually. There are many existing remote management tools on the market today, but they are cost prohibitive for the Edwards Campus IT budget, requiring both a server license and a per seat client license which must be renewed on a yearly basis. Thus it was decided to develop our own software program to gather this information across the network and display the results for analysis and action.

Objectives of the project

The management tool that has been developed will help meet both the university and IT goals. It will ensure the quality of resources available to students and faculty. It increases the level of infrastructure support and efficiency without having funds diverted to hiring additional staff. Computers will have the most recent upgrades, service packs, and anti-virus protection reducing the possibility of PC failure. This will reduce TCO in operation and maintenance and will have the added benefit of inventory and licensing checks.

Critical Assumptions and constraints

1. The tool must run on existing hardware
2. The tool must be user-friendly
3. The network administrator must be involved in development as the end-user and must have full buy-in

Analysis of options

1. Do nothing and hope for more hires in the future so that computers can be manually maintained
2. Look for funding to purchase an already developed software package
3. Design and implement an in-house management tool

Preliminary Project Requirements

The software program and GUI to be developed should provide the network administrator with hardware and software information for each machine. It should run from a server and the GUI should provide options for displaying the data. Specifically, the deliverables include:

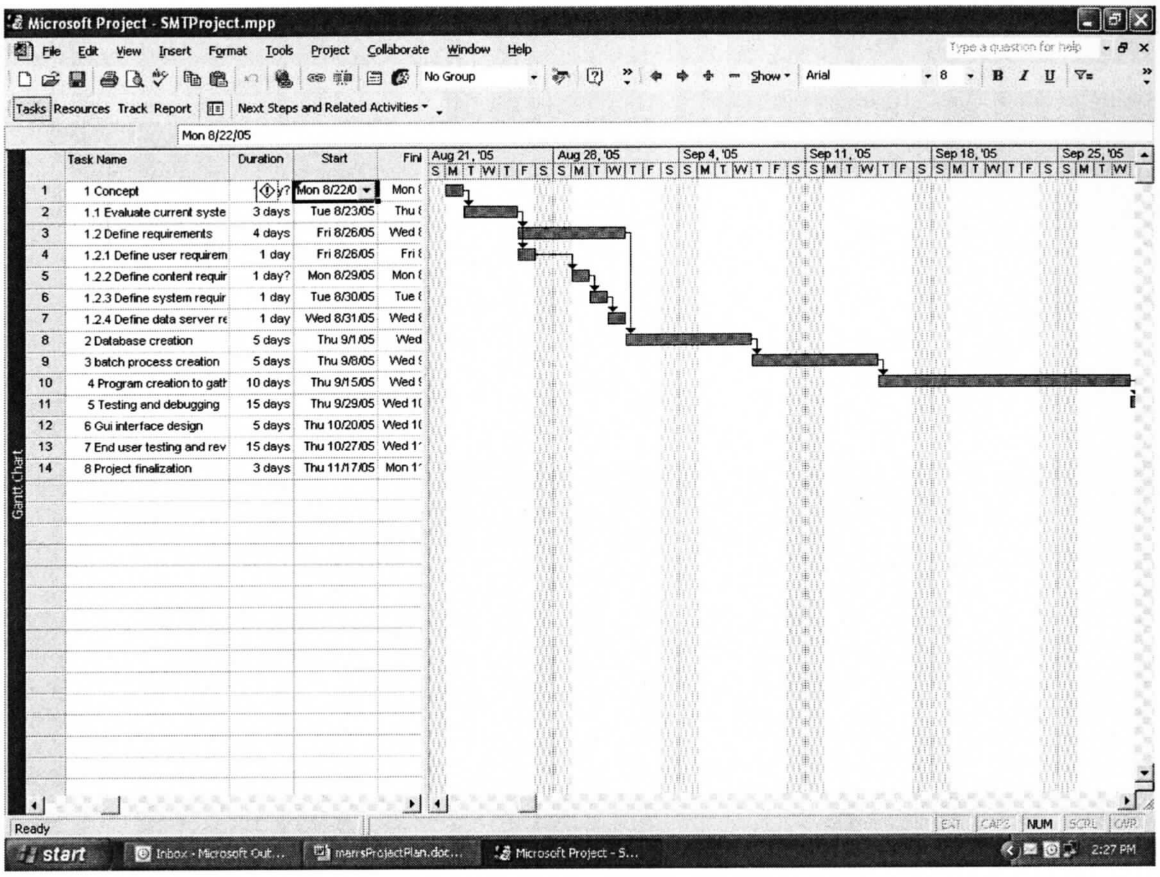
1. Network command to contact every PC within the local area network
2. Program to gather data on each PC
    - o Software including version
    - o OS version and patches
    - o Anti-virus updates
    - o Harddrive info – RAM, HD, CPU class, available disk space
3. Database to store gathered data and return reports
4. GUI interface
5. Other features suggested by users

Budget Estimate and Financial analysis

A preliminary estimate of the cost for the entire project is $3600 but is in terms of projects deferred by staff working on the proposed project rather than on actual cash flow. The estimate is based on 1 person working 3 hours/day for 3 months at $20/hour. Projected benefits are based on a reduction in hours spent physically attending to each PC in two buildings. On average, two people spend 20 hours per quarter manually installing service packs at an annual cost of $3200. It is estimated that using a management tool would reduce this time by half or ¾ resulting in an annual savings of $1600-2400, or to manage twice as many PCs in the same time frame.

Schedule Estimate

The sponsor would like to see the project finished in three months.



Mon 8/22/05

	Task Name	Duration	Start	Fini
1	1 Concept	y?	Mon 8/22/0	Mon
2	1.1 Evaluate current syste	3 days	Tue 8/23/05	Thu
3	1.2 Define requirements	4 days	Fri 8/26/05	Wed
4	1.2.1 Define user requirem	1 day	Fri 8/26/05	Fri
5	1.2.2 Define content requir	1 day?	Mon 8/29/05	Mon
6	1.2.3 Define system requir	1 day	Tue 8/30/05	Tue
7	1.2.4 Define data server re	1 day	Wed 8/31/05	Wed
8	2 Database creation	5 days	Thu 9/1/05	Wed
9	3 batch process creation	5 days	Thu 9/8/05	Wed
10	4 Program creation to gath	10 days	Thu 9/15/05	Wed
11	5 Testing and debugging	15 days	Thu 9/29/05	Wed 1(
12	6 Gui interface design	5 days	Thu 10/20/05	Wed 1(
13	7 End user testing and rev	15 days	Thu 10/27/05	Wed 1'
14	8 Project finalization	3 days	Thu 11/17/05	Mon 1'

Aug 21, '05 | Aug 28, '05 | Sep 4, '05 | Sep 11, '05 | Sep 18, '05 | Sep 25, '05

Figure 1. Work breakdown structure

Potential Risks

The greatest risk to this project is that the network administrator will not buy in and will not utilize the management tool, effectively eliminating any cost benefit. However, the project sponsor is also the IT Director and will manage/require the use of the tool. A secondary risk is the load to the network but will be countered as much as possible in code refinement.

CHAPTER 2

LITERATURE REVIEW

The campus goal is to grow and offer more quality programs while maintaining the same level of customer service and student support, ultimately requiring more IT infrastructure, without growing its operational costs or increasing staff.

According to David, Schuff and Louis in "Managing your IT Total Cost of Ownership" (2002), there are two methods that can accomplish this end – centralization and standardization. As an educational institution, it is not possible to simply replace older equipment although there is a structured method for upgrades. Lab computers get replaced first. The old lab computers move to the classrooms, the old classroom computers move to faculty offices. As might be guessed, the largest area of one-on-one support is in the faculty offices. Magee (2004) lists desktops visits as the highest cost area for desktop support as shown in figure 28:

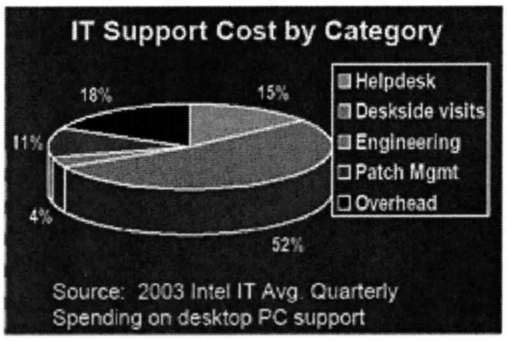


Figure 2. IT support cost by category (Magee, 2004)

This is due to the fact that older equipment tends to run slower and break down more often. In addition, because they are older, they sometimes require special configurations. Forbath, Kalaher and Schenof, in "New Insights on PC Management: Benefits of controlled PC Hardware Diversity" (2004), state that "extending PC lifecycles typically increases software deployment efforts by increasing the total number of deployed PC configurations" (2). The resulting increase in time and effort is shown in Figure 3.

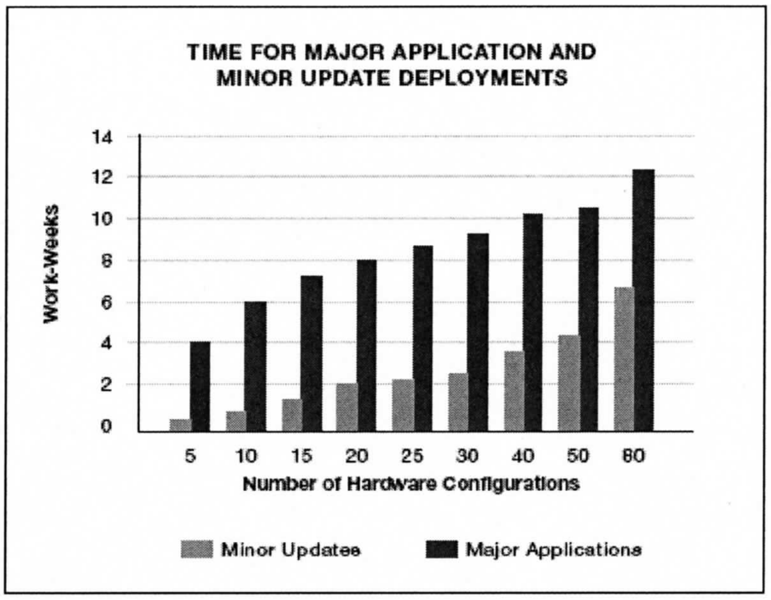


Figure 3. Time needed for deployments (Forbath et all, 2004, p. 2)

The amount of work increases significantly with even just 5 hardware configurations. Another way to look at it is in terms of cost, as shown in Figure 4.

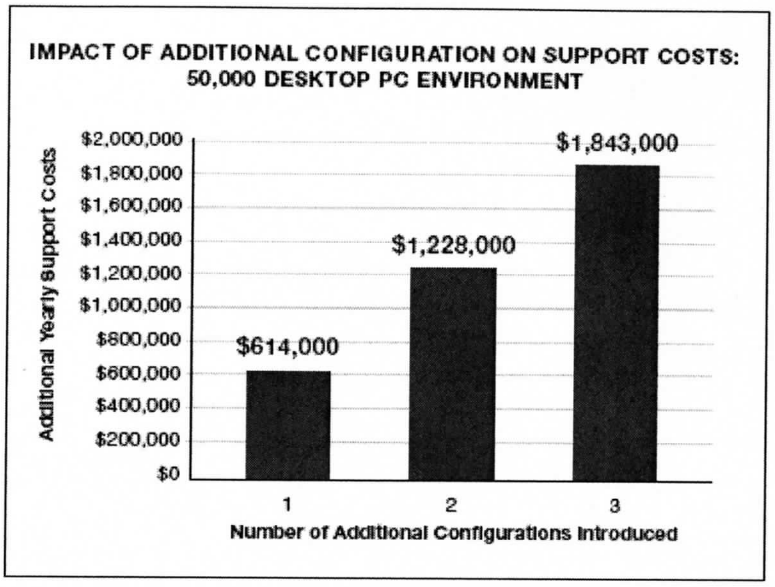


Figure 4. Impact of additional configurations (Forbath et all, 2004, p. 17)

Using these figures, 1 configuration costs $12.28 per year per PC and the cost of support doubles to $24.56 for 2 configurations. Translated into the KU Edwards environment of approximately 200 PCs, the additional cost is $2,456 per year due to the lack of standardization.

Any total cost of ownership analysis for IT lists the bulk of expenses for IT in the operating costs. In fact, up to 80% of the cost of ownership is after the purchase and includes support, installations, upgrades, training, viruses, downtime, etc (David et all, 2002, p.102). The KUEC department has standardized the software and PC configurations as much as possible but the hardware cannot be fully standardized, leaving the other solution for improved TCO – centralization.

The department is now looking at centralization options like remote deployment of software, upgrades and service packs, as well as an efficient means to inventory hardware and software. The goal here is to decrease total cost of ownership as we increase inventory so that hiring additional staff is not necessary. An additional benefit is inventory control and software licensing checks.

There are many network managing systems available on the market. For example, there is Remote Scope which includes the following features:

- Full control of client
- Windows explore type interface
- Drag and drop function
- Auto-run functions to do remote backups and defragmentation
- Remote install of updates, service packs, anti-virus software
- Hardware inventory reports which include hardware, software, and system resource information
- A report creator that allows the user to customize and combine various reports

There are many other similar products on the market including NetSupportMgr 9.0 ($9700), Belarc, Remote-Scope ($3000), LANDesk Management Suite, Vector's PC-Duo Enterprise ($2500 for two modules), Kaseya Computer Audit and Discovery ($2,000 for inventory module only). These products require server and client licensing which would add a renewal yearly cost in the IT budget. For the most part, these products do more than what we really need and cost more than we have funding for. The advantage of developing in-house is that the only true cost is what other projects are delayed while this project is being developed.

CHAPTER 3

SYSTEM DESIGN

It was determined that the Java 2 platform would be the best programming language to use for this program because it has network commands built into existing classes, works well in the PC environment with existing hardware and software, and has an easy-to-use GUI design interface. A basic use-case model was created:

Table 1. Use case model

Use case name: System Information requested	**Importance:** High
Primary Actor: Network Administrator	
Stakeholders and interests: Network administrator, user, vendor, IT director, University	
Trigger: License audit	
Relationships: all internal objects are included	
Normal Flow of events 1. Network administrator receives request for licensing information 2. Network administrator opens program 3. Network administrator clicks on "Scan Now" 4. Server broadcasts request for data 5. Each client PC retrieves data and returns data summary object to server 6. GUI displays data 7. Network administrator chooses software program and gets total count and location of install	
Alternate/exception flows: if PC is turned off, the retrieval program will skip this PC and the GUI will read the last known data saved to the server for this PC.	

The same use-case can be used for all possible requests which include location of special software installs, system summary for any individual PC, total number of installs in the LAN for any software program, and identification of machines needing updates.

Using the methodology described in *Systems Analysis and Design* (Dennis, et al). the project was use-case driven and used a form of phased development – the project was divided into functional areas and developed one at a time, with analysis, design, implementation, testing and user review for each area. The use case model was created to demonstrate the functional view/behavior the user would see. The first functional area to be developed was the network communication piece. This would run from the server, broadcast a request for data across the network from a specific IP and port. The client would return an object which would be saved to the server hard drive. Using a test object of "date", this piece was tested and fully functional before moving on to the second functional area, the data acquisition. Data acquisition developed in three steps: retrieval of software information, retrieval of system and hardware information, data manipulation. Finally, the GUI was developed and the three areas were joined. As each piece was developed it was demonstrated to the project sponsor and revisions made where necessary.

Static View

The following classes were developed, organized by functional group:

Table 2. List of classes developed, by functional group

Network Communication	Data Acquisition	GUI
Client	Acquire	Autumn
Client Thread	Asset	Sort Comparator
Server	Data Store	Sort Model
Server Thread		Soft Mouse Adapter
		Sort Renderer

Dynamic View

There are two install options – a server and client, or just a client. The client runs continuously on all PCs and does not have a GUI. When the program is opened by the network administrator (Autumn Object), it immediately creates a client and a server object. These extend thread and continue to run and "listen" for a message from the network as long as the GUI is open. The server object broadcasts a message across the LAN and listens for a response from a client. The packet contains a specific IP and port number. It also creates a Server Thread. The server thread will receive the Summary object sent from the client PC, serialize it and save it to the server hard drive. The client object running on each PC in the network receives the packet and this activates the client to create a Client Thread object. The client thread will stop when its methods are done.

The client thread gathers the IP address, port number, and date and creates the Summary object. The summary object creates the Acquire object to gather the data from the PC. We discovered that "winmsd" used in the command line, returned all system and hardware information (as well as a great quantity of additional data on software and internet settings). Acquire captures only the data we need from this area (rather than doing a full dump, to save processing time). It then does some extensive data cleanup, using XML to identify tags/categories, breaking items up into individual items, creating an Asset object for each item. For example, the CD-ROM drive is an Asset, with the following attributes:

- Description
- Media Loaded
- Media Type
- Name
- Manufacturer
- Status
- Transfer Rate
- SCSI Target ID
- PNP Device ID

And these attributes have corresponding values all of which become one Asset. The acquire object creates a new asset for each hardware device, and one for the system summary. It then moves on to applications. This information can be found in many places but the most comprehensive location was found in the registry key at HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall. Each software application becomes an asset. The summary object uses the acquire object to gather each asset as it is created and saves it to a single file which will be sent to the server. The class diagram on the next page illustrates how the classes interact.

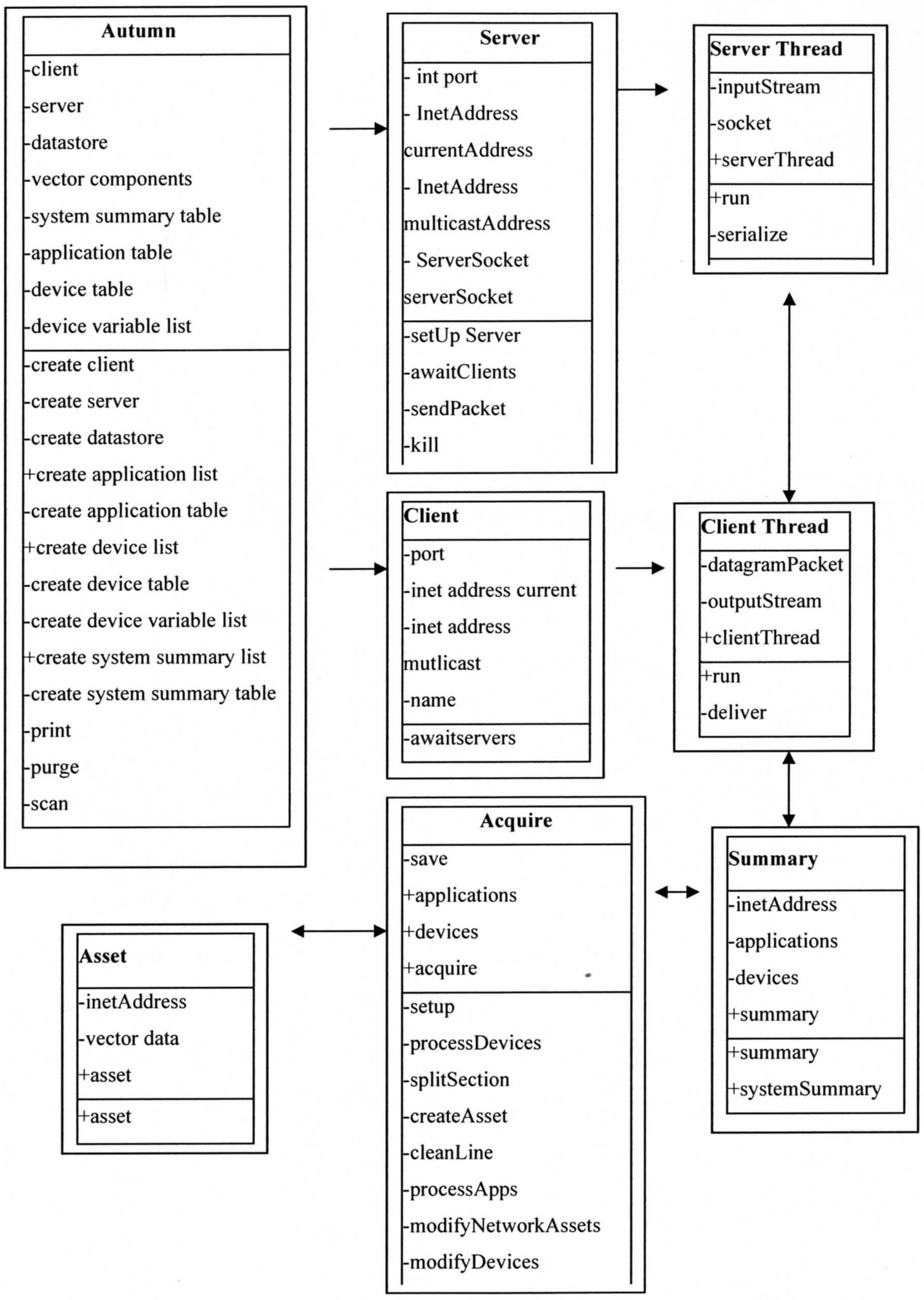


Figure 5. Class diagram 1

The Autumn object also creates a DataStore object. The server thread object receives the summary object from each PC and serializes it a ".data" file so it can be stored to the hard drive using the PC IP address as a file name. The dataStore object is then used to deserialize the data back into a Summary object so it can be read in the GUI. The dataStore object creates the SortModel which creates the SortMouseAdapter, the SortRenderer, and SortComparator all of which controls the behavior of the customized JTable in the GUI as it responds to the mouse (user interface). With these objects, the user can move columns about, sort any column in descending or ascending order, and select which variables will be viewed.

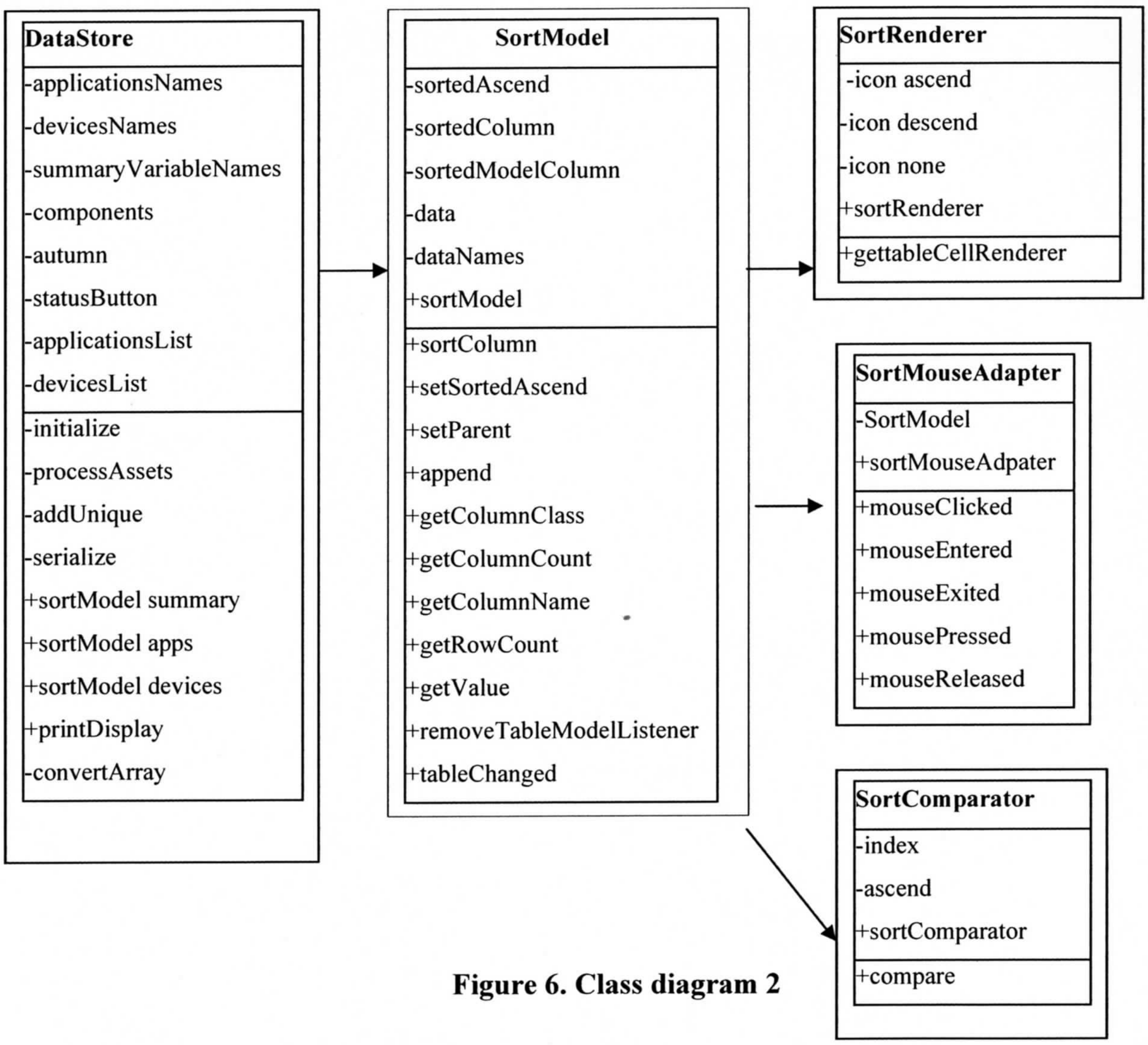


Figure 6. Class diagram 2

CHAPTER 4

RESULTS AND DISCUSSION

Graphical User Interface (GUI) and functions

The GUI was developed with four buttons: scan now, which broadcasts the message to return data; update display, which loads the files into the display table; purge data, which deletes all the data files on the server; and print, which opens the display table into a webpage so it can be printed from the browser. The user clicks on the .jar files to open the GUI, then the scan now button to begin. To view existing files, the user can click on update display. When the update display button is clicked the status of the upload is shown to the right of the print button. When scan and display are finished, status shown in menu bar at top 100% and 1st tab opens be default. The attributes for system display are shown in the left.

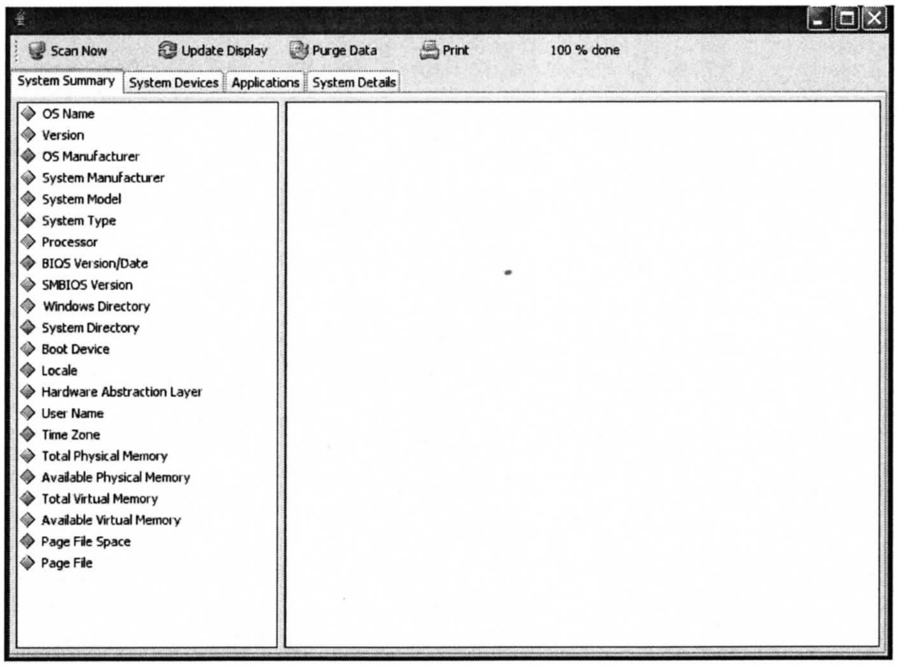
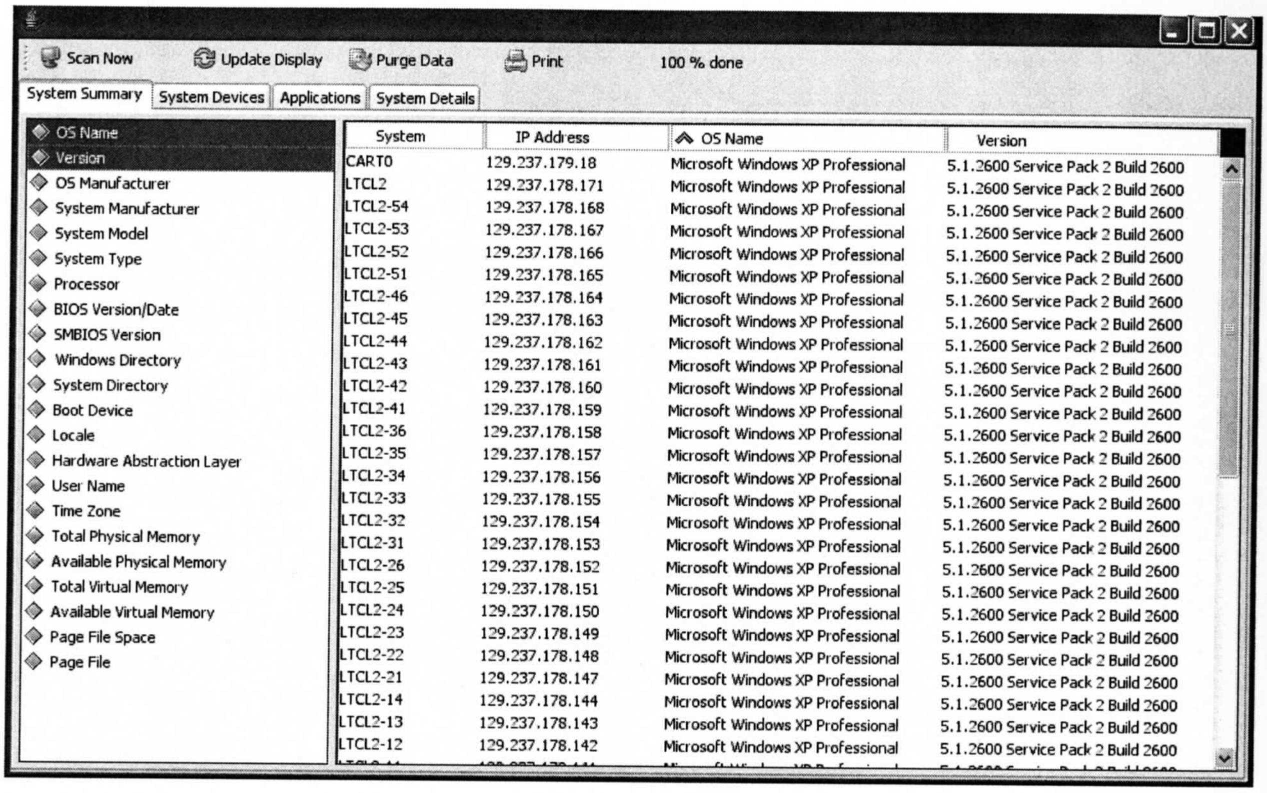


Figure 7. Default view of GUI

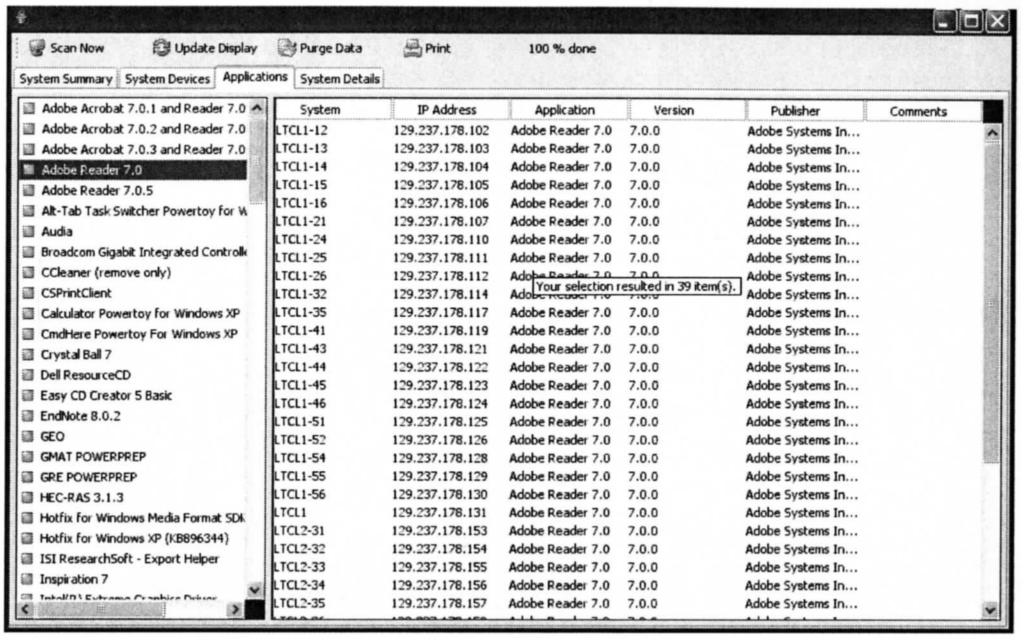
The user can then click on one or multiple attributes and all those attributes for all computers in the network are displayed in the right pane. For every display pane, the net-bios name (system) and the IP address are always shown so that if an inconsistency is found, it is easy to know the name and location of the specific PC.



System	IP Address	OS Name	Version
CARTO	129.237.179.18	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2	129.237.178.171	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-54	129.237.178.168	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-53	129.237.178.167	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-52	129.237.178.166	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-51	129.237.178.165	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-46	129.237.178.164	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-45	129.237.178.163	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-44	129.237.178.162	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-43	129.237.178.161	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-42	129.237.178.160	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-41	129.237.178.159	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-36	129.237.178.158	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-35	129.237.178.157	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-34	129.237.178.156	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-33	129.237.178.155	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-32	129.237.178.154	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-31	129.237.178.153	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-26	129.237.178.152	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-25	129.237.178.151	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-24	129.237.178.150	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-23	129.237.178.149	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-22	129.237.178.148	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-21	129.237.178.147	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-14	129.237.178.144	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-13	129.237.178.143	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600
LTCL2-12	129.237.178.142	Microsoft Windows XP Professional	5.1.2600 Service Pack 2 Build 2600

Figure 8. GUI view of system summary

The application tab shows all software found on any machine. Selecting a specific software, shows the publisher, version and which PCs it is installed on. When you mouse over any column in the right pane, it also gives you a total count.



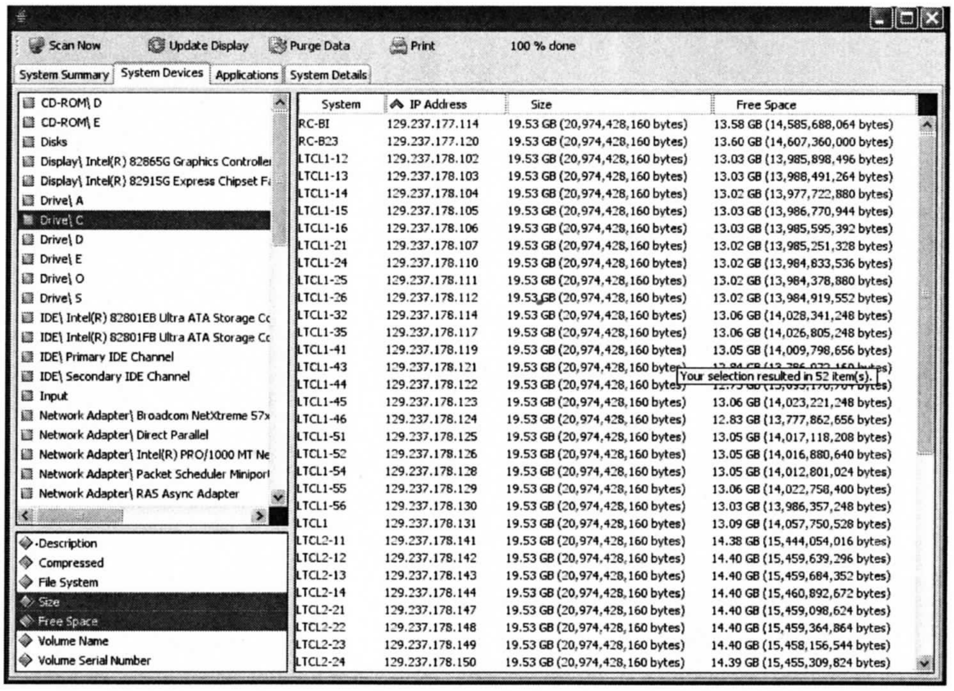
- Adobe Acrobat 7.0.1 and Reader 7.0
- Adobe Acrobat 7.0.2 and Reader 7.0
- Adobe Acrobat 7.0.3 and Reader 7.0
- Adobe Reader 7.0
- Adobe Reader 7.0.5
- Alt-Tab Task Switcher Powertoy for W
- Audia
- Broadcom Gigabit Integrated Controll
- CCleaner (remove only)
- CSPrintClient
- Calculator Powertoy for Windows XP
- CmdHere Powertoy For Windows XP
- Crystal Ball 7
- Dell ResourceCD
- Easy CD Creator 5 Basic
- EndNote 8.0.2
- GEO
- GMAT POWERPREP
- GRE POWERPREP
- HEC-RAS 3.1.3
- Hotfix for Windows Media Format SDK
- Hotfix for Windows XP (KB896344)
- ISI ResearchSoft - Export Helper
- Inspiration 7

System	IP Address	Application	Version	Publisher	Comments
LTCL1-12	129.237.178.102	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-13	129.237.178.103	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-14	129.237.178.104	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-15	129.237.178.105	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-16	129.237.178.106	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-21	129.237.178.107	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-24	129.237.178.110	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-25	129.237.178.111	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-26	129.237.178.112	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-32	129.237.178.114	[illegible]	[illegible]	Adobe Systems In...	
LTCL1-35	129.237.178.117	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-41	129.237.178.119	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-43	129.237.178.121	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-44	129.237.178.122	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-45	129.237.178.123	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-46	129.237.178.124	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-51	129.237.178.125	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-52	129.237.178.126	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-54	129.237.178.128	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-55	129.237.178.129	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1-56	129.237.178.130	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL1	129.237.178.131	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL2-31	129.237.178.153	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL2-32	129.237.178.154	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL2-33	129.237.178.155	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL2-34	129.237.178.156	Adobe Reader 7.0	7.0.0	Adobe Systems In...	
LTCL2-35	129.237.178.157	Adobe Reader 7.0	7.0.0	Adobe Systems In...	

Your selection resulted in 39 item(s).

Figure 9. GUI view of Applications

Selecting system device gives you three panes, one for the device, one for the device attributes, and the right pane for the resulting data.



- CD-ROM\ D
- CD-ROM\ E
- Disks
- Display\ Intel(R) 82865G Graphics Controller
- Display\ Intel(R) 82915G Express Chipset Fa
- Drive\ A
- Drive\ C
- Drive\ D
- Drive\ E
- Drive\ O
- Drive\ S
- IDE\ Intel(R) 82801EB Ultra ATA Storage Cc
- IDE\ Intel(R) 82801FB Ultra ATA Storage Cc
- IDE\ Primary IDE Channel
- IDE\ Secondary IDE Channel
- Input
- Network Adapter\ Broadcom NetXtreme 57x
- Network Adapter\ Direct Parallel
- Network Adapter\ Intel(R) PRO/1000 MT Ne
- Network Adapter\ Packet Scheduler Miniport
- Network Adapter\ RAS Async Adapter

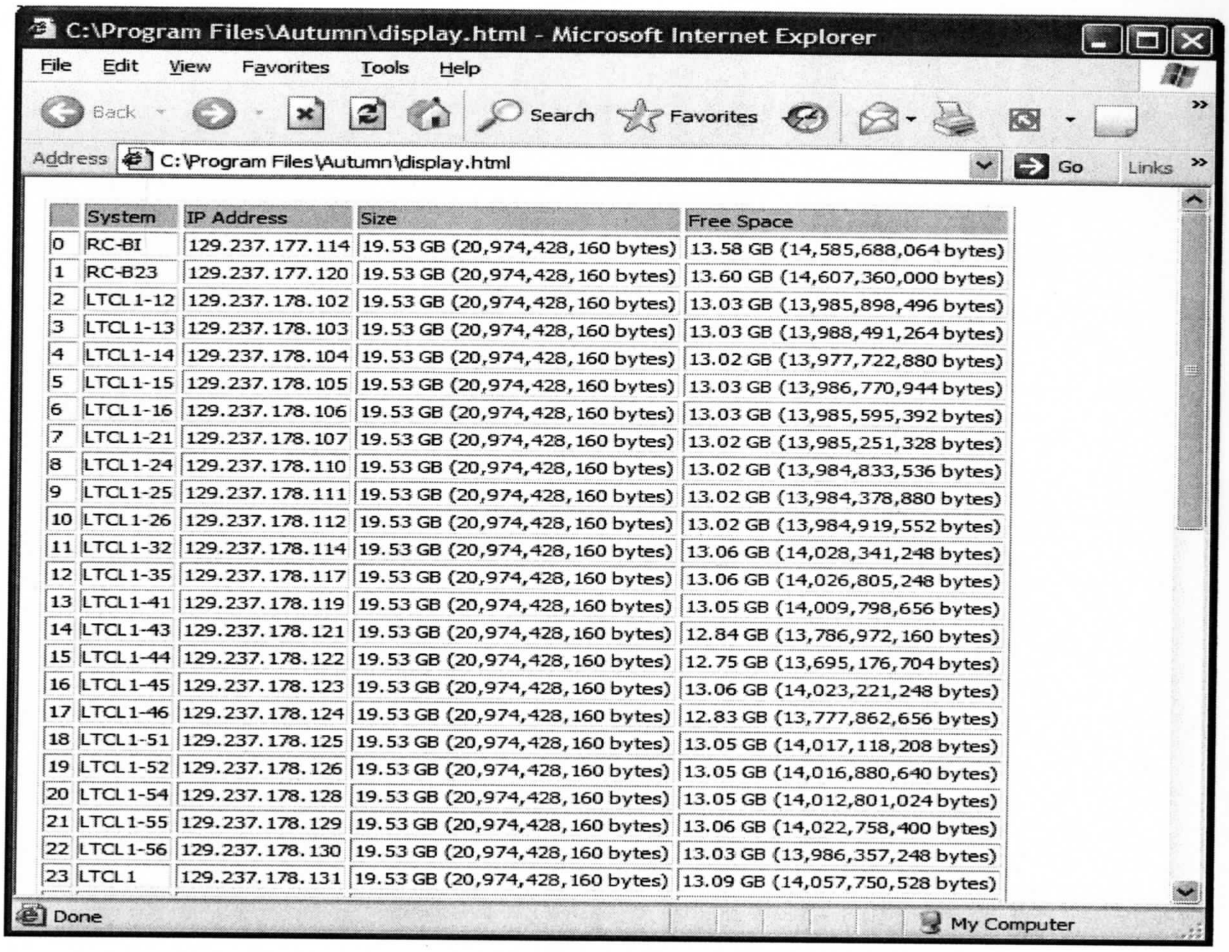
- Description
- Compressed
- File System
- Size
- Free Space
- Volume Name
- Volume Serial Number

System	IP Address	Size	Free Space
RC-BI	129.237.177.114	19.53 GB (20,974,428,160 bytes)	13.58 GB (14,585,688,064 bytes)
RC-B23	129.237.177.120	19.53 GB (20,974,428,160 bytes)	13.60 GB (14,607,360,000 bytes)
LTCL1-12	129.237.178.102	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,985,898,496 bytes)
LTCL1-13	129.237.178.103	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,988,491,264 bytes)
LTCL1-14	129.237.178.104	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,977,722,880 bytes)
LTCL1-15	129.237.178.105	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,986,770,944 bytes)
LTCL1-16	129.237.178.106	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,985,595,392 bytes)
LTCL1-21	129.237.178.107	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,985,251,328 bytes)
LTCL1-24	129.237.178.110	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,984,833,536 bytes)
LTCL1-25	129.237.178.111	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,984,378,880 bytes)
LTCL1-26	129.237.178.112	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,984,919,552 bytes)
LTCL1-32	129.237.178.114	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,028,341,248 bytes)
LTCL1-35	129.237.178.117	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,026,805,248 bytes)
LTCL1-41	129.237.178.119	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,009,798,656 bytes)
LTCL1-43	129.237.178.121	19.53 GB (20,974,428,160 bytes)	[illegible]
LTCL1-44	129.237.178.122	19.53 GB (20,974,428,160 bytes)	[illegible]
LTCL1-45	129.237.178.123	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,023,221,248 bytes)
LTCL1-46	129.237.178.124	19.53 GB (20,974,428,160 bytes)	12.83 GB (13,777,862,656 bytes)
LTCL1-51	129.237.178.125	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,017,118,208 bytes)
LTCL1-52	129.237.178.126	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,016,880,640 bytes)
LTCL1-54	129.237.178.128	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,012,801,024 bytes)
LTCL1-55	129.237.178.129	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,022,758,400 bytes)
LTCL1-56	129.237.178.130	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,986,357,248 bytes)
LTCL1	129.237.178.131	19.53 GB (20,974,428,160 bytes)	13.09 GB (14,057,750,528 bytes)
LTCL2-11	129.237.178.141	19.53 GB (20,974,428,160 bytes)	14.38 GB (15,444,054,016 bytes)
LTCL2-12	129.237.178.142	19.53 GB (20,974,428,160 bytes)	14.40 GB (15,459,639,296 bytes)
LTCL2-13	129.237.178.143	19.53 GB (20,974,428,160 bytes)	14.40 GB (15,459,684,352 bytes)
LTCL2-14	129.237.178.144	19.53 GB (20,974,428,160 bytes)	14.40 GB (15,460,892,672 bytes)
LTCL2-21	129.237.178.147	19.53 GB (20,974,428,160 bytes)	14.40 GB (15,459,098,624 bytes)
LTCL2-22	129.237.178.148	19.53 GB (20,974,428,160 bytes)	14.40 GB (15,459,364,864 bytes)
LTCL2-23	129.237.178.149	19.53 GB (20,974,428,160 bytes)	14.40 GB (15,458,156,544 bytes)
LTCL2-24	129.237.178.150	19.53 GB (20,974,428,160 bytes)	14.39 GB (15,455,309,824 bytes)

Your selection resulted in 52 item(s).

Figure 10 – GUI view of System Devices

Finally, the print button takes the results from the right pane, puts it in a table with borders and opens up in a webpage for printing from the browser.


C:\Program Files\Autumn\display.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
C:\Program Files\Autumn\display.html
Go
Links
Done
My Computer

	System	IP Address	Size	Free Space
0	RC-BI	129.237.177.114	19.53 GB (20,974,428,160 bytes)	13.58 GB (14,585,688,064 bytes)
1	RC-B23	129.237.177.120	19.53 GB (20,974,428,160 bytes)	13.60 GB (14,607,360,000 bytes)
2	LTCL1-12	129.237.178.102	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,985,898,496 bytes)
3	LTCL1-13	129.237.178.103	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,988,491,264 bytes)
4	LTCL1-14	129.237.178.104	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,977,722,880 bytes)
5	LTCL1-15	129.237.178.105	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,986,770,944 bytes)
6	LTCL1-16	129.237.178.106	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,985,595,392 bytes)
7	LTCL1-21	129.237.178.107	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,985,251,328 bytes)
8	LTCL1-24	129.237.178.110	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,984,833,536 bytes)
9	LTCL1-25	129.237.178.111	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,984,378,880 bytes)
10	LTCL1-26	129.237.178.112	19.53 GB (20,974,428,160 bytes)	13.02 GB (13,984,919,552 bytes)
11	LTCL1-32	129.237.178.114	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,028,341,248 bytes)
12	LTCL1-35	129.237.178.117	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,026,805,248 bytes)
13	LTCL1-41	129.237.178.119	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,009,798,656 bytes)
14	LTCL1-43	129.237.178.121	19.53 GB (20,974,428,160 bytes)	12.84 GB (13,786,972,160 bytes)
15	LTCL1-44	129.237.178.122	19.53 GB (20,974,428,160 bytes)	12.75 GB (13,695,176,704 bytes)
16	LTCL1-45	129.237.178.123	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,023,221,248 bytes)
17	LTCL1-46	129.237.178.124	19.53 GB (20,974,428,160 bytes)	12.83 GB (13,777,862,656 bytes)
18	LTCL1-51	129.237.178.125	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,017,118,208 bytes)
19	LTCL1-52	129.237.178.126	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,016,880,640 bytes)
20	LTCL1-54	129.237.178.128	19.53 GB (20,974,428,160 bytes)	13.05 GB (14,012,801,024 bytes)
21	LTCL1-55	129.237.178.129	19.53 GB (20,974,428,160 bytes)	13.06 GB (14,022,758,400 bytes)
22	LTCL1-56	129.237.178.130	19.53 GB (20,974,428,160 bytes)	13.03 GB (13,986,357,248 bytes)
23	LTCL1	129.237.178.131	19.53 GB (20,974,428,160 bytes)	13.09 GB (14,057,750,528 bytes)

Figure 11 – view of Print function results

Performance

Performance was a major issue so much time was spent in development trying to limit data and processes to a minimum. After finding the application information in the registry, we had to determine how to only export the specific registry key we needed. The same was true for the system and hardware information from winmsd. In addition, we discovered that winmsd actually uses msinfo32.exe and that calling that process directly speeded up the data acquisition significantly – about 20 seconds per PC. The registry information was fairly clean and saved to a text file, but the winmsd data dump was very messy as a text file. We discovered that winmsd used XML tags so we used these tags to reduce the amount of time the Asset object spent cleaning up this section of data. Because we limited the data collection, the final summary object that gets transferred across the network is, on average, only 85k.

We also ended processes as soon as they completed their respective tasks and made as many variables as possible “private”. The only processes that continue to run are the client and server process. The autumn object can read existing saved files without even running the server. The server process does not start until the “Scan Now” button is clicked, activating the network call for new data. Each time this occurs, the data files are overwritten, saving hard drive space.

Delays

There was some delay in the schedule due to additional requests. One request was to also obtain office updates. This posed two problems. The first was that the data is stored in a different registry key. While it was possible to obtain it, it would another group of assets to the overall package. The most significant problem however was that the registry stores office updates using KB numbers. There is no “human interpretation” of what these are. If the KB information was displayed in the GUI, the network administrator would have to know what they meant and whether that was the last update available for “Word”, for example. A possible solution would be to maintain a database of Office updates and have the program check against that, but then that database would have to be maintained manually. Some commercial products interface with Microsoft directly but this was beyond the scope of this project. After spending days investigating this additional request, it was decided not to include it. Instead, the IT department built a Windows update server which will handle this.

Another delay was the additional request for a systray icon. This piece is still being worked on. Opensource code was found on www.limewire.com but pieces of it are embedded in many folders and it will take weeks to pull out what is needed. A copyrighted solution was also found which can be used as long as our product remains non-commercial and contains the copyright information in the code. Because the client side does not have a GUI, the only way to see if it is running is by using a systray icon. There was also a request for the system details tab. This tab, seen in the GUI, is not yet functional. It will display all the information for a single PC. These two items will be implemented at a later date.

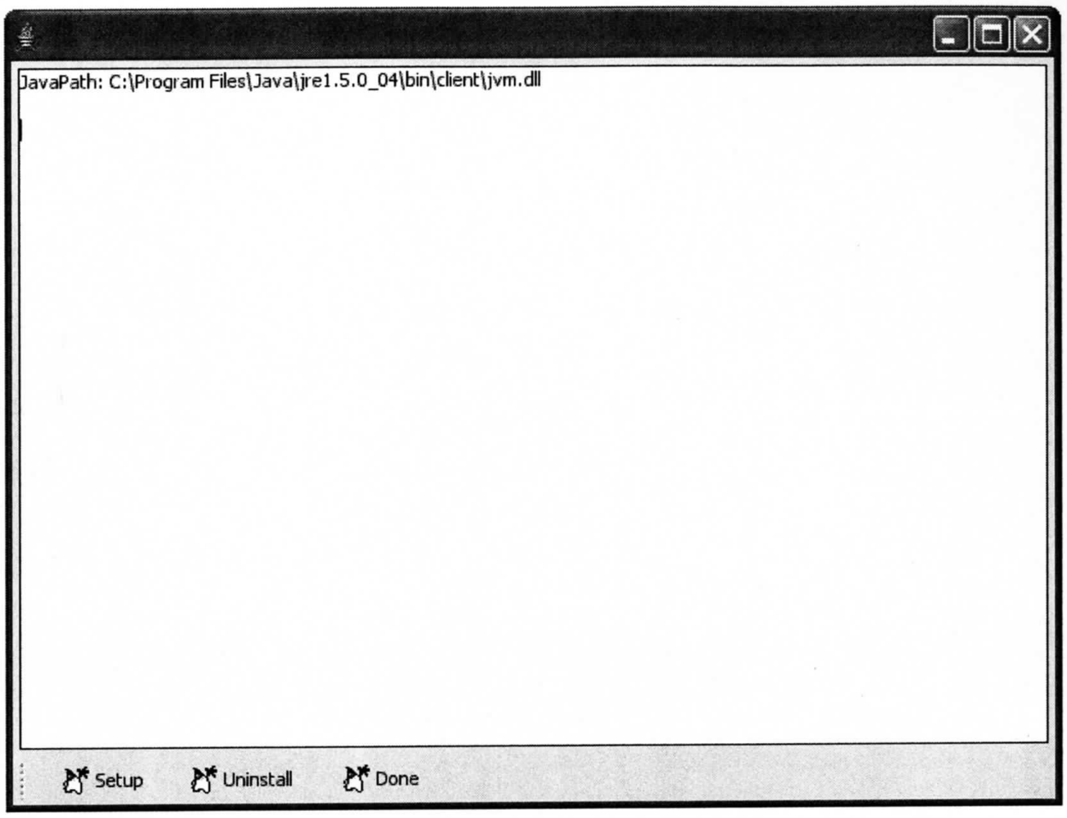
Testing

The project has been tested on up to sixty-nine PCs and responds well with low network load. It takes about 20 seconds to broadcast the command for files and to receive the files back to the server for 69 machines. In addition, the data files were copied until we had 200 data files in order to test the GUI's ability to read and manipulate the views. Originally, when the program was written, it read the data directly from memory and with 60 files no problem was found. When we had 200 files, we discovered that the RAM was overwhelmed and froze the program. The entire section on data storage and manipulation had to be rewritten.

The program initially de-serializes each data file into a Summary object. In order to accommodate more than 200 ~250 systems, Summary objects are no longer stored in memory. As each Summary is processed, each asset is serialized to a data file in a subdirectory containing assets of the same name. This results in a longer initial data loading time – it takes about one minute to read 200 files. Since we don't want the GUI to freeze during this process, the process is now executed in a separate thread, and the GUI is updated with the % done progress indicator. The percentage is calculated from the number of deserialized data files vs. the total number of Summary data files. Once the initial data load is complete, the loading process sets the GUI's applications list, system summary list, and devices list. Clicking on an item in a primary list kicks off an action. For example, clicking on an application causes the program to deserialize all of the serialized assets in the folder of the same name. This results in satisfactory performance in generating tables, even with 512 and 1024 entries for that asset.

In the original scope, we thought a database would be needed. However after discussing this piece during status reports, the project sponsor decided he would never use archived data. The purpose of the program is to identify where software is installed right now, which computers needs updates today, and how many licenses are in use now. The data files are stored on the server and can be read without a new scan. Clicking the scan button pulls real time data so this requirement was removed from the scope.

Four additional items were added to the scope and have been completed. The first was the print function. The next was a direct result of rewriting the data files. Because the scan now takes some time to do the initial read of the files, a progress bar was added so the user will know when the program is ready to be manipulated. The third addition was to run the program as a windows service so that it starts on startup and shows up in the task manager. The final addition was to create another application for the install.



```
JavaPath: C:\Program Files\Java\jre1.5.0_04\bin\client\jvm.dll
```

Figure 12. GUI view of install program

The project sponsor is satisfied with the additions and the delay in implementation is not a problem. A summary of the timeline and completion of tasks is shown below:

Table 3. Milestone chart

Milestone	**Date**	**Status**	**Responsible**	**Comments**
Initiating				
Determine Project Manager	2/15/05	Completed	Diana	
Business Case Completed	4/22/05	Completed	Diana	
Planning				
Scope Statement Completed	4/28/05	Completed	Diana	

WBS Completed	4/28/05	Completed	Diana	
Executing				
Evaluate current system	8/22/05	Completed	Diana	
Define requirements	8/26/05	Completed	Diana	
Database creation	NA	NA	Diana	Removed from scope
Batch process created	9/18/05	Completed	Adam	Revised to Java network communication
Program development	10/28/05	Completed	Adam and Diana	1 item remaining (addition to scope)
Testing and Debugging	11/18/05	Completed	Adam and Diana	Ongoing throughout development
GUI design	11/22/05	Completed	Adam and Diana	
End user testing	12/15/05		Network administrator	
Project Finalization	12/20/05		Diana	

Security

The autumn.jar file will run from a network folder which only IT administrators have access to. This file is the only way to open the GUI and run the scan and display functions. The client runs continuously in the background on each PC but cannot be opened as there is no GUI for this piece. The computer the files get stored on is only a data storage server, not running typical vulnerabilities like IIS or as an SQL server. During installation, the IP address of the server and the port number to listen to is hard-coded. The client will only dump its information when it receives a message from this specific port and IP and will deliver the summary file only back to this specific IP.

CHAPTER 5

CONCLUSIONS

I believe we have achieved our objective. We are very near the completion, only lacking the systray icon and the system details tab. The program gathers the data the IT department needs to more efficiently manage PCs on the LAN. This product in combination with the windows update server and the already existing ability to push software remotely means that PCs can be maintained from an office desk rather than by going to each office, classroom and lab. It has an easy-to-use interface that allows the user to look at one machine, all machines and all variables, or all machines and selected variables. The GUI is so simple that a user manual probably won't be needed. Installation directions will be provided as well as commented code when the user documentation has been completed after final testing which will occur as soon as the program is running on every PC in the LAN. Given the response time of reading the data files, we may be near a maximum limit for this configuration. In a few years, when the campus expands again, adding a third and fourth building, the issue of including a SQL database may need to be revisited as a more robust system for large record numbers.

From a project manager perspective I learned a lot about running a project. I think working in-house makes it much more difficult to control the scope. When your boss says to add something to the scope, you follow the directive where possible unless faced with a request like the Office updates where the resulting data would not be useful. Also, in many projects, the team is dedicated to working on the project to completion. In this environment, the project was put on hold anytime the daily operations and duties of the staff required immediate attention so the timeline was really a rough estimate. Since time was not critical and cost was only in terms of other projects deferred, adding to the scope to enhance the end product was beneficial. When an external client adds something to the scope, you can discuss time and cost constraints and rewrite the contract if needed. Finally, as a participant in the programming, I learned a great deal about where and how PCs store information and about native java classes. Our plan is to install the client program on all computers by mid-December and do final testing, then continue working on the final two items, which will be deployed at a later date.

REFERENCES

David, J.S., Schuff, D., & St. Louis, R.(2002, January).Managing Your IT Total Cost of Ownership. *Communications of the ACM 45*(1), 101-106.

Dennis, A., Wixom, B.H., & Tegarden, D. (2002). Systems Analysis & Design: An Object-Oriented Approach with UML. *John Wiley & Sons, Inc.*

Forbath, T., Kalaher, P., & Schenof, J. (2004). New Insights on PC Management: Benefits of Controlled PC Hardware Diversity. *Wipro NerveWire.*

Magee, M. (2004, November 29). Intel offers hope for corporate PC management. http://www.theinquirer.net/?article=19937

NetSupport Manager. http://www.netsupportinc.com/nsm/netsupport_manager_features.htm (2005).

Remote-Scope. http://www.millennium-solutions.co.uk/network-management-tool/ (2005).

Schwalbe, K. (2004) Information Technology Project Management, 3rd Ed. *Thompson Course Technology.*

APPENDICES

APPENDIX A: USERS' MANUAL

Welcome to your system management tool.

1. Open install.jar and click on the setup button. When the display shows the install is completed, close this window by clicking the Done button. You can also uninstall the SMT from here.
2. Open the autumn.jar file found in the IT network folder\Autumn\dist
3. For the first run, click on the "Scan Now" button.
4. Click on the "Update Display" button – you will see a progress bar to the right of the Print button. When it reads 100%, you can begin viewing files.
5. To view files, click on a tab. Then select the items from the left panel that you want to view. The results will show in the right panel.
6. For future views, you can choose to view existing files by opening autumn.jar and clicking on Update Display, or you can get new current data by starting with the Scan Now button.
7. Close window when finished.

APPENDIX B: SYSTEM TECHNICAL DOCUMENTATION

N:\..... \src\autumn\Acquire.java

```
//  Acquire.java
//  Created on November 10, 2005, 7:49 PM

package autumn;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.Vector;

public class Acquire
{

    private File save;
    private Vector applications;
    private Vector devices;

    public Acquire()
    {
        this.setup();
        this.applications = new Vector();
        this.devices = new Vector();
        this.processApps();
        this.processDevices();
        this.modifyNetworkAssets();
        this.modifyDevices();
        System.gc();
    }

    // Create the temporary output file
    private void setup()
    {
        try
        {
            String saved = this.toString();
            String[] parts = saved.split("@");
            saved = parts[0] + parts[1];
            File parent = new File("C:\\Program Files\\Autumn");
            parent.mkdirs();

            save = new File("C:\\Program Files\\Autumn\\" + saved);
        }
        catch (Exception e)
        {   System.out.println("Acquire.setup\n" + e);  }
    }


    private void processDevices()
    {
        try
        {
            String command =
"c:\\windows\\system32\\dllcache\\msinfo32.exe /categories +Components
   /nfo ";
            command = command + "\"" + save.toString() + "\"";
            Runtime runTime = Runtime.getRuntime();
            Process process = runTime.exec(command);

            int exitVal = process.waitFor();
            System.out.println("     ExitValue: " + exitVal + "\n\n");

            save = new File(save.toString() + ".nfo");
            FileInputStream inp = new FileInputStream(save);
            BufferedReader input = new BufferedReader(new
        InputStreamReader(inp));

            String line = null;
            String dataItem = "Device\t";

            while ((line = input.readLine()) != null)
            {
                line = cleanLine(line);

                if (line.startsWith("<Category"))
                    dataItem = dataItem + line.substring(16,
   line.length() - 2) + "\n";

                else if (line.startsWith("<Item><![CDATA["))
                    dataItem = dataItem + line.substring(15,
   line.length() - 10) + "\t";

                else if (line.startsWith("<Value><![CDATA["))
                    dataItem = dataItem + line.substring(16,
   line.length() - 11) + "\n";

                else if (line.startsWith("</Category"))
                {
                    splitSection(dataItem);
                    dataItem = "Device\t";
                }
            }
            input.close();
            inp.close();
            save.delete();
        }
        catch (Exception e)
        {   System.out.println("Acquire.processDevices\n" + e); }
    }


    private void splitSection(String input)
    {
        if (input.startsWith("Device\tNetwork"))
            createAsset(input, "Network Adapter\\", "Name");

        else if (input.startsWith("Device\tStorage"))
            createAsset(input, "Drive\\", "Drive");

        else if (input.startsWith("Device\tIDE"))
            createAsset(input, "IDE\\", "Name");

        else if (input.startsWith("Device\tWinSock"))
            createAsset(input, "WinSock\\", "File");

        else if (input.startsWith("Device\tProtocol"))
            createAsset(input, "Protocol\\", "Name");

        else if (input.startsWith("Device\tCD-ROM"))
            createAsset(input, "CD-ROM\\", "Drive");

        else if (input.startsWith("Device\tDisplay"))
            createAsset(input, "Display\\", "Name");

        else if (input.startsWith("Device\tSound Device"))
            createAsset(input, "Sound Device\\", "Name");

        else
        {
            Asset asset = new Asset(input);
            if (asset.dataSize() > 1)
                devices.add(asset);
        }

    }


    private void createAsset(String input, String subCat, String identifier)
    {
        Asset asset;
        String section = "";
        String[] split = input.split("\n");      //  break into lines

        for (int i = 0; i < split.length; i++)       //  loop through all lines
        {
            if (split[i].startsWith(identifier))     //  if a line starts with....
            {
                asset = new Asset(section);
                if (asset.dataSize() > 1)
                    devices.add(asset);

                String[] line = split[i].split("\t");
                if (line.length > 1)
                {
                    line[0] = "Device";
                    line[1] = subCat + " " + line[1];
                    split[i] = line[0] + "\t" + line[1];
                }
                section = split[i] + "\n";
            }
            else
                section = section + split[i] + "\n";
        }
        asset = new Asset(section);
        if (asset.dataSize() > 1)
            devices.add(asset);
    }

    private String cleanLine(String input)
    {
        String cleaned = "";
        for (int i = 0; i < input.length(); i++)
        {
            char current = input.charAt(i);
            cleaned = cleaned;

            if ((int) current > 0)
                cleaned = cleaned + current;
        }
        return cleaned;
    }


    private void processApps()
    {
        try
        {
            String key =
"HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall";
            String command = "cmd.exe /C reg query " + key + " /s > " +
"\"" + save.toString() + "\"";
            Runtime runTime = Runtime.getRuntime();
            Process process = runTime.exec(command);

            int exitVal = process.waitFor();
            System.out.println("    ExitValue: " + exitVal + "\n\n");

            FileInputStream inp = new FileInputStream(save);
            BufferedReader input = new BufferedReader(new
InputStreamReader(inp));

            String line = null;
            String section = "";
            while ((line = input.readLine()) != null)
            {
                if (line.startsWith("HKEY"))
                {
                    Asset asset = new Asset(section);
                    if (asset.query("Application").equals("") == false)
                        applications.add(asset);
                    section = "";
                }
                else
                {
                    String[] parts = line.split("\t");
                    if (parts.length > 2)
                    {
                        parts[0] = parts[0].substring(4);
                        if (parts[0].equalsIgnoreCase("DisplayName"))
                            parts[0] = "Application";
                        section = section + parts[0] + "\t" + parts[2]
+
            "\n";
                    }
                }
            }
            input.close();
            inp.close();
            save.delete();
        }
        catch (Exception e)
        {   System.out.println("Aqcuire.processApps\n" + e);     }
    }

    private void modifyNetworkAssets()
    {
        for (int i = 0; i < devices.size(); i++)
        {
            Asset asset = (Asset) devices.elementAt(i);
            if (asset.query("Device").startsWith("Network"))
                asset.set("Device", "Network Adapter\\ " +
    asset.query("Product Type"));
        }
    }


    private void modifyDevices()
    {
        for (int i = 0; i < devices.size(); i++)
        {
            Asset asset = (Asset) devices.elementAt(i);
            Vector data = asset.data();
            for (int k = 0; k < data.size(); k++)
            {
                String[] array = (String[]) data.elementAt(k);
                if (array[0].equals("Device"))
                {
                    String[] parts = array[1].split(":");
                    array[1] = "";
                    for (int m = 0; m < parts.length; m++)
                        array[1] = array[1] + parts[m];
                }
            }
        }
    }

    //  method called by Summary
    public Vector applications()
    {   return applications;    }

    public Vector devices()
    {   return devices;     }
}

```

N:\.....\src\autumn\Asset.java

```java
// Asset.java
// Created on November 3, 2005, 2:04 PM

package autumn;

import java.io.Serializable;
import java.net.InetAddress;
import java.util.Arrays;
import java.util.Vector;

public class Asset implements Serializable
{
    private InetAddress inetAddress;
    private Vector data;

    public Asset()
    {    }

    public Asset(String input)
    {
        String[] lines = input.split("\n");
        String[] parts = null;
        data = new Vector();

        int iteration = 0;
        while (iteration < lines.length)
        {
            parts = lines[iteration].split("\t");
            iteration = iteration + 1;
            if (parts.length > 1)
            {
                data.add(parts);
            }
        }

        try
        {   this.inetAddress = InetAddress.getLocalHost();  }
        catch (Exception e)
        {   System.out.println(e);  }
    }


    public void print()
    {
        int iteration = 0;
        while (iteration < data.size())
        {
            String[] current = (String[]) data.elementAt(iteration);
            System.out.println(current[0] + "\t\t\t" + current[1]);

            iteration = iteration + 1;
        }
    }


    public String query(String input)
    {
        int iteration = 0;
        while (iteration < data.size())
        {
            String[] current = (String[]) data.elementAt(iteration);
            iteration = iteration + 1;
            if (current[0].equalsIgnoreCase(input))
                return current[1];
        }
        return "";
    }


    public void set(String variable, String value)
    {
        int iteration = 0;
        while (iteration < data.size())
        {
            String[] current = (String[]) data.elementAt(iteration);
            iteration = iteration + 1;
            if (current[0].equalsIgnoreCase(variable))
                current[1] = value;
        }
    }


    public int dataSize()
    {   return data.size(); }


    public Vector data()
    {   return this.data;   }


    public InetAddress inetAddress()
    {   return inetAddress; }

}

```

N:\IT\Adam\Java\Autumn\src\autumn\Autumn.java

```
/*
 * Autumn.java
 *
 * Created on November 15, 2005, 10:38 AM
 */

package autumn;

import java.awt.Color;
import java.awt.Component;
import java.awt.Container;
import java.util.Collections;
import java.util.Vector;
import javax.swing.JButton;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JViewport;
import javax.swing.JTable;
import javax.swing.UIManager;


public class Autumn extends javax.swing.JFrame
{

    private Client client;
    private Server server;
    private DataStore dataStore;
    private Vector components;

    /** Creates new form Autumn */
    public Autumn()
    {
        initComponents();
        initMyComponents();

        components = new Vector();
    }

    private void initMyComponents()
    {
        Color dark = new Color(50, 50, 50);
        Color white = new Color(255, 255, 255);
        Color selectColor = new Color(100, 100, 100);

        Component[] components = subComponents(this.getComponent(0));
        for (int i = 0; i < components.length; i++)
        {
            if (components[i] instanceof JViewport)
                ((JViewport) components[i]).setBackground(white);

            if (components[i] instanceof JList)
                ((JList)
components[i]).setSelectionBackground(selectColor);

            if (components[i] instanceof JButton)
                this.modifyButton((JButton) components[i]);

            if (components[i] instanceof JScrollPane)
                ((JScrollPane) components[i]).setBackground(dark);
        }

        try
        {   detailsEditorPane.setText("Not Yet Implemented\n\nThis
feature is beyond initial project scope.");  }
        catch (Exception e)
        {   System.out.println(e);  }

    }


    private Component[] subComponents(Component input)
    {
        Vector sub = new Vector();
        sub.add(input);
        Component[] subArray = ((Container) input).getComponents();

        if (subArray.length < 1)     //  Component has no subComponents
            sub.add(input);

        else                         //  Component has subComponents
        {
            for (int i = 0; i < subArray.length; i++)
            {
                Component[] subSet = subComponents(subArray[i]);
                for (int k = 0; k < subSet.length; k++)
                    sub.add(subSet[k]);
            }
        }

        Component[] output = new Component[sub.size()];
        for (int i = 0; i < sub.size(); i++)
            output[i] = (Component) sub.elementAt(i);

        return output;
    }



    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents()
    {
        autumnToolBar = new javax.swing.JToolBar();
        serverButton = new javax.swing.JButton();
        updateDisplayButton = new javax.swing.JButton();
        purgeDataButton = new javax.swing.JButton();
        printButton = new javax.swing.JButton();
        statusButton = new javax.swing.JButton();
        appTabbedPane = new javax.swing.JTabbedPane();
        summaryPane = new javax.swing.JPanel();
        summarySplitPane = new javax.swing.JSplitPane();
        summaryScrollPane0 = new javax.swing.JScrollPane();
        summaryList = new javax.swing.JList();
        summaryScrollPane1 = new javax.swing.JScrollPane();
        devicesPane = new javax.swing.JPanel();
        devicesSplitPane = new javax.swing.JSplitPane();
        devicesScrollPane0 = new javax.swing.JScrollPane();
        devicesSelectSplitPane = new javax.swing.JSplitPane();
        devicesSelectScrollPane0 = new javax.swing.JScrollPane();
        devicesDeviceList = new javax.swing.JList();
        devicesSelectScrollPane1 = new javax.swing.JScrollPane();
        devicesVariableList = new javax.swing.JList();
        devicesScrollPane1 = new javax.swing.JScrollPane();
        applicationsPane = new javax.swing.JPanel();
        applicationsSplitPane = new javax.swing.JSplitPane();
        applicationsScrollPane0 = new javax.swing.JScrollPane();
        applicationsList = new javax.swing.JList();
        applicationsScrollPane1 = new javax.swing.JScrollPane();
        detailsPane = new javax.swing.JPanel();
        detailsSplitPane = new javax.swing.JSplitPane();
        detailsScrollPane0 = new javax.swing.JScrollPane();
        detailsList = new javax.swing.JList();
        detailsScrollPane1 = new javax.swing.JScrollPane();
        detailsEditorPane = new javax.swing.JEditorPane();


    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
;
        setFont(new java.awt.Font("Tahoma", 0, 11));
        setName("autumn");
        autumnToolBar.setRollover(true);
        serverButton.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/autumn/~Server.png")
));
        serverButton.setText("Scan Now");
        serverButton.setToolTipText("Create a Server process and query
    active Client processes.");

    serverButton.setHorizontalAlignment(javax.swing.SwingConstants.LEFT
);
        serverButton.setMaximumSize(new java.awt.Dimension(110, 25));
        serverButton.setMinimumSize(new java.awt.Dimension(110, 25));
        serverButton.setPreferredSize(new java.awt.Dimension(110, 25));
        serverButton.addActionListener(new
    java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                serverButtonActionPerformed(evt);
            }
        });

        autumnToolBar.add(serverButton);

        updateDisplayButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/autumn/~Update.gif")));
        updateDisplayButton.setText("Update Display");
        updateDisplayButton.setToolTipText("Update display information from stored data.");
        updateDisplayButton.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        updateDisplayButton.setMaximumSize(new java.awt.Dimension(110, 25));
        updateDisplayButton.setMinimumSize(new java.awt.Dimension(110, 25));
        updateDisplayButton.setPreferredSize(new java.awt.Dimension(110, 25));
        updateDisplayButton.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                updateDisplayButtonActionPerformed(evt);
            }
        });

        autumnToolBar.add(updateDisplayButton);

        purgeDataButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/autumn/~Clean.gif")));
        purgeDataButton.setText("Purge Data");
        purgeDataButton.setToolTipText("Purge all stored data. [Currently Disabled]");
        purgeDataButton.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        purgeDataButton.setMaximumSize(new java.awt.Dimension(110, 25));
        purgeDataButton.setMinimumSize(new java.awt.Dimension(110, 25));
        purgeDataButton.setPreferredSize(new java.awt.Dimension(110, 25));
        purgeDataButton.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                purgeDataButtonActionPerformed(evt);
            }
        });

        autumnToolBar.add(purgeDataButton);

        printButton.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/autumn/~Print.png"))
);
        printButton.setText("Print");
        printButton.setToolTipText("Print the displayed table.");

    printButton.setHorizontalAlignment(javax.swing.SwingConstants.LEFT)
;
        printButton.setMaximumSize(new java.awt.Dimension(110, 25));
        printButton.setMinimumSize(new java.awt.Dimension(110, 25));
        printButton.setPreferredSize(new java.awt.Dimension(110, 25));
        printButton.addActionListener(new
java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                printButtonActionPerformed(evt);
            }
        });

        autumnToolBar.add(printButton);


    statusButton.setHorizontalAlignment(javax.swing.SwingConstants.LEFT
);
        statusButton.setMaximumSize(new java.awt.Dimension(110, 25));
        statusButton.setMinimumSize(new java.awt.Dimension(110, 25));
        statusButton.setName("statusButton");
        statusButton.setPreferredSize(new java.awt.Dimension(110, 25));
        autumnToolBar.add(statusButton);

        getContentPane().add(autumnToolBar,
java.awt.BorderLayout.NORTH);

        summaryPane.setLayout(new java.awt.BorderLayout());

        summaryPane.setBorder(new javax.swing.border.EmptyBorder(new
    java.awt.Insets(4, 4, 4, 4)));
        summaryPane.setName("summaryPane");
        summarySplitPane.setBorder(null);
        summarySplitPane.setDividerLocation(220);
        summarySplitPane.setDividerSize(6);
        summarySplitPane.setMaximumSize(new java.awt.Dimension(1600,
    1600));
        summarySplitPane.setMinimumSize(new java.awt.Dimension(0, 0));
        summarySplitPane.setPreferredSize(new java.awt.Dimension(0,
0));
        summaryScrollPane0.setBorder(new
    javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
        summaryList.setToolTipText("Select Variables");
        summaryList.addListSelectionListener(new
    javax.swing.event.ListSelectionListener()
        {
            public void
valueChanged(javax.swing.event.ListSelectionEvent
    evt)
```

```
                {
                    summaryListValueChanged(evt);
                }
            });

            summaryScrollPane0.setViewportView(summaryList);

            summarySplitPane.setLeftComponent(summaryScrollPane0);

            summaryScrollPane1.setBorder(new
        javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            summarySplitPane.setRightComponent(summaryScrollPane1);

            summaryPane.add(summarySplitPane,
java.awt.BorderLayout.CENTER);

            appTabbedPane.addTab("System Summary", summaryPane);

            devicesPane.setLayout(new java.awt.BorderLayout());

            devicesPane.setBorder(new javax.swing.border.EmptyBorder(new
        java.awt.Insets(4, 4, 4, 4)));
            devicesPane.setName("devicesPane");
            devicesPane.setPreferredSize(new java.awt.Dimension(900, 600));
            devicesPane.setRequestFocusEnabled(false);
            devicesSplitPane.setBorder(null);
            devicesSplitPane.setDividerLocation(220);
            devicesSplitPane.setDividerSize(6);
            devicesSplitPane.setMaximumSize(new java.awt.Dimension(1600,
        1600));
            devicesSplitPane.setMinimumSize(new java.awt.Dimension(0, 0));
            devicesScrollPane0.setBorder(null);
            devicesScrollPane0.setMaximumSize(new java.awt.Dimension(1600,
        1600));
            devicesScrollPane0.setMinimumSize(new java.awt.Dimension(0,
0));
            devicesScrollPane0.setPreferredSize(new java.awt.Dimension(200,
        200));
            devicesSelectSplitPane.setBorder(null);
            devicesSelectSplitPane.setDividerLocation(400);
            devicesSelectSplitPane.setDividerSize(6);

        devicesSelectSplitPane.setOrientation(javax.swing.JSplitPane.VERTIC
AL_
        SPLIT);
            devicesSelectSplitPane.setMaximumSize(new
        java.awt.Dimension(1600, 1600));
            devicesSelectSplitPane.setMinimumSize(new java.awt.Dimension(0,
        0));
            devicesSelectSplitPane.setPreferredSize(new
java.awt.Dimension(0,
        0));
            devicesSelectScrollPane0.setBorder(new
        javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            devicesSelectScrollPane0.setMaximumSize(new
        java.awt.Dimension(1600, 1600));
        devicesSelectScrollPane0.setMinimumSize(new java.awt.Dimension(0, 0));
        devicesSelectScrollPane0.setPreferredSize(new java.awt.Dimension(0, 0));
        devicesDeviceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        devicesDeviceList.setToolTipText("Select A Device");
        devicesDeviceList.setAutoscrolls(false);
        devicesDeviceList.setMaximumSize(new java.awt.Dimension(1600, 1600));
        devicesDeviceList.setVisibleRowCount(20);
        devicesDeviceList.addListSelectionListener(new javax.swing.event.ListSelectionListener()
        {
            public void valueChanged(javax.swing.event.ListSelectionEvent evt)
            {
                devicesDeviceListValueChanged(evt);
            }
        });

        devicesSelectScrollPane0.setViewportView(devicesDeviceList);

devicesSelectSplitPane.setTopComponent(devicesSelectScrollPane0);

        devicesSelectScrollPane1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
        devicesSelectScrollPane1.setMaximumSize(new java.awt.Dimension(1600, 1600));
        devicesSelectScrollPane1.setMinimumSize(new java.awt.Dimension(0, 0));
        devicesSelectScrollPane1.setPreferredSize(new java.awt.Dimension(0, 0));
        devicesVariableList.setToolTipText("Select Variables");
        devicesVariableList.setAutoscrolls(false);
        devicesVariableList.setMaximumSize(new java.awt.Dimension(1600, 1600));
        devicesVariableList.setVisibleRowCount(10);
        devicesVariableList.addListSelectionListener(new javax.swing.event.ListSelectionListener()
        {
            public void valueChanged(javax.swing.event.ListSelectionEvent evt)
            {
                devicesVariableListValueChanged(evt);
            }
        });

        devicesSelectScrollPane1.setViewportView(devicesVariableList);

        devicesSelectSplitPane.setBottomComponent(devicesSelectScrollPane1)
;

            devicesScrollPane0.setViewportView(devicesSelectSplitPane);

            devicesSplitPane.setLeftComponent(devicesScrollPane0);

            devicesScrollPane1.setBorder(new
        javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            devicesScrollPane1.setAutoscrolls(true);
            devicesScrollPane1.setMaximumSize(null);
            devicesScrollPane1.setMinimumSize(null);
            devicesScrollPane1.setPreferredSize(null);
            devicesSplitPane.setRightComponent(devicesScrollPane1);

            devicesPane.add(devicesSplitPane,
java.awt.BorderLayout.CENTER);

            appTabbedPane.addTab("System Devices", devicesPane);

            applicationsPane.setLayout(new java.awt.BorderLayout());

            applicationsPane.setBorder(new
javax.swing.border.EmptyBorder(new
        java.awt.Insets(4, 4, 4, 4)));
            applicationsPane.setMinimumSize(new java.awt.Dimension(0, 0));
            applicationsPane.setName("applicationsPane");
            applicationsPane.setPreferredSize(new java.awt.Dimension(0,
0));
            applicationsSplitPane.setBorder(null);
            applicationsSplitPane.setDividerLocation(220);
            applicationsSplitPane.setDividerSize(6);
            applicationsSplitPane.setMaximumSize(new
java.awt.Dimension(1600,
        1600));
            applicationsSplitPane.setMinimumSize(new java.awt.Dimension(0,
        0));
            applicationsSplitPane.setPreferredSize(new
java.awt.Dimension(0,
        0));
            applicationsScrollPane0.setBorder(new
        javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            applicationsScrollPane0.setPreferredSize(new
        java.awt.Dimension(240, 400));

        applicationsList.setSelectionMode(javax.swing.ListSelectionModel.SI
NGL
        E_SELECTION);
            applicationsList.setToolTipText("Select An Application");
            applicationsList.setName("applicationsList");
            applicationsList.addListSelectionListener(new
        javax.swing.event.ListSelectionListener()
            {
                public void
valueChanged(javax.swing.event.ListSelectionEvent
        evt)
```

```
                {
                    applicationsListValueChanged(evt);
                }
            });

            applicationsScrollPane0.setViewportView(applicationsList);

            applicationsSplitPane.setLeftComponent(applicationsScrollPane0);

            applicationsScrollPane1.setBackground(new java.awt.Color(255, 255, 255));
            applicationsScrollPane1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            applicationsSplitPane.setRightComponent(applicationsScrollPane1);

            applicationsPane.add(applicationsSplitPane, java.awt.BorderLayout.CENTER);

            appTabbedPane.addTab("Applications", applicationsPane);

            detailsPane.setLayout(new java.awt.BorderLayout());

            detailsPane.setBorder(new javax.swing.border.EmptyBorder(new java.awt.Insets(4, 4, 4, 4)));
            detailsPane.setName("detailsPane");
            detailsSplitPane.setBorder(null);
            detailsSplitPane.setDividerLocation(220);
            detailsSplitPane.setDividerSize(6);
            detailsSplitPane.setMaximumSize(new java.awt.Dimension(1600, 1600));
            detailsSplitPane.setMinimumSize(new java.awt.Dimension(0, 0));
            detailsSplitPane.setPreferredSize(new java.awt.Dimension(0, 0));
            detailsScrollPane0.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            detailsScrollPane0.setViewportView(detailsList);

            detailsSplitPane.setLeftComponent(detailsScrollPane0);

            detailsScrollPane1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(50, 50, 50)));
            detailsEditorPane.setEditable(false);
            detailsScrollPane1.setViewportView(detailsEditorPane);

            detailsSplitPane.setRightComponent(detailsScrollPane1);

            detailsPane.add(detailsSplitPane, java.awt.BorderLayout.CENTER);

            appTabbedPane.addTab("System Details", detailsPane);

            getContentPane().add(appTabbedPane, java.awt.BorderLayout.CENTER);

            pack();
```

```
    }
    // </editor-fold>

    private void printButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        this.printDisplay();
    }

    private void purgeDataButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        //  not yet implemented - Adam
    }


    private void serverButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        try
        {   server.kill();  }
        catch (Exception e)
        {   System.out.println("\nAutumn.serverButtonActionPerformed\n" + e);   }

        this.createServer();
    }


    private void summaryListValueChanged(javax.swing.event.ListSelectionEvent evt)
    {
        this.createSystemSummaryTable();
    }


    private void devicesVariableListValueChanged(javax.swing.event.ListSelectionEvent evt)
    {
        this.createDevicesTable();

    }


    private void devicesDeviceListValueChanged(javax.swing.event.ListSelectionEvent evt)
    {
        this.createDevicesTable();
        this.createDevicesVariablesList();
    }


    private void
    applicationsListValueChanged(javax.swing.event.ListSelectionEvent evt)
    {
        this.createApplicationsTable();

    }


    private void updateDisplayButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        components.add(this);
        this.dataStore = new DataStore(components);
        dataStore.start();
    }


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        System.setProperty("swing.aatext", "true");
        try
        {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
        catch (Exception e)
        {   System.out.println(e);  }

        final String[] options = args;

        java.awt.EventQueue.invokeLater(new Runnable()
        {
            public void run()
            {
                Client client = new Client();
                client.start();

                if (options.length < 1)
                {
                    Autumn autumn = new Autumn();
                    autumn.setVisible(true);
                }
            }
        });
    }


    private void createServer()
    {
        server = new Server();
        server.setDaemon(true);
        server.start();
    }

    private void createClient()
    {
        client = new Client();
        client.setDaemon(true);
        client.start();
    }


    // Variables declaration - do not modify
    private javax.swing.JTabbedPane appTabbedPane;
    private javax.swing.JList applicationsList;
    private javax.swing.JPanel applicationsPane;
    private javax.swing.JScrollPane applicationsScrollPane0;
    private javax.swing.JScrollPane applicationsScrollPane1;
    private javax.swing.JSplitPane applicationsSplitPane;
    private javax.swing.JToolBar autumnToolBar;
    private javax.swing.JEditorPane detailsEditorPane;
    private javax.swing.JList detailsList;
    private javax.swing.JPanel detailsPane;
    private javax.swing.JScrollPane detailsScrollPane0;
    private javax.swing.JScrollPane detailsScrollPane1;
    private javax.swing.JSplitPane detailsSplitPane;
    private javax.swing.JList devicesDeviceList;
    private javax.swing.JPanel devicesPane;
    private javax.swing.JScrollPane devicesScrollPane0;
    private javax.swing.JScrollPane devicesScrollPane1;
    private javax.swing.JScrollPane devicesSelectScrollPane0;
    private javax.swing.JScrollPane devicesSelectScrollPane1;
    private javax.swing.JSplitPane devicesSelectSplitPane;
    private javax.swing.JSplitPane devicesSplitPane;
    private javax.swing.JList devicesVariableList;
    private javax.swing.JButton printButton;
    private javax.swing.JButton purgeDataButton;
    private javax.swing.JButton serverButton;
    private javax.swing.JButton statusButton;
    private javax.swing.JList summaryList;
    private javax.swing.JPanel summaryPane;
    private javax.swing.JScrollPane summaryScrollPane0;
    private javax.swing.JScrollPane summaryScrollPane1;
    private javax.swing.JSplitPane summarySplitPane;
    private javax.swing.JButton updateDisplayButton;
    // End of variables declaration


    public void createApplicationsList()
    {
        applicationsList.setCellRenderer(new
    OurListCellRenderer("asset"));
        applicationsList.setListData(dataStore.applicationsNames());
    }


    private void createApplicationsTable()
    {
        applicationsScrollPane1.setViewportView(null);
        if (applicationsList.getSelectedIndex() > -1)
        {
            String selected = (String) applicationsList.getSelectedValue();
            SortModel sortModel = dataStore.appSortModel(selected);
            JTable table = new JTable(sortModel);
            sortModel.setParent(table);
            table.setToolTipText("Your selection resulted in " + table.getRowCount() + " item(s).");
            applicationsScrollPane1.setViewportView(table);
        }
    }


    public void createDevicesList()
    {
        devicesDeviceList.setCellRenderer(new OurListCellRenderer("asset"));
        devicesDeviceList.setListData(dataStore.devicesNames());
    }


    private void createDevicesTable()
    {
        devicesScrollPane1.setViewportView(null);
        if (devicesDeviceList.getSelectedIndex() > -1)
        {
            String device = (String) devicesDeviceList.getSelectedValue();
            Object[] variables = devicesVariableList.getSelectedValues();
            SortModel sortModel = dataStore.devicesSortModel(device, variables);
            JTable table = new JTable(sortModel);
            sortModel.setParent(table);
            table.setToolTipText("Your selection resulted in " + table.getRowCount() + " item(s).");
            devicesScrollPane1.setViewportView(table);
        }
    }


    private void createDevicesVariablesList()
    {
        String selected = (String) devicesDeviceList.getSelectedValue();
        Vector variables = (Vector) dataStore.deviceVariables(selected);
        variables.remove("Device");
        devicesVariableList.setCellRenderer(new OurListCellRenderer("variable"));
        devicesVariableList.setListData(variables);
    }


    public void createSystemSummaryList()
    {
        summaryList.setCellRenderer(new
OurListCellRenderer("variable"));
        summaryList.setListData(dataStore.summaryVariableNames());
    }


    private void createSystemSummaryTable()
    {
        summaryScrollPane1.setViewportView(null);
        if (summaryList.getSelectedIndex() > -1)
        {
            Object[] variables = summaryList.getSelectedValues();
            SortModel sortModel =
dataStore.summarySortModel(variables);
            JTable table = new JTable(sortModel);
            sortModel.setParent(table);
            summaryScrollPane1.setViewportView(table);
        }
    }


    private void modifyButton(JButton input)
    {
        final JButton button = input;
        final Color normal = button.getBackground();
        final Color hover = new Color(normal.getRed() - 50,
    normal.getGreen() - 50, normal.getBlue() -50);

        button.setMinimumSize(new java.awt.Dimension(110, 25));
        button.setMaximumSize(new java.awt.Dimension(110, 25));
        button.setPreferredSize(new java.awt.Dimension(110, 25));
        button.setHorizontalAlignment(JButton.LEFT);

        button.addMouseListener(new java.awt.event.MouseAdapter()
        {
            public void mouseEntered(java.awt.event.MouseEvent
    mouseEvent)
            {   button.setBackground(hover);    }

            public void mouseExited(java.awt.event.MouseEvent
mouseEvent)
            {   button.setBackground(normal);   }
        });
    }


    private void printDisplay()
    {
        try
        {
            JPanel panel = (JPanel)
appTabbedPane.getSelectedComponent();

            Component[] components = subComponents(panel);
            for (int i = 0; i < components.length; i++)
            {
                if (components[i] instanceof JTable)
```

```
                    {
                        SortModel sortModel = null;

                        if (panel.getName().equals("summaryPane"))
                            sortModel = (SortModel) ((JTable)
    components[i]).getModel();

                        else if (panel.getName().equals("devicesPane"))
                            sortModel = (SortModel) ((JTable)
    components[i]).getModel();

                        else if
(panel.getName().equals("applicationsPane"))
                            sortModel = (SortModel) ((JTable)
    components[i]).getModel();

                        dataStore.printDisplay(sortModel);
                    }
                }
            }
            catch (Exception e)
            {   System.out.println("Autumn.printDisplay\n" + e);    }
        }


        public void setStatus(String input)
        {   statusButton.setText(input);    }

}
```

N:\IT\Adam\Java\Autumn\src\autumn\DataStore.java

```
//  DataStore.java
//  Created on November 15, 2005, 9:32 AM

package autumn;

import java.awt.Component;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.util.Collections;
import java.util.Vector;
import javax.swing.JButton;
import javax.swing.JList;
import javax.swing.JFrame;
import javax.swing.JToolBar;


public class DataStore extends Thread
{
    private Vector applicationsNames;
    private Vector devicesNames;
    private Vector summaryVariableNames;
    private Vector components;

    private Autumn autumn;
    private JButton statusButton;
    private JList applicationsList;
    private JList devicesList;


    public DataStore(Vector components)
    {
        this.components = components;
        this.applicationsNames = new Vector();
        this.devicesNames = new Vector();
        this.summaryVariableNames = new Vector();

        for (int i = 0; i < components.size(); i++)
        {
            Component current = (Component) components.elementAt(i);
            if (current.getName().equals("autumn"))
                autumn = (Autumn) current;
        }

        initialize();
    }


    //  uses            processAssets()
```

```
        private void initialize()
        {
            Thread thread = new Thread()
            {
                public void run()
                {
                    try
                    {
                        File parent = new File("C:\\Program
Files\\Autumn");
                        File[] directory = parent.listFiles();

                        for (int i = 0; i < directory.length; i++)
                        {
                            autumn.setStatus(((i + 1) * 100) /
        directory.length + " % done");
                            try
                            {
                                if (directory[i].isFile())
                                {
                                    FileInputStream inStream = new
        FileInputStream(directory[i]);
                                    ObjectInputStream inputStream = new
        ObjectInputStream(inStream);
                                    Summary summary = (Summary)
        inputStream.readObject();
                                    processAssets(summary);
                                }
                            }
                            catch (Exception e)
                            {
        System.out.println("DataStore.initialize[inner try]\n" + e);     }
                        }
                    }
                    catch (Exception e)
                    {   System.out.println("DataStore.initialize\n" + e);
}

                    autumn.createApplicationsList();
                    autumn.createDevicesList();
                    autumn.createSystemSummaryList();
                }
            };
            thread.start();
        }


        //  uses            asset.query()
        //  uses            summary.applications()
        //  uses            summary.devices()
        //  uses            addUnique()
        //  used by     initialize()
        private void processAssets(Summary summary)
        {
            Vector applications = summary.applications();
            Vector devices = summary.devices();

            for (int i = 0; i < applications.size(); i++)
            {
                Asset asset = (Asset) applications.elementAt(i);
                this.addUnique(applicationsNames,
       asset.query("Application"));
                this.serialize(asset);
            }

            for (int i = 0; i < devices.size(); i++)
            {
                Asset asset = (Asset) devices.elementAt(i);
                if (asset.query("Device").equals("System Summary") ==
false)
                    this.addUnique(devicesNames, asset.query("Device"));
                this.serialize(asset);
            }

            if (summaryVariableNames.size() < 1)
            {
                Asset systemSummary = summary.systemSummary();
                Vector data = systemSummary.data();
                for (int i = 0; i < data.size(); i++)
                {
                    String[] array = (String[]) data.elementAt(i);
                    summaryVariableNames.add(array[0]);
                }
                summaryVariableNames.remove("Device");
                summaryVariableNames.remove("System Name");
            }

            Collections.sort(applicationsNames);
            Collections.sort(devicesNames);
        }


        //  used by     processAssets
        private void addUnique(Vector vector, String input)
        {
            if (vector.contains(input))
                return;
            else
                vector.add(input);
        }


        private void serialize(Asset input)
        {
            try
            {
                File dataOut = assetPath(input);
                File parent = new File(dataOut.getParent());
                parent.mkdirs();
                FileOutputStream outStream = new FileOutputStream(dataOut);
                ObjectOutputStream outputStream = new
       ObjectOutputStream(outStream);
                outputStream.writeObject(input);
                outputStream.flush();
            outputStream.close();
        }
        catch (Exception e)
        {
            // System.out.println("DataStore.serialize\n" + e);
        }

    }


    private File assetPath(Asset asset)
    {
        String path = "C:\\Program Files\\Autumn\\";
        Vector data = asset.data();
        for (int i = 0; i < data.size(); i++)
        {
            String[] array = (String[]) data.elementAt(i);
            if (array[0].equals("Application"))
                path = path + array[0] + "\\" + array[1] + "\\";

            if (array[0].equals("Device"))
                path = path + array[0] + "\\" + array[1] + "\\";
        }
        path = path + asset.inetAddress().getHostAddress() + ".data";

        try
        {   return new File(path);  }
        catch (Exception e)
        {   System.out.println(e);  }

        return null;
    }


    //  uses:   variables(String string);
    //  uses:   assets();
    public SortModel summarySortModel(Object[] variables)
    {
        Vector rows = new Vector();
        Vector columns = summaryVariableNames;

        if (variables.length > 0)
            columns = convertArray(variables);

        columns.insertElementAt("IP Address", 0);
        columns.insertElementAt("System", 0);

        try
        {
            File parent = new File("C:\\Program
    Files\\Autumn\\Device\\System Summary");
            File[] directory = parent.listFiles();

            for (int i = 0; i < directory.length; i++)
            {
                try
                {
```

```
          if (directory[i].isFile())
          {
            FileInputStream inStream = new
  FileInputStream(directory[i]);
            ObjectInputStream inputStream = new
  ObjectInputStream(inStream);
            Asset asset = (Asset) inputStream.readObject();
            Vector line = new Vector();


  line.add(asset.inetAddress().getHostName().toUpperCase());
            line.add(asset.inetAddress().getHostAddress());

            for (int k = 2; k < columns.size(); k++)
              line.add(asset.query((String)
  columns.elementAt(k)));

            rows.add(line);
          }
        }
        catch (Exception e)
        {  System.out.println(e);  }
      }
    }
    catch (Exception e)
    {  System.out.println("DataStore.summarySortModel\n" + e); }

//  columns.remove("Device");
//  columns.remove("System Name");

    return new SortModel(rows, columns);
  }




  public SortModel appSortModel(String input)
  {
    Vector columns = new Vector();
    columns.add("System");
    columns.add("IP Address");
    columns.add("Application");
    columns.add("Version");
    columns.add("Publisher");
    columns.add("Comments");

    Vector rows = new Vector();

    try
    {
      File parent = new File("C:\\Program
  Files\\Autumn\\Application\\" + input);
      File[] directory = parent.listFiles();

      for (int i = 0; i < directory.length; i++)
      {
        try
```

```
                {
                    if (directory[i].isFile())
                    {
                        FileInputStream inStream = new
    FileInputStream(directory[i]);
                        ObjectInputStream inputStream = new
    ObjectInputStream(inStream);
                        Asset asset = (Asset) inputStream.readObject();
                        Vector line = new Vector();


    line.add(asset.inetAddress().getHostName().toUpperCase());
                        line.add(asset.inetAddress().getHostAddress());
                        line.add(asset.query("Application"));
                        line.add(asset.query("DisplayVersion"));
                        line.add(asset.query("Publisher"));
                        line.add(asset.query("Comments"));
                        rows.add(line);
                    }
                }
                catch (Exception e)
                {   System.out.println(e);  }
            }
        }
        catch (Exception e)
        {   System.out.println("DataStore.summarySortModel\n" + e); }
        return new SortModel(rows, columns);
    }


    public SortModel devicesSortModel(String device, Object[]
variables)
    {
        Vector rows = new Vector();
        Vector columns = new Vector();

        if (variables.length < 1)
            columns = this.deviceVariables(device);
        else
        {
            for (int i = 0; i < variables.length; i++)
                columns.add((String) variables[i]);
        }
        columns.insertElementAt("IP Address", 0);
        columns.insertElementAt("System", 0);
        // columns.remove("Device");
        columns.remove("System Name");

        try
        {
            File parent = new File("C:\\Program
Files\\Autumn\\Device\\"
     + device);
            File[] directory = parent.listFiles();

            for (int i = 0; i < directory.length; i++)
            {
```

```
            try
            {
                if (directory[i].isFile())
                {
                    FileInputStream inStream = new
FileInputStream(directory[i]);
                    ObjectInputStream inputStream = new
ObjectInputStream(inStream);
                    Asset asset = (Asset) inputStream.readObject();
                    Vector line = new Vector();


line.add(asset.inetAddress().getHostName().toUpperCase());
                    line.add(asset.inetAddress().getHostAddress());

                    for (int k = 2; k < columns.size(); k++)
                        line.add(asset.query((String)
columns.elementAt(k)));

                    rows.add(line);
                }
            }
            catch (Exception e)
            {   System.out.println(e);  }
        }
    }
    catch (Exception e)
    {   System.out.println("DataStore.summarySortModel\n" + e); }

    return new SortModel(rows, columns);
}


public void printDisplay(SortModel sortModel)
{
    try
    {
        Vector columns = sortModel.columns();
        Vector rows = sortModel.data();
        String sep = System.getProperty("line.separator");
        File dataOut = new File("C:\\Program
Files\\Autumn\\display.html");
        FileWriter outPut = new FileWriter(dataOut);
        outPut.write("<html>" + sep);
        outPut.write("<style type=text/css>\n" + sep);
        outPut.write("table\n" + sep);
        outPut.write("{" + sep);
        outPut.write("font-family: Tahoma;" + sep);
        outPut.write("font-size: 8pt;" + sep);
        outPut.write("}" + sep);
        outPut.write("td    { height: 15px;     }" + sep);
        outPut.write("</style>" + sep);
        outPut.write("<table border=1>" + sep);

        String title = "<tr>" + sep;
        columns.insertElementAt("&nbsp", 0);
        for (int i = 0; i < columns.size(); i++)
```

```
            {   title = title + "<td valign=top bgcolor=cccccc>" +
    (String) columns.elementAt(i) + "</td>" + sep;   }
            title = title + "</tr>" + sep;
            outPut.write(title);

            for (int i = 0; i < rows.size(); i++)
            {
                Vector row = (Vector) rows.elementAt(i);
                String line = "<tr>" + sep + "<td valign=top>" + i +
    "</td>" + sep;
                for (int k = 0; k < row.size(); k++)
                {   line = line + "<td valign=top>" + (String)
    row.elementAt(k) + "</td>" + sep;         }
                line = line + "</tr>" + sep;
                outPut.write(line);
            }

            outPut.write("</table>" + sep);
            outPut.write("</html>" + sep);
            outPut.close();

            String command = "cmd /C \"" + dataOut.getAbsoluteFile() +
    "\"";
            Runtime runTime = Runtime.getRuntime();
            Process process = runTime.exec(command);
        }
        catch (Exception e)
        {   System.out.println("DataStore.printDisplay\n" + e);     }
    }


    private Vector convertArray(Object[] array)
    {
        Vector out = new Vector();
        for (int i = 0; i < array.length; i++)
            out.add(array[i]);

        return out;
    }


    public Vector deviceVariables(String input)
    {
        Vector out = new Vector();

        try
        {
            File parent = new File("C:\\Program
Files\\Autumn\\Device\\"
    + input);
            File[] directory = parent.listFiles();

            for (int i = 0; i < directory.length; i++)
            {
                try
                {
                    if (directory[i].isFile())
                    {
                        FileInputStream inStream = new
    FileInputStream(directory[i]);
                        ObjectInputStream inputStream = new
    ObjectInputStream(inStream);
                        Asset asset = (Asset) inputStream.readObject();
                        Vector data = asset.data();
                        for (int k = 0; k < data.size(); k++)
                        {
                            String[] array = (String[])
    data.elementAt(k);
                            this.addUnique(out, array[0]);
                        }
                        out.remove("Device");
                    }
                }
                catch (Exception e)
                {   System.out.println(e);  }
            }
        }
        catch (Exception e)
        {   System.out.println(e);  }

        return out;
    }


    //  used by:    Autumn
    //  purpose:    returns a vector of strings naming the types of
    devices
    public Vector devicesNames()
    {   return this.devicesNames;   }


    public Vector applicationsNames()
    {   return this.applicationsNames;  }


    public Vector summaryVariableNames()
    {   return this.summaryVariableNames;   }

}
```

N:\IT\Adam\Java\AutumnSetup\src\autumnsetup\AutumnSetup.java

```
/*
 * AutumnSetup.java
 *
 * Created on November 30, 2005, 4:57 AM
 */

package autumnsetup;

import java.io.File;
import java.io.FileWriter;
import java.util.Vector;
import javax.swing.UIManager;

/**
 *
 * @author  Oliver
 */
public class AutumnSetup extends javax.swing.JFrame
{
    private String JavaPath;
    private String JarPath;
    private String Service;
    private String ServicePath;
    private String SourcePath;
    private Vector setup;
    private Vector uninstall;


    /** Creates new form AutumnSetup */
    public AutumnSetup()
    {
    initComponents();
    initMyComponents();
    }

    private void initMyComponents()
    {
    try
    {
        setup = new Vector();
        uninstall = new Vector();
        textArea.setFont(new java.awt.Font("Tahoma", 0, 11));

        SourcePath = "\\\\kuec-ad1\\Autumn";
        Service = "Autumn";
        ServicePath = "\\\\kuec-ad1\\Autumn\\Autumn.exe";
        JarPath = "\\\\kuec-ad1\\Autumn\\Autumn.jar";
        JavaPath = System.getProperty("sun.boot.library.path") +
"\\client\\jvm.dll";

        textArea.append("JavaPath: " + JavaPath + "\n\n");
        File JavaVM = new File(JavaPath);

        setup.add("cmd /C netsh firewall add portopening ALL 50000 \"" + Service + "  Port 50000\" ENABLE ALL");
        setup.add("cmd /C netsh firewall add allowedprogram " + ServicePath + " " + Service + " ENABLE ALL");
        setup.add("cmd /C " + ServicePath + " -install " + Service + " \"" + JavaPath + "\" -Djava.class.path=" + JarPath + " -start autumn.Autumn -params client");
        setup.add("cmd /C net start " + Service);


        uninstall.add("cmd /C net stop " + Service);
        uninstall.add("cmd /C " + ServicePath + " -uninstall " + Service);
        uninstall.add("cmd /C netsh firewall delete portopening ALL 50000 ALL");
        uninstall.add("cmd /C netsh firewall delete allowedprogram " + ServicePath + " ALL");

    }
    catch (Exception e)
    {
        textArea.append("\nAutumnSetup.initMyComponents\n" + e);
    }
    }


    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents()
    {
        toolBar = new javax.swing.JToolBar();
        setupButton = new javax.swing.JButton();
        uninstallButton = new javax.swing.JButton();
        doneButton = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        textArea = new javax.swing.JTextArea();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setFont(new java.awt.Font("Tahoma", 0, 11));
        toolBar.setRollover(true);
        setupButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/autumnsetup/~Java.gif")));
        setupButton.setText("Setup");
        setupButton.setMaximumSize(new java.awt.Dimension(80, 26));
        setupButton.setMinimumSize(new java.awt.Dimension(80, 26));
        setupButton.setName("setupButton");
        setupButton.setPreferredSize(new java.awt.Dimension(80, 26));
        setupButton.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
                {
                    setupButtonActionPerformed(evt);
                }
            });

            toolBar.add(setupButton);

            uninstallButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/autumnsetup/~Java.gif")));
            uninstallButton.setText("Uninstall");
            uninstallButton.setMaximumSize(new java.awt.Dimension(80, 26));
            uninstallButton.setMinimumSize(new java.awt.Dimension(80, 26));
            uninstallButton.setName("uninstallButton");
            uninstallButton.setPreferredSize(new java.awt.Dimension(80,
26));
            uninstallButton.addActionListener(new
java.awt.event.ActionListener()
            {
                public void actionPerformed(java.awt.event.ActionEvent evt)
                {
                    uninstallButtonActionPerformed(evt);
                }
            });

            toolBar.add(uninstallButton);

            doneButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/autumnsetup/~Java.gif")));
            doneButton.setText("Done");
            doneButton.setMaximumSize(new java.awt.Dimension(80, 26));
            doneButton.setMinimumSize(new java.awt.Dimension(80, 26));
            doneButton.setName("doneButton");
            doneButton.setPreferredSize(new java.awt.Dimension(80, 26));
            doneButton.addActionListener(new
java.awt.event.ActionListener()
            {
                public void actionPerformed(java.awt.event.ActionEvent evt)
                {
                    doneButtonActionPerformed(evt);
                }
            });

            toolBar.add(doneButton);

            getContentPane().add(toolBar, java.awt.BorderLayout.SOUTH);

            jPanel1.setLayout(new java.awt.BorderLayout());

            jPanel1.setBorder(new javax.swing.border.EmptyBorder(new
java.awt.Insets(4, 4, 4, 4)));
            jPanel1.setMaximumSize(new java.awt.Dimension(1600, 1600));
            jPanel1.setMinimumSize(new java.awt.Dimension(0, 0));
            jPanel1.setPreferredSize(new java.awt.Dimension(600, 400));
            jScrollPane1.setBorder(new javax.swing.border.LineBorder(new
java.awt.Color(50, 50, 50)));
            jScrollPane1.setViewportView(textArea);

```

```
        jPanel1.add(jScrollPane1, java.awt.BorderLayout.CENTER);

        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        pack();
    }
    // </editor-fold>

    private void doneButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
    System.exit(0);
    }

    private void uninstallButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
    this.uninstallService();
    }

    private void setupButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
    this.setupService();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
    System.setProperty("swing.aatext", "true");
    try
    {   UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());    }
    catch (Exception e)
    {   System.out.println(e);  }

    final String[] options = args;

    java.awt.EventQueue.invokeLater(new Runnable()
    {
        public void run()
        {
        new AutumnSetup().setVisible(true);
        }
    });
    }


    private void setupService()
    {
    Thread thread = new Thread()
    {
        public void run()
        {
        try
        {
            setupButton.setEnabled(false);
            uninstallButton.setEnabled(false);
            doneButton.setEnabled(false);

            textArea.append("\n\nInstalling Service Autumn\n");

            Runtime runTime = Runtime.getRuntime();
            for (int i = 0; i < setup.size(); i++)
            {
            textArea.append(setup.elementAt(i) + "\n");
            Process process = runTime.exec((String)
setup.elementAt(i));
            int exitVal = process.waitFor();
            textArea.append("    ExitValue: " + exitVal + "\n");

            }
            textArea.append("Done Installing Service Autumn\n");

            setupButton.setEnabled(true);
            uninstallButton.setEnabled(true);
            doneButton.setEnabled(true);
        }
        catch (Exception e)
        {
            System.out.println("\nAutumnSetup.setup\n" + e);
        }
        }
    };
    thread.start();
    }


    private void uninstallService()
    {
    Thread thread = new Thread()
    {
        public void run()
        {
        try
        {
            setupButton.setEnabled(false);
            uninstallButton.setEnabled(false);
            doneButton.setEnabled(false);

            textArea.append("\n\nUninstalling Service Autumn\n");
            String sep = System.getProperty("line.separator");
            File dataOut = new File("C:\\Autumn.bat");
            FileWriter outPut = new FileWriter(dataOut);

            for (int i = 0; i < uninstall.size(); i++)
            {
            outPut.append((String) uninstall.elementAt(i) + sep);
            }
            outPut.close();
            Runtime runTime = Runtime.getRuntime();
            Process process = runTime.exec("cmd /C c:\\autumn.bat");
```

```
                int exitVal = process.waitFor();
                textArea.append("    ExitValue: " + exitVal + "\n");
                textArea.append("Done Uninstalling Service Autumn\n");

                setupButton.setEnabled(true);
                uninstallButton.setEnabled(true);
                doneButton.setEnabled(true);
            }
            catch (Exception e)
            {
                System.out.println("\nAutumnSetup.uninstall\n" + e);
            }
            }
        };
        thread.start();
        }


        // Variables declaration - do not modify
        private javax.swing.JButton doneButton;
        private javax.swing.JPanel jPanel1;
        private javax.swing.JScrollPane jScrollPane1;
        private javax.swing.JButton setupButton;
        private javax.swing.JTextArea textArea;
        private javax.swing.JToolBar toolBar;
        private javax.swing.JButton uninstallButton;
        // End of variables declaration

}
```