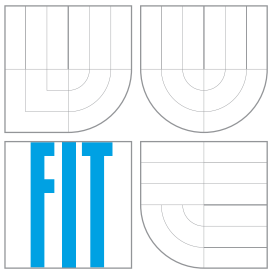
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

ACTIVE DIRECTORY MANAGEMENT DASHBOARD

NÁSTROJ PRO SPRÁVU ACTIVE DIRECTORY

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE
Bc. SAMUEL RADIMÁK

SUPERVISOR
VEDOUCÍ PRÁCE
Ing. TOMÁŠ FIEDOR

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Radimák Samuel, Bc.**
Obor: Informační systémy
Téma: **Nástroj pro správu Active Directory**
Active Directory Management Dashboard
Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte technologie související s doménovými službami Active directory.
2. Analyzujte požadavky pro správu domén Active Directory a navrhněte nástroj umožňující jednoduchou správu uživatelů, jejich atributů, auditování domény a jejího stavu.
3. Navržený nástroj implementujte v .NET jako samostatný software.
4. Ověřte implementovaný nástroj na všech základních případech užití, různých konfigurací systému a domény.

Literatura:

- Domovská stránka frameworku Microsoft .NET. URL: http://www.microsoft.com/net

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Fiedor Tomáš, Ing.**, UITS FIT VUT
Konzultant: Najvárek Jan, Ing., Ph.D., ARTIN
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstract

This thesis is focused on the main concepts of Active Directory and the creation of an application allowing basic management tasks. It introduces the logical as well as physical components and provides an overview of existing servers that are using the services of Active Directory. The functionality of existing management applications is discussed and desired properties of management applications are discovered. On these grounds, a new application concept is introduced and the benefits of the new application over the existing ones is shown. According to the concept, a new application is developed supporting the management of users and groups and implementing additional features such as profile photo editing and a definition of customized object creation process. This application is also tested on different levels and possibilities of future improvements are given.

Abstrakt

Jedným z kľúčových faktorov v oblasti informačných technológií sú ľudia. Ľudia v zamestnaní často používajú rôzne zdroje, ktoré im daná firma ponúka, pričom tieto zdroje je možné nájsť v rôznych tvaroch a veľkostiach – počítače, tlačiarne, telefóny, e-maily, softvér atď. Avšak každá správna firma potrebuje korektne nastaviť politiku správy daných zdrojov. Adresárové služby sú dlhodobo známym konceptom a dnes sú už široko používaným štandardom firiem po celom svete, kde pomáhajú v efektívnom riadení a zoskupovaní dostupných zdrojov firmy. Táto práca je zameraná na adresárové služby Microsoft Active Directory. Ponúka hlavné koncepty a základy technológie Active Directory.

V práci sú predstavené logické a fyzické komponenty Active Directory. Active Directory je pôvodná technológia spoločnosti Microsoft, avšak časom sa začali objavovať systémy podporujúce Active Directory na linuxových operačných systémoch. Táto práca sa zaoberá rozdielmi medzi Active Directory v operačných systémoch Windows Server a ďalších operačných systémov založených na Linuxe. Najdôležitejšie riešenie zo všetkých alternatív k Microsoft, Samba4, bol rozšírený o podporu Active Directory a úspešne kopíruje originálnu funkcionalitu. Stáva sa tým legitímnou alternatívou k použitiu Windows Server, s ktorým má v súčasnej dobe porovnateľnú funkcionalitu. V práci sa nachádza rozbor danej funkcionality a opísané sú aj obmedzujúce faktory daných systémov. Ako ďalšia alternatíva k predchádzajúcim riešeniam je poskytnutý aj popis systému Zentyal, ktorý je komerčnou implementáciou systému Samba4 a preto disponuje podobnými vlastnosťami ako Samba4.

Hlavným cieľom tejto práce je skutočná správa objektov v Active Directory, preto sú v práci predstavené existujúce aplikácie umožňujúce správu týchto objektov. Sám Microsoft ponúka balík základných nástrojov pre správu, ktoré sú porovnané s výberom pokročilejších komerčných nástrojov. Zobrazený je aj zoznam výhod a nevýhod každej z daných aplikácií.

Na základe popisu funkcionality existujúcich aplikácií pre správu domény sú identifikované niektoré chýbajúce vlastnosti takýchto aplikácií. Podľa týchto vlastností je predstavený nový koncept aplikácie, ktorý je porovnaný s existujúcimi riešeniami a ukazuje benefity tohto konceptu. Tento návrh je následne implementovaný, pričom nová aplikácia podporuje správu používateľov a skupín a takisto ponúka rozšírenia ako úpravu profilových fotiek a prispôsobenie procesu vytvárania objektov.

Na základe výsledkov testovania aplikácie sú predstavené možnosti budúceho rozširovania tohto projektu o nové vlastnosti.

Keywords

Active Directory, directory service, LDAP, application, domain, user, group, Microsoft Windows, Windows Server, Samba4

Klíčová slova

Active Directory, adresárová služba, LDAP, aplikácia, doména, používateľ, skupina, Microsoft Windows, Windows Server, Samba4

Reference

RADIMÁK, Samuel. *Active Directory Management Dashboard.* Brno, 2016. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Fiedor Tomáš.

Active Directory Management Dashboard

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Tomáš Fiedor. The supplementary information was provided by the representatives from the company ArtIn, namely Mr. Jiří Štěrba, Jan Najvárek, Lukáš Konečný and Petr Vyhňák. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Samuel Radimák
May 24, 2016

Acknowledgements

I would like to express my gratitude to my supervisor Ing. Tomáš Fiedor and Ing. Jan Fiedor, who guided me through the process of writing this thesis. I would also like to thank the representatives from ArtIn for their technical leadership and many insights that lead to the higher quality.

Contents

Chapter 1

Introduction

Information technology is one of the most important areas of interest among professionals around the world and is still experiencing a revolutionary developments in creating values a lot easier. One of the key factors in information technology are people. An information technology company always employs a number of people that create a value for the company and make all the developments possible. People in the company use different resources available in the company, resources that are necessary to create a value for the customer. Resources in a company can come in different shapes and sizes — people, computers, printers, phones, emails, software, etc. — and every company needs to make sure, that all those resources are properly managed.

Directory services are a long known concept today and are broadly used by a variety of companies around the globe to efficiently manage and group all resources available in a company. This thesis focuses on the directory service called Microsoft Active Directory. It offers a concise look at the basics of the Active Directory technology and offers a list of suggested literature that offer more comprehensive information with a broader scope. Nevertheless, this thesis describes the most fundamental concepts used in Active Directory and should be sufficient to understand the overall concept of Active Directory that every administrator should already have knowledge of.

Active Directory is originally a Microsoft technology, but over the time, also Linux-based systems tried to incorporate this technology. This thesis discusses differences between Active Directory in Windows Server operation systems and other Linux-based operation systems. The most important of all services, Samba4, was developed to copy the functionality of Active Directory and to act as a legitimate Active Directory service comparable to Windows Server. Also commercial implementation called Zentyal is discussed as a possibility to be a part of an Active Directory.

The main focus of this thesis is the actual management of Active Directory, therefore a number of the most known and used management applications for Active Directory is discussed. As Windows offers a package of basic management tools, they are compared to the selection of more advanced tools used today. A list of pros and cons of every application is also given.

Looking at the descriptions and reviews of existing management applications for Active Directory, a new concept of an application has risen. The design as well as implementation and testing of this application is the most important part of this work.

This diploma thesis uses the results of previously created study that was part of the semester project. Chapters 2, 3, 4 and 5, containing theory about Active Directory as well as the analysis of currently used applications were taken from the semester project, edited and served as the basis for this thesis. Chapters 6, 7 and 8 is the result of this study.

Chapter 2

Directory Services and Directory Server

Directory services are software systems that store, organize and provide access to directory information in order to unify network resources[1]. Directory service maps the names of network resources to network addresses and define a naming structure for networks.

The directory service provides transparency to protocols and network topology, permitting users to access resources without having to be aware of the physical location of the devices. It is an important component of the network operating system and is a central information repository for a service delivery platform.

Furthermore, specific directory services called *naming services* map the names of resources in the network to the respective network address. They provide a simple method to locate the resource based only on the provided name of the resource. Therefore, the physical address of a network resource is not needed. In addition, directory services define namespaces for networks, which hold one or more objects as name entries.

Directory services identify every resource such as an e-mail address, a peripheral device or a computer on the network, and make these resources accessible to users and applications. They hold shared information infrastructure to administer, manage, locate and organize common items and network resources. Each resource on the network is considered to be an object on the directory server.

Directory Server provides a central repository for storing and managing information, which makes it one of the core components of most corporate networks[15]. Almost any kind of information can be stored there, from identity profiles and access privileges to information about network resources, printers, network devices and manufactured parts. Information stored in Directory Server can be used for the authentication, authorization and audit of users, therefore to enable secure and manageable access to any kind of service or application within the corporate network. Directory Server is extensible, can be integrated with existing systems and enables the consolidation of employee, customer, supplier, and partner information. Directory servers are available from a number of vendors, and in their most basic form consist of key-value lookups on structured information. This information is often organized into a hierarchy.

The predominant directory access technology in use today is *Lightweight Directory Access Protocol* (LDAP)[12][18], a descendant of the X.500 directory standard from the International Telecommunication Union (ITU), and a component of the full Open Systems Interconnection (OSI) networking model[3].

Chapter 3

What is Active Directory?

Active directory (AD) is a directory service implementation that Microsoft developed for Windows domain networks. It originates from the LDAP/X.500 directory and is included in most of the Windows Server operating systems as a collection of services [9][14]. It is used to manage identity and access for and to resources on a network. AD makes use of LDAP versions 2 and 3, Microsoft's version of Kerberos, and Domain Name System (DNS).

AD stores information about objects on the network and makes this information easy for administrators and users to find, use and manage. One of the most important parts of AD infrastructure is a dedicated server or a set of servers called *Active Directory Domain Controller*. Domain controller (DC) is a server with Active Directory Domain Services (AD DS) installed, which provides all directory services (authentication, authorization, searching, central configuration using Group Policy, etc.) and stores all information needed for the services. It is also used for enforcing security policies for all resources in the domain. For example, when a user logs into the system, AD checks for the submitted password and determines the role of the user (administrator or normal user)[7].

3.1 Logical structure

The logical structure of Active Directory is flexible and provides methods for creating a directory hierarchy that makes it easier for administrators to manage it. Logical structure is independent of the physical location of servers and other components. In the most basic view, Active Directory contains objects and attributes, all of which are hierarchically arranged. In order to use and manage Active Directory, a competent person needs to know the logical structure in detail and get acquainted with the different layers of Active Directory. The main areas of Active Directory's structure include: Domains, Trees, Forests and Organizational Units.

3.1.1 Objects

Data is stored hierarchically, much like the registry's logical organization, where container objects can store other objects, including other container objects[7]. Everything that AD tracks is considered to be an object, more specifically, any user, system, resource or service tracked within AD is considered to be an object. Every object has a set of *attributes* that describe the object and are defined by a schema. One of the main advantages this design allows is a unique identification of an object. Active Directory allows for each object to

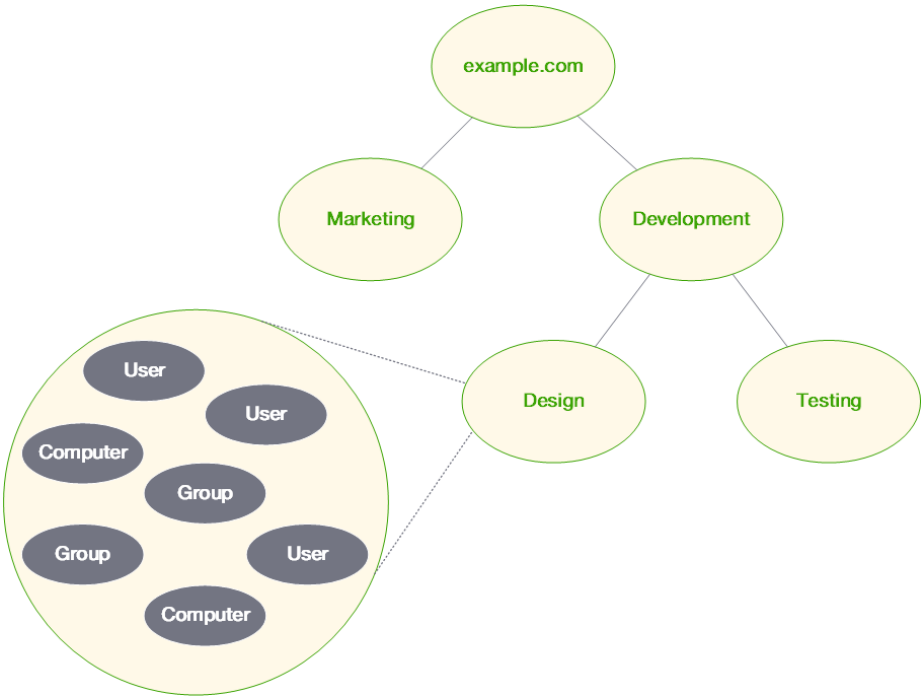


Figure 3.1: Objects hierarchy

be uniquely locatable and identifiable. For this purposes, there are two ways of uniquely identifying an object[2]:

- **Globally unique identifier (GUID)**: GUID is a universally unique identifier (UUID) in the form of a 128-bit number. It is automatically created and assigned to an object by the system at creation. GUID is not guaranteed to be unique, rather statistically improbable to be duplicated before the year 3400[4]. GUID is connected with the object for the whole life of the object until it is deleted and is universally unique (across all Active Directory instances). This number is hard to remember and does not conform to the hierarchy structure, therefore the second identification is more commonly used.
- **Distinguished name (DN)**: DN copies the hierarchical structure of Active Directory and is used to uniquely reference any object regardless of its type. DNs are defined as a means of referring to any object in the directory[16].

Since DN is more commonly used, we can see an example for a DN of a path to the root object from the Figure 3.1:

- *dc=example,dc=com*

DNs are made up of names and prefixes separated by the equals sign (=). In this example, dc stands for domain component and is used to specify domain or application partition objects. Table 3.1 provides the overview of the most often used attribute types in directory services, but more attribute types exist[17]. If we pretended that all containers in the Figure 3.1 are organizational units, we can see a more sophisticated example for a user named John Snow and a group called Managers:

Key	Attribute
CN	Common name
L	Locality name
ST	State or province name
O	Organization name
OU	Organizational unit name
C	Country name
DC	Domain component
UID	User ID

Table 3.1: Basic attribute types

- *cn=John Snow,ou=Design,ou=Development,dc=example,dc=com*
- *cn=Managers,ou=Design,ou=Development,dc=example,dc=com*

3.1.2 Schema

Schema is actually a set of classes and attributes to which objects belong[13]. They are available for a particular object type and therefore determine types of objects that can be stored in AD, where they belong in the database and what attributes belong to what objects. The schema is extendible in terms of adding new object classes and new attributes to define those objects. However, an object can only be deactivated and never deleted. Deletion of an object is not possible and even the deactivation or extension is fundamental change to the system, therefore every single change has to be always properly planned in advance[2].

Schema information is actually stored in the *schema partition* of the AD database, which allows administrators to modify it and let the changes be distributed to the whole domain without the need to restart any domain controllers. As a security precaution, schema modifications can be performed only on the domain controller holding the role of *Schema master* (more information about Schema master can be found in the subsection 3.1.9). Schema master is a forest-wide role, therefore there can only be one schema for each AD forest.

3.1.3 Domain

AD with objects can be viewed at a number of levels. The logical divisions in an AD network are domain, tree and forest. Domain is a logical grouping of users, computers, and other Active Directory objects based on administrative and security needs. It defines a separate namespace, a separate security structure, a separate management structure, and a separate naming context. An Active Directory Domain consists of four main components[2]:

- Hierarchical structure of containers and objects.
- A DNS domain name as a unique identifier.
- A security service, which enables authentication and authorization to access any resources in the domain.
- Policies that define the restrictions placed on users and machines.

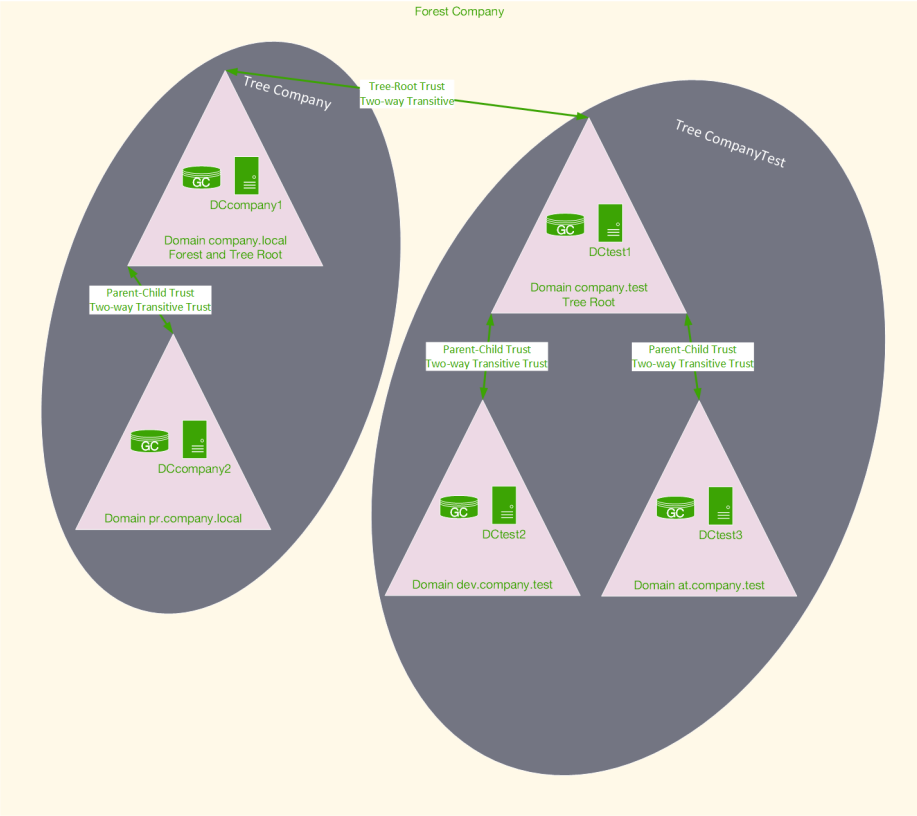


Figure 3.2: Domain, tree and forest relationship

Active Directory domain name is usually used as a full DNS name, but every domain has also a pre-Windows 2000 name for the purposes of compatibility called NetBIOS domain name. The domain name example from the Figure 3.1 is *example.com* and is in fact the root of the domain tree.

A *domain controller* (DC) hosts domain, while there is a restriction that prohibits configuration of more than one domain on a single DC. When a new domain is created and tree and forest does not exist yet, they are created automatically.

3.1.4 Tree

Domain tree is a collection of domains and forms a hierarchical structure of domain names connected together while using contiguous naming scheme. If *example.com* from the Figure 3.1 was to add more domains called Europe, Asia and Africa, then the new domain names would be *europe.example.com*, *asia.example.com* and *africa.example.com*. The domain name is comprised of their own name and the name of the parent domain recursively up to the root of the tree.

Another example of a tree can be found on the Figure 3.2. On the figure, there are actually two different trees – *Company* and *CompanyTest*. Tree Company includes two domains, where *pr.company.local* is a child domain of the *company.local* domain. Every domain has its own domain partition located on all DCs of the domain. Domains in such setting are trusting each other fully, therefore an administrator could be changing access privileges of any user to any resource in the tree. There is always only one root domain,

therefore when a new one is needed, there must be also a whole new tree created.

3.1.5 Forest

Forest is a collection of domain trees that share common logical structure, global catalog, schema, configuration as well as automatic two-way transitive trust relationships[8]. All domains share the same schema within the same forest and every single domain has its own domain partition. Furthermore, trees in a forest all trust each other, so objects in these trees are available to all users, if the security allows it[5]. The first domain that is created is typically known as the *Forest Root Domain*. Only one Forest Root Domain exists in the forest, it is the first domain created in the respective forest and it never looses this role. This domain is very important and contains groups Enterprise Admins and Schema Admins.

In the Figure 3.2, there is a Forest called Company. It contains two trees – *Company* and *CompanyTest*. *Company.local* is the forest root domain. Both trees now exist together and share their resources. Even though this kind of setup is possible, best practice for the new forest design is almost always a single-domain forest.

3.1.6 Organizational Units

Objects held within a domain can be grouped into Organizational Units (OUs). OUs are container objects and can be nested in other OUs or containers (second type of container object is called container). Microsoft alone recommends using OUs rather than domains for structure and to simplify the implementation of policies and administration. That makes it easier to locate and manage objects within one OU, which enhances the support for administrative purposes[10]. There is also a possibility to delegate the authority to manage an OU.

OUs provide multiple levels of administrative authority and are the smallest scope or unit to which *Group Policy* settings can be assigned or to which administrative authority can be delegated. This delegation simplifies the task of managing these objects, enables an administrator to structure Active Directory to fit the organization's requirements and can be scaled to any size. After installing a domain, a number of default OUs and containers is created, including the Users container, Computers container and the Domain Controllers OU. New OUs or containers can be created, but OUs still stay as the recommended container object to create.

3.1.7 Group Policy

The goal of the policy-based administration is to ease the work for an administrator and to assure the control of resources. Administrator typically defines environment for users, computers and objects only once by defining policies and then relies on the system, who will make sure, that those policies are enforced. With Group Policy, administrators can actually control the behavior of every machine, every workstation and every server as well as manage the end user's desktop environment.

Group Policy settings are typically available for computer configuration and user configuration. There are tens of thousands of settings that can be applied: programs available to users, programs to appear on the desktop, screen saver options, desktop backgrounds, timings, power management, options for the Start menu, etc.

Group Policy settings create a Group Policy Object (GPO) that is associated with other objects — typical Active Directory objects such as sites, domains or OUs. Therefore Group Policy can be applied to users, client computers, member servers, domain controllers or any other Microsoft Windows 2000 computers within a forest. More about Group Policy can be found on the Microsoft Developer Network (MSDN) web pages[1] or in any book about Active Directory[2].

3.1.8 Global catalog

Some domain controllers in AD serve as global catalog servers. They contain full replica of all AD objects in their domain and a partial replica of objects from other domains, that are in the same forest. Partial replica means that the global catalog server is aware of the objects and contains only a part of all attributes. Attributes in global catalog are members of the *partial attribute set* (PAS) and can be added to or removed from PAS by modifying the `attributeSchema` for the respective attribute in the schema.

Since global catalog contains every object in the forest[13][2], they are used for the forest-wide search purposes. They can be also used and are necessary for user logon. There always has to be at least one domain controller, that will serve as the global catalog and a server or servers used for this can be chosen manually by an administrator.

Global catalogs are read-only and cannot be updated directly. For the purposes of communication serves the LDAP port 3268 or LDAP/SSL port 3269.

3.1.9 Flexible single master operation (FSMO) roles

Even in AD there are some situations, where only one domain controller should perform certain functions. In that case, AD assigns the role to one selected domain controller to serve as a master for such a function. In AD there are five such functions that need this special care. A server for the function is known as the Flexible Single Master Operator (FSMO) role owner and a server typically holds more than one of these roles. There are three domain-wide FSMOs and two forest-wide FSMOs:

Schema master (forest-wide)

Schema master server is the domain controller allowed to make changes to the schema. No other domain controller is allowed to make these updates to schema, except the one holding the role of Schema master. Changes to schema are generally planned well in advance, therefore it is used very rarely and only for this purpose.

Domain naming master (forest-wide) This is a server that is responsible for the changes to forest-wide namespace. Typical use for this server is for addition and removal of domains. It is also required for the successful renaming or moving of domains within a forest, adding application partitions or changing their replicas. These functions are generally planned well in advance, just as the schema master role functions and therefore used very rarely.

PDC emulator (domain-wide)

Domain controller with this role provides backwards compatibility for Windows NT and acts as the primary domain controller (PDC)[2]. The role is important, because

[1] MSDN web page for Group Policy: `https://msdn.microsoft.com/en-us/library/windows/desktop/aa373783(v=vs.85).aspx`

the server has many different additional purposes. It maintains the latest password for any account and all DCs immediately forward password changes to the PDC, which in return helps to support PDC chaining functions used in authentication chaining. It is also the main server used in Group Policy management and lowers the possibility of changing the policy on several domain controllers by several administrators at the same time. PDC is also used as the master time server within every domain. It also handles external trusts, the DFS consistency check and authorizes domain controller cloning operations.

RID master (domain-wide)

Relative identifier (RID) master is a server role that deals with the domain security. Every security principal in a domain has a security identifier (SID), that consists of several parts and the RID is among them. SID is typically used for the unique identification of objects for security permission[2].

SID is unique across the whole domain. RID master therefore serves the role of an object that allocates pools of unique identifiers to domain controllers, that are used for object creation. When a new domain controller is added to the domain, a predefined number of values from the RID pool is allocated for its own use. New DC then uses these values to create SID that will uniquely identify an object within the domain. This way RID master makes sure that every SID in the domain is unique and has a unique RID value. The pool of values assigned to a DC is decreased over time and when the number of values available reaches the predefined threshold, the DC will ask the RID master for another set of unique values.

Infrastructure master (domain-wide)

The Infrastructure master role is one of the important roles in AD. It is used to maintain references to objects that are not created in the same domain as the source of reference[2]. When referencing an object from another domain, the Infrastructure master of the domain maintains the references on those remote objects (phantoms). Phantoms are not manageable or even visible from the reference source domain and are a construct that serves to maintain the consistency. This is used typically for the cross-domain group membership. Infrastructure master is designed to continually maintain all references to objects, whenever object is renamed, moved or changed in any other way, consistency has to be assured. The reference is represented by the GUID, the SID (for security principals) and the DN of the object.

The role of the Infrastructure master has its dependencies and rules and as a result of those, this role cannot be on the same server as the Global Catalog. This applies to a specific setup with multi-domain forests, where not all DCs are a Global Catalog servers.

3.2 Physical structure

This section describes the physical structure of Active Directory. The main parts of the physical structure of Active Directory are sites, site links, bridgehead server and site link bridges.

3.2.1 Sites

A site in Active Directory is a physical grouping of computers. A site generally refers to a certain geographic location in which all computers are connected together and are located in one or more well-connected AD subnets.

In the real life, a site can refer not only to one specific geographic location, but can span several locations. The main reason for this is that AD sees sites in terms of connectivity. Sites are expected to be based on well-connected TCP/IP networks, where the term well-connected is the most important one. AD assumes well-connected networks to have fast, reliable, abundant and inexpensive bandwidth[13]. This is very important, because once a site is established, AD expects network bandwidth to have these qualities at all times and uses this knowledge in the replication flow and resource location boundaries definition.

The main reason for sites to exist in AD is for traffic (user logon, query or service) and replication purposes, where sites basically define the boundaries. An extensive bandwidth needed for these purposes is not yet able to be assured over long distances. If the traffic between different geographical locations would be assured to be as fast as the traffic in LAN, there would not be any need for sites. Sites and site links assure, that the traffic between different sites is kept at minimum.

3.2.2 Site links

An AD assumes that each site is connected to at least one another site through some kind of WAN connection. AD does not assume, that WAN connections are fast and reliable, but expects for such a connection to exist. As the database replication across a number of sites has to be preserved in some way, site links were introduced. Site link is a way for an administrator to create information about the connection between sites and to set the preferences for this connection. This additional information is necessary for an AD to determine how to control replication data.

There are three items that help the AD determine how replication should be handled over the site link[13]:

- **Cost** is considered to be the logical cost of the link in which it defines the bandwidth usage.
- **Frequency** is a parameter used to determine how often should replication occur over the link (using pull requests to determine replication changes).
- **Schedule** offers the possibility to define the time interval when the site link can be used.

3.2.3 Bridgehead server

Once an Active Directory site is created, the role of bridgehead server is assigned automatically to one of the domain controllers in the new site. The bridgehead server takes care of the communication with the world outside the current site. It receives and sends replication data for a site and is the only entry and exit point for the communication with other sites. A bridgehead server receives data and redistributes this data to other domain controllers in the site so that all domain controllers in the same site have data at their disposal.

The bridgehead server is selected automatically among domain controllers within a site. This can be manually changed and another domain controller can be chosen as the

bridgehead server. In case of failure of bridgehead server, another domain controller is again automatically selected for this role[13]. This can be also manually configured and an administrator can choose additional bridgehead servers. In case of a failure, next bridgehead server in the pool of preselected backup servers can be chosen.

3.2.4 Site Link Bridge

Many sites in Active Directory and site links between them can create a complicated network of sites. Site Link Bridge uses site links to ease the communication for different sites. Active Directory automatically bridges site links together in case they are using the same protocol (such as IP). Therefore site link bridges are made to create transitive connections to other sites. The cost of such transitive site link bridge is automatically determined by addition of costs of all site links that create this logical link. This way the physical site link has always lower cost as the newly created logical link and the correct traffic routing is assured[13].

3.3 Trusts

Communication between domains occurs through trusts. Trust relationships must be established to enable remote domain resource access. Trust relationships in Active Directory enable a user in a domain to access resources in another domain. Six types of trusts exist in Active Directory, from which two are implicitly created[6].

Parent-child trust (default)

By default, for every newly created child domain in an existing domain tree, parent-child trust is created. This way all authentication requests from child domain flow upward through the parent to the trusting domain. This trust is transitive and two-way, therefore all domains in the domain tree trust each other implicitly, see an example in the Figure 3.2.

Tree-root (default)

By default, when adding a new tree to the existing forest, a new tree-root trust is created. These trusts are created automatically, are transitive and two-way. The forest therefore holds the default boundaries of trust, even though additional trusts can be added to extend the trust relationships to other forests. See an example in the Figure 3.2.

External trust

External trust type is used to establish a trust relationship with domains outside the current forest. They provide access to resources that are located on a Windows NT 4.0 domain or a domain that is located in a separate forest that is not joined by a forest trust. This type of trust can be either one-way or two-way and is always nontransitive. After creating a trust, a foreign security principal object is created in the internal domain to represent each security principal from the trusted external domain and these principals can afterward become members of local domain groups in the internal domain.

Realm trust

Realm trust is a special kind of trust and can be established between any non-Windows Kerberos v5 realm and Windows Server domain. This can be one-way or two-way relationship, either transitive or nontransitive. Relationships can switch from transitive to nontransitive and back over time. It can be used to allow cross-platform usability with UNIX-based or MIT-based implementations of Kerberos v5.

Forest trust

Forest trust is always created between two forest root domains only. This way every domain in one forest has a one-way or two-way transitive relationship with every other domain in another forest. This kind of trust can be useful in a number of typical situations – merger or acquisition of a company, collaborating businesses or Application Service Providers. This trust can be only created between two forests and cannot be implicitly extended to the third one.

Shortcut trust

Shortcut trusts are one-way or two-way transitive trusts that are typically created between two domains in the same forest. As their name says, they create a shortcut relationship between two domains. This is regularly used by administrators to reduce time to authenticate one user in another domain, where authentication otherwise takes a long time to perform. This kind of relationship makes sense in cases, where trust path between domain trees in large multi-domain forests takes too much time and the creation of such shortcut trust is beneficial.

Chapter 4

Active Directory Solutions

Even though Active Directory is created by Microsoft, it is currently not the only operating system that supports Active Directory. Active Directory can be nowadays used also in a Linux-based operation systems; thanks to Samba4 that can act almost as a fully compatible Active Directory service. Windows Server, Samba4 and Zentyal server are discussed as the potential candidates to use as an Active Directory server and the limitations of these servers are mentioned, as they can play a big role in the selection of the server, which can potentially lead to incompatibilities between controllers.

4.1 Windows Server

Windows Server is a series of server operating systems developed by Microsoft Corporation. Windows servers are more powerful versions of desktop operating systems and designed to handle enterprise-level management, corporate networking, internet or intranet hosting, data storage, enterprise-scale messaging and similar functions in a more efficient way.

The first server version of operating system was Windows NT 3.1, but the first operating system from the Windows Server brand to be released was Windows Server 2003. Windows 2000 Server was the first server version to feature things like Active Directory, DNS Server, DHCP Server, Group Policy and many other features that are extensively used today. Since Windows Server 2003, more versions were released over the years including Windows Server 2003 R2, 2008, 2008 R2, 2012 and 2012 R2, where new features were added with every release and the functionality was extended (more details can be found on the MSDN web page[1]). Latest version of server operating system is Windows Server 2012 R2 and focuses on cloud computing.

All versions of Windows Server support Active Directory as a collection of services. Figure 4.1 shows the overview of these services along with the basic functionality of every service. Every Active Directory service is used to work with different objects and manage different aspects of Active Directory:

- **AD Domain Services (AD DS)**: Manages Users, Computers and Policies. It is both the directory information source and the service that makes the information available and usable. This includes information about servers, users, clients, network devices, applications and email servers. AD DS provides manageability, security and

[1]MSDN web page for Windows Server: `https://msdn.microsoft.com/en-us/library/dn636873(v=vs.85).aspx`

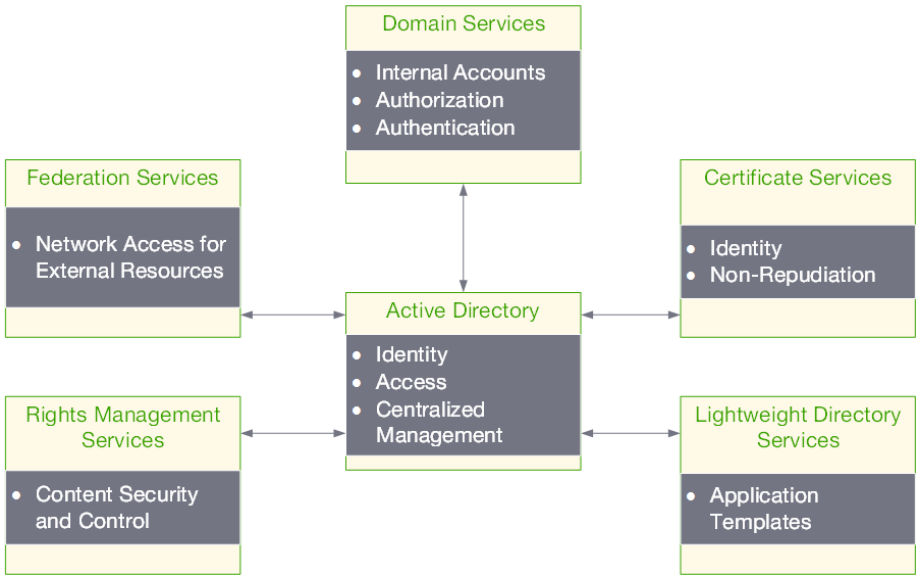


Figure 4.1: Overview of Active Directory Services

interoperability in a centralized, much more easily managed way. It stores and manages information about network resources and allows for centralized management and delegation of that centralized management. It also provides support for directory-enabled applications such as Microsoft Exchange Server.

- **AD Certificate Services (AD CS)**: Handles Service, Client, Server and User identification. AD CS is the Microsoft implementation of Public Key Infrastructure (PKI). PKI is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates. It provides customizable services for issuing and managing digital certificates using a number of tools including Certification Authorities, CA Web Enrollment or Certificate Enrollment Web Service. More information can be found on the Microsoft pages[2].

- **AD Federation Services (AD FS)**: Manages resource access across traditional boundaries. AD FS is a software component that facilitates the cross-organizational access of systems and applications. This service lets IT administrators either share resources out to the world or allow users to access resources from another organization. The AD FS server role provides simplified, secured identity federation and Web single sign-on (SSO) capabilities. It enables the creation of trust relationships between two organizations, provides access to applications between organizations and provides SSO between two different directories for Web-based applications.

- **AD Rights Management Services (AD RMS)**: Maintains security of data. AD RMS is an information protection technology that works with applications to safeguard digital information. It allows an author to create a content such as a word document or an e-mail and it allows him to protect this content using AD RMS aware applications. It lets the author set permissions for tasks which can be performed on the document such as read, print, copy and forward. It also allows individuals and

[2]Microsoft technet web page for AD CS: `https://technet.microsoft.com/en-us/library/hh831740.aspx`

administrators to specify access permissions to documents, workbooks, and presentations and prevents sensitive information from being printed, forwarded, or copied by unauthorized personnel. It makes sure that access and usage restrictions are enforced no matter where the information is located.

- **AD Lightweight Directory Services (AD LDS)**: It copies the structure of AD DS. AD LDS is a hierarchical file-based directory store. It is both the directory information source and the service that makes the information available and usable. This includes information about users, network devices, applications, servers, e-mail servers. AD LDS provides manageability, security and interoperability in the centralized, much more easily manageable way. It uses Lightweight Directory Access Protocol (LDAP) that provides flexible support for directory-enabled applications, without dependencies and domain-related restrictions of AD DS. It offers directory services for directory-enabled applications without incurring the overhead of domains and forests. There is also no requirement for a single schema throughout a forest, which means that AD LDS is customizable.

4.2 Samba4 + AD module

Samba is an Open Source/Free Software suite that provides seamless file and print services to SMB/CIFS clients[3]. Samba is freely available, unlike other SMB/CIFS implementations, and allows for interoperability between Linux/Unix servers and Windows-based clients.

Samba is a software package that gives network administrators flexibility and freedom in terms of setup, configuration, and choice of systems and equipment. Because of all that it offers, Samba has grown in popularity, and continues to do so, every year since its release in 1992.

As a result of ten years' work, the Samba Team introduced the first compatible free software implementation of Microsoft's Active Directory protocols. Samba 4.0[11] comprises an LDAP directory server, Heimdal Kerberos authentication server, a secure Dynamic DNS server, and implementations of all necessary remote procedure calls for Active Directory. Therefore Samba 4.0 has everything necessary to serve as an AD compatible DC and can be even used by all versions of Microsoft Windows clients available. Samba 4.0 Server can also serve as an additional DC in AD and can be easily added to an existing AD domain. As a fully compatible AD DC, other Microsoft AD domain controllers can easily join already existing domain with Samba 4.0 as an original DC. This shows true peer-to-peer interoperability of the Microsoft and Samba implementations of the AD protocols. Samba 4.0 ships with an improved winbind, which allows Samba 4.0 file servers to easily integrate into existing Active Directory services as member servers. Both Microsoft Active Directory and Samba 4.0 Active Directory Compatible servers are supported.

In addition, Samba 4.0 AD supports standard features of AD such as Group Policy, Roaming Profiles, Windows Administration tools and can integrate with Microsoft Exchange compatible services. It is suitable for low-power and embedded applications, with the possibility to the scale of large clusters and provides a Python programming interface and administration toolkit.

The key in the development of the Samba Active Directory functionality was the documentation and interoperability labs that Microsoft provided and Microsoft continues with

[3]Main web page of Samba: `https://www.samba.org/`

the commitment to support for interoperability across these platforms. Samba Active Directory was also tested with the help of Microsoft engineers and can be considered as a fully compatible AD domain server comparable to the Microsoft AD servers, even though it still possesses some limitations:

- Inability to join a Microsoft Windows 2012/2012R Server as a Domain Controller to a Samba Active Directory.
- Missing 2012/2012R2 AD schema support in Samba (Samba only with AD schema version 47).
- Samba does not support multiple domains, therefore only one domain in the forest is supported.
- Cross-forest trusts are not supported (Samba can be trusted, but cannot trust).

Samba 4.0 is naturally famous not only for an addition of Active Directory to it. It consists of a lot of valuable improvements that are out of the scope of this thesis and more information about it can be found on the project's home page[4] or their wiki page[5].

4.3 Zentyal server

Zentyal started as an open source project, providing companies and organizations with a complete Linux-based, full-featured server. Zentyal is now a company based in Zaragoza, Spain offering two main products – Zentyal Cloud and Zentyal Server.

Zentyal Server is focused on the implementation of Microsoft Exchange and Active Directory protocols on Linux and became one of the most popular business-led Open Source projects in the world released under the GNU General Public License[6]. Zentyal offers two types of Zentyal Server – community and commercial edition. Commercial versions of Zentyal run on top of Ubuntu[7] server edition, always on LTS (Long Term Support), whereas community version may be based on standart Ubuntu server editions.

Zentyal Server connects two main functionalities – mail and directory server. It also incorporates DNS Server, DHCP Server, NTP Server, Certification Authority and VPN Server & Client. On top of these services, it offers some basic networking management, such as configuration of static and DHCP interfaces, Objects & Services, packet filtering and port forwarding. The main features are mail and directory services:

- **Mail service**: Zentyal offers native compatibility with Microsoft Exchange Server Protocols and support for Microsoft Outlook 2007 and 2010. It also provides native compatibility with Microsoft Active Directory and makes use of multiple virtual mail domains. Mail service provides basic operations with email, calendars, contacts, offers Webmail functionality and synchronization with mobile devices (ActiveSync support). On top of this email functionality, it introduces also antivirus, antispam and supports Extension and MIME type filters.

[4]Home page of Samba version 4: `https://www.samba.org/samba/news/releases/4.0.0.html`
[5]Wiki page fro Samba: `https://wiki.samba.org/index.php/Main_Page`
[6]GNU General Public License: `http://www.gnu.org/licenses/gpl.html`
[7]Main page of Ubuntu: `http://www.ubuntu.com/`

- **Domain & Directory service**: Zentyal offers central domain directory management, including operations with objects such as OUs, users, security groups, distribution lists and contacts. It supports operating systems Windows XP, Windows Vista, Windows 7 and Windows 8. Other functionalities offered are file sharing in Windows environments (CIFS), access control lists and antivirus with integrated quarantine for file server.

Zentyal integrates Samba4 as a Directory Service, implementing Windows domain controller functionality and printer/file sharing. The domain concept in Zentyal is based on the Microsoft Active Directory implementation and depends fully on Samba4 capabilities. It also provides file sharing possibilities using SMB/CIFS protocol to maintain compatibility with Microsoft clients. Integrating Samba4 technologies, Zentyal is able to act as a standalone domain server or to become an additional controller of an existing domain, joining a Windows Server or any Samba4-based controller. Zentyal Server can also take any of the FSMO roles. Integration of Samba4 however comes with some known limitations:

- Samba does not support multiple domains, therefore only one domain in the forest is supported.
- Hostname cannot match NETBIOS name.
- Functional Domain level of the forest and the domain has to be min. 2003R2, but max. 2012.
- Trust relationships between domains and forests are not supported.
- GPOs cannot be synced from Zentyal to Windows servers(only Zentyl to Zentyal and Windows to Zentyal).

Chapter 5

Management applications

No matter the size of an organization, Active Directory management is a necessity. Quite a lot of applications were developed to help AD administrators with their job, some of which are simple and easy to use, other are complicated and offer a lot of functionality. This section will have a look at the basic tools given by Microsoft along with several of the best commercial applications on the market.

5.1 Windows remote server administration tools

Remote Server Administration Tools (RSAT) lets IT administrators manage Windows Server from a remote computer running the Windows operation system. RSAT is available for almost any version of Windows and provides basic set of application that can be used by administrators to manage an Active Directory.

5.1.1 Active Directory Users and Computers

Active Directory Users and Computers is a Microsoft Management Console (MMC) snap-in included in the RSAT. It is used to administer and publish information in the directory and serves as the primary application used for the management of user, computer and group objects. It provides basic CRUD operations on objects in Active Directory and allows administrators to perform some basic management tasks, such as:

- **User account** – adding new users, changing passwords, granting rights, etc.
- **Group** – creating security groups, changing rights, setting membership, etc.
- **Computer account** – creating a computer, changing properties, etc.
- **Domain** – connecting to a specific domain, managing domain properties, etc.
- **Organizational Units** – creating a new container or OU, managing properties, etc.

This tool is runnable on Windows operation system only and is free of charge, but can be used to manage any Active Directory server – compatible with Microsoft and Samba4.

Even though this tool can be very handy, it is unusable for managing a larger environment. It allows for a management of only a few parts of Active Directory and working with directory containing a larger amount of objects such as users or groups can take a long time and can be uncomfortable.

5.1.2 Active Directory Domains and Trusts

Active Directory Domains and Trusts is a Microsoft Management Console (MMC) snap-in included in the RSAT. This tool does not offer the same level of functionality as the Active Directory Users and Computers and is rather focused on the tasks that can be performed globally on domains. It is used to administer domain trust relationships, domain and forest functional levels, and user principal name (UPN) suffixes. It allows administrators to perform some basic management tasks, such as:

- **Domain trust** – create a trust, verify a trust, create a shortcut, realm or external trust, etc.
- **Forest trust** – create a forest trust, manage routing for specific name suffixes, etc.
- **Domain functional level** – select a mode, etc.
- **Forest functional level** – select a mode, etc.
- **UPN** – create a UPN suffix, etc.

This tool is runnable on Windows operation system only and is free of charge, but can be used to manage any Active Directory server – compatible with Microsoft and Samba4.

Even though this tool can be very handy, it is unusable for managing a larger environment. It allows for a management of only a few parts of Active Directory and working with directory containing a great number of users, domains, forests can take a long time and can be uncomfortable and challenging.

5.1.3 Active Directory Sites and Services

Active Directory Sites and Services is a Microsoft Management Console (MMC) snap-in included in the RSAT. It is used to administer the replication of directory data among all sites in an Active Directory Domain Services (AD DS) forest. This snap-in also provides a view of the service-specific objects that are published in AD DS. It allows to perform some basic management tasks such as:

- **Server tasks** – enable global catalog, select query policy, check replication topology, etc.
- **Site tasks** – cache universal group membership, create a site or a subnet, create GPO, etc.
- **Site replication tasks** – creating a site link or a site link bridge, adding a connection, etc.

This tools is for the most part only used by organizations with relatively complex Active Directory hierarchies. On a regular basis, this tool is typically used by organizations that have a need of multiple sites, domains, forests, partitions, etc.

This tool is runnable on Windows operation system only and is free of charge, but can be used to manage any Active Directory server – compatible with Microsoft and Samba4.

Even though this tool can be very handy, after some time and with a bigger complexity, it is unusable for managing a larger environment. It allows for a management of only a few parts of Active Directory and working with directory containing a great number of domains, sites, forests can take a long time and can be uncomfortable and challenging.

5.1.4 Active Directory Administrative Center

Active Directory Administrative Center (ADAC) is a more advanced tool used in Active Directory management and is also a part of the RSAT. It provides improvements for the management of large and complex environments. It offers all the functionality of previously mentioned RSAT applications combined together to one tool:

- AD Users and Computers functionality
- AD Domains and Trusts functionality
- AD Sites and Services functionality
- Complex LDAP queries
- Multi-domain management
- Multi-forest management

The application is also very flexible and provides quick panels with most recently used tasks and customizable panels for an administrator to define tasks that are important for him. Even though it is a very complex application and can be used very well on the Windows Server domains, it does not support the management of Samba4. ADAC requires at least one domain controller to run Active Directory Web Services that are not included in the Samba4.

It is Windows PowerShell-based, therefore is runnable only on Windows operation systems. On Linux systems, the application would need a PowerShell interpreter for Linux to run, but unfortunately it is not yet developed to fully support it.

5.2 Samba-tool

Samba-tool is the main Samba administration tool developed by Samba and for Samba. It is runnable only on the Linux based operation system and provides only a basic administration capabilities. Mainly it is intended to be used by administrators for a simple and basic administration tasks of Samba and Samba Active Directory[1]:

- **Users** – create a new user, disable or enable user, reset password, etc.
- **Groups** – create a new group, add members, etc.
- **Domains** – show basic info, raise function level, set password, etc.
- **Sites** – create and remove a site

Samba-tool does not support all the tasks available from RSAT and therefore is unsuitable to perform even the most basic tasks such as creating a new user. Also it is runnable only on the Linux based operation system.

[1]Manual pages of Samba-tool: `https://www.samba.org/samba/docs/man/manpages/samba-tool.8.html`

5.3 Adaxes

Softerra Adaxes[2] is a comprehensive solution for management, administration and monitoring of Active Directory. It enables an administrator to automate and secure user provisioning in Active Directory environments. Adaxes provides Role-based security, rules-based automation, approval-based workflow and AD Reports. Adaxes offers functionality of all previously mentioned applications and more. It can be used in small, medium or large sized businesses and can be customized for different management roles. It also provides additional functionalities such as Web interface, User Self-Service and Self Password Reset. They also offer an Annual Maintenance and Support.

Adaxes can be used to manage both Microsoft Server and Samba4 and can be seen on every operational system thanks to having a web interface. Application also uses PowerShell for specific automation tasks. But this application is not completely free of charge. It only offers a free trial and at the time the minimal price for a full application to manage up to 100 users is 1600 dollars plus minimal costs for maintenance and support 480 dollars.

5.4 ADManager Plus

ADManager Plus[3] is a simple, easy to use Active Directory Management and Reporting Solution. It helps Active Directory administrators and desk technicians with their everyday tasks. It is presented as a centralized and intuitive web-based UI management tool, but offers also a mobile applications to perform important user management tasks right on a mobile device. ADManager Plus offers functionality of all RSAT tools combined in one tool and can be used in small, medium or large sized businesses.

Except from the trivial management tasks, it also handles a variety of complex tasts like Bulk Management of User accounts and other AD objects, delegation of role-based access to help desk technicians and generation of an exhaustive list of AD Reports. It can reduce repetitive and complex tasks, automate routine activities for administrators and support object management in bulk.

ADManager Plus can be used to manage both Microsoft Server and Samba4. But this application is not free of charge. It comes with two editions. The standard edition for 1 domain and restricted to 100 domain objects is free of charge. Other editions are paid and the price starts at 495 dollars.

[2]Home page of Adaxes: `http://www.adaxes.com/`

[3]Home page of ADManager Plus: `https://www.manageengine.com/products/ad-manager/`

Chapter 6

ADaMAN – a new application design

Management of Active Directory sometimes comes hand in hand with quite complicated procedures and applications for AD management are rather complicated and diverse. There is a lot of tools available, some quite powerful, but not all tools offer a complete management of all aspects of Active Directory. Such solutions are difficult to administer, because for many tasks administrators have to use several tools in order to accomplish a single management task. It would be beneficial to have a tool that simplifies management of Active Directory and could be used by administrators on a daily basis. Usually, companies use a combination of MS utility tools that are a part of MS Administrative tools, specifically Active Directory Users and Computers, Active Directory Domains and Trusts and Active Directory Sites and Services. These are just the basic three applications used in AD management, but their functionality is not always sufficient. Therefore, many administrators look for solutions from other companies, such as Cjwdev Limited[1] with their tools *AD Photo Edit* and *AD Permissions Reporter*. Previously mentioned applications alone are five different tools that can be used by administrators.

These difficulties with management of Active Directory have been an inspiration to create a list of capabilities that an ideal application for AD management should have:

- **Users and Groups management**: The application should provide a simple way to manage User and Group objects in Active Directory. It should be able to cover the needed functionality of Active Directory Users and Computers.
- **Profile photo management**: A lot of services within a company are using a profile photo for their users as a simple and quick way to recognize a person. Since AD is often used as a provider of information about a user to these services, it is only logical to have their profile photos saved within AD. The application should offer a way to edit these photos and perform simple operations on them before saving them to AD.
- **Customized creation process**: A creation of new users (or other objects) in the company domain may come with some specific setting needs. As an example, many company administrators need to create a new user, change a number of attributes that are specific to the company environment and add a profile photo. All this is typically done with different tools and since making all these changes can be quite

[1]Home page of Cjwdev: `http://cjwdev.co.uk/Software.html`

time consuming, the new tool should be prepared to do all this in more simpler, pleasant and straight-forward way for the administrators.

- **Attribute Editor**: An editor tool for every single attribute of any object is available in all Microsoft AD management tools and therefore must be also included in the new application. It provides a way to select any object in AD and manually add or remove a value of any attribute of this object.
- **Modularity**: The new application should be prepared to serve administrators and help them to stop using many different small tools that are in use today. Therefore it should combine the functionality of many different small tools and provide a way to include any new functionality that needs to be added in the future.
- **Support any AD solution**: Despite many companies using Windows Server as their AD domain controller, there are still many other companies that are looking for linux-based domain controllers and they usually use Samba4 server. There is also a commercial product based on the implementation of Samba4 called Zentyal Server, that is internally using Samba4. The goal of this application should also be to provide a way to support any implementation of Active Directory.
- **Be free and open-source**: Advanced applications for AD management do not typically offer a free version. Even if they do, they cut the functionality possibilities to a minimum. This new application should be free to use and the source code should be available to anyone to incorporate new functionalities and customize the application.

A new application concept has risen from difficulties with the management of users, their rights and the general Active Directory related problems. The capabilities mentioned above show several points that are crucial for an AD management application to satisfy and these needs serve as a motivation to create this new management application — ADaMAN.

6.1 Assessment of required functionality

This section provides a high level view on the main functionality of the tool. The tool is designed to be easily extensible so that the new functions can be integrated in a simple way. Figure 6.1 shows the basic functionality of the application (normal coloring) together with some additional functionality that may be useful, however, is out of the scope of this thesis (transparent coloring).

Functionality on the Figure 6.1 is divided into four main areas: Logical Infrastructure Management, Object Management, Physical Infrastructure Management and Extensions. First three named areas show mostly the functionality that will be a part of this work. The last one offers a set of sub-areas that can serve as an inspiration for the future work. The explanation of every sub-area is given in the following parts.

Users & Profiles

Correct user and profile settings play one of the most important roles in the company resource management. The flow of employees must be managed on the professional level and this tool should offer administrators the capability of doing so. Administrator should have the power to perform CRUD (create/read/update/delete) tasks on all the user accounts. Part of these tasks is enable, disable, lock, unlock or reset the

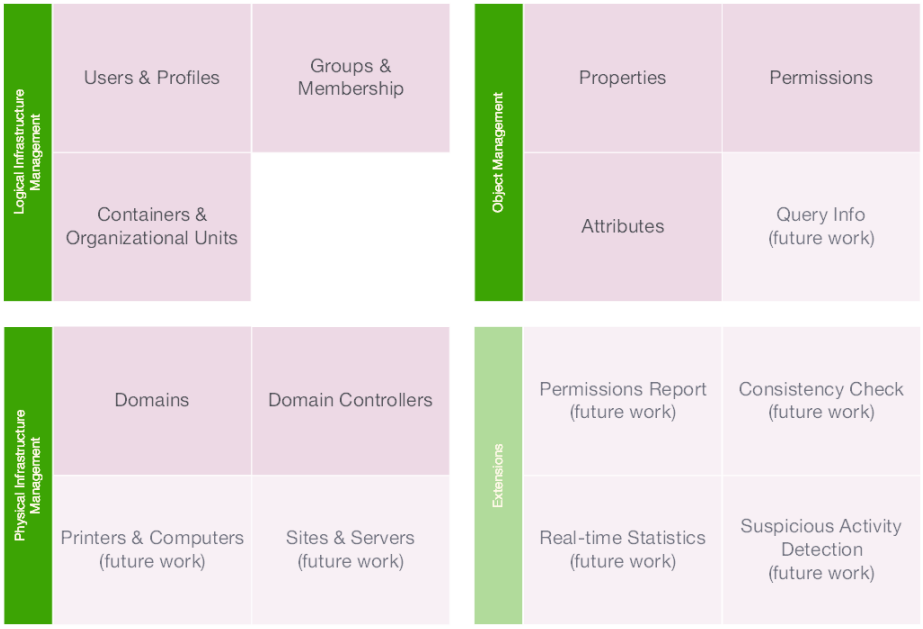


Figure 6.1: Overview of the functionality with possible future work.

user account. The capability to manage user profile pictures should be added along with the correct picture transformation tasks such as rotation, compressing, resizing or cropping.

Groups & Membership

Traditionally users are divided into groups, where a group has special permissions to access special company resources. Administrator should have a possibility to easily do CRUD tasks on the security groups and define the membership of a user to the group. Groups then offer permissions, that are applied to users or other objects included in the group, such as VPN connection ability, maximum login count or rights to file servers and documents. These permissions are assigned according to the company security policy.

Containers & Organizational Units

All objects in Active Directory are structured with the use of container objects and organizational units. Administrator's responsibility is to structure the objects in a simple and yet effective way. These tasks should be easily performed in a special designed view.

Properties, Permissions, Attributes

Properties, permissions and attributes of any object provide the most detailed look on a single Active Directory object. Administrator should assign all the attributes for an object in the most detailed view and all attributes should be found on a single place.

Domains

The authentication of a user in a domain is the main pre-requisite of every other functionality for this tool. An application user should be able to use any selected

username and password for the connection to an AD DC and authenticate against it. User should be also able to change the connection to different domains while running the program.

Domain Controllers

Working with domain controllers is an important management functionality. User with proper rights should be able to change the current domain controller settings on the particular selected domain controller. User should be able to perform common DC tasks such as setting the domain naming operations master and enabling or disabling a global catalog server.

Additional functionality (future work):

The functionality described below may be useful for some users, however, as most users barely need it, it is out of the scope of this work.

- **Printers & Computers**: Printers and Computers management is one of the additional functionalities, that is designated to change the properties of such objects. It should offer possibilities to enable or disable a computer, reset a computer, delegate control or set the permissions for the use of a printer.
- **Sites & Servers**: Sites and Servers application view should be designed in a way to allow the experience of easy resource sharing and moving. Raising the functional level, changing the forest, selecting operations master or adding UPN suffixes, sites, sub-sites and services should be an easy task to do.
- **Query Info**: Query info service is a designated service for special purposed, pre-defined views on the Active Directory object. Application should allow to define special queries and perform search according to any combination of given parameters.
- **Permissions Report**: Often when migrating from one system to another or when expanding your business, the extensive check on the permissions is needed. The application should be able to perform the permissions check on the domain and show possible security holes and problems with permissions.
- **Consistency Check**: The user should be able to check the consistency of the domain and identify missing settings that are not set according to the company's security policy.
- **Real-time Statistics**: The application will provide real-time statistics about the usage of the company domain and provide information about user activities. This should include the number of users that are currently logged in or real-time usage of the network bandwidth.
- **Suspicious Activity Detection**: The application can also include a module designed for auditing. It could monitor the user activity and notify administrator about any suspicious activity that is currently happening. It can also notify about the extensive use of the resource of the company.

6.2 Usage

This application will be used by two types of users:

- **User with administrator rights** will be the main type of a user, for whom is this application developed. This user will have full rights to manage all objects and their attributes in the domain.
- **User without administrator rights** will be the secondary user type. He will have limited rights for objects and attributes. He can access the objects and make changes that he is authorized to do.

The difference between these types of users and their rights to manage objects will be saved in Active Directory as a permission and the application will reflect these settings in the application automatically.

The specific information a user needs for the application to serve its purpose is:

- **User profile** created in the Active Directory domain with all rights set properly,
- **Username and password** for the user to connect to the domain.

Very important for the application is the presence and cooperation with the Active Directory Domain Controller, therefore there is a need for an AD DC to be running and all the required conditions to communicate with it are being met. This application has to be also able to communicate with any type of AD server, namely with Microsoft Server AD and Zentyal Samba4 AD.

6.3 Phases of development

As a result of the basic overview of the application functionality, a rather comprehensive list of possible application tasks was created in order to specify the functionality in more detail. This list serves as the main information resource for the application design. List can be found as a set of tables in the Appendix A.

Development of the application will be divided into four main phases according to the importance of individual tasks, where finishing functionality tasks from one phase is usually required for the tasks implemented in the next phase.

Phase 1

In the first phase, tasks necessary to manage the application and to perform the basic operations will be implemented. All these tasks are shown in the Figure 6.2. The most important is the *General* package that contains tasks for changing settings of the application. Since the application is based on modules to support the future extensibility, a system for modules must be created. User will be able to change between different views. Application settings such as connection credentials can also be set.

In this phase, also tasks that are needed for managing Active Directory will be implemented. These tasks are shown in *Domain/Domain Controllers* package and include tasks for establishing the connection to a DC, authentication of the user and loading directory content. Those are prerequisites for almost all the other tasks and therefore need to be implemented first.

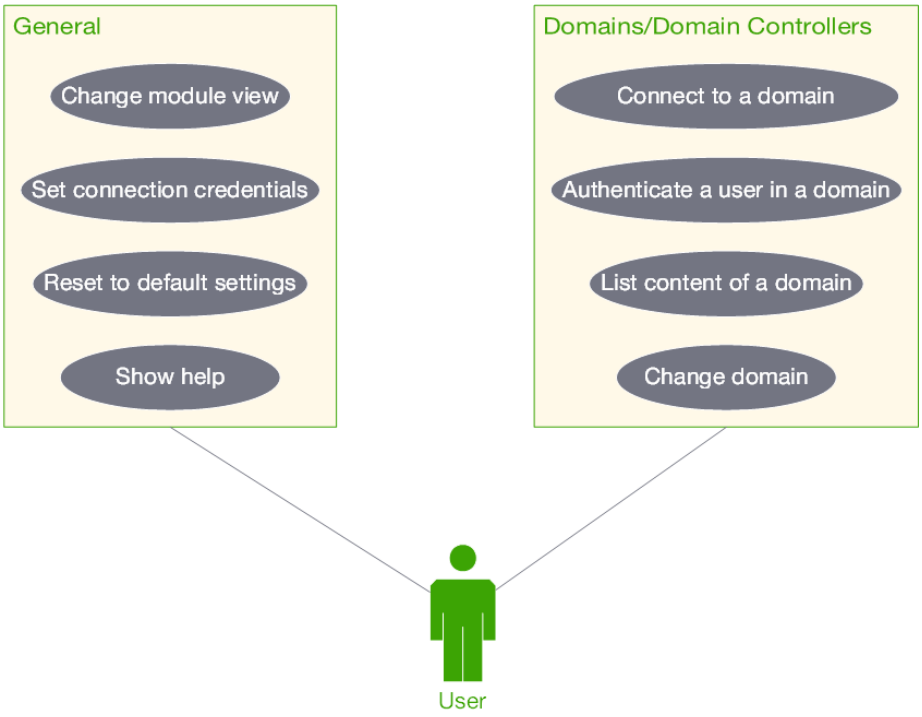


Figure 6.2: Use case diagram of the first phase.

Phase 2

After the successful confirmation of the usability of tasks from the Phase 1, the functionality needs to be extended. The next logical step is to show all the attributes that a selected object has with the possibility to easily change the attributes value. These cases can be seen in the *Object* package on the Figure 6.3. Also two other basic cases are included – moving an object to another organizational unit and deleting a selected object. The next step is the possibility to control and manage OUs. Application settings to create a default creation attributes and their appropriate values can be set. This is set for a new organizational unit and is also used withing the process of creation a new OU.

A user is able to create a new OU with the possibility to edit pre-defined attributes from settings along with the standard creation attributes. Properties of an OU are able to be shown and their values can be changed.

Phase 3

When tasks for the basic AD management are available, the need falls down to the actual management of users. All possible actions for users are shown in the Figure 6.4. Application settings for default creation attributes and their appropriate values can be set. These are later used while creating a new User object. An administrator is able to create a new User object in AD and set its standard attributes along with the predefined ones. Administrator can also edit these attributes while creating a new User object. After the creation, a user is also able to see all User object properties and change their values if needed. Application settings also allow to define the default settings for creating a new profile photo of a user. These include size, width and

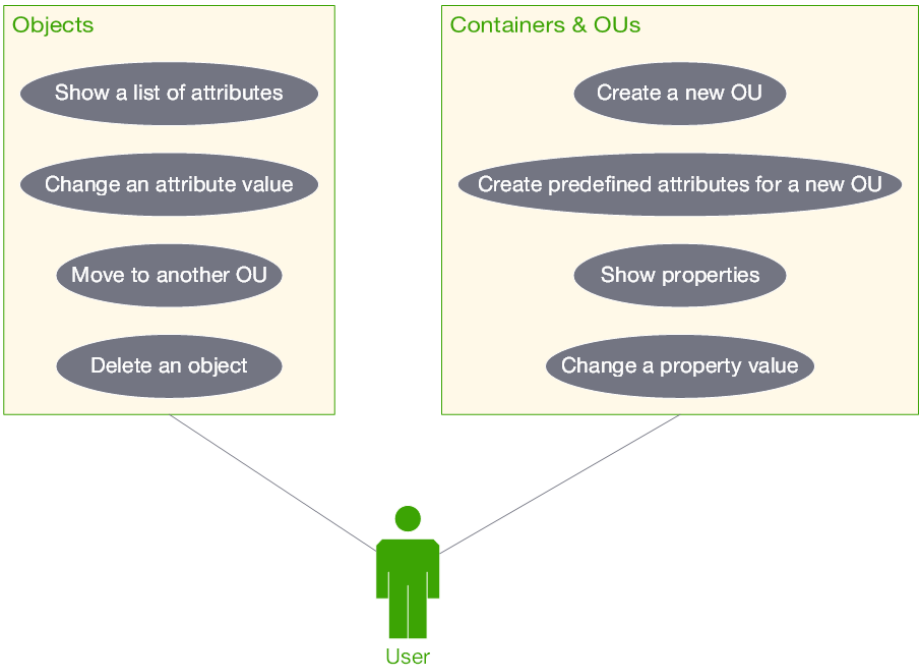


Figure 6.3: Use case diagram of the second phase.

height dimensions and attribute for a photo. The creation of a user profile photo is added to the creation process as well as to the properties view of the selected User object. Main tasks for user objects are added as well – enabling and disabling an account, resetting a password, unlocking a user account.

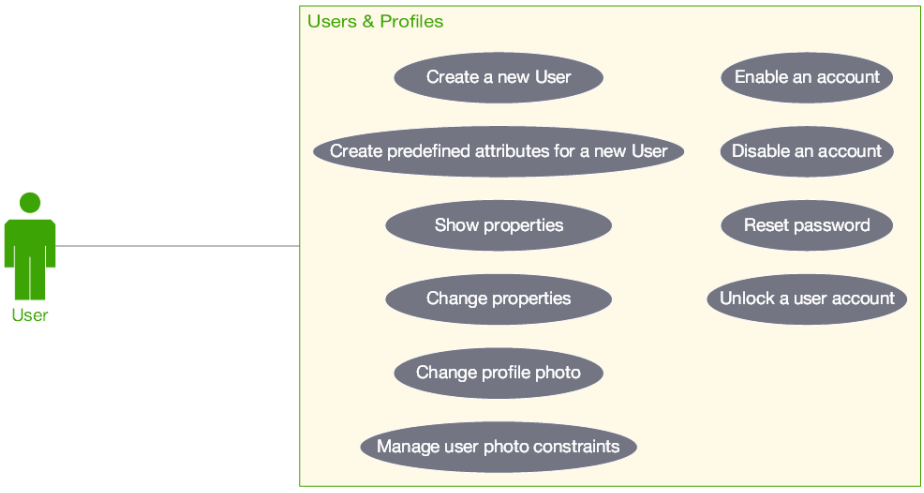


Figure 6.4: Use case diagram of the third phase.

Phase 4

The management of group objects is added in the last phase. Use cases can be found in the Figure 6.5 and include creation of a new group, showing properties of a selected group object and a possibility to change the value of a property. Also the memberships of this group can be changed in the properties view for this object.

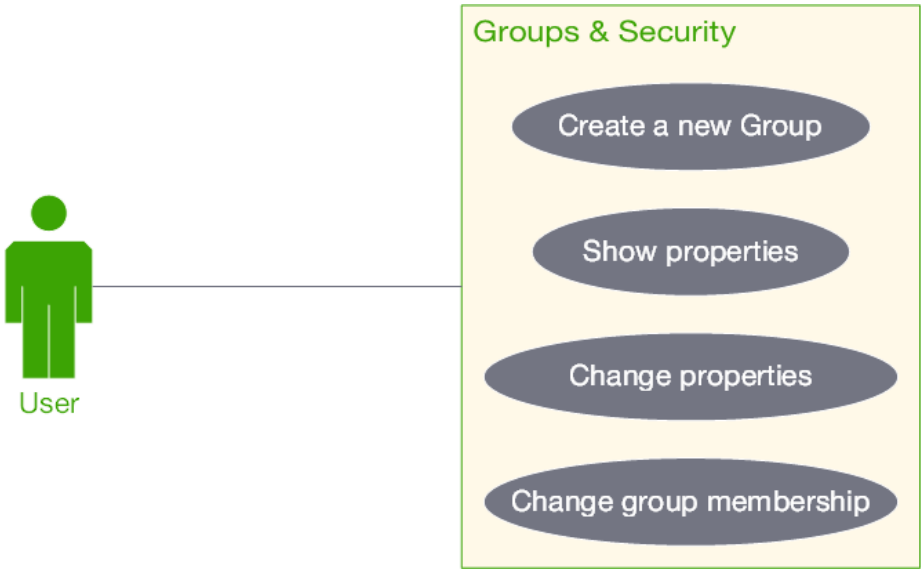


Figure 6.5: Use case diagram of the fourth phase.

6.4 Use case descriptions

This subsection provides a detailed description of a few of the most important use cases that are often needed while managing an Active Directory.

6.5 Graphical users interface design

The design of graphical users interface (GUI) is based on the previously analyzed and prepared functionality. Figure 6.6 shows the mock-up for the main home page of the application. On the left, you can see the panel with different tabs. Every tab provides a specially configured view for a dedicated functionality package. The top bar shows the commonly used functionalities in the appropriate package. On the top right, there is a panel showing the information about the connection. There is also a possibility to re-connect and change domains. The main part of the window is dedicated to the contents of the Active Directory, properties and attributes of objects.

Figure 6.7 shows a mock-up for the user module. The main view is filtered and only user objects and paths to the domain root are shown. The right panel also shows different tabs for properties and attribute management. Also the top action bar is changed and custom actions are added. This provides a simple and fast way to change some properties of a user. The properties tab on the right includes a number of sub-tabs that offer a look on different properties, grouped by some level of similarity.

Figure 6.8 shows a window used for the creation of a new user. This window is tabbed and the figure shows a tab where the change of the profile photo is possible. A user has possibility to upload a new photo and perform basic transformation operations such as rotation or resizing.

Use Case Number	1.1.
Use Case Name	Create a new user
Use Case Description	Creation of a new user object in Active Directory and setting up the standard and optional (pre-defined) attributes for a user.
Primary Actor	User
Preconditions	Active Directory must be running, user has to have rights to create a new user and an AD object (OU) has to be selected.
Trigger	User selects the new user creation.
Basic Flow	1. System shows the location for a new user and possibility to change standard attributes associated with the name of the new user object. 2. User adds necessary information for the user object and confirms. 3. System shows possibilities to add a password, to change password at logon, possibility of changing a password by user, password expiration and to make account disabled after creation. 4. User fills in or chooses required fields and confirms the information. 5. System shows optional attributes and a possibility to edit their values. 6. User edits values for these attributes (if any) and confirms the information. 7. System shows a possibility to add a profile photo to the account. 8. User adds a profile photo and confirms information. 9. User confirms the user creation. 10. System will create a new user in Active Directory with the associated information.

Table 6.1: Use case description for creating a new user.

Use Case Number	1.2.
Use Case Name	Change user properties
Use Case Description	Changing of properties associated with the selected user object.
Primary Actor	User
Preconditions	Active Directory must be running and User has to have rights to change properties of other objects.
Trigger	User selects a user object.
Basic Flow	1. System shows properties view of selected user object. 2. User selects a property he wants to change. 3. System shows a possibility to change the property. 4. User edits a value of the property and confirms changes. 5. System sets the property value in AD and refreshes the view.

Table 6.2: Use case description for changing user's properties.

Use Case Number	1.3.
Use Case Name	Reset password
Use Case Description	User changes the password for a selected user in Active Directory.
Primary Actor	User
Preconditions	Active Directory must be running, admin has to have rights to change user attributes and a user has to be selected.
Trigger	User selects the reset password option.
Basic Flow	1. System shows a possibility to change the password. 2. User will change required information and confirms changes. 3. System updates information about a user in Active Directory and refreshes the view.

Table 6.3: Use case description for resetting user's password.

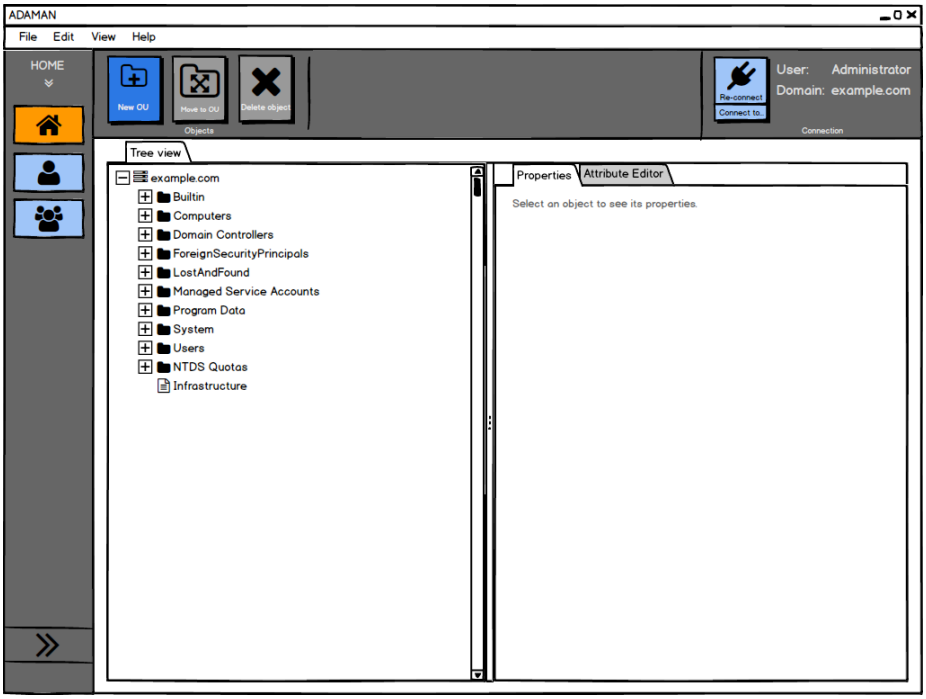


Figure 6.6: GUI mock-up for Home view.

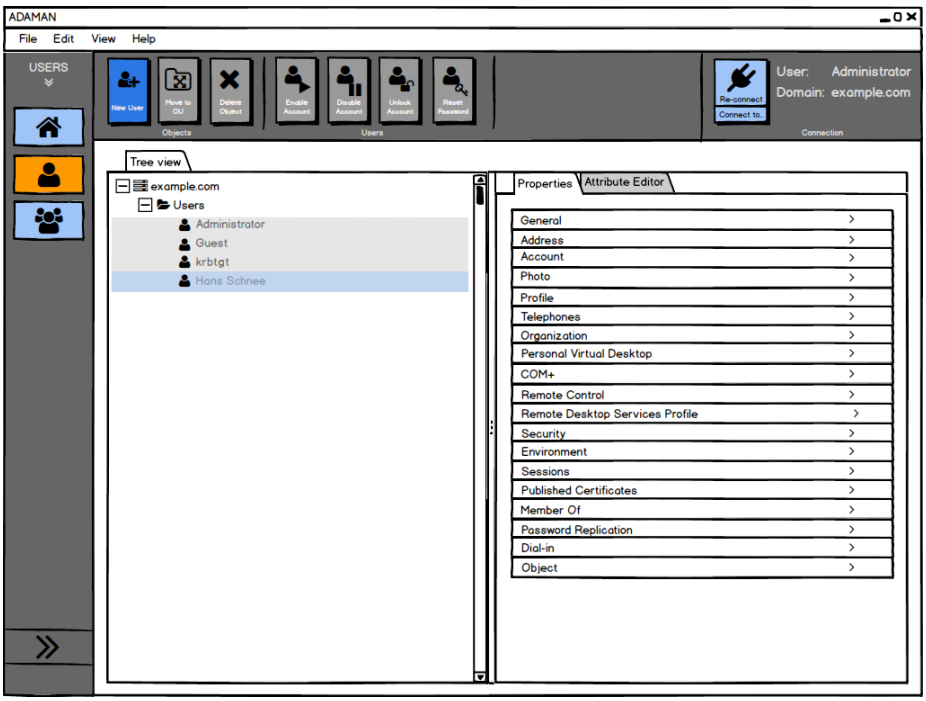


Figure 6.7: GUI mock-up for Users view.

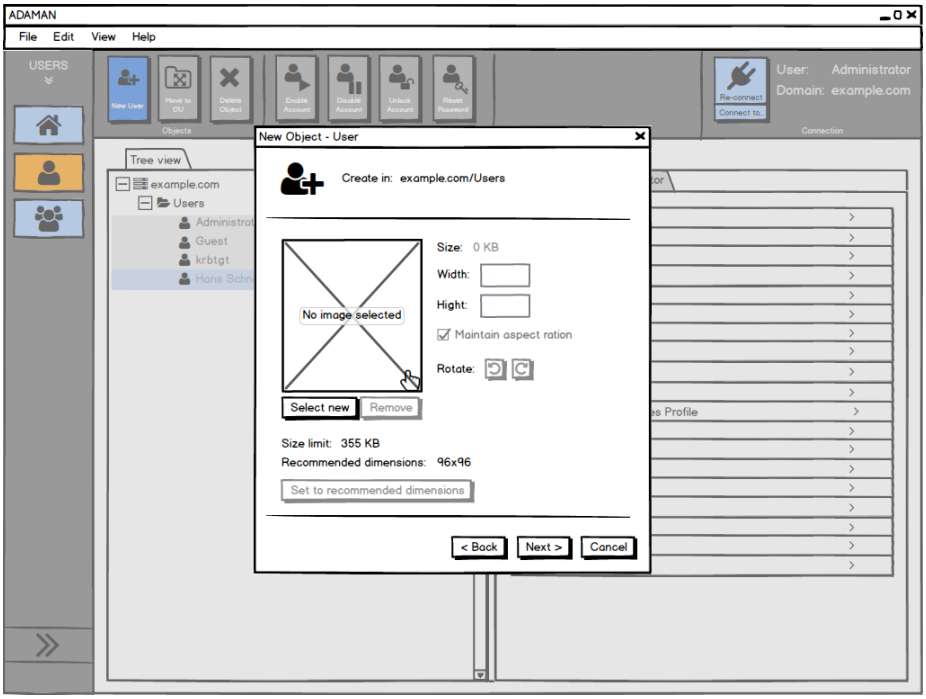


Figure 6.8: GUI mock-up for a new user photo.

6.6 Additional design information

Microsoft Windows was chosen as the main operation system, where the application will be used with the exception of the use of PowerShell. PowerShell is typically prohibited in the company network and the application has to work without using it.

As the .NET framework currently offers a broad number of prepared ways to work with Active Directory, this tool will be developed using this framework and C# programming language. This combination offers a lot of features and optimizations for the use on Windows operating systems.

6.7 Comparison to existing solutions

Design of the new application provides a great opportunity to compare the proposed application functionality and capabilities to the functionalities of existing applications that were mentioned in the previous chapter. Table 6.4 shows a comparison between the new application and all the existing applications in terms of the usage, licenses and the need to use other AD or windows services. Since the requirement binds us not to use powershell, a couple of solutions are flagged red.

The downfall of commercial products might also be their price. Every single commercial product offers only a limited functionality in the free version and that is not sufficient for the proper management usage. Therefore the paid versions are considered and all falls down to the price of those products.

Table 6.5 shows the functionality of different solutions. Active Directory Domains and Trusts and Active Directory Sites and Service are focused on different aspects of management, therefore are flagged red in the table. Also the lack of functionality flags Samba-tool

	Runs on Windows	Runs on Linux	Manages Windows	Manages Samba4	Powershell	AD Web Services (ADWS)	Free version	Paid version
Active Directory Users and Computers	YES		YES	YES			YES	
Active Directory Domains and Trusts	YES		YES	YES			YES	
Active Directory Sites and Services	YES		YES	YES			YES	
Active Directory Administrative Center	YES		YES		Required	Required	YES	
Samba-tool		YES		YES			YES	
Adaxes	YES		YES	YES	YES			From 1600$/100 users
ADManager Plus	YES		YES	YES	YES		For 1 domain and up to 100 domain objects	From 495$/1 domain and 2 help Technicians
ADaMAN	YES	MAYBE	YES	YES			YES	

Table 6.4: General comparison between the new application and other existing management applications.

red and therefore removes it from the consideration for comparison. As mentioned before, Active Directory Administrative Center and both commercial solutions use other services, that we do not want to use in order to support the requirements. The functionality of the final application is therefore comparable to Active Directory Users and Computers. The difference is that ADaMAN offers a possibility to predefine optional attributes and ease the creation of different objects in this way. Also the possibility to add/remove/edit profile photos of user object is added and the modularity of ADaMAN allows one to easily extend it with additional functionality.

	Manage Users	Manage Groups	Manage Computers	Manage Domains	Manage OU	Manage Forests	Manage Trusts	Manage Sites	Manage Subnets	Manage Servers	Manage Site links	Edit profile photos	Define additional creation	Pre-defined creation attribute
Active Directory Users and Computers	YES	YES	YES	YES	YES									
Active Directory Domains and Trusts				YES		YES	YES							
Active Directory Sites and Services								YES	YES	YES	YES			
Active Directory Administrative Center	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES			
Samba-tool	YES	YES		YES				YES						
Adaxes	YES	YES	YES	YES	YES					YES				
ADManager Plus	YES	YES	YES	YES	YES									
ADaMAN	YES	YES		YES	YES							YES	YES	YES

Table 6.5: Functionality comparison between the new application and other existing management applications.

Chapter 7

Implementation

Implementation is a crucial part of this thesis, therefore a great deal of time was allocated to divide the expected functionality into different logical components and to create a clear communication schema between these components.

The design consists of several main logical components that comprise classes and namespaces. Figure 7.1 shows the most important parts of the application from which three serve as the basis: *GUI*, *Business model* and *Data Service*. These components are also a mapping of the Model-View-ViewModel (MVVM) pattern.

MVVM is an architectural pattern that separates the development of the GUI from the development of the business logic. The ViewModel serves as a value converter and is responsible for converting data from the Model to the representation that is suitable for management and presentation in the View. View's display logic is mostly handled by the ViewModel and uses *data binding* as a way to bring data from the code to the View. In data binding, when data in the ViewModel is updated, the View is automatically informed and appropriately changed.

This pattern has proven to be helpful when creating an application. It simplifies the complexity of testing and helps to make the application more manageable by lowering the coupling between components. The application uses MVVM Light Framework that has all necessary base classes for implementing MVVM.

The next important component in the Figure 7.1 is the Active Directory Services component that encloses the actual Active Directory Service. It is treated as a blackbox in terms of communication with the application and therefore is independent of the actual implementation of Active Directory Service. More detailed information on all parts from the Figure 7.1 is given in the subsequent sections.

7.1 Basic components

This section describes the basic components required by other components to function properly. They play different supporting roles.

Resources class is used to group all used resources in the application. It is globally accessible and defines text in a specific language. It is used to easily incorporate other languages than the default one by creating a copy of the resource file and correct naming of the resource file.

Settings class is similar to Resources class. It supports the default settings of the application and maintaining the state of it even when the application is closed and reopened again.

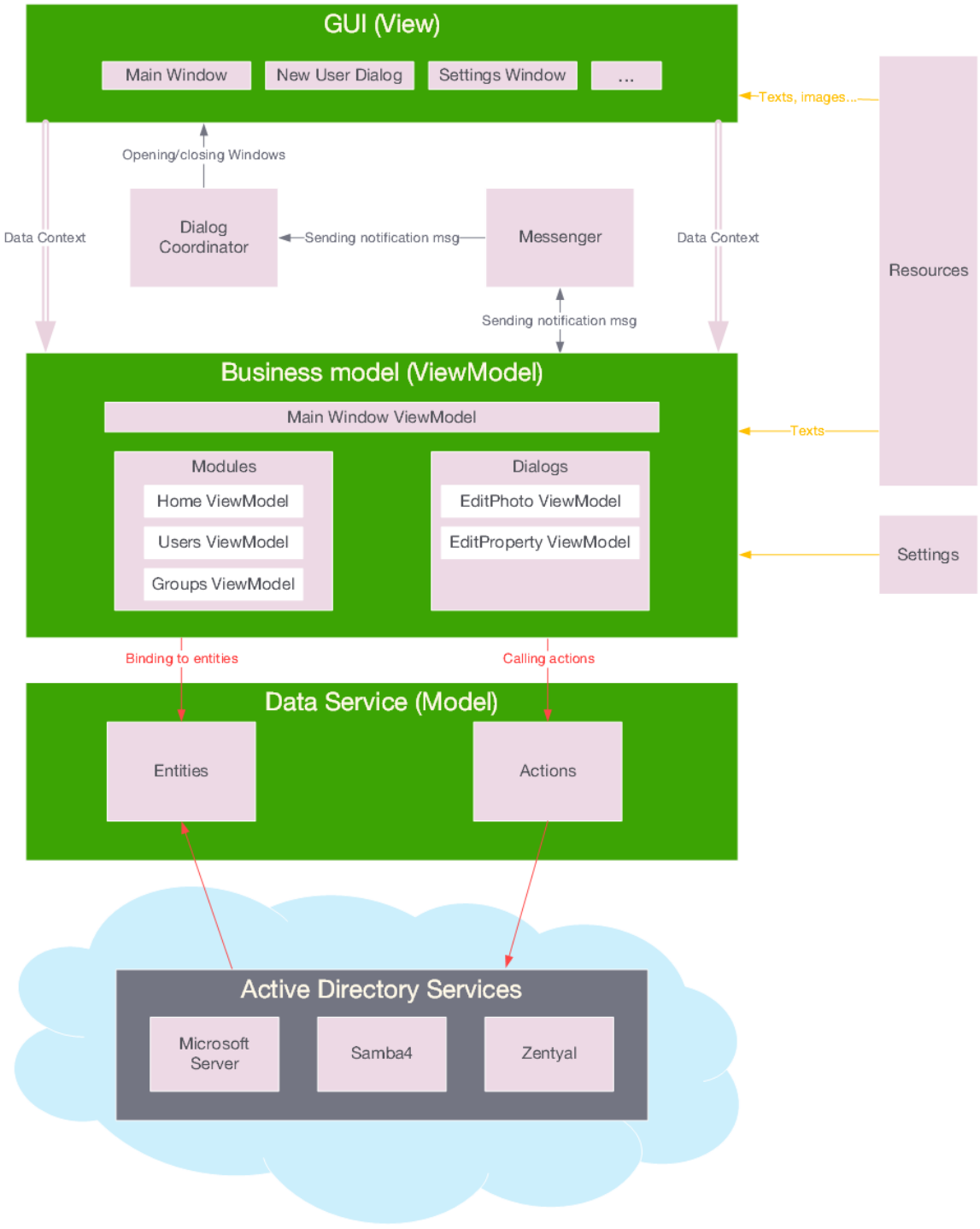


Figure 7.1: Overview of the application structure.

Messenger class is used to support messaging between different components. It is mainly used for notification messages between different ViewModels, notification messages for DialogCoordinator and IDataService. It is based on the observer design pattern, where any class can subscribe to listen for any message.

DialogCoordinator class is used as a single point of managing Views. It is notified via Messenger class and depending on the type of message and the type of window currently opened, it opens or closes both modal and non-modal windows and dialogs.

7.2 Active Directory Service

Active Directory Service in the Figure 7.1 represents a server running Active Directory Domain Services. It represents a blackbox from the application point of view, with clearly defined interface it provides. It is not important what implementation of it is running as long as it supports the protocol used by the Active Directory Domain Services. The most commonly used servers that are considered to be a suitable solution for this component can be found in the Chapter 4.

This component functions as a source of all data. Application connects to this service and authenticates a user against it. If provided credentials are accepted, it enables a user with sufficient permissions to retrieve any data he asks for.

This component communicates only with Data Service component. Therefore, Data Service is responsible for all communication and proper manipulation with data retrieved from AD.

7.3 Data Service (Model)

This component plays a role of a mediator between the Active Directory Service and the Business model. It translates objects from Active Directory Service to an internal representation used by the application and handles communication with the Active Directory Service. It comprises of two logical parts – *Entities* and *Operations*. The Data Service component provides the `IDataService` interface that is used by the Business Model, offering a number of various operations for managing AD. It also holds Entities retrieved from the underlying AD Service component, transforms them and passes them up for binding purposes[1].

Data Service makes use of .NET framework – especially `DirectoryEntry` class. This class is used to create a connection (when URI of an AD object and credentials of a user are provided) and to retrieve the attributes of an object.

Entities

Entities are internal objects that represent an object retrieved from the AD Service. These Entities are offered to and used by the Business Model component to provide information about properties and attributes of AD objects. Each entity class inherits from the `ADObject` class. This is a generic class that holds information about the connection to the original AD object and contains a list of all possible properties as

[1]Data binding is general technique that binds two data sources together and maintains synchronization of data. The main motive of data binding is to ensure that the UI is always synchronized with the internal object structure automatically.

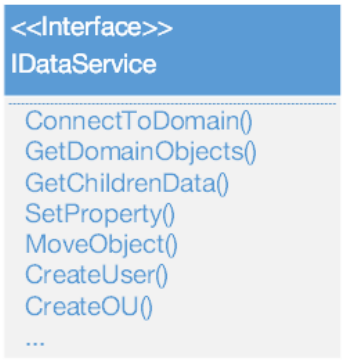


Figure 7.2: Operations accessible from the IDataService interface.

defined in the AD schema along with their actual values. It serves as a base class and all the other classes inherit from it.

The application supports the recognition of a number of typical AD objects: Domain, User, Container, Organizational Unit, Computer and Group. All these inherit from the `ADObject` class and extend the list of available custom properties that is used for binding purposes on the Business Model side.

Operations

`IDataService` is an interface that contains all implemented operations in the AD Service. A typical method requires an `ADProperty` or an `ADObject` in which the change should occurs, a new value and a callback action signaling an error that might happenwhen calling the Active Directory Service. Some of the implemented operations can be seen on the Figure reffig:idataservice.

7.4 Business model (ViewModel)

Business model is the main part of the application that binds the functionality of all components together. It consists of ViewModels that define data used by Views for binding purposes. ViewModels process commands from Views and apply business logic depending on the command invoked. A ViewModel contains properties that are used for binding in Views. They are also connected to the Messenger from which they can receive notification messages or send a notification message[2] to the Messenger.

MainWindowViewModel

This ViewModel is the most important one, since it is used for binding purposes in the `MainWindowView` component. It holds logic common for the whole application and also holds the context for different ViewModels of specific modules such as Users and Groups. It implements business logic for extending the application with new modules and manages visibility of currently selected module. The next important functionality of the `MainWindowViewModel` is that it contains a logic for connection to Active Directory. It loads the default settings for a connection and enables the current user to define login information for the connection to Active Directory. When

[2]Notification messages are in fact calls of the callback functions that were previously set by the receiving component

a user wants to connect to a new domain or refresh the currently loaded content of a domain, it invokes operations on the Data Service component. This component then refreshes its content and loads the current state of all entities within the domain.

Modules

Modules represent a group of ViewModels that is used for binding data to controls included in the View. They group functionality and business logic of one specific module. It is used as an easy way to add a new functionality to the application. Adding a new module to the application is done on the source-code level, therefore to add a new module, a View and a ViewModel for such a module has to be created and link to this module has to be added to the `MainWindowView` and the `MainWindowViewModel`. Modules are always connected to the `MainWindowViewModel` and this is the only connection they have to the application. Other than that, it makes use of the `Messenger` that allows this module to subscribe to particular notification messages or to send a notification message. To every ViewModel, there is also a View that is used as a control added to the main part of MainWindow. It also holds a list of other ViewModels that are represented as actions menu buttons. Menu buttons in action menu are completely controlled by this ViewModel and MainWindow only holds the view representation of these buttons. ViewModel retrieves a list of initial root AD objects to display. These objects then contain methods to obtain their children. This representation is well suited for a View to display these items in the `TreeView` later.

Several ViewModels that serve as a module has been already implemented, the most important of which are described in more detail:

- **HomeViewModel** is a ViewModel that serves as one of the modules. It contains all AD objects retrieved from the domain. It exposes these objects to be later displayed by the appropriate View. It also holds a list of menu objects. `HomeViewModel` offers only a selected number of operations that are possible from the menu. These operations include creation of a new OU, deletion of the selected object, movement of the selected object to another location in AD. These are the basic operations available for all objects in AD plus a general attribute editor with which any attribute of any object is editable.
- **UsersViewModel** is a module working with User objects. It holds the possibilities to manipulate User objects in AD. It contains a list of User objects and containers holding these objects as well as the root Domain object. This way, the View can only show actual User objects or paths from the root. It also holds a list of menu objects that allow manipulation of a selected user. They contain definition of operations such as creation of a new User, deletion of the selected object, movement of the selected object to another location in AD. It also contains operations that are specific for a User object – enable account, disable account, unlock account or reset password.
- **GroupsViewModel** is very similar to the `UsersViewModel` and holds a list of Group objects, containers holding these objects and the root Domain object of AD. It also holds currently selected AD object and a list of menu object offering operations on the selected AD object.

Dialogs

This group of ViewModels is used as a business logic for several dialog Views to which they are also connected for the purpose of binding data. They receive all important data in their constructor and then work only with this data. They are also not allowed to change the data and therefore after a successful confirmation of the window or after closing the window, the ViewModel invokes a callback method containing the result. The result data contains data changed in the ViewModel. It is then the responsibility of one of the modules or the MainWindow to perform actions on other components such as Data Service.

The application contains several Dialog Windows and the most important ones are described in more depth:

- **EditPhotoWindowModel** is used to edit one of the photo properties of an AD object that is holding data of the photo. It takes a photo property that is currently set on the AD object and a callback function as parameters. The callback function is used to signalized either an error or a success. The `EditPhotoWindowModel` also returns a newly created property with edited photo and it is the responsibility of parent component to ensure the change in the data. Working with photos is one of the key new features that is introduced in this application. It uses the *Settings* component to load pre-defined recommendations on photos. Main recommendations include format of a photo with its size, width, and height limit. There are two possible formats of a photo – Jpeg and Png. Other attributes are all configurable using the `SettingsView`. The logic offers a number of operations that are possible on the photo. It loads a photo from the storage and allows changing the width and height, rotating the picture and setting the picture to the recommended dimensions.
- **EditPropertyWindowModel** is used as a generic ViewModel for changing the value of any attribute of an AD object. It can be used to edit any attribute of an `ADObject` in case application supports the editing of such attribute. ViewModel requires a property of an AD object that is currently set and a callback function as parameters. The callback function signalizes the error or successful end of editing of the property. It provides a public property that returns a copy of the `ADProperty` with the performed changes. Other components that were calling this ViewModel are then able to perform other actions on this `ADProperty`, such as calling the Data Service to perform changes on the actual Active Directory Service.

7.5 Views

A View represents a part of the application that is most visible to the end users. They reflect the designed GUI and must offer a comfortable, fashionable and yet effective way to interact with the application.

Views use the *MahApps.Metro*[3] design framework that is open source. It provides an extensive list of additional controls and redesigns of almost all original controls of WPF. This framework was selected based on the open source license availability and the easiness to incorporate the framework into the application. It has the support for different color

[3]Home page of MahApps.Metro: `http://mahapps.com/`

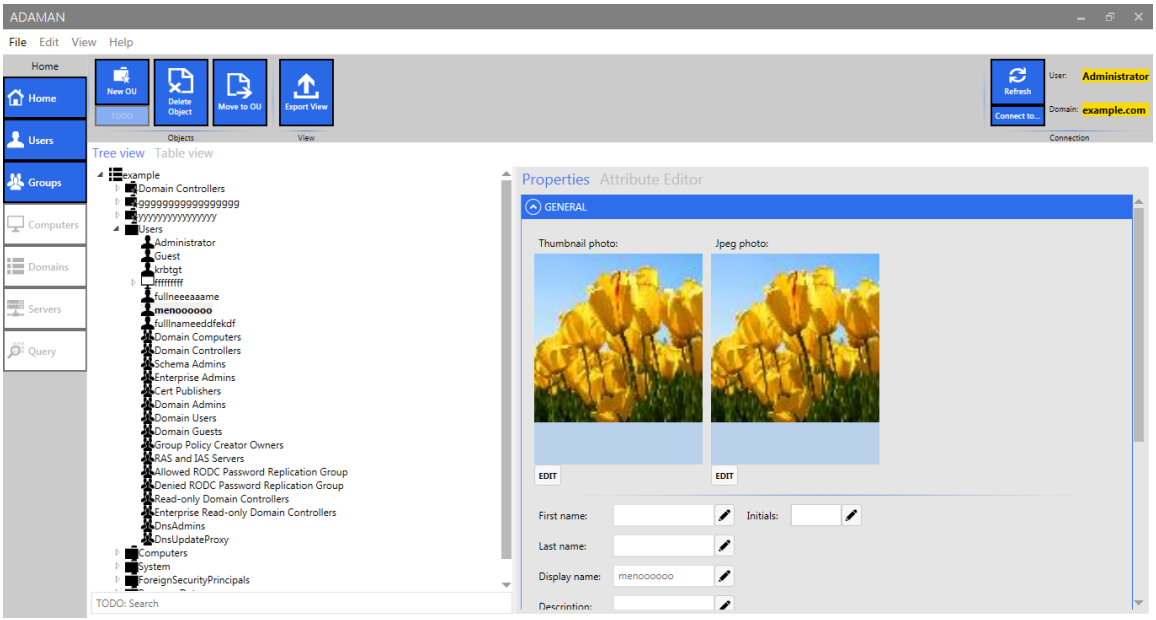


Figure 7.3: MainWindow of the application with properties view.

themes and with its continual development as a free open source project, it is well suited for the needs of this application. It provides Metro-like design of all windows, controls and uses a one-window only option for showing the additional dialogs.

MainWindow

It is the only window instance of the application. It is always the only window that is opened at all times and all the other dialog windows are incorporated into the main one and shown as an overlay. `MainWindow` has a typical menu bar on the top of the window as seen on the Figure 7.3, where application actions such as settings or help can be found. On the left side there is a module menu. It contains buttons for every module that the application can offer and serves as a way to switch between modules. It also offers a way to add new modules. When a new module is added to the application, a new menu button is added as well and all functionality of the new module is easily accessible to the user. Top part of the window also contains actions menu bar that is divided into the module action menu bar (left part) and application action menu (right part). The application actions menu contains only one button and a few labels. It is used to connect to AD and shows useful information about what user name is used in the connection and what domain is it currently connected to. This action is application-wide, so that the connection to AD is a concern of the application and not the individual modules. The main part of the window is a placeholder for a selected module. Module therefore has two basic spaces to use and those are the main part of the window and the actions menu.

HomeView

`HomeView` represents a home module's View. It takes all available main space in the `MainWindow` as shown in the Figure 7.4. It consists of two parts – the left part with a `TreeView` and the right part with the View for the selected AD object. The `TreeView` is used to show the content of AD. It is supposed to display all objects from

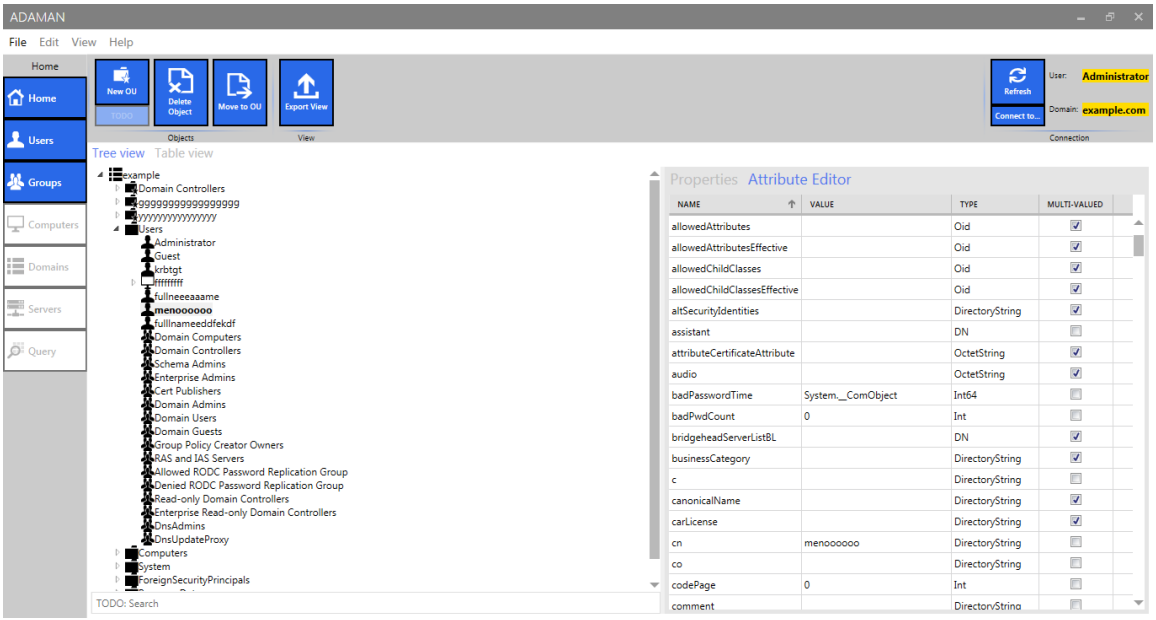


Figure 7.4: HomeView of the application with attribute editor.

AD and is intended to be used as way to change attributes for individual AD object by professionals with the knowledge of attribute naming. After a successful selection of an AD object from the `TreeView`, the right part of the view is updated to reflect the properties and attributes of the selected object. This part offers two ways to see the actual attributes of an object.

Properties tab is used to handle the most common cases and is easier to use. It offers more descriptive names and labels for selected number of attributes and should be used to change the most common attributes. Currently there is a possibility to see Properties only for a selected number of types of objects – OU, Container, Domain, User and Group. These are however the most commonly used objects in AD and are currently sufficient for the application.

These and also all the other objects attributes can be seen in the second type of a view – *Attribute Editor*. This simple editor shows a table of all attributes that can be assigned to an object along with their actual values (if they currently have a value set). It also shows a type of the attribute and information, whether this attribute is single- of multi-valued. To change the value of an attribute, user needs to double-click on the selected attribute and a new dialog is opened. Currently supported types that can be changed are `Bool`, `DirectoryString`, `DNString`, `GeneralizedTime`, `Int64`, `Int` and `OctetString`. This is not a complete list of all available types for an attribute value, but it represents the most commonly used ones. Other types are more or less used and set internally by AD. After the successful confirmation of changes, the actual responsibility of changing the value is sent to the connected ViewModel and visible components are updated appropriately.

UsersView and **GroupsView**

These are very similar to `HomeView`. The position of elements is the same as in the `HomeView` with the exception of their connection to the ViewModel. They have separate ViewModels offering a different set of data that is bound to the View. They both also offer the properties view of attributes and an Attribute Editor.

7.6 Communication

Communication between previously mentioned logical blocks is an important aspect, because high coupling can lead to an unmaintainable code. Therefore, three basic communication parts are recognizable:

1. **Data Service communication** – In order to reduce coupling between elements, Data Service layer communicates only with two other components – Business model and Active Directory Services. A clear communication flow is given. Business model accesses the interface of Data Service that includes all actions that it offers. After calling an action, the Data Service evaluates the type of the call and can decide to either change the Entities values or to perform a request on the Active Directory Services to retrieve or create a new information. After retrieving or creating a new information in Active Directory Services, changes to Entities are performed. Business model is bound to these Entities and retrieves the changed information automatically as the `PropertyChanged` event is raised.

2. **Business model communication** – A clear way of working with Views and their correspondent Viewmodels is often hard to find. This application uses services of two extra components – the Messenger and the Dialog Coordinator. Messenger is also used in the inter-ViewModel communication. To open a new window, a notification message with necessary parameters is sent to the Messenger. Messenger then evaluates to which components this message is supposed to be redistributed based on the type of the message and the Messenger's logic. In case of opening a new window, it sends a message to the Dialog Coordinator. Then the evaluation of the message continues and necessary changes with the View are performed. Communication from the View to the ViewModel is secured by binding the changes in the View to the commands in the ViewModel and calling these commands. All changes in the View are sent to the ViewModel, where appropriate updates of data is performed and the View is automatically changed to reflect data in the ViewModel.

3. **Resources and Settings** – Settings and Resources exist as global objects, where every part of application has an access to them. Nevertheless, to mitigate the coupling, Settings are only bound to ViewModels and in case of need are redistributed from there. Resources are bound both to Views and to ViewModels and contain texts and other localization data in the selected language.

Chapter 8

Testing

Testing is a process used to help identify the correctness, completeness and quality of developed computer software. In simple words, software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. With that in mind, testing can never completely establish the correctness of computer software.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. Therefore more approaches to testing are applied to different components of this application and on different levels of use. The main differentiation of testing is done according to the size of tested parts and therefore is testing divided into three subsequent parts – testing components, testing the application logic and compliance with the requirements.

8.1 Testing components

This section describes the testing of selected application components using unit tests. In general, this type of testing is performed by developers before the setup is handed over to the testing team. Unit testing is performed by the respective developers on the individual units of source code areas and the developers use test data that is different from the test data of the testing team. However, there is a limit to the number of scenarios and test data that a developer can use to verify a source code. Therefore unit testing cannot catch each and every bug in an application and it is impossible to evaluate every execution path in a software application.

Testing Active Directory Service

This component is a third party solution and is considered to be a blackbox and therefore is not specially tested. The application is only concerned about the operations that this component offers and presumes a functioning connectivity to an AD server. Nevertheless, the overall implemented application was tested on different versions of AD server to ensure the consistent multi-version functionality:

- Windows Server 2008 R2
- Windows Server 2012 R2
- Zentyal 4.1 Commercial Edition
- Zentyal 4.2 Development Edition

Testing Data Service

Unit tests are appropriate for the testing of this component. Tests are created to ensure that this component functions correctly. Testing of this component requires a functioning connectivity to the underlying Active Directory Service in order to perform all the unit tests. Unit tests cover the whole functionality that the representation of this component, interface `IDataService`, offers.

Testing Business model

Testing of this component is complicated mainly because only a small change of the Business model can lead to the major changes in tests. Business model is however implemented in the form of a number of ViewModels as described in the section 7.4. There are sophisticated ways of testing the ViewModel, but none were used in this thesis because of their complexity. A few unit tests were however created to test ViewModels for dialog windows, since they are easier to test then the complex `MainViewModel`. The Business Model logic is however a part of tests for the whole application. The Business Model is a heart of the application and connects all the other modules together, therefore the application could not properly function without this component.

8.2 Testing the application logic

This subsection turns its focus on the testing of the whole application logic. It is expected at this point, that every single application component functions properly. This part of the work is testing the application logic from the user's point of view based on the specification of the tested software.

The type of black-box testing that is based on the specification is called *functional testing*. The application is tested by providing input and then the results of execution are compared to the expected functionality of the application. Functional testing is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. For this purposes, several testing stories were created (the full description can be found in the Appendix B):

1. Create an OU
2. Change OU description
3. Create a user
4. Create a group
5. Add user group membership
6. Change country
7. Create an OU 2
8. Change OU description 2
9. Move a user
10. Add custom attributes

11. Create a user 2
12. Change user picture
13. Move an OU
14. OU protected from deletion
15. Delete an object

Usability testing

This technique is an important black-box testing technique that is used to identify any errors and improvements in the software by observing the users through their usage and operation. Previously defined testing stories were used and provided to the users for the purposes of testing. There were two techniques used in the usability testing: *remote usability testing* and *expert review*.

Remote usability testing consists of automatic collection of user's click streams, user logs of critical incidents, whereas the *expert review* involves the usage of the application by users with expert knowledge in the field. As a result of the expert review, their subjective feedback was collected and evaluated. All the collected data was processed and divided according to the testing story. These results can be found in the Appendix B.

Results from the usability testing show, that the application is not yet a fully suitable solution for Active Directory management. Even though an administrator is able to use this application to do basic management tasks on Active Directory and the functionality from the requirements analysis is met, it misses some of the functionality that is very useful, sometimes even needed. This is a functionality such as autocomplete when choosing an AD object or support for searching for various AD objects. Also there is a missing possibility to change the parent container while creating a new AD object. Once a creation window is visible, there is no way to change the parent container of this newly created object.

UI testing

UI testing involves testing the Graphical User Interface of the Software. UI testing ensures that the GUI functions according to the requirements and is tested in terms of color, alignment, size and other properties. In this part, testing of the correct functionality of the UI was conducted using *exploratory testing* technique. This technique is described as simultaneous learning, test design and test execution. In this part a tester or a user is encouraged to discover all the functionality of the application and to explore the whole application. This can lead to the discovery of previously unknown properties of the application.

This testing was part of the continuous testing and took place in every part of the development stage mainly by the tester. This was of course also part of the testing done by a user – usability testing. When finished with the prepared testing stories, a user had some time to fully explore all possibilities that this application has and to express their concerns or improvements to the UI.

Feedback from the prepared testing stories and from testing subjects suggest, that there are still a lot of changes that can be done for this application to be suitable for these users. From the notes taken during testing:

- Missing signalization of the selected module
- Hardly recognizable, which object is currently selected
- Layout problems with some dialog windows: `SettingsView`, `EditPropertyView` etc.

Even though these flaws were found, the users identified the application as very intuitive. They also liked the color scheme and the fact that dialog windows are shown within the same one window.

8.3 System testing

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested to see that it meets the specified business requirements. The important business factors that are to be considered were defined in one of the first phases of the application design. Chapter 6 contains the requirements that are being evaluated:

Users and Groups management – Testing stories were created and performed to test the basic operations for user and group management. The application supports tasks for managing the User and Group objects in Active Directory. It covers the needed functionality derived from Active Directory Users and Computers.

Profile photo management – Part of the testing was also performed to demonstrate the ability to set a profile photo for a user. This is supported in the application along with a way to do a selected number of basic photo transformations. The editing of a photo of an existing user is accompanied by the possibility to set a profile photo during the user creation process.

Customized creation process – This is one of the most important additional functionalities that this application provides. One of the testing stories contains the setting and the use of such a customized creation process. The application supports the definition of any additional attributes for a creation process.

Attribute Editor – The application implements an attribute editor that is used to change any value of any attribute. The functionality is to some point similar to the one used in Attribute Editor included in Active Directory Management Tools.

Modularity – The modularity is partially supported. The application provides a place in the source code to include other functionalities in the form of modules that include their own View and action menu. This must be done on the source code level of the application, where no support for the on-the-run inclusion in the form of a plugin is implemented. Nevertheless this might be considered as a next step in the future.

Support any AD solution – The application was successfully tested on four different implementations that support Active Directory Domain Services and therefore can be considered to comply with this requirement.

Be free and open-source – Since this application is a part of the thesis, it is free, open-source and accessible in digital as well as printed format.

Chapter 9

Conclusion

The goal of this thesis was to design and implement a new application for managing Active Directory. This application covers the functionality which usually requires to use several free tools of to buy costly third-party programs which can do that.

This thesis explained the basic concepts of Active Directory. After presenting the logical and physical components, the usage of these concepts was described. A number of servers using these services and a plenty of applications for the management of Active Directory proves, that this concept is widely used even today. New possibilities to ease the management of Active Directory domain were discovered and a new concept of management application was defined. Design includes the assessment of required functionality, a number of use case diagrams along with the propositions for the graphical users interface. This design is then compared to currently existing applications and offers an easy-to-understand view on the benefits of the new application. New application was designed to completely cover the functionality of other existing applications and to be prepared for extensions and future changes. According to the concept, a new application was developed supporting the management of users and groups and implementing additional features such as profile photo editing and a definition of customized object creation process.

The application was tested on five different servers supporting Active Directory. Testing of the application indicates, that the application is suitable for managing Active Directory on all tested servers and successfully implements additional functionality. Even if it does not implement some supporting features, the feedback from testing was positive in terms of intuitiveness. Still, it also shown a number of additional features that are missing in order to make the use more efficient.

The application is undertaking a thorough review by the representatives of the company ArtIn, which is considering the application as their main Active Directory management application. The development of this application is therefore expected to continue.

Bibliography

[1] DeSimone, A.: *Directory Services: The Foundation for Web Portals*. 11 2002. [Online; Accessed 2016-05-20].
Retrieved from: `https://net.educause.edu/ir/library/pdf/eqm0240.pdf`

[2] Desmond, B.; et al.: *Active Directory*. O'Reilly Media, Inc.. 2013. ISBN 978-1-449-32002-7.

[3] Isode Ltd.: *LDAP and X.500*. [Online; Accessed 2016-05-20].
Retrieved from: `http://www.isode.com/whitepapers/ic-6033.html`

[4] Leach; et al.: *RFC4122: A Universally Unique IDentifier (UUID) URN Namespace*. 07 2005. [Online; Accessed 2016-05-20].
Retrieved from: `http://www.ietf.org/rfc/rfc4122.txt`

[5] Microsoft Corporation: *Active Directorys*. 1999. [Online; Accessed 2016-05-20].
Retrieved from: `https://msdn.microsoft.com/en-us/library/bb742424.aspx`

[6] Microsoft Corporation: *Active Directory Domains and Trusts*. 03 2012. [Online; Accessed 2016-05-20].
Retrieved from: `https://technet.microsoft.com/en-us/library/cc770299.aspx`

[7] Microsoft Corporation: *Active Directory Collection*. 11 2014. [Online; Accessed 2016-05-20].
Retrieved from: `https://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx#w2k3tr_ad_over_qbjd`

[8] Microsoft Corporation: *What Are Domains and Forests?* 11 2014. [Online; Accessed 2016-05-20].
Retrieved from:
`https://technet.microsoft.com/en-us/library/cc759073(v=ws.10).aspx`

[9] Microsoft Corporation: *Directory System Agent*. 2016. [Online; Accessed 2016-05-20].
Retrieved from:
`https://msdn.microsoft.com/en-us/library/ms675902(v=vs.85).aspx`

[10] Microsoft Corporation: *Organizational Units*. 2016. [Online; Accessed 2016-05-20].
Retrieved from: `https://technet.microsoft.com/en-us/library/cc978003.aspx`

[11] Samba Team: *Samba Team Releases Samba 4.0*. 12 2012. [Online; Accessed 2016-05-20].
Retrieved from: `https://www.samba.org/samba/news/releases/4.0.0.html`

[12] Sermersheim, J.: *RFC4511: Lightweight Directory Access Protocol (LDAP): The Protocol*. 06 2006. [Online; Accessed 2016-05-20].
Retrieved from: `https://tools.ietf.org/html/rfc4511`

[13] Simmons, C.: *Active Directory Bible*. IDG Books Worldwide, Inc.. 2001. ISBN 0-7645-4762-3.

[14] Solomon, D. A.; et al.: *Microsoft Windows Internals: Microsoft Windows Server 2003, Windows XP, and Windows 2000*. Microsoft Press. 2005. ISBN 0-7356-1917-4.

[15] Sun Microsystems, Inc.: *Introduction to Directory Services and Directory Server*. 2005. [Online; Accessed 2016-05-20].
Retrieved from: `https://docs.oracle.com/cd/E19396-01/817-7619/intro.html`

[16] Wahl; et al.: *RFC2251: Lightweight Directory Access Protocol (v3)*. 12 1997. [Online; Accessed 2016-05-20].
Retrieved from: `https://www.ietf.org/rfc/rfc2251.txt`

[17] Wahl; et al.: *RFC2253: Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. 12 1997. [Online; Accessed 2016-05-20].
Retrieved from: `https://tools.ietf.org/html/rfc2253`

[18] Wahl, et. al.: *RFC2253: Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. 12 1997. [Online; Accessed 2016-05-20].
Retrieved from: `https://tools.ietf.org/html/rfc2253`

Appendices

List of Appendices

Appendix A

List of application tasks

- **General tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Change configuration	X			X			
Delete configuration	X			X			
Set view options	X			X			
Export view	X				X		
Filter objects	X				X		
Show help	X			X			

- **Domain/domain controller tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Connect to a domain controller	X			X			
Authenticate a user in a domain	X			X			
List content of a domain	X			X			
Find objects in AD domain services	X				X		
Change domain	X			X			
Change domain controller	X			X			
Delegate control	X				X		
Change Operations Masters	X			X			
Pre-create read-only domain controller account			X				
Raise domain functional level		X					X
Create shortcut trust		X					X
Create realm trust		X					X
Create external trust		X					X
Verify a trust		X					X
Remove a trust		X					X
Select the scope of authentication for users		X					X

- **Object tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Manage properties	X			X			
Manage attributes	X			X			
Manage permissions	X				X		
Add to a group	X				X		
Remove from a group	X				X		
Move to another OU	X			X			
Find objects in AD domain services	X				X		
Create a new object	X			X			
Rename an object	X			X			
Delete an object	X			X			

- **Object-Computer tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Create new computer	X				X		
Add a computer to a group	X				X		
Delete a computer	X				X		
Move a computer	X				X		
Reset a computer account	X					X	
Enable a computer account	X					X	
Disable a computer account	X					X	
Name mappings			X				
Manage computer			X				

- **Object-Group tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Create a new group	X				X		
Add a member to a group	X				X		
Change group scope	X				X		
Delete a group	X				X		
Manage group membership	X				X		
Assign user rights	X				X		
Send mail		X			X		

- **Object-Organizational Unit tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Create an OU	X			X			
Move an OU	X			X			
Delete an OU	X			X			
Delegate control	X				X		

- **Object-User tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Copy user	X			X			
Create a new user	X			X			
Delete a user	X			X			
Enable account	X			X			
Disable account	X			X			
Reset password	X			X			
Open home page		X			X		
Send Mail		X			X		
Unlock a user account	X			X			
Delegate authority	X				X		
Allow users to use VPN	X			X			
Grant rights to file servers	X			X			
Control when user can use the network	X			X			
Add photo	X			X			
Delete photo	X			X			
Change photo	X			X			
Set photo criteria	X			X			
Rotate photo	X			X			
Compress photo	X			X			
Resize photo	X			X			
Crop photo	X			X			
Name mappings			X				
Set logon hours			X				
Setting login and logout scripts			X				

- **Forest tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Change forest		X					
Raise forest functional level		X				X	
Add UPN suffixes			X				X
Delete UPN suffixes			X				
Show a list of UPN suffixes			X				
Create a forest trust		X					X
Change the routing status of a name suffix			X				
Enable an existing name suffix from routing			X				
Disable an existing name suffix from routing			X				
Exclude name suffixes from routing to a local forest			X				
Remove a forest trust		X					X
Select the scope of authentication for users		X					X

- **Site tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Delegate control		X					X
Create a new site		X				X	
Rename a site		X				X	
Delete a site		X				X	
Create a subnet		X				X	
Associate a subnet with site		X				X	
Delete a subnet		X				X	
Select another licensing computer			X				
Cache universal group memberships			X				
Reveal the services node			X				

- **Site replication tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Create a site link		X				X	
Delete a site link		X				X	
Create a site link bridge		X				X	
Delete a site link bridge		X				X	
Configure intersite replication availability		X					X
Configure site link cost		X					X
Configure intersite replication frequency		X					X
Ignore replication schedules			X				
Enable site link bridges		X					X
Disable site link bridges		X					X
Add a site to a site link		X				X	
Remove a site from a site link		X				X	
Manually add connections			X				
Force replication over a connection			X				

- **Server tasks:**

	Required	Desired	No need	Phase 1	Phase 2	Phase 3	Phase4
Select a query policy		X					X
Add a global catalog server	X					X	
Remove a global catalog server	X					X	
Check replication topology	X						X
Delete extinct server metadata			X				

Appendix B

Testing stories

1. **Create an OU**

   As an... Administrator

   I want to... create a new OU

   so that... it is named "Testing"

Needed help	—
Minimal number of clicks	5
Average number of clicks	6
Average number of incidents	0
Additional feedback	—

2. **Change OU description**

   As an... Administrator

   I want to... change a description property of OU named "Testing"

   so that... the new value is "This is OU for testing"

Needed help	—
Minimal number of clicks	4
Average number of clicks	5
Average number of incidents	0
Additional feedback	—

3. **Create a user**

   As an... Administrator

   I want to... create a new User

   so that... it is located in OU "Testing" and his name is "Jon Snow"

Needed help	—
Minimal number of clicks	11
Average number of clicks	18
Average number of incidents	2 – not all required fields were completed
Additional feedback	Not able to change custom location from Users module.

4. **Create a group**

   As an... Administrator

   I want to... create a new Group

   so that... it is located in OU "Testing" and has "TestGroup" as a name

Needed help	—
Minimal number of clicks	6
Average number of clicks	11
Average number of incidents	1 – not all required fields were completed
Additional feedback	Not able to change custom location from Groups module. Required fields are not specially coloured.

5. **Add user group membership**

   As an... Administrator

   I want to... add a group membership of user "Jon Snow"

   so that... user is a member of group "TestGroup"

Needed help	—
Minimal number of clicks	11
Average number of clicks	17
Average number of incidents	1 – add as member button was not pushed
Additional feedback	Member of is harder to find. Add as member button is not visible to be required to push.

6. **Change country**

   As an... Administrator

   I want to... change the country property of user "Jon Snow"

   so that... the new country value is "Albania"

Needed help	—
Minimal number of clicks	6
Average number of clicks	6
Average number of incidents	0
Additional feedback	—

7. **Create an OU 2**

   As an... Administrator

   I want to... create a new OU

   so that... it is named "TestMove"

Needed help	—
Minimal number of clicks	5
Average number of clicks	5
Average number of incidents	0
Additional feedback	—

8. **Change OU description 2**

   As an... Administrator

   I want to... change a description property of OU named “TestMove”

   so that... the new value is “Testing object move.”

Needed help	—
Minimal number of clicks	4
Average number of clicks	4
Average number of incidents	0
Additional feedback	—

9. **Move a user**

   As an... Administrator

   I want to... move a user named “Jon Snow”

   so that... it is relocated to OU named “TestMove”

Needed help	—
Minimal number of clicks	5
Average number of clicks	6
Average number of incidents	0
Additional feedback	—

10. **Add custom attributes**

    As an... Administrator

    I want to... change settings

    so that... while a new user is being created, setting a “description” attribute is possible

Needed help	—
Minimal number of clicks	7
Average number of clicks	12
Average number of incidents	1 – not hitting add button
Additional feedback	—

11. **Create a user 2**

    As an... Administrator

    I want to... create a new User

    so that... it is located in OU “TestMove”, his name is “John Smith” and description is “Description set on user creation”

Needed help	—
Minimal number of clicks	12
Average number of clicks	13
Average number of incidents	0
Additional feedback	—

12. **Change user picture**

    As an... Administrator

    I want to... change a photo of user “John Smith”

so that... photo property "thumbnailPhoto" is set to any picture, recommended dimensions of a photo are set and photo is saved in the jpeg format

Needed help	—
Minimal number of clicks	7
Average number of clicks	10
Average number of incidents	0
Additional feedback	—

13. **Move an OU**

    As an... Administrator

    I want to... move OU named "MoveTest"

    so that... it is located in OU "Testing"

Needed help	—
Minimal number of clicks	4
Average number of clicks	4
Average number of incidents	0
Additional feedback	—

14. **OU protected from deletion**

    As an... Administrator

    I want to... set property of OU "Testing"

    so that... it is no longer protected from deletion

Needed help	—
Minimal number of clicks	4
Average number of clicks	5
Average number of incidents	0
Additional feedback	—

15. **Delete an object**

    As an... Administrator

    I want to... delete an OU named "Testing"

    so that... it no longer exists

Needed help	—
Minimal number of clicks	2
Average number of clicks	2
Average number of incidents	0
Additional feedback	Missing dialog window whether you really want to delete this object