ABSTRACT

Microsoft’s Active Directory (AD) services provide methods of unifying an entire network of devices and applications. AD is a central collection of users, groups and computers, enabling single sign-on (SSO) for devices and applications joined to the AD domain. However, the tools provided by Microsoft that helps to manage the AD services do not offer the capabilities of bulk user, group and computer management.

Existing tools provide mechanisms for managing user properties, passwords and group membership in bulk; however, each tool solves an individual problem. In this project, the Active Directory Management Tool (ADMT) combines several of these mechanisms into a single application. Integrating these tools aids system administrators by providing a single tool to manage multiple components of AD.

Additionally, this tool manages group membership based on a user’s role within a business. This will aid system administrators in defining the level of access that a particular person with a given business role should have. Also, the tool includes a logical filter. Microsoft’s tools include a filter that utilizes an object’s string attributes to determine if the attribute’s value contains a specified phrase. The logical filter proposed will allow system administrators to utilize additional attributes to determine if the attribute’s value falls within a certain range.
# TABLE OF CONTENTS

ABSTRACT ....................................................................................................................................... ii

TABLE OF CONTENTS .................................................................................................................. iii

LIST OF FIGURES ....................................................................................................................... vi

LIST OF TABLES ........................................................................................................................... viii

1. BACKGROUND AND RATIONALE ..................................................................................... 1
   1.1 Active Directory ............................................................................................................. 1
       1.1.1 Uses of Active Directory ...................................................................................... 2
   1.2 Prior Work ..................................................................................................................... 3
       1.2.1 Microsoft’s Active Directory Solutions .............................................................. 3
       1.2.2 SolarWind’s Active Directory Solutions .............................................................. 4
       1.2.3 WiseSoft’s Active Directory Solutions ............................................................... 4
       1.2.4 ManageEngine’s Active Directory Solutions ....................................................... 5

2. NARRATIVE .............................................................................................................................. 6
   2.1 Problem Statement .......................................................................................................... 6
   2.2 Motivation ....................................................................................................................... 7
   2.3 Parent Objective ............................................................................................................ 8
   2.4 Functionalities of the Project ....................................................................................... 8
       2.4.1 Bulk Management ................................................................................................. 8
       2.4.2 Logical Filtering ................................................................................................... 9
       2.4.3 Role Based Group Membership ......................................................................... 9

3. PROPOSED SYSTEM DESIGN ............................................................................................ 11
3.1 Environment ............................................................................................................ 11
3.2 Lightweight Directory Access Protocol ................................................................. 11
3.3 .NET Framework Classes ....................................................................................... 12
   3.3.1 DirectoryEntry Class ..................................................................................... 12
   3.3.2 DirectoryEntryConfiguration Class ............................................................. 13
   3.3.3 DirectorySearcher Class .............................................................................. 13
   3.3.4 SearchResult Class ..................................................................................... 13
   3.3.5 DirectoryServicesCOMException Class .................................................... 14
   3.3.6 DirectoryContext Class ............................................................................... 14
   3.3.7 Domain Class ................................................................................................ 14
   3.3.8 PrincipalContext Class .............................................................................. 14
   3.3.9 UserPrincipal Class ................................................................................... 15
   3.3.10 GroupPrincipal Class .............................................................................. 15
   3.3.11 ComputerPrincipal Class .......................................................................... 15
   3.3.12 ActiveDirectorySecurity Class ................................................................. 15
   3.3.13 ActiveDirectoryAccessRule Class ............................................................. 16
3.4 File Formats .......................................................................................................... 16
   3.4.1 Import Files ................................................................................................ 16
   3.4.2 Dictionary File ............................................................................................ 17
   3.4.3 Filters File ................................................................................................ 17
4. Implementation and Results ...................................................................................... 18
   4.1 Authenticating to the Domain .......................................................................... 18
   4.2 Building the Tree ............................................................................................. 19
LIST OF FIGURES

Figure 1.1. Desktop Operating System Market Share (NetMarketShare 2015) .......... 1
Figure 3.1. Distinguished Name for the User JSmith (Object Naming 2015) .......... 12
Figure 4.1. Active Directory Management Tool’s Login Interface ...................... 18
Figure 4.2. Visual Representation of Active Directory’s Structure .................... 20
Figure 4.3. Dialog For Adding a Single User .................................................. 21
Figure 4.4. Results of Importing a List of Users .............................................. 21
Figure 4.5. Accidental Deletion Protection Prompt .......................................... 23
Figure 4.6. Results of Deleting Multiple Objects ............................................ 24
Figure 4.7. Job Title Manager ..................................................................... 25
Figure 4.8. Resulting Group Memberships Due to Job Title Assignment .......... 26
Figure 4.9. Filter Manager ......................................................................... 27
Figure 4.10. Filter Editor ............................................................................ 28
Figure 4.11. Grouping Similar Objects ......................................................... 29
Figure 5.1. Sample CSV File for Importing Users ........................................ 30
Figure 5.2. Sample CSV File for Importing Groups ....................................... 30
Figure 5.3. Sample CSV File for Importing Computers ................................. 31
Figure 5.4. Result of the Importing the Sample Users CSV File .................... 31
Figure 5.5. Result of Importing the Sample Groups CSV File ......................... 32
Figure 5.6. Result of Importing the Sample Computers CSV File ................. 32
Figure 5.7. Display of All Imported Active Directory Objects ....................... 33
Figure 5.8. Result of Deleting Active Directory Objects in Bulk .................... 34
Figure 5.9. Job Title Manager ..................................................................... 34
Figure 5.10. Result of Assigning the Computer Science Professor Job Title ...... 35
Figure 5.11. Result of Assigning the Math Professor Job Title ....................... 36
Figure 5.12. Filter Manager ....................................................................... 37
Figure 5.13. Filter Editor .......................................................................... 37
Figure 5.14. Results of Filter with Flat View Enabled ......................................................... 38
Figure 5.15. Logon Using the Wrong Domain Name ........................................................... 38
Figure 5.16. Logon Using the Wrong Username ................................................................. 39
Figure 5.17. Attempting to Add Users with No Information ............................................ 40
Figure 5.18. Adding Blank Job Titles .............................................................................. 41
Figure 5.19. Adding a Job Title with a Duplicate Name ................................................... 41
Figure 5.20. Message Displayed When No Users Are Selected for Job Titles .............. 42
Figure 5.21. Filters Are Not Allowed to Have Blank Names ......................................... 42
Figure 5.22. Filters Are Not Allowed to Have Duplicate Names .................................. 43
LIST OF TABLES

Table 5.1. User Friendliness Evaluation ................................................................. 44
1. BACKGROUND AND RATIONALE

1.1 Active Directory

As shown below in Figure 1.1, Microsoft's Windows operating system is used widely throughout industry. Companies use Microsoft's Active Directory (AD) service to make single sign-on (SSO) available in company networks and to provide a central point for user and computer management. AD considers users and computers as objects, where each object is contained in a tree structure built using organizational units (OU). Users and computers are also members of groups within AD, and computers are also assigned to a site that represents a local area network (LAN). Based either on OU, group, or site membership, AD provides mechanisms for securing the company's network. However, the tools Microsoft provides for managing users and membership to OUs, groups or sites do not allow system administrators to perform tasks in bulk. To perform bulk operations, system administrators currently must create their own batch and PowerShell scripts utilizing AD commands and commandlets.

Figure 1.1. Desktop Operating System Market Share (NetMarketShare 2015)

The goal of this project is to implement a management tool for AD that allows system administrators to perform bulk operations. The proposed solution would aid
system administrators in determining objects that may need attention by filtering based on specified attributes, such as: PasswordLastSet and LastLogonTimeStamp. As a person's role within a company changes, their access to company systems and services also change. Taking advantage of an attribute available in each user's AD profile, this project proposes a method of managing group membership based on each user's business role. System administrators would be tasked to create a dictionary defining the group membership that each business role would be provided.

1.1.1 Uses of Active Directory

AD is a specialized database that is able to operate as a control mechanism to allow or disallow users access to the following within a company's network:

- Windows services, such as: application services, file shares, print services, Hyper-V virtual environments, terminal services, simple network management protocol (SNMP) monitoring, telnet, file transfer protocol (FTP) services, virtual private network (VPN) services, and more
- Microsoft services, such as: Microsoft SQL (MSSQL) Server, SharePoint, Lync, OneNote, Azure's cloud services, Exchange, Visual Studio's Team Foundation services, and more
- Wireless networks with WPA2-Enterprise Remote Authentication Dial In User Service (RADIUS) authentication
- Management consoles for networking equipment enabled with Authentication, Authorization, and Accounting's (AAA) RADIUS authentication
- Confidential section of the company's website
• A Microsoft Windows or UNIX based machine that is joined to the AD domain

1.2 Prior Work

Microsoft has developed tools to manage AD, however, realizing the shortcomings of these tools provided by Microsoft, various companies have developed their own tools to offset the shortcomings. Companies such as SolarWinds, ManageEngine and WiseSoft have created tools that are capable of performing individual tasks within AD, such as: computer and user account auditing, bulk user management, password management, and more (ManageEngine 2015, SolarWinds 2015, WiseSoft 2015). The disadvantages of current solutions are that most tools provide a means of accomplishing an individual task, and each company’s collection of tools does not integrate with each other.

1.2.1 Microsoft’s Active Directory Solutions

Microsoft provides GUI-based tools for managing different components within AD, such as users, computers, groups, group policies, sites, domains, forests, trusts, and certificates. The tools provided by Microsoft are called Active Directory Users and Computers, Active Directory Domains and Trusts, Active Directory Sites and Services, Certificate Authority and Group Policy Management. Each component may also be managed using the Windows command line or PowerShell commandlets, which also allows system administrators to script an operation that needs to be performed within the AD environment.
Active Directory Users and Computers is the tool Microsoft provides to manage AD objects. It allows system administrators to add or remove objects such as users, contacts, printers, computers, shared folders and groups. System administrators may also use this tool to manage group membership or the structure of AD. It offers filtering functionality that accepts multiple constraints to identify AD objects that have attributes containing a given string value.

1.2.2 SolarWind’s Active Directory Solutions

SolarWinds has several tools for managing AD, including Inactive User Account Removal Tool, Inactive Computer Account Removal Tool and User Import Tool. The Inactive User Account Removal tool allows system administrators to identify users that have been inactive for a given period of time, and remove the inactive users. The Inactive Computer Account Removal Tool is similar, however, it can identify and remove computers that are a specified age or older. The User Import Tool reads a CSV file and creates users based on the specified attributes.

1.2.3 WiseSoft’s Active Directory Solutions

WiseSoft’s solutions include applications, such as Account Management Spreadsheet, Bulk AD Users and Password Control. Similar to SolarWind’s applications, these tools have individual purposes. Account Management Spreadsheet, for example, allows system administrators to create AD user accounts in bulk. The Bulk AD Users tool can also perform bulk user account creation in addition to other bulk tasks, such as: password reset, password generation, user account deletion and the modification of user attributes. Password Control provides help desk employees a method of resetting a user’s
password, enabling or disabling the user’s account or forcing the user to change their password at next logon.

1.2.4 ManageEngine’s Active Directory Solutions

ManageEngine has created a software program called ADManager Plus, a single application to assist system administrators with a vast array of AD management tasks. The application is capable of performing bulk tasks, such as AD object creation, and AD object modification. These tasks include adding a list of users to a group in bulk, creating computer accounts and modifying user attributes. Individual filters may be applied to identify AD objects that meet specific criteria. For example, a filter can be applied to identify users that have dial-in access to the network.
2. NARRATIVE

2.1 Problem Statement

As mentioned earlier, current solutions do not integrate with each other in order to manage users, groups and computers within AD. ManageEngine’s ADManager Plus does provide a vast array of functionality for system administrators; however, the filters provided by ManageEngine cannot be combined. For example, if a system administrator wanted to identify users that have been inactive for ninety days and currently have expired accounts, two separate reports would have to be generated. The system administrator would have to review the reports for any users that are found in both reports in order to identify the users that have inactive and expired accounts.

One solution to get around this problem is to use scripts. For example, a PowerShell script could be written to take in the reports generated by ManageEngine and perform an administrative task. The script could also be written to perform the desired filtering without the use of ManageEngine’s tool. This, however, requires that the system administrator creates the required scripts. System administrators are not always fluent in a scripting language. If an inexperienced system administrator writes a script with little knowledge of the scripting language, he/she may cause damage to the AD service.  

When managing users and groups, system administrators typically create groups to represent business roles that users would perform. Although this method provides the desired outcome, the AD structure becomes cluttered with these groups as these groups are mixed in with other groups that are not named after business roles. Also, not every business role could be associated with more than a single user, which would eliminate the
need for a group related to that business role. Additionally, when groups are created as business roles, auditing access to systems can become a challenge as it adds an additional layer that the system administrator or security administrator must trace to determine which users have access to which systems.

The ADMT eliminates the need for groups that represent business roles by instead using the AD attribute to associate a user with a set of groups. This technique is also beneficial as it allows system administrators to identify the group memberships that are necessary to perform a given duty. For instance, the user jdoe is a programmer for a company, however, jdoe also participates in the company’s research practices. As a result, jdoe belongs to the “programmer” and “research” groups. When jdoe leaves the company, jsmith is hired to take jdoe’s place, however, jsmith does not need access to the “research” group.

In similar scenarios, system administrators currently identify the group memberships that the previous user had, and attempts to determine the group memberships that the new user will need based on the previous user. The ADMT prevents human error in identifying necessary group memberships, therefore, increasing system security using the principal of least privilege.

2.2 Motivation

System and security administrators have an assortment of tools to perform various AD management tasks. The tools provided by Microsoft are not always adequate for the task at hand, leaving potentially inexperienced system and security administrators to write their own scripts. System and security administrators may also download an
application to perform the task, and in some instances, multiple applications may be required to complete the given task.

2.3 Parent Objective

The ADMT integrates bulk user, group and computer creation with logical filtering components, allowing system administrators to filter for users or computers that meet specified constraints, and then perform a task on the resulting objects in bulk. The ADMT also allows system administrators to create a dictionary defining each business role’s group membership. When a user is assigned to a particular business role, the user will automatically be added to the corresponding groups that the business role is defined to be a member of.

2.4 Functionalities of the Project

One goal of the ADMT is to mimic the interface used by Microsoft’s Active Directory Users and Computers since this is the standard utility for managing AD. This will decrease the learning curve necessary for understanding how to use the ADMT. The ADMT goes beyond the standard utility by creating bulk user, group and computer management functionality. As well, the tool has a logical filter that is able to base comparisons on more than string comparisons, but also integer, date, time and boolean comparisons. Finally, this tool simplifies the process of managing group membership by allowing a system administrator to specify the business role that a user performs in order to manage the groups that the user is a member of.

2.4.1 Bulk Management
At times, system administrators must perform a single task to a collection of AD objects. With the ADMT, system administrators are able to select a collection of user accounts, and with the click of a button change the password of each of the selected user accounts. If a collection of computers are located at a single physical site, the attribute in AD specifying the physical location could be modified for each computer within the collection.

2.4.2 Logical Filtering

The filter provided in Active Directory Users and Computers is only capable of performing string comparisons, such as equal to, begins with, ends with and contains. The logical filter in ADMT allows system administrators to perform comparisons on strings as well as integers, dates, times and boolean values, then specify the comparison operator, such as: less than or equal to or greater than or equal to. For example, the logical filter allows system administrators to perform tasks such as filter for users that have not signed-in within the past ninety days.

In comparison to the filters provided by ManageEngine, the logical filter in the ADMT allows multiple filters to be applied. Building on top of the previous example that allowed a system administrator to filter for users that have not signed-in within the past ninety days, an additional filter may then be applied to determine which of those users accounts have expired. The system administrator may then choose to delete each of the user accounts.

2.4.3 Role Based Group Membership

Each user object has an attribute used for describing the business role that the person has within the company. This field is commonly unused; however, the ADMT
uses this field to store the dictionary word describing the business role. The business role, having been predefined by a system administrator, will specify the groups that a user with a given role should be a member of. When a system administrator changes a user’s business role via the ADMT, the tool will add the user to his/her new groups based on the definition provided in the dictionary without removing the user from his/her current group memberships.
3. PROPOSED SYSTEM DESIGN

3.1 Environment

During development of the ADMT, an AD server was required in order to have a functioning service to test against. Microsoft has built the AD service into their Windows Server line of products, requiring the use of a Windows Server operating system. For this project, Windows Server 2008 and Windows Server 2012 were used. The Windows Server operating system was installed in a virtual environment using VMware Workstation. VMware provides the ability to take snapshots, allowing quick and easy restoration in case of any potential malfunction.

Additionally, the ADMT was built using the C# programming language. C# is built on top of Microsoft’s .NET Framework, which provides deep access into the Windows environment. .NET Framework version 4.5 was used for this project, and as an IDE, Visual Studio was used for the development of the program as it provides debugging features for C# development.

3.2 Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is used to access the AD database. An LDAP query requests a particular AD object by specifying the object’s common name (CN), organizational unit (OU) and domain controller (DC) in that order. The purpose for these objects being in the order of CN, followed by OU, then followed by DC is due to how AD is structured. In AD, CN objects can be contained in either an OU or a DC object. OU objects are contained within either OU or DC objects. DC
objects may only be contained within other DC objects. When the names of these objects are combined, the central object’s distinguished name is created. In Figure 3.1, the distinguished name for the user object, JSmith, is created.

![Diagram showing the structure of a distinguished name]

**Figure 3.1. Distinguished Name for the User JSmith** (Object Naming 2015)

As shown above, the resulting distinguished name for user JSmith is

\[
CN=JSmith,OU=Promotions,OU=Marketing,DC=noam,DC=reskit,DC=com.
\]

Using the distinguished name, it is possible to create an LDAP query to then perform various tasks on the user’s account.

### 3.3 .NET Framework Classes

Microsoft’s .NET Framework has a multitude of classes available for C# developers, including several classes for AD programming. The following are classes necessary for the ADMT.

#### 3.3.1 DirectoryEntry Class

The DirectoryEntry class is used to perform LDAP queries against the AD service. The constructor accepts the distinguished name of the object to be queried, the username, the password and the authentication type, such as anonymous or secure.
When an instance is created, the instance object contains some, but not all, attributes related to the queried AD object. Additional attributes can still be accessed or modified using the instance object by utilizing functions built-in to the DirectoryEntry class (DirectoryEntry 2015).

### 3.3.2 DirectoryEntryConfiguration Class

Using the DirectoryEntryConfiguration class, it is possible to modify search options that are not related to the query. For instance, to ensure a secure method of communication is being used to send the password to the server, the DirectoryEntryConfiguration property PasswordEncodingMethod can be set to PasswordEncodingSsl (DirectoryEntryConfiguration 2015). If the DirectoryEntry instance has already specified similar options when it was initialized, the options do not have to be set using the DirectoryEntryConfiguration class.

### 3.3.3 DirectorySearcher Class

The DirectorySearcher class makes basic filtering possible. This class allows AD object attributes to be compared to user specified values. If the user does not need all attributes associated with each object with matching values, the DirectorySearcher can request specific attributes to be returned from the query. The query results will contain all objects and their attributes that match the specified constraints (DirectorySearcher 2015).

### 3.3.4 SearchResult Class

When a query is placed using an instance of the DirectorySearcher class, the query results are returned as a SearchResultCollection object, similar to an array. Each individual result in the collection is a SearchResult item. The elements of the collection
may be used to create a DirectoryEntry instance of the matched item, thus, allowing access to the matched item’s attributes (SearchResult 2015).

3.3.5 DirectoryServicesCOMException Class

When an error occurs while attempting to access an AD object, an exception is thrown. The exception is a subset of the DirectoryServicesCOMException class. The exception contains extended information about the error that occurred and provides details to trace the issue (DirectoryServicesComException 2015).

3.3.6 DirectoryContext Class

In order to access additional attributes related the AD domain, an instance of the DirectoryContext class must first be created. For the purpose of the project, the DirectoryContext would be configured to operate in a domain as a given user; however, this class could also be configured to operate on an AD forest or directory server (DirectoryContext 2015).

3.3.7 Domain Class

The Domain class represents the AD domain that would be managed by the ADMT. To identify the domain, an instance of the DirectoryContext class is utilized. The instance of the Domain class is then be used to gain additional information regarding the domain (Domain 2015).

3.3.8 PrincipalContext Class

The PrincipalContext class is used to provide principal objects (i.e., UserPrincipal, GroupPrincipal, ComputerPrincipal and Principal) a context in which they can operate (PrincipalContext 2015). In order to access objects within AD, the context is set for the AD domain. This is specified using the ContextType enumeration (ContextType 2015).
3.3.9 UserPrincipal Class

The UserPrincipal class offers access to a user object and its attributes. This also provides functionality to change a user’s password, force a user’s password to expire, determine group memberships, delete a user, get a DirectoryEntry instance of the user, and more. Instances of this class utilize an instance of the PrincipalContext class to determine the context in which it should operate (UserPrincipal 2015).

3.3.10 GroupPrincipal Class

Similar to the UserPrincipal class, the GroupPrincipal class offers access to a group object and its attributes. This also provides functionality to determine the groups that a group is a member of, enumerate the current members of a group, add members to a group, delete a group, get a DirectoryEntry instance of a group, and more. Instances of this class also utilize an instance of the PrincipalContext class to determine the context in which it should operate (GroupPrincipal 2015).

3.3.11 ComputerPrincipal Class

The ComputerPrincipal class offers access to a computer object and its attributes. This also provides functionality to determine a computer’s group memberships, delete a computer, get a DirectoryEntry instance of a computer, and more. Instances of this class also utilize an instance of the PrincipalContext class to determine the context in which it should operate (ComputerPrincipal 2015).

3.3.12 ActiveDirectorySecurity Class

Each object in AD has an access control list associated with it that controls the level of access necessary to view, modify or delete the object. An instance of the ActiveDirectorySecurity class can be obtained from a DirectoryEntry instance. This is
used to then determine the effective permissions associated with an object in AD

3.3.13 ActiveDirectoryAccessRule Class

An ActiveDirectoryAccessRule is an individual access control list entry obtained
from an instance of the ActiveDirectorySecurity class. This is used to identify and delete
existing access control list entries that prevent object deletion

3.4 File Formats

In the ADMT, there are several instances where files would be utilized. When a
user imports a list of users, groups or computers, a comma separated value (CSV) file is
used. The dictionary defining business roles is stored using a custom format, and the
filters are saved in a binary format.

3.4.1 Import Files

When importing a list of group or computer objects to be created by the ADMT,
the CSV file only requires a single column containing the name of the object. The CSV
file for users, however, requires the following columns: username, first name, last name,
password, the true or false value if the user must change their password at next logon, the
true or false value if the user cannot change their password, the true or false value if the
password never expires and the true or false value if the user’s account is disabled. For
instance, the user John Smith may have the following configuration:

Username: jsmith
First Name: John
Last Name: Smith
Password: Password1!
User Must Change Password at Next Logon: False
User Cannot Change Password: False
Password Never Expires: True
Account is Disabled: False

The following is an example line using the CSV format that would create the user
John Smith from the parameters above:

    jsmith, John, Smith, Password1!, false, true, false, false

3.4.2 Dictionary File

The dictionary for business roles uses a similar format to CSV files. The
dictionary term, A.K.A. the business role, is separated from its definition(s), A.K.A. the
group memberships, by a colon. Each individual group membership is separated by a
comma. For example, the dictionary term “CS Professor” may need to be a member of
the groups “Domain Users”, “Development Server” and “CSSLab”. In the dictionary file,
this entry would be represented by the following line:

    CS Professor: Domain Users, Development Server, CSSLab

3.4.3 Filters File

Filters are saved using an instance of the BinaryFormatter class. This class can
serialize and deserialize any object in a binary format (BinaryFormatter 2015). This
allows the ADMT to save the data directly from an instance of an object to a file, and
read the file back into an instance of the same object.
4. Implementation and Results

4.1 Authenticating to the Domain

As users of this tool perform various tasks, their domain level credentials are necessary to access AD. These credentials are obtained from the user in the form shown in Figure 4.1, which is displayed when the program begins execution. 

![Active Directory Management Tool's Login Interface](image)

**Figure 4.1. Active Directory Management Tool’s Login Interface**

When the user clicks the “Login” button, the form information is used to create an instance of a DirectoryEntry object, as shown in the code snippet below. However, the AD server is not queried until the program attempts to get the “Name” value for the DirectoryEntry instance. If this query fails, a catch statement will be able to determine the reason, and the program will make the user aware of the error so that the user may resolve the issue.

```csharp
DirectoryEntry dirEntry = new DirectoryEntry(
    "LDAP://" + Domain,
    Username,
    Password,
    AuthenticationTypes.Secure);
```
string dirName = dirEntry.Name;

4.2 Building the Tree

The AD structure is similar to that of a tree. To give users a visual representation of this tree, AD is recursively queried to obtain a listing of each AD object and its children objects. As shown in the code segment below, each AD object is checked for child objects. Every child object is added to the list of child nodes, and then its information is forwarded to the recursive function, “getADObjects”, to query if it has any children of its own.

DirectoryEntry dirEntry = new DirectoryEntry(
    "LDAP://" + DomainName,
    User,
    Pass,
    AuthenticationTypes.Secure);

foreach (DirectoryEntry x in dirEntry.Children)
{
    Tree node = new Tree(
        x.Path.ToString().Split('/').Last(),
        DomainName,
        x.SchemaClassName);

    ADObs[0].childNode.Add(node);

    getADObjects(ref node,
        x.Path.ToString().Split('/').Last());
}

After the logical tree is built, it is possible to create the visual representation of this tree structure, as illustrated in the left portion of Figure 4.2. Since users managing AD services are familiar with Microsoft’s Active Directory Users and Computers management interface, a similar design was followed in the making of this tool. As demonstrated in Figure 4.2, in the visual representation of the tree structure, it would
appear that the selected AD object, “Users”, does not have any child objects. However, the right portion of Figure 4.2 lists a number of children that exist under the “Users” container. In AD, there are six object types that are considered to be leaf objects: computers, contacts, groups, printers, users and volumes. Since these are leaves, these objects are not represented in the visual tree structure, similar to how Microsoft has designed Active Directory Users and Computers.

![Figure 4.2. Visual Representation of Active Directory’s Structure](image)

4.3 Object Management

This tool provides the user the ability to manage objects in AD. Several functions include adding objects, deleting objects and adding members to a group. The ADMT utilizes a dictionary of business roles to manage group memberships.

4.3.1 Adding Objects

When adding an object, such as a user, group or computer, the user has the ability to add a single object, as shown in Figure 4.3. The user also has the ability to import a list of objects from a CSV formatted file, as illustrated in Figure 4.4. After adding an object, a status message is provided notifying the user of any issues. This can be seen in the left column of the grid shown in Figure 4.4.
Figure 4.3. Dialog For Adding a Single User

Figure 4.4. Results of Importing a List of Users
A function has been created to handle the creation of each kind of object. This function has been designed to build each object in a similar fashion to how the object would normally be created within AD. For example, when adding an object, the snippet of code below is used:

```csharp
    dirEntry.Children.Add(objectName, objectType);
```

However, if the object being added were a computer, the “samAccountName” attribute associated with the computer object should be the object’s name followed by the “$” symbol. This can be assigned using an instance of the ComputerPrincipal class that is associated with the new computer object.

User objects also have additional attributes that must be set, and similar to computer objects, these attributes can be set using an instance of the UserPrincipal class that is associated with the new user object. Below is a sample of code demonstrating how these attributes are modified using an instance of a UserPrincipal named “userAccount”.

```csharp
    userAccount.SamAccountName = username;
    userAccount.GivenName      = firstName;
    userAccount.Surname        = lastName;
```

### 4.3.2 Deleting Objects

As opposed to the add function, this tool allows the user to delete any object in AD, not just users, groups and computers. The user can delete a single object or multiple objects at once. The code snippet below demonstrates how an object is deleted by getting two instances of the DirectoryEntry class, the first for the parent of the object being deleted and the second is for the object itself. The object is then removed from the list of the parent’s children, thus, deleting the object from AD.

```csharp
    DirectoryEntry dirEntry = new DirectoryEntry(
LDAPPath_Container, username, password, AuthenticationTypes.Secure);

DirectoryEntry object = new DirectoryEntry(
  LDAPPath_Object, username, password, AuthenticationTypes.Secure);

dirEntry.Children.Remove(object);

Objects can be protected from accidental deletion in AD. To verify if the user wants to delete the object, a prompt is provided, as shown in Figure 4.5. If the user chose to delete a single object, a message box will appear with the status of the deletion process. Otherwise, a window will appear with the statuses of each deleted object, as shown in Figure 4.6.

![Accidental Deletion Protection Prompt](image)

**Figure 4.5. Accidental Deletion Protection Prompt**
4.3.3 Adding Members to a Group Based on Business Roles

Groups are maintained in the ADMT by assigning job titles to users. Before a job title can be assigned, the Job Title Manager must be used to define the group memberships associated with a job title. As shown in Figure 4.7, the “New Job Title” field is used to create a common phrase that represents a business role. To prevent duplicate entries, the Job Title Manager will not allow job titles with the same name to be added. Once a job title has been added, the user can associate groups with the job title.
Figure 4.7. Job Title Manager

The ADMT can then be used to assign a job title to a user. This process will join the user to the groups associated with the job title. The window shown in Figure 4.8 will provide the status of joining each user to each group after the assignment of job titles.
4.4 Filtering

Filters can be created using the Filter Manager shown in Figure 4.9. This window gives the user the ability to flatten the view of the tree structure. In other words, this allows the user to see all AD objects, excluding those that are filtered out, in a single list. The user can deselect the “Flat View” box to return to the tree structure. To use the flat view, the scope of the filtering performed on the domain is set to “Subtree” as shown in the code snippet below.

```csharp
dirSearcher.SearchScope = SearchScope.Subtree;
```

When the user switches back to the tree structure, the scope of the filtering is set to “OneLevel” as shown below in the code snippet.

```csharp
dirSearcher.SearchScope = SearchScope.OneLevel;
```
The user also has the ability to create a new filter and provide it with a name to easily derive the filter’s purpose.

![Filter Manager](image)

**Figure 4.9. Filter Manager**

The filter is then edited, as shown in Figure 4.10, so that the user can place constraints on the displayed AD objects. The purpose of the filter shown in Figure 4.10, for example, is to limit the objects displayed in ADMT to only the users that were last logged into the domain between April 20, 2015 12:00:00 AM and April 20, 2015 11:59:59 PM and the computers that were last logged into between April 20, 2015 12:00:00 AM and April 20, 2015 11:59:59 PM.
When the example filter above is applied, the LDAP filter is generated. This filter would be written using prefix notation, as follows:

\[(|(&(objectCategory=user)(lastLogon>=130739796000000000)(lastLogon<=130740659990000000))(&(objectCategory=computer)(lastLogon>=130739796000000000)(lastLogon<=130740659990000000)))\]

The date and time stamps have been converted to an integer value that represents the number of 100-nanosecond intervals that have occurred since January 01, 1601 00:00:00 AM UTC (DateTime 2015). This is necessary for comparisons since AD stores date and time values using this format. The code snippet below demonstrates the conversion process from a date and time stamp to a long integer.

```csharp
DateTime dateTimeStamp = DateTime.Now;
long convertedDTS = dateTimeStampToFileTimeUtc();
```
4.5 Grouping

The data grid used to display the AD objects has a grouping feature that allows users to group similar objects by the properties displayed in the grid. For instance, grouping objects by their class (i.e., user, computer, group, etc.), then by their job title, then by whether or not they are disabled is illustrated in Figure 4.11. This allows system administrators to focus on particular groupings of AD objects, such as the “Computer Science Professor” whose account is disable as shown below in Figure 4.11.

![Figure 4.11. Grouping Similar Objects](image)

Figure 4.11. Grouping Similar Objects
5. Testing and Evaluation

5.1 Testing

Testing has been performed on the various features offered by the ADMT. The following sections demonstrate these tests.

5.1.1 Testing Bulk Operations

Sample CSV files containing users, groups and computers have been made, as shown in Figure 5.1, Figure 5.2, and Figure 5.3.

![Figure 5.1. Sample CSV File for Importing Users](image)

![Figure 5.2. Sample CSV File for Importing Groups](image)
Figure 5.3. Sample CSV File for Importing Computers

These files were then used to create each of the objects in bulk, as shown in Figure 5.4, Figure 5.5, and Figure 5.6.

Figure 5.4. Result of the Importing the Sample Users CSV File
Figure 5.5. Result of Importing the Sample Groups CSV File

Figure 5.6. Result of Importing the Sample Computers CSV File
The final result after importing the users, groups and computers in bulk is shown in Figure 5.7.

<table>
<thead>
<tr>
<th>commonName_Property</th>
<th>accountEnabled_Property</th>
<th>schemaClass_Property</th>
<th>title_Property</th>
<th>distinguishedName_Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>cslab-01</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=cslab-01,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>cslab-02</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=cslab-02,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>cslab-03</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=cslab-03,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>cslab-04</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=cslab-04,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>cslab-05</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=cslab-05,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>mathlab-05</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=mathlab-05,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>mathlab-04</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=mathlab-04,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>mathlab-03</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=mathlab-03,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>mathlab-02</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=mathlab-02,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>mathlab-01</td>
<td>Enabled</td>
<td>computer</td>
<td></td>
<td>CN=mathlab-01,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>WiFiAccess</td>
<td>group</td>
<td></td>
<td></td>
<td>CN=WiFiAccess,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>CSLab</td>
<td>group</td>
<td></td>
<td></td>
<td>CN=CSLab,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>MathLab</td>
<td>group</td>
<td></td>
<td></td>
<td>CN=MathLab,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Nick Jackson</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Nick Jackson,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Mike Lewis</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Mike Lewis,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Kim Silvas</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Kim Silvas,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Ryan Daranger</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Ryan Daranger,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>John Smith</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=John Smith,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>John Doe</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=John Doe,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Janice Coleman</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Janice Coleman,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Bill Young</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Bill Young,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Lilly Walker</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Lilly Walker,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
<tr>
<td>Amber Greene</td>
<td>Enabled</td>
<td>user</td>
<td></td>
<td>CN=Amber Greene,OU=TAMUCC,DC=dev,DC=lab</td>
</tr>
</tbody>
</table>

**Figure 5.7. Display of All Imported Active Directory Objects**

The ADMT can also perform bulk deletion of objects. Figure 5.8 demonstrates the result of deleting the previously created AD objects.
5.1.2 Testing Group Management via Business Roles

Business Roles were created using the Job Title Manager shown in Figure 5.9.

![Job Title Manager](image)

**Figure 5.9. Job Title Manager**
From the imported users created during testing, the users Amber Greene, John Doe, Kim Silvas, Nick Jackson and Ryan Daranger are assigned the “Computer Science Professor” job title. The users Mike Lewis, John Smith, Janice Coleman, Bill Young and Lilly Weber are assigned the “Math Professor” job title. Figure 5.10 shows the groups that each user was added to as a result of assigning the “Computer Science Professor” job title, and Figure 5.11 shows the groups that each user was added to as a result of assigning the “Math Professor” job title.

Figure 5.10. Result of Assigning the Computer Science Professor Job Title
Figure 5.11. Result of Assigning the Math Professor Job Title

As shown, the users that have been assigned job titles have been joined to the groups that a person that functions as a “Computer Science Professor” or “Math Professor” should be a member of.

5.1.3 Testing Filters

The filters can be managed using the Filter Manager shown in Figure 5.12. New filters can be created and saved for future use.
Once a filter has been made, the Filter Editor shown in Figure 5.13 can be used to edit a filter.

The result of this filter when applied with the flat view is shown below in Figure 5.14. A single computer, W2K8R2-X64, and a single user, Administrator, are shown to have logged in on April 20, 2015.
5.2 Error Handling

During the development of the project, bad data would be provided to the ADMT to evaluate its reaction. For instance, the logon screen would be provided with a bad domain, username or password. In return, it provides an error message with details as to why the logon failed, as shown in Figure 5.15 and Figure 5.16.

Figure 5.14. Results of Filter with Flat View Enabled

Figure 5.15. Logon Using the Wrong Domain Name
Figure 5.16. Logon Using the Wrong Username

When creating users, groups or computers, error checking is put in place to ensure that the necessary information is provided, as illustrated in Figure 5.17.
Figure 5.17. Attempting to Add Users with No Information

Figure 5.18 and Figure 5.19 illustrate that job titles cannot be blank, and must not be duplicated.
As shown in Figure 5.20, the ADMT does not have the ability to add a job title to any kind of AD object other than users, since AD does not have a title field for any kind of object other than users.
Figure 5.20. Message Displayed When No Users Are Selected for Job Titles

Similar to the Job Title Manager, the Filter Manager requires a non-blank and unique name in order to create a new filter, as shown in Figure 5.21 and Figure 5.22.

Figure 5.21. Filters Are Not Allowed to Have Blank Names
5.3 User Evaluations

Volunteers that are familiar with managing AD services were asked to perform a series of tasks using the ADMT. Tasks consisted of adding users, groups and computers, deleting users, groups and computers, creating a job title with associated group memberships, assign job titles to users, filter for specific job titles, and group the filtered users by job title and whether or not their account was enabled. The following table lists the average of the volunteers’ opinions of how each function within the ADMT ranks in terms of whether or not it is user friendly at first glance:
Table 5.1. User Friendliness Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk Operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group Management via Job Titles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filtering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each volunteer noted that after a few minutes of learning how the ADMT works, it became simple to accomplish complex tasks.
6. Conclusion and Future Work

The ADMT developed builds on top of the functionality provided by Microsoft’s Active Directory Users and Computers. ADMT is capable of bulk creation of users, groups and computers, bulk deletion of AD objects, managing group memberships based on a user’s business role, logical filtering on string, integer, date, time and boolean values and grouping of AD objects based on displayed properties. Through the use job titles, adding new users to groups helps to ensure the principal of least privilege is satisfied. Additionally, the use of multiple logical filters provides system administrators and security administrators more flexibility to identify objects in the domain that may need attention without the use of scripting languages.

In the future, ADMT could be broadened to include other features provided by Microsoft’s Active Directory Users and Computers, such as: group management without assigning job titles, assigning new passwords to users, management of AD object properties, and the creation of any kind of AD object. Also, the user interface could include additional features that a user familiar with Microsoft’s Active Directory Users and Computers would be accustomed to, such as: context menus, and the option to use a menu bar. Finally, the speed at which tasks are performed may be improved with the use of threading.
BIBLIOGRAPHY AND REFERENCES


