Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Systém automatickej distribúcie softvéru na pracovné stanice s OS Windows 7

Bakalárska práca

2014 **Martin Strapko**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Systém automatickej distribúcie softvéru na pracovné stanice s OS Windows 7

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 2508 Informatika
Školiace pracovisko: Katedra Informatiky FMFI
Vedúci práce: RNDr. Jaroslav Janáček, PhD.

2014 **Martin Strapko**

Poďakovanie

Zatiaľ sebe.

Abstrakt

Slovenský abstrakt.

Abstract

English abstract.

Obsah

List of listings

Zoznam obrázkov

Úvod

Pravidelnou prácou systemového administratora je inštalácia a príprava počítačov na prácu pre ľudí menej zbehlých v informatických technológiach. Toto zahŕňa všetko od umiestnenia a zapojenia počítaču do elektrickej siete až po inštaláciu programov.

V tejto baklárskej práci sa budeme zaoberať riešením problematiky hromadnej inštalácie počitačového softwaru. Orientovať sa budeme na platformu Windows 7, keďže mnoho firiem a školských inštitúcii pracuje práve s týmto operačným systémom.

Cieľom tejto bakalárskej práce je analýza problému hromadnej inštalácie, popísať možné riešenia tohto problému a implementovať aplikáciu, ktorá bude túto problematiku riešiť. V prvej kapitole sa budeme venovať špecifikácii problému a návrhu aplikácie. V nasledujúcej kapitole objasníme použité technológie. V tretej kapitole, ktorá tvorí jadro práce, vysvetlíme jej implementáciu, popíšeme jednotlivé triedy a ukážeme kľúčové časti kódu. Poslednú kapitolu bude tvoriť dokumentácia k inštalácii a používaniu aplikácie.

KAPITOLA 1

Špecifikácia problému a návrh riešenia

1.1 Zadefinovanie Problému

V dnešnej dobe najdôležitejším nástrojom produktívného zamestnanca vo firme je počítač. Avšak aj napriek tomuto faktu mnoho z týchto zamestnancov počítačom nerozumie a vyžadujú pomoc firemného administrátora pri inštalácii počítaču alebo niektorých programov. Táto úloha je síce z pohľadu informatika triviálna, avšak zaberá obrovské množstvo času vzhľadom na jej obťažnosť. Administrátor dokáže stráviť niekoľko dní nad prípravou novokúpených počítačov, pred tým ako ich odovzdá zamestnancom. Toto je spôsobené hlavne veľkým množstvom ovládačov programov bez ktorých je počítač takmer nepoužiteľný. Problém, ktorému sa v tejto práci budeme venovať a výsledný program ho má čo najefektívnejšie riešiť, je ako presunúť čo najviac práce s prípravou počítaču do funkčného stavu z administrátora na užívateľa s tým, že od užívateľa nebudeme vyžadovať žiadne nové znalosti, ani mu nebudeme musieť prideľovať vyššie prístupové práva.

1.2 Návrh riešenia

Bohužiaľ inštalácie na operačnom systéme Windows nie sú homogénne. Veľa výrobcov softvéru si programuje svoje vlastné inštalačné programy a preto výroba aplikácie, ktoré by vedela automatický obslúžiť akúkoľvek inštaláciu, by bola takmer nemožná. Preto v rámci riešenia problému si budeme generovať vlastné inštalácie, ktoré budú mať spoločnú predlohu a vďaka tejto vlastnosti budeme môcť vyrobiť aplikáciu, ktorá tieto inštalácie bez zásahu užívateľa nainštaluje. Toto riešenie vyrobí balíčky, ktoré následne presunie na cieľovú stanicu, kde sa rozbalia a tým nainštalujú želaný program. Celý

tento proces môže byť pre používateľa neviditeľný. Implementáciu a finálnu aplikáciu vieme rozdeliť na tieto časti:

- Odchytenie a uloženie priebehu inštalácie do balíčku
- Inštalácia balíčku na cieľovej stanici
- Prenos balíčkov medzi stanicou a serverovým úložiskom

1.2.1 Príprava balíčkov

Priamočiare riešenie Priamočiary prístup k tomuto problému pozostáva z vyrobenia zoznamu súborov na disku pred a po inštalácii a nasledovné vytvorenie rozdielu medzi týmito zoznamami. Avšak pri výrobe rozdielu by bolo treba kontrolovať aj čas poslednej úpravy pre súbory, ktoré neboli inštaláciou vytvorené ale len zmenené. V dnešnej dobe, kedy veľkostí pevných diskov sa pohybujú v stovkách gigabajtov, je tento proces zdĺhavý a neefektívny.

Pomocou obalenia inštalácie Ďalším možným riešením zisťovania zmien, ktoré inštalácia vykonala, je vytvorenie aplikácie, ktorá obalí inštalačný program a bude reagovať na jeho volania operačnému systému na vytvorenie alebo zmenu súborov. Tento prístup je v ohľade na rýchlosť vykonávania nesmierne rýchly, avšak nie dokonalý. Pri implementácií by sme museli filtrovať volania operácií na dočasných súboroch, ktoré si inštalácia môže vytvárať počas svojho behu. Implementácia tohto prístupu by bola viazaná na konkrétny proces inštalácie a museli by sme ošetrovať špeciálne prípady, kedy inštalácia používa pomocné procesy, ktorých volania by nemuseli byť odchytené pôvodnou aplikáciou.

Odchytením zmien na súborovom systéme Posledné riešenie, ktoré spomenieme a ktoré budeme implementovať, je pomocou štandardných knižníc .NET frameworku pre systémy Windows. V týchto knižniciach nájdeme mnoho nástrojov, ktoré nám pomôžu narábať so súborovým systémom. V tomto konkrétnom prípade použijeme triedu FileSystemWatcher, ktorá čaká na udalosti zo súborového systému a posiela ich ďalej do programu. Týmto spôsobom sa vyhneme vytváraniu kompletného zoznamu súborov na disku a zároveň zistíme zmeny v systéme vytvorené akýmkoľvek procesom v danom čase. S týmto prichádza aj mnoho negatív, ako zachytávanie udalostí od samotného operačného systému alebo programov bežiacich na pozadí nesúvisiacich s inštaláciou. Doteraz sme riešili ako zachytiť zmenu v súborovom systéme, avšak dôležitou súčasťou mnohých programov sú aj Windows registre. Spomínaný .NET framework obsahuje triedy na počúvanie udalostí v registroch, avšak tieto triedy neobsahujú dostatok informácií kde a aká zmena v registroch nastala aby sme ich mohli použiť. Preto pri zisťovaní

zmien v registroch použijeme ich základne funkcie a to výpis podstromu registrov do súboru a načítanie podstromu registrov zo súboru. Zmeny, ktoré vznikli počas inštalácie získame z rozdielu výpisov registrov pred a po inštalácií. Zistením zoznamu súborov a registrov, ktoré sa vyrobili alebo zmenili počas inštalácie dostaneme balíček, ktorý obsahuje všetky potrebné veci pre fungovanie daného programu.

1.2.2 Inštalácia balíčkov

Po tom ako pripravíme balíčky vyššie uvedeným spôsobom, treba spraviť inštaláciu programov z daných balíčkov. Balíčky majú nami zadefinovanú štruktúru, ktorá obsahuje nasledovné informácie:

- Súbory programu
- Výpis registrov, ktoré program vyžaduje
- Zoznam súborov programu a miesto ich inštalácie
- Balíčky, ktoré treba mať nainštalované

Toto sú všetky informácie, ktoré sú potrebné na replikáciu inštalácie na iné stanice. Okrem týchto si v zozname všetkých balíčkov budeme pamätať zoznam spustiteľných súborov na ktoré sa ma vytvoriť odkaz a nastavenie typu inštalácie pri štarte systému alebo na vyžiadanie. Úlohou tejto časti aplikácie bude rozbaliť súbory z balíčka na miesto, ktoré bolo určené pri inštalácii, načítanie hodnôt registrov podľa výpisu v balíčku a vytvorenie odkazu na spustiteľné súbory, ktoré by užívateľa mohli zaujímať.

1.2.3 Prenos balíčkov

Ako posledné budeme v implementácií riešiť prenášanie balíčkov na cieľové stanice. Vychádzať budeme z predpokladu, že naša aplikácia bude používaná v prostredí obsahujúcom sieťové spojenie medzi stanicami a stanicu na ktorej sa budú skladovať všetky dostupné balíčky, pre potreby tejto práce ju budeme nazývať server. Užívateľská aplikácia vždy zo serveru stiahne zoznam balíčkov s odkazmi, ktoré ma pripraviť pre užívateľa, a neskôr pri inštalácií bude zo serveru sťahovať súbory inštalovaného balíku. Čo sa týka obnovovania zoznamu balíkov na servery, bude administrátorovou povinnosťou presunúť súbor so zoznamom na server.

1.2.4 Sériové kľúče

Väčšina programov vyžaduje kúpenú licenciu, ktorá sa bežne kontroluje pomocou sériového kľúča. Tento kľúč sa zadáva pri inštalácií alebo prvom spustení. Ak sa zadáva pri prvom spustení, našu aplikáciu to neovplyvní, len administrátor bude zodpovedný za

spustenie programu na každej stanici a zadaniu kľúča. Ak sa kľúč zadáva pri inštalácií naša aplikácia okopíruje program s jedným kľúčom na všetky stanice kam bude program nainštalovaný pomocou balíčka. Keďže v dnešnej dobe čím ďalej tým viac firiem poskytuje možnosť hromadného nákupu licencií, ktorým prislúcha len jeden sériový kľúč, sme sa rozhodli tento problém a jeho riešenie nechať na administrátorovi, ktorý bude musieť zariadiť aby boli dodržané všetky licenčné podmienky, ktoré sa týkajú kľúča.

1.2.5 Užívateľské nastavenia

Mnoho programov si v dnešnej dobe zapisuje nastavenia do užívateľskej zložky umiestnenej na systémovom disku - C:\<meno užívateľa>\. Táto zložka sa samozrejme mení podľa užívateľa a preto treba aj pri inštalácií balíčka na toto myslieť a nainštalovať ho do užívateľskej zložky práve prihlaseného užívateľa. Toto vyriešíme jednoducho keď pri zapisovaní cesty skontroľujeme či neukazuje do uživateľskej zložky a nahradíme ju našou značkou, ktorú neskôr nahradíme práve prihlaseným užívateľom.

KAPITOLA 2

Technológie

2.1 C#

Jazyk C# sme zvolili kvôli podpore .NET frameworku. Ten nám poskytuje funckie na narábanie s mnohými časťami operačného systému Windows. Programy napísané v tomto frameworku v aplikačnom virtuálnom stroji, ktorý sa stará o bezpečnosť, správu pamäte a odchytávania výnimiek. Vďaka nemu budeme schopný odchytiť a zreplikovať inštalácie na staniciach, poskytuje nám triedy na prácu so súbormi, zložkami a registrami operačného systému.

2.2 HTTPS

Na prenos dát medzi užívateľmi a serverom budeme používať technológiu Secure Sockets Layer (SSL) zakomponovanú v protokole HTTPS.

KAPITOLA 3

Implementácia

3.1 Štruktúra aplikácie

Aplikácia bude mať dve spustiteľné zložky, administrátorskú a užívateľskú. Zatiaľ čo administrátorská časť bude obyčajná Windows aplikácia, ktorú bude treba manuálne spustiť, užívateľská bude reagovať na špeciálne vytvorené odkazy aplikácií

3.1.1 Administrátor

Po spustení tejto časti aplikácie, bude administrátor prezentovaný aplikáciou obsahujúcou ovládanie na vytváranie balíčkov, spustenie/vypnutie zachytávania zmien v súborovom systéme a chybový výpis. Táto časť aplikácie vyžaduje administrátorské práva na spustenie, keďže sú potrebné pre spustenie FileSystemWatcher triedy na súborovom systéme.

3.1.2 Užívateľ

Odkazy vytvorené našou aplikáciou, budú smerovať k tejto časti aplikácie, ktorá skontroluje či už daný program existuje a rozhodne, či treba spustiť balíček na inštaláciu alebo samotný program. Keďže aj táto časť aplikácie bude narábať s registrami alebo kopírovať súbory do špeciálnych zložiek v súborovom systéme je potrebné pustiť ju s administrátoskými právami. Avšak zároveň táto časť aplikácie je určená pre základných užívateľov bez administrátoských práv, preto potrebujeme odkazy púšťať ako Administrátor a nevyžadovať prihlásenie od používateľa.

3.1.3 Konfigurácia

Všetky nastavenia, ktoré aplikácia potrebuje budú uložené vo Windows registroch, pri prvom spustení si aplikácia všetky tieto nastavenia vypýta, neskôr budú meniteľné len

pomocou regedit. Tieto nastavenia zahŕňajú:

- Cesta k aplikácii
- Cesta k balíčkom
- Cesta k odkazom na programy
- Adresa serveru

3.1.4 Zmeny súborového systému

Základom našej aplikácie je .NET trieda FileSystemWatcher, ktorú na začiatku inicializujeme a nastavíme na počúvanie udalosti na všetkých pevných diskoch počítaču, ukážka kódu v kóde 3.1. Táto trieda generuje udalosti na ktoré reaguje zvyšok programu, existujú dve kategórie udalostí na ktoré treba reagovať. Zmena súboru, ktorá zahrňuje vytvorenie, zmenu a zmazanie súboru. Druhá kategória je premenovanie súboru, ktoré sa uskotoční iba pri zmene názvu súboru.

Zmena súboru

Pri tomto type udalostí najprv zistíme, ktorá z troch vecí nastala. Pri vytvorení súboru si zapíšeme miesto, kde bol tento súbor vytvorený, do zoznamu súborov tohto balíku. Túto informáciu využijeme neskôr pri koprírovaní súborov do balíku a inštalácií balíku. Ak bola táto udalosť zavolaná pri zmene súboru, skontrolujeme či už tento súbor v tomto balíku monitorujeme, ak nie tak si ho zapíšeme do zoznamu akoby bol novo vytvorený aby sme odzalohovali zmeny, ktoré v ňom nastali. Nakoniec ak táto udalosť nastala pri zmazaní súboru, zistíme jeho výskyt v doterajšom zozname súborov a prípadne ho vymažeme aby sme sa neskôr vyhli chybe pri kopírovaní neexistujúceho súboru. Zoznam doteraz monitorovaných súborov si pamätáme v hašovacej mape, keďže nám poskytuje rýchle riešenie vyhľadávania a kontroly prítomnosti prvku v poli.

Premenovanie súboru

Pri tomto type události prejdeme náš doterajši zoznam nájdeme pôvodný názov a vymažeme ho, zároveň pridáme do zoznamu nový záznam s novým menom súboru. Tento zložitý postup je zapričinený použitím hašovacej mapy, ktorej hašovacia funkcia dostane ako vstup cestu k súboru a preto haš pre starý a nový názov sú odlišné.

3.1.5 Rozdiel registrov

K hodnotám registrov, ktoré program vytvoril alebo zmenil sa dostaneme pomocou vytvorenia rozdielu pôvodných registrov a registrov po nainštalovaní programu. K tomuto použijeme algoritmus na nájdenie najdlhšej spoločnej sekvencie, tento algoritmus

nebudeme implementovať ale použijeme časti voľne šíriteľného programu, ktorý vypíše užívateľovi rozdiely medzi dvomi textovými súbormi. V našej aplikácií dodáme tomuto programu na vstup výpisy registrov v štandardnom formáte windows vytvorené pomocou programu regedit. Ako výstup dostaneme kľúče registrov, ktoré pribudli od poslednej inštalácie.

3.1.6 Inštalácia balíčkov

Na každej podstanici bude po štarte systému spustená naša aplikácia, ktorá ma za úlohu nastaviť potrebné cesty do registrov, udržovať aktuálny zoznam balíčkov zo serveru a nainštalovať alebo spustiť vyžiadaný balíček. Vrámci inštalácie, aplikácia prejde cez zoznam súborov, ktoré daný balíček obsahuje a skontroluje či existujú na miestách kde majú, ak zistí že neexistuje tak ho tam nakopíruje z balíku. Táto časť aplikácia bere pri spustení 0, 1 alebo 2 parametre. Všetky odkazy vytvorené našou aplikáciou budú smerovať na tento spustiteľný súbor a budú mať aplikáciou nastavené potrebné parametre k spusteniu.

0 parametrov

Pri spustení aplikácie bez parametrov, aplikácia skontroluje Windows Registry a zistí, či obsahujú všetky potrebné údaje pre aplikáciu (cesty k dôležitým zložkám) a následne stiahne zo serveru najnovší zoznam balíčkov a podľa neho vytvorí potrebné odkazy. Odkazy vytvorí na jednom z dvoch miest.

- Balíčky, ktoré sa spustia pri spustení systému, odkazy budú umiestnené do zložky Po spustení prihlaseného užívateľa
- Balíčky, ktoré sa spustia na vyžiadanie uživateľa, odkazy budú umiestnené do zložky zapísanej v registroch

1 parameter

Ak do aplikácia zadáme len jeden parameter, tento parameter musí byť názov existujúceho balíčku, ktorý aplikácia stiahne zo serveru a nainštaluje.

2 parametre

Pri dvoch parametroch aplikácia dostane prvý parameter názov balíčku, ktorý ma nainštalovať a ako druhý parameter očakáva cestu k súboru, ktorý ma spustiť po dokončení inštalácie.

3.2 Popis Tried

MainWindow Trieda ovládajúca hlavné okno administráčného podprogramu a logiky na jeho pozadí

DepedenciesDialog Dialog v ktorom si administrátor vybere balíčky na ktorých je práve vytvorený balíček závislý

InstallTypeDialog Trieda, ktorá poskytuje dialogové okno na vybratie typu inštalácie balíčku, po spustení alebo na vyžiadanie

ExecutableDialog Dialog na výber prítomnosti odkazu na spustiteľný súbor

Hashtable .NET implementácia kolekcie typu heš mapa, ktorá si pamätá prvky podľa hešu kľúča

Directory Trieda poskytujúca rozhranie na narábanie so zložkami na disku

File Používa sa na prácu so súbormi na disku

ServicePointManager Trieda v .NET starajúca sa o manažovanie HTTP spojení, použitá na overovanie https cetifikátu

WebClient Obaľovacia trieda v našom prípade na http spojenie, ale dokáže pracovať s ľubovoľným spojením cez URI

DriveInfo Poskytuje nám informácie o pevných diskoch v počítači

DiffEngine Trieda zodpovedná za implementáciu algoritmu na zistenie rozdielov medzi dvomi súbormi

FileSystemWatcher .NET rozhranie poskytujúce prístup k událostiam v súborovom systéme.

UserApp Hlavná trieda úžívateľskej aplikácie zodpovednej na inštalovanie balíkov

Process Trieda, ktorá nám pomáha spúšťať potrebné programy priamo z našej aplikácie

Registry Rozhranie na pohyb po registroch a ich čítanie a zápis

RegistryKey Obsahuje informácie o jednom z kľúčov v registroch

3.3 Ukážky častí kódu

3.3.1 Inicializácia pozorovateľov

Na každom pevnom disku spustíme pozorovateľa udalosti v súborovom systéme a nastavíme funkcie, ktoré sa majú volať pri daných udalostiach

```
1  DriveInfo[] allDrives = DriveInfo.GetDrives();
2  foreach (DriveInfo d in allDrives)
3  {
4      if (d.DriveType == DriveType.Fixed)
5      {
6          FileSystemWatcher watcher = new FileSystemWatcher();
7          watcher.Path = d.Name;
8          watcher.IncludeSubdirectories = true;
9          watcher.NotifyFilter = NotifyFilters.LastWrite
10             | NotifyFilters.FileName | NotifyFilters.DirectoryName;
11
12         watcher.Changed += new FileSystemEventHandler(OnChanged);
13         watcher.Created += new FileSystemEventHandler(OnChanged);
14         watcher.Deleted += new FileSystemEventHandler(OnChanged);
15         watcher.Renamed += new RenamedEventHandler(OnRenamed);
16
17         watcher.EnableRaisingEvents = true;
18         watchers.Add(watcher);
19     }
20 }
```

Zdrojový kód 3.1: Inicializácia

3.3.2 Kontrola registrov

Pri spustení uživateľskej aplikácie bez parametrov sa vykoná kontrola registrov, ktoré sú potrebné na inštalovanie balíkov. Ak daný kľúč neexistuje tak sa spýtame užívateľa alebo zapíšeme tam potrebnú hodnotu (ako v prípade zložky kde je program nainštalovaný)

3.3.3 Tvorba odkazov

Po kontrole registrov sa zo serveru stiahne zoznam dostupných balíčkov a vytvoria sa potrebné odkazy. Odkazu nastavíme cestu k spustiteľnému súboru našej aplikácie a ako argumenty pošleme názov balíku, ktorý ma nainštalovať a cestu k spustiteľnému súboru, ktorý sa má pustiť ak už je balík nainštalovaný

```
installDir = (string)Registry.GetValue(keyName, "installDir", "Not Exist");
if (installDir == "Not Exist")
{
    RegistryKey key = Registry.CurrentUser.OpenSubKey("Software", true);
    key = key.OpenSubKey("SetItUp", true);
    key.SetValue("installDir", Application.StartupPath+"\\UserApp.exe");
    installDir = (string)Registry.GetValue(keyName, "installDir", "Not Exist");
}
```

Zdrojový kód 3.2: Kontrola registrov

```
var wsh = new IWshRuntimeLibrary.IWshShell_Class();
path = shortcutDir + "\\" + tmp[1] + ".lnk";
IWshRuntimeLibrary.IWshShortcut shortcut = wsh.CreateShortcut(shortcutDir + "\\"
shortcut.Arguments = "/user:Wryxo-PC\\AdminTest /savecred \"" + installDir + "\"
shortcut.TargetPath = "C:\\Windows\\System32\\runas.exe";
shortcut.Save();
FileSecurity fs = File.GetAccessControl(path);
fs.AddAccessRule(new FileSystemAccessRule(adminName, FileSystemRights.FullControl
fs.AddAccessRule(new FileSystemAccessRule(userName, FileSystemRights.ReadAndExecu
fs.SetAccessRuleProtection(true, false);
File.SetAccessControl(path, fs);
```

Zdrojový kód 3.3: Vytvorenie odkazu

3.3.4 Export registrov

Registre exportujeme použitím programu regedit, ktorý vytvorí textový súbor v štandardnom formáte.

3.4 Problémy pri implementácií

Pri implementácií sa vyskytli dva väčšie problémy, ktoré by mohli spôsobiť zlé fungovanie našej aplikácie. Prvý je zapríčinený nedostatkom informácia o udalostiach generovaných triedou FileSystemWatcher. Jedna z informácií, ktorá nám chýba je identita procesu, ktorý udalosť vyvolaľ. Bez tejto informácie monitorujeme a reagujeme aj na zmeny spôsobené inými procesmi počas inštalácie a môžu nastať prípady, kedy medzi súbormi označenými ako patriace k balíku sa vyskytnú súbory, ktoré nemajú s daným balíkom nič spoločné a ocitli sa tam náhodou. Tento problém sme dočasne trochu vyriešili zavedením zakázaných slov, ktoré filtrujú najväčšie zdroje, ktoré sme pri testovaní objavili, týchto zmien ako sú napríklad dočasné súbory a cookies z prehliadača. Druhou z väčších chýb vzniká pri generovaní rozdiel medzi registrami. Na túto úlohu používa apikácia algoritmus najdenia najdlhšej spoločnej sekvencie, ktorý ako výstup vráti riadky v ktorých nastala zmena. Keďže rozdiel, ktorý chceme dostať musí spĺňať

```
1  public void ExportKey(string RegKey, string SavePath)
2          {
3              string path = "\"" + SavePath + "\"";
4              string key = "\"" + RegKey + "\"";
5
6              var proc = new Process();
7              try
8              {
9                  proc.StartInfo.FileName = "regedit.exe";
10                 proc.StartInfo.UseShellExecute = false;
11                 proc = Process.Start("regedit.exe", "/e " + path + " " + key + "
12
13                 if (proc != null) proc.WaitForExit();
14             }
15             finally
16             {
17                 if (proc != null) proc.Dispose();
18             }
19         }
```

Zdrojový kód 3.4: Exportovanie registrov

štandarný windows registry formát potrebujeme aby každá hodnota zapísaná v súbore mala hlavičku s kľúčom do ktorého patrí. Problém nastáva ak sa zmení hodnota v kľúči, ktorý už existuje. V tom momente algoritmus nájde len riadok s hodnotou ale bude mu chýbať hlavička s informáciou o kľúči ku ktorému táto hodnota patrí. Riešenie tohto problému nie je triviálne a preto sme po prieskume usúdili, že zatiaľ nie je potreba tento problém vyriešiť, keďže množstvo programov, ktoré menia vrámci inštalácie kľúče, ktoré boli vytvorené pred ňou je málo. Okrem týchto dvoch problémov sa počas implementácie nevyskytli problémy, ktoré by zabrali veľa času. Väčšina vznikala nedôslednosťou autora, ako pristupovanie k zložke cez triedu File alebo podobne. Jedna z posledných vecí, čo môže byť problém je overovanie certifikátu servera, zatiaľ je tento problém vyriešený funckiou, ktorá vždy vracia úspešné overenie certifikátu.

3.5 Možné zlepšenia v budúcnosti

Služba Užívateľská aplikácia by sa dala pretransformovať na službu do operačného systému, ktorej by odkazy posielali príkazy na inštalovanie alebo odinštalovanie balíku. Takisto by vedela periodický zisťovať od serveru zmeny v zozname balíkov a tento zoznam dynamický upravovať.

Aktualizácie Pridať balíkom informáciu o ich aktuálnej verzií. Následne pri každom spustení sa skontroluje lokálna verzia balíka s verziou na serveri, v prípade nezhody

by sa stiahla verzia zo serveru a nainštalovala sa. Takisto by trebalo do administráčného programu pridať možnosť aktualizovať určitý balík, čím by aplikácia automatický zvýšila hodnotu verzie balíku

Odinštalovanie Vytvoriť algoritmus, ktorý nájde všetky súbory a registre spojené s daným balíkom a odstráni ich, takisto upozorní používateľa o možných chýbach v programoch závislých od daného balíčku.

Sériové kódy Pridať do administračnej aplikácie nástroje na manažovanie sériových kódov. Administrátor by vedel k balíku pridať množinu sériových kódov, ktoré by program manažoval a každej stanici, ktorá by si balík inštalovala poslal jeden kľúč z množiny nepoužitých. Týmto by sa vyriešila otázka licencovania. Toto riešenie by avšak vyžadovalo zistenie umiestnenia kľúča po inštalácii a jeho zmenu podľa potreby.

KAPITOLA 4

Dokumentácia

4.1 Inštalácia

4.1.1 Prerekvizity

Aplikácia vyžaduje aby na každej podstanici bol nainštalovaný .NET framework verzia 4.0, v ktorom je aplikácia vyvíjaná. Táto verzia frameworku je automatický nainštalovaná v operačnom systéme Windows 7 alebo Windows 8. Ďalej aplikácia potrebuje aby stanica bola na spoločnej sieti so serverom, z ktorého bude sťahovať balíky. Tento server musí mať zložku s balíkmi prístupnu cez HTTP protokol a zároveň mať nainštalovaný SSL certifikát, ktorým sa bude identifikovať užívateľovi pri požiadavke na stiahnutie balíku.

4.1.2 Aplikácia

Zdrojovy kód aplikácie je dostupný na stránke https://github.com/Wryxo/bakalarka. Avšak je doporučené stiahnuť si inštalačný program zo stránky <TODO>, ktorý nainštaluje všetky potrebné súbory našej aplikácie a spustí prvotnú konfiguráciu. Takisto vytvorí odkaz po spustení, ktorý stiahne zo serveru obnovený zoznam balíkov.

4.2 Konfigurácia

Po skončení inštalácie sa spustí aplikácia, ktorá si vypýta cesty k zložkám potrebným pre správne fungovanie a adresu súborov na servery. Po konfigurácií sa aplikácia pokúsi stiahnuť zoznam balíčkov zo serveru, ktorý administrátor zadal.

4.3 Spúšťanie

V inštalašnej zložke programu sa nachádzajú 2 spustiteľné súbory:

- setitup-admin.exe - slúži pre administrátora na vytváranie balíkov. Vyžaduje administrátorské práva na spustenie, kvôli sledovaniu události na súborom systéme
- setitup.exe - aplikácia, ktorá sa spúšťa keď užívateľ použije odkaz vytvorený našou aplikáciou. Aplikácia nainštaluje potrebný balík alebo spustí program vyžiadaný od užívateľa

4.4 Správa balíkov

4.4.1 Vytvorenie balíku

Po spustení administrátorskej aplikácie, sa zobrazí rozhranie v ktorom administrátor dokáže spustiť sledovanie inštalácie a kontrolovať v konzole priebeh sledovania a chybové správy. Po skončení sledovania bude prezentovaný formulárom na výber odkazov, ktoré majú byť dostupné užívateľovi, výberom balíkov na ktorých je práve vytváraný balík závislý a zvolením kedy má byť balík inštalovaný.

4.4.2 Zoznam balíkov

Zoznam balíkov je zapísaný v súbore na servery alebo v zložke s balíkmi. Administrátor môže manuálne meniť nastavenia balíkov, zmenou textových súborov v zložke s balíkmi. V týchto konfiguračných súbroch je zapísany zoznam balíkov, odkazov, ktoré sa majú vytvoriť a dokonca aj zoznam súborov určitého balíku.

4.4.3 Inštalácia balíku

Užívateľ môže nainštalovať balík spustením odkazu zo zložky s odkazmi. Tento odkaz zistí, či už je daný balík nainštalovaný a podľa toho sa rozhodne o ďalšom kroku. Ak balík ešte nebol nainštalovaný aplikácia stiahne potrebné súbory zo serveru a nakopíruje ich na potrebné miesto. V prípade, že balík bol nainštalovaný aplikácia spustí požadovaný program. Toto celé sa vykoná bez vedomia užívateľa.

Literatúra