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**

77900676

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Strapko
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Systém automatickej distribúcie softvéru na pracovné stanice s OS Windows 7 / *Automatic Software Distribution System for Windows 7 Workstations*

Cieľ: Cieľom práce je analyzovať možnosti automatizovanej inštalácie softvéru, konfigurácie a vykonávania definovaných úloh na pracovných staniciach s Windows 7 a navrhnúť a implementovať systém, ktorý umožní automatickú inštaláciu softvéru na základe vzorovej inštalácie na jednom počítači. Systém má umožniť inštalovať softvér "na požiadanie", ako aj automaticky po štarte počítača. Zároveň má umožniť centrálne nastavovanie konfiguračných parametrov a vykonávanie definovaných úloh. Zmyslom výsledného diela je automatizácia práce administrátora veľkého počtu pracovných staníc. Pri návrhu a implementácii systému je potrebné zohľadniť aj požiadavky na bezpečnosť.

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 19.10.2012

Dátum schválenia: 21.10.2013 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Zatiaľ sebe.

Abstrakt

Slovenský abstrakt.

Abstract

English abstract.

Obsah

Úvod **1**

1 Špecifikácia problému a návrh riešenia **2**
1.1 Zadefinovanie Problému 2
1.2 Návrh riešenia 2
1.2.1 Príprava balíčkov 3
1.2.2 Inštalácia balíčkov 4
1.2.3 Prenos balíčkov 4
1.2.4 Sériové kľúče 5
1.2.5 Užívateľské nastavenia 5
1.2.6 Kategorizovanie inštalácií 5

2 Technológie **7**
2.1 C# 7
2.2 HTTPS 7

3 Implementácia **8**
3.1 Štruktúra aplikácie 8
3.1.1 Administrátor 8
3.1.2 Používateľ 8
3.1.3 Služba 8
3.1.4 Konfigurácia 9
3.1.5 Zmeny súborového systému 9
3.1.6 Rozdiel registrov 10
3.1.7 Inštalácia balíčkov 10
3.2 Ukážky častí kódu 11
3.2.1 Inicializácia pozorovateľov 11
3.2.2 Kontrola registrov 11
3.2.3 Tvorba odkazov 11
3.2.4 Logovanie chýb služby 12
3.2.5 Export registrov 13
3.2.6 Redundancia v registroch 13

3.3 Problémy pri implementácii 13

4 Dokumentácia **16**
4.1 Inštalácia 16
4.1.1 Požiadavky 16
4.1.2 Detailný postup inštalácie 16
4.2 Konfigurácia 17
4.2.1 packages.txt 17
4.2.2 depedencies.txt 18
4.2.3 <názov balíku>.txt 18
4.2.4 blacklist.txt 18
4.2.5 Registre 19
4.3 Spúšťanie 19
4.3.1 Administrátor 19
4.3.2 Používateľ 19
4.4 Správa balíkov 19
4.4.1 Vytvorenie balíku 19
4.4.2 Zoznam balíkov 20
4.4.3 Inštalácia balíku 20

5 Diskusia **21**
5.1 Možné zlepšenia v budúcnosti 21
5.2 Experiment 22

Záver **24**

Literatúra **25**

Zoznam listingov

3.1 Inicializácia 11
3.2 Kontrola registrov 12
3.3 Vytvorenie odkazu 12
3.4 Logovanie sluzby 12
3.5 Exportovanie registrov 13
3.6 Zavedenie redundancie 14
3.7 Pred a po zavedení redundancie 15
4.1 Štruktúra packages.txt 17

Zoznam obrázkov

4.1 Výber typu inštalácie 18
4.2 Výber závislostí 18
4.3 Hlavné okno 19
4.4 Výber odkazov na vytvorenie 20

Úvod

Pravidelnou prácou systémového administrátora je inštalácia a príprava počítačov na prácu pre ľudí menej zbehlých v informačných technológiách. Toto zahŕňa všetko od umiestnenia a zapojenia počítača do elektrickej siete až po inštaláciu programov.

V tejto bakalárskej práci sa budeme zaoberať riešením problematiky inštalácie počítačového softvéru na väčšie množstvo staníc. Existuje viacero spoločností, ktoré vyvíjajú nástroje na riešenie tohto problému. Po dlhšom výskume sme zistili, že každé riešenie má svoje plusy a mínusy, a záleží na potrebách inštitúcie, ktoré riešenie zvolí. Avšak ani jedno z týchto riešení nie je vhodné pre našu prácu, keďže našim cieľom je vytvorenie jednoduchej aplikácie, ktorá má minimálne požiadavky na nastavenia. Z pomedzi známejších mien v tomto obore by sme spomenuli ako príklad **ZENworks 10 Configuration Management** od firmy Novell, ktoré na svoju prácu vyžaduje špeciálny server, čo pre mnoho menších inštitúcii ako sú základné a stredné školy je takmer nesplniteľná požiadavka. Práve tieto inštitúcie, ktoré nepotrebujú ťažkopádne riešenia sú cieľovou skupinou našej aplikácie.

Cieľom tejto bakalárskej práce je analýza problému inštalácie, popísať možné riešenia tohto problému a implementovať jednoduchú aplikáciu, ktorá bude túto problematiku riešiť. Orientovať sa budeme na platformu Windows 7, keďže mnoho firiem a školských inštitúcii pracuje práve s týmto operačným systémom. 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 je často dôležitým nástrojom produktívneho zamestnanca vo firme počítač. Avšak aj napriek tomuto faktu mnoho z týchto zamestnancov sa do počítačov nevyzná a vyžadujú pomoc firemného administrátora pri inštalácii počítača alebo niektorých programov. Táto úloha je síce z pohľadu informatika triviálna, avšak zaberá obrovské množstvo vynaloženého času vzhľadom na jej nízku obtiažnosť. Administrátor musí stráviť niekoľko dní nad prípravou novokúpených počítačov, pred tým ako ich odovzdá zamestnancom. Takisto v školách sa začínajú hromadne využívať počítače. Tu vzniká problém, že študenti sú zvedaví a ich zvedavosť často končí počítačom v nefunkčnom stave. Znova nainštalovanie počítaču z obrazu, ktorý obsahuje všetky programy by zabralo zbytočne veľa času, preto sa naša aplikácia sústredí na koncept inštalácie programov na vyžiadanie bez prítomnosti administrátora. Problém, ktorému sa v tejto práci budeme venovať a ktorého výsledný program ho má čo najefektívnejšie riešiť, je ako automatizovať čo najviac práce s prípravou počítaču do funkčného stavu. Najčastejšie sa tento problém rieši pomocou klonovania disku, avšak toto riešenie je použiteľné len vtedy ak chceme mať rovnakú sadu programov na každej stanici. Takisto v takomto obraze disku sa ťažko robia dodatočné zmeny.

1.2 Návrh riešenia

Bohužiaľ inštalácie na operačnom systéme Windows nie sú homogénne. Dali by sa kategorizovať, bližšie popísane v sekcii 1.2.6, avšak výroba aplikácie, ktorá by vedela automaticky 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 používateľa nainštaluje. Toto riešenie vyrobí balíčky, ktoré následne presunie na cieľovú stanicu, kde sa rozbalia a tým nainštalujú želané programy. Celý tento proces môže byť pre používateľa neviditeľný. Naša aplikácia bude takisto umožňovať inštaláciu balíka na vyžiadanie. Vďaka tejto funkcionalite základ systému, ktorý sa nainštaluje napríklad klonovaním, môže byť menší. Implementáciu a finálnu aplikáciu vieme rozdeliť na tieto časti:

- Odchytenie a uloženie priebehu inštalácie do balíčka
- Inštalácia balíčka na cieľovej stanici

1.2.1 Príprava balíčkov

Ako prvé potrebujeme vytvoriť balíčky, ktoré naša aplikácia neskôr použije na inštalovanie programov. Aby sme vedeli inštalácie čo najpresnejšie zopakovať potrebujeme si pamätať súbory a registry, ktoré inštalácia vytvorila alebo upravila, zoznam programov, na ktorých je balík závislý a zoznam spustiteľných súborov, ktoré používa program obsiahnutý v našom balíku.

Porovnanie stavu pred a po inštalácií Tento prístup k problému pozostáva z vytvorenia 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 veľkosti pevných diskov sa pohybujú v stovkách gigabajtov, je tento proces zdĺhavý a neefektívny.

Pomocou zabalenia inštalácie Ďalším možným riešením zisťovania zmien, ktoré inštalácia vykonala, je vytvorenie aplikácie, ktorá obalí inštalačný program a bude reagovať na jeho volania operačného 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 zachytené pôvodnou aplikáciou.

Zachytením zmien na súborovom systéme Posledné riešenie, ktoré spomenieme a ktoré budeme implementovať, je pomocou štandardných knižníc .NET framework 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 v registroch, 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, načítať hodnoty registrov podľa výpisu v balíčku a vytvoriť odkazy na spustiteľné súbory, ktoré by používateľa 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 má pripraviť pre používateľa, a neskôr pri inštalácií bude zo serveru sťahovať súbory inštalovaného balíku. Čo sa týka obnovovania zoznamu balíkov na serveri, bude administrátorskou 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 zadanie 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, rozhodli sme sa 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 používateľskej zložky umiestnenej na systémovom disku - C:\Users\<meno používateľa>\. Táto zložka sa samozrejme mení podľa používateľa a preto treba aj pri inštalácií balíčka na toto myslieť a nainštalovať ho do používateľskej zložky práve prihláseného používateľa. Toto vyriešime jednoducho keď pri zapisovaní cesty skontrolujeme či neukazuje do používateľskej zložky a nahradíme ju našou značkou, ktorú neskôr nahradíme práve prihláseným používateľom.

1.2.6 Kategorizovanie inštalácií

Inštalácie v operačnom systéme Windows by sa dali rozdeliť do 3 kategórií:

Rozbalenie archívu Najjednoduchším typom inštalácií sú programy, ktoré sa šíria v archívoch typu .zip, .rar alebo .tar.gz. Pre nainštalovanie takéhoto programu, nám stačí rozbaliť archív na správne miesto pomocou jedného z mnohých špecializovaných programov.

Windows Installer Komponent určený na inštaláciu, údržbu a odstraňovanie softvéru na systéme Windows. Informácie o inštalácií a súbory programu sú zabalené v inštalačných balíkoch, ktoré sú vďaka ich koncovke známe ako MSI súbory. Pri tvorbe

MSI balíku môže programátor pripraviť používateľské rozhranie v ktorom dopytuje informácie, ktoré sa neskôr použijú pri inštalácií ako sú dôležité cesty, časti balíku ktoré by používateľ chcel nainštalovať a podobne. Ďalšia vlastnosť týchto balíkov je odvolanie zmien v prípade chyby alebo zrušenia inštalácie na pokyn používateľa. Pre všetky štandardné operácie nad systémom má balík vygenerované opačné operácie, ktoré vrátia systém do stavu pred inštaláciou. Tieto balíky obsahujú mnohé ďalšie funkcie ako tichú inštaláciu, bez zobrazenia používateľského rozhrania, inštaláciu na vyžiadanie, balík vytvorí odkaz na program, avšak samotný program nainštaluje až keď sa používateľ rozhodne daný program spustiť. Takisto je možné počas inštalácie spúšťať vlastné skripty, napríklad na kontrolu sériového kľúča.

Vlastná inštalácia Mnohé spoločnosti, ktoré vyvíjajú softvér, si implementujú vlastné inštalačné programy. Tieto programy sú navrhnuté špeciálne pre danú aplikáciu a často vyvíjané v programovacích jazykoch s natívnou podporou na rôznych operačných systémoch. Mávajú v sebe zakomponované algoritmy na kontrolu sériových kľúčov, overovacie požiadavky na vzdialené serveri ako ochranu proti pirátstvu a ďalšie funkcie prispôsobené požiadavkám spoločnosti.

KAPITOLA 2

Technológie

2.1 C#

C# sme zvolili kvôli podpore .NET framework. Ten nám poskytuje knižnice na narábanie s mnohými časťami operačného systému Windows. Programy napísané v tomto frameworku bežia v aplikačnom virtuálnom stroji, ktorý sa stará o bezpečnosť, správu pamäte a zachytávanie výnimiek. Poskytuje nám triedy na prácu so súbormi, zložkami a registrami operačného systému. Vďaka ním dokážeme zachytiť a neskôr zopakovať inštalácie programov.

2.2 HTTPS

Zabezpečený hypertextový prenosový protokol je zabezpečená verzia komunikačného protokolu HTTP. Namiesto používania textovej komunikácie, komunikácia protokolu HTTPS je šifrovaná pomocou protokolu SSL alebo TLS. Táto komunikácia je závislá na digitálnych certifikátoch, bez ktorých sa stráca bezpečnosť protokolu HTTPS. V našej práci budeme tento protokol používať na prenos balíkov a konfiguračných súborov medzi používateľskými stanicami a serverom. Na šifrovanie použijeme protokol SSL.

KAPITOLA 3

Implementácia

3.1 Štruktúra aplikácie

Aplikácia bude mať tri zložky. Administrátorskú časť zodpovednú za vytváranie balíkov, používateľskú časť zodpovednú za rozhodovanie, či treba balík nainštalovať alebo spustiť výsledný program a nakoniec službu, ktorá sa bude starať o inštalovanie súborov a registrov z balíčkov na požiadavku od užívateľskej aplikácie.

3.1.1 Administrátor

Po spustení tejto časti aplikácie, bude mať administrátor možnosť spustiť sledovanie zachytávania zmien. Takisto bude mať prístup k informačnému výpisu, v ktorom sú informácie o jednotlivých zachytených súboroch a prípadných chybách. Po ukončení sledovania, bude mať administrátor možnosť vybrať odkazy, ktoré sa majú vytvoriť, balíky na ktorých je daný balík závislý a typ inštalácie.

3.1.2 Používateľ

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 na čo sú potrebné administrátorské práva, táto aplikácia zavolá službu s danými parametrami.

3.1.3 Služba

Táto služba bude zaregistrovaná do systému pri nainštalovaní našej aplikácie a bude sa spúšťať pri zapnutí stanice. Jej úlohou je počúvať na požiadavky od používateľskej aplikácie a následne nakopírovať súbory a registre z balíčka na miesto kam patria.

3.1.4 Konfigurácia

Všetky nastavenia, ktoré aplikácia potrebuje budú uložené vo Windows registroch, pri prvom spustení užívateľskej časti si aplikácia všetky tieto nastavenia vypýta, neskôr budú meniteľné len pomocou programu regedit. Tieto nastavenia zahŕňajú:

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

3.1.5 Zmeny súborového systému

Základom našej aplikácie je .NET trieda FileSystemWatcher, ktorú na začiatku inicializujeme a nastavíme na počúvanie udalosti na všetkých pevných diskoch počítača. Táto trieda generuje udalosti na ktoré reaguje zvyšok programu. Existujú dve kategórie udalostí na ktoré treba reagovať. Prvá kategória zahrňuje vytvorenie, zmenu a zmazanie súboru. Druhá kategória sa stará o premenovanie súboru, ktoré sa uskutoční iba pri zmene názvu súboru.

Zmena súboru

Pri tomto type udalostí najprv zistíme, ktorá z troch vecí prvej kategórie nastala. Pri vytvorení súboru si zapíšeme miesto, kde bol tento súbor vytvorený, do zoznamu súborov tohto balíku. Túto informáciu využijeme neskôr pri kopírovaní súborov do balíku a inštalácií balíku. Ak bola táto udalosť volaná 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 zálohovali 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 súborov, ktoré doteraz monitorujeme, si pamätáme v hašovacej mape, keďže nám poskytuje rýchle riešenie vyhľadávania a kontroly prítomnosti prvku v poli.

Premenovanie súboru

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

3.1.6 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 používateľovi 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. Keďže tento algoritmus nefunguje efektívne, rozhodli sme sa registre exportovať po jednotlivých kľúčoch prvé stupňa. Týmto kompletné porovnanie registrov pozostáva z veľa porovnaní na malých súboroch. Pri generovaní výsledného zoznamu registrov pred každú zmenenú hodnotu napíšeme kľúč do ktorého patrí. Táto redundancia nám ošetrí problém zmeny hodnoty bez vytvorenia kľúča.

3.1.7 Inštalácia balíčkov

Na každej stanici bude po štarte systému spustená naša služba, ktorá ma za úlohu skontrolovať potrebné cesty v registroch, udržovať aktuálny zoznam balíčkov zo serveru a nainštalovať vyžiadaný balíček. V rámci inštalácie, aplikácia prejde cez zoznam súborov, ktoré daný balíček obsahuje a skontroluje, či existujú na miestach kde majú. Ak zistí že niektorý súbor chýba, tak ho nakopíruje z balíku na jeho miesto v systéme. Táto časť aplikácia berie pri spustení 0, 1 alebo 2 parametre. Odkazy vytvorené našou aplikáciou spustia spustiteľný súbor UserApp.exe s potrebnými parametrami.

0 parametrov

Pri spustení aplikácie bez parametrov, aplikácia skontroluje Windows registre a zistí, či obsahujú všetky potrebné údaje pre aplikáciu (cesty k dôležitým zložkám).

1 parameter

Ak do aplikácie zadáme len jeden parameter, tento parameter musí byť názov existujúceho balíčku, ktorý aplikácia pošle ako požiadavku službe, ktorá sa rozhodne či treba daný balíček nainštalovať

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 Ukážky častí kódu

3.2.1 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

3.2.2 Kontrola registrov

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

3.2.3 Tvorba odkazov

Po kontrole registrov sa zo serveru stiahne zoznam dostupných balíčkov a vytvoria sa potrebné odkazy. Každému odkazu nastavíme cestu k spustiteľnému súboru našej aplikácie a ako argumenty pošleme názov balíku, ktorý sa má 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);
    installDir = (string)Registry.GetValue
                        (keyName, "installDir", "Not Exist");
}
```

Zdrojový kód 3.2: Kontrola registrov

```
path = shortcutDir + "\\" + tmp[1] + ".lnk";
var wsh = new IWshRuntimeLibrary.IWshShell_Class();
IWshRuntimeLibrary.IWshShortcut shortcut =
    wsh.CreateShortcut(shortcutDir + "\\" + tmp[1] + ".lnk")
    as IWshRuntimeLibrary.IWshShortcut;
shortcut.Arguments = package + " \"" + tmp[2] + "\"";
shortcut.TargetPath = installDir + "UserApp.exe";
shortcut.Save();
```

Zdrojový kód 3.3: Vytvorenie odkazu

3.2.4 Logovanie chýb služby

V rámci inicializácie služby, ktorá má za úlohu inštalovať balíky, nastavíme tejto službe zapisovanie udalostí do Windows logu, kde ich v prípade potreby môže administrátor nájsť.

```
if (!System.Diagnostics.EventLog.SourceExists("SetItUp"))
{
    System.Diagnostics.EventLog.CreateEventSource(
        "SetItUp", "DebugLog");
}
eventLog1.Source = "SetItUp";
eventLog1.Log = "DebugLog";
eventLog1.WriteEntry("Spustam sluzbu");
```

Zdrojový kód 3.4: Logovanie sluzby

3.2.5 Export registrov

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

```
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
12                           ("regedit.exe", "/e " + path + " " + key + "");
13
14                 if (proc != null) proc.WaitForExit();
15             }
16             finally
17             {
18                 if (proc != null) proc.Dispose();
19             }
20         }
```

Zdrojový kód 3.5: Exportovanie registrov

3.2.6 Redundancia v registroch

Na ošetrenie prípadov, kedy sa zmenila hodnota v kľúči, ktorý existoval už pred danou inštaláciou, zavádzame do výsledného zoznamu registrov redundanciu. Ukážeme si kus kódu, ktorý prechádza výsledkom diferenčného algoritmu a zapisuje výsledný súbor so zoznamom registrov.

V nasledujúcej ukážke z balíka pre webový prehliadač Mozilla Firefox vidíme rozdiel medzi štandardným výpisom z registrov a nami upraveným výpisom v ktorom sme adresu kľúča zapísali pred každou hodnotou.

3.3 Problémy pri implementácii

Pri implementácii sa vyskytli dva väčšie problémy, ktoré by mohli spôsobiť zlé fungovanie našej aplikácie. Prvý je zapríčinený nedostatkom informácií o udalostiach generovaných triedou FileSystemWatcher. Jedna z informácií, ktorá nám chýba je identita

```
foreach (DiffResultSpan drs in DiffLines)
{
    switch (drs.Status)
    {
        case DiffResultSpanStatus.AddDestination:
            for (i = 0; i < drs.Length; i++)
            {
                tmp = ((TextLine)destination.GetByIndex(drs.DestIndex + i)).Line
                if (tmp != "") {
                    if (tmp.StartsWith("[HKEY_"))
                    {
                        lastKey = tmp;
                    }
                    else if (tmp.StartsWith("\"") || tmp.StartsWith("@"))
                    {
                        res.Add(lastKey);
                        res.Add(tmp);
                        res.Add("");
                    }
                    else
                    {
                        res.Add(tmp);
                    }
                }
            }
            break;
        case DiffResultSpanStatus.NoChange:
            for (i = 0; i < drs.Length; i++)
            {
                tmp = ((TextLine)destination.GetByIndex(drs.DestIndex + i)).Line
                if (tmp.StartsWith("[HKEY_")) lastKey = tmp;
            }
            break;
    }
}
```

Zdrojový kód 3.6: Zavedenie redundancie

procesu, ktorý udalosť vyvolal. 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 týchto zmien, ktoré sme pri testovaní objavili, ako sú napríklad dočasné súbory a cookies z prehliadača. Tento zoznam je uložený v súbore blacklist.txt nachádzajúcom sa v zložke s balíkmi. Ďalšia z väčších chýb vznikla pri generovaní rozdielu medzi registrami. Na túto úlohu

```
##### PRED ZAVEDENIM REDUNDANCIE #####
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\FirefoxURL]
@="Firefox URL"
"FriendlyTypeName"="Firefox URL"
"URL Protocol"=""
"EditFlags"=dword:00000002

##### PO ZAVEDENI REDUNDANCIE #####
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\FirefoxURL]
@="Firefox URL"

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\FirefoxURL]
"FriendlyTypeName"="Firefox URL"

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\FirefoxURL]
"URL Protocol"=""

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\FirefoxURL]
"EditFlags"=dword:00000002
```

Zdrojový kód 3.7: Pred a po zavedení redundancie

používa aplikácia algoritmus nájdenia najdlhšej spoločnej sekvencie, ktorý ako výstup vráti riadky v ktorých nastala zmena. Keďže rozdiel, ktorý chceme dostať musí spĺňať štandardný formát Windows registra, 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. Tento problém sme vyriešili pomocou zapisovanie adresy kľúča ku každej hodnote, ktorá sa v danom kľúči nachádza. Ukážku riešenia tohto problému sme spomenuli v zdrojovom kóde 3.6. 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 a podobne.

KAPITOLA 4

Dokumentácia

4.1 Inštalácia

4.1.1 Požiadavky

Aplikácia vyžaduje aby na každej stanici bol nainštalovaný .NET framework verzia 4.5, v ktorom je aplikácia vyvíjaná. Ďalej aplikácia potrebuje aby stanica dokázala komunikovať pomocou sieťovej komunikácie so serverom, z ktorého bude sťahovať balíky. Tento server musí mať zložku s balíkmi prístupnú cez HTTP protokol a zároveň musí mať nainštalovaný SSL certifikát, pomocou ktorého sa bude identifikovať používateľovi pri požiadavke na stiahnutie balíku.

4.1.2 Detailný postup inštalácie

V tejto sekcii dôkladne popíšeme postup, ktorým uvedieme aplikáciu do funkčného stavu.

1. Z priloženého CD skopírujeme alebo stiahneme z internetu súbor SetItUp.zip
2. Tento súbor rozbalíme do zložky, kde chceme mať našu aplikáciu nainštalovanú
3. S administrátorskými právami spustíme UserApp.exe a vyplníme cesty k významným priečinkom - priečinok s balíkmi, priečinok s odkazmi, adresa vzdialeného serveru
4. Spustíme SetItUpService.exe s argumentom install, ktorý nám do systému nainštaluje potrebnú službu. Taktiež vyžaduje administrátorské práva
5. Následne spustíme Administration.exe a môžme začať monitorovať inštalácie programov

6. Po každom vytvorenom balíku odkopírujeme súbor packages.txt a <názov balíku>.zip na server. Súbor packages.txt obsahuje všetky balíky, ktoré boli zapísané na serveri pri spustení aplikácie nasledované balíkmi, ktoré sme vytvorili v rámci tejto inštancie aplikácie.

4.2 Konfigurácia

Aplikácia si k správnemu behu potrebuje nakonfigurovať cesty k dôležitým priečinkom ako je priečinok s balíčkami, s odkazmi a adresa serveru na ktorom sú uložené balíky. Táto konfigurácia prebehne pri prvom spustení aplikácie pomocou UserApp.exe, ktorá si vypýta cesty k týmto zložkám. Program taktiež udržuje dôležité informácie o balíčkoch v 3 konfiguračných súboroch. Zároveň existuje štvrtý konfiguračný súbor, ktorý nesie informáciu o slovách, ktoré ignorujeme pri tvorbe balíku.

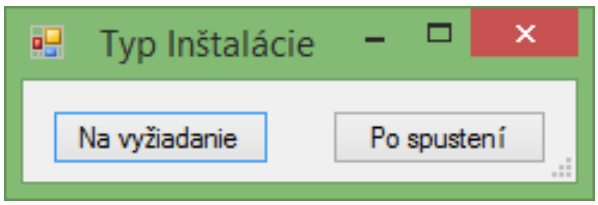
4.2.1 packages.txt

Tento súbor je uložený na serveri a obsahuje zoznam všetkých balíčkov. Kópie na používateľských počítačoch sa pri každom spustení služby alebo aplikácie Administration nahradia tou zo serveru. V tomto súbore sú zapísané aj odkazy na balíčky, ktoré sa majú vytvoriť a cesta k spustiteľnému súboru v prípade, že balíček je nainštalovaný. Odkazy sú zapísané pri zozname balíčkov, aby služba, ktorá sa stará o vytváranie odkazov, nemusela rozbaľovať všetky balíky. V súbore namiesto medzier používame znaky ~~, aby sme vedeli rozlíšiť medzery vytvorené našim programom a medzery v názve súboru alebo priečinku. V prvom riadku máme písmeno P ako identifikátor

```
1 pm~~PowerISO
2 s~~PowerISO~~D:\Bakalar\Power ISO\PowerISO.exe
3 pa~~netFramework
```

Zdrojový kód 4.1: Štruktúra packages.txt

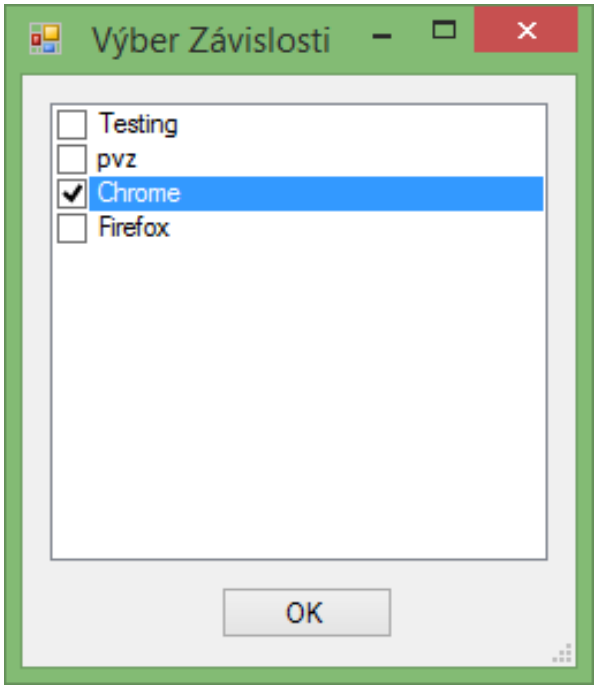
riadku s názvom balíku a následne samotný názov balíku. Písmeno M alebo A, ktoré sa nachádza na druhom mieste, označuje či je balík nainštalovaný na vyžiadanie alebo automaticky pri štarte systému. Po tomto riadku nasledujú riadky označené písmenom S, ktoré popisujú aké odkazy majú byť vytvorené k danému balíku. Ako prvý je zadaný názov odkazu, ktorý sa ma vytvoriť a za ním je zapísaná cesta k spustiteľnému súboru reprezentovanému týmto odkazom.



Obr. 4.1: Výber typu inštalácie

4.2.2 depedencies.txt

V tomto súbore, ktorý sa nachádza v každom balíku zvlášť, sú zapísané informácie o balíkoch, ktoré musia byť nainštalované spolu s daným balíkom. Jeho štruktúra je jednoduchá, na každom riadku je 1 názov balíku, ktorý musí byť nainštalovaný.



Obr. 4.2: Výber závislostí

4.2.3 <názov balíku>.txt

Je to hlavný súbor v balíku, ktorý v sebe drží informácie o miestach kam majú byť súbory prenesené. Jeho štruktúra je taktiež jednoduchá, na každom riadku si pamätá cestu k jednému súboru. Keď z tejto cesty odoberieme názov disku, dostaneme relatívnu cestu k danému súboru v zložke balíku.

4.2.4 blacklist.txt

Je súbor, ktorý sa nachádza lokálne v zložke s balíkmi, potrebný pri vytváraní balíkov. V ňom nájdeme zoznam slov, ktorých výskyt v ceste k súboru spôsobí, že daný súbor nebude zapísaný medzi súbory daného balíku. V momente kontroly týchto slov sa všetko

zmení na malé písmena, preto aj v tomto konfiguračnom súbore je všetko zapísané malými písmenami.

4.2.5 Registre

Počas tvorby balíku aplikácia kontroluje zmeny vo Windows registroch v podstrome *HKEY_LOCAL_MACHINE/SOFTWARE/*, všetky zmeny budú zapísané v súboroch v priečinku SetItUp$_R$*egistryuloenomvbalku.Takistomonitorujeme HKEY_CURRENT_USER/S*

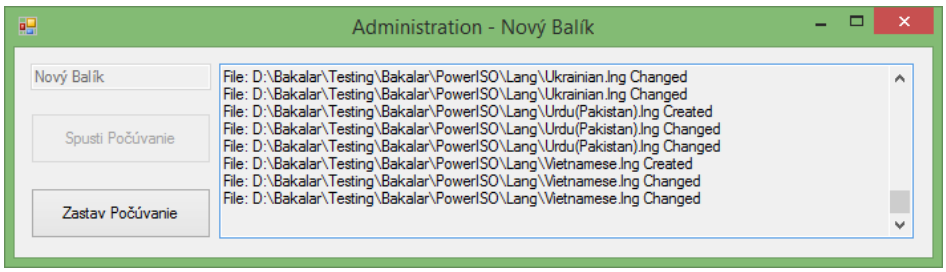
4.3 Spúšťanie

4.3.1 Administrátor

Po nainštalovaní aplikácie spustí UserApp.exe a SetItUpService.exe aby nakonfiguroval aplikácie a pripravil službu, ktorá sa stará o aktualizáciu zoznamu balíkov a odkazov. Následne bude pomocou Administration.exe vytvárať balíky, ktoré budú neskôr prístupné používateľovi.

4.3.2 Používateľ

V ponuke Štart bude mať špeciálne odkazy na programy, pomocou ktorých sa mu daná program nainštaluje a následne spustí. Ostatok aplikácie vyžaduje administrátorské práva a preto s ním obyčajný používateľ nebude môcť narábať.

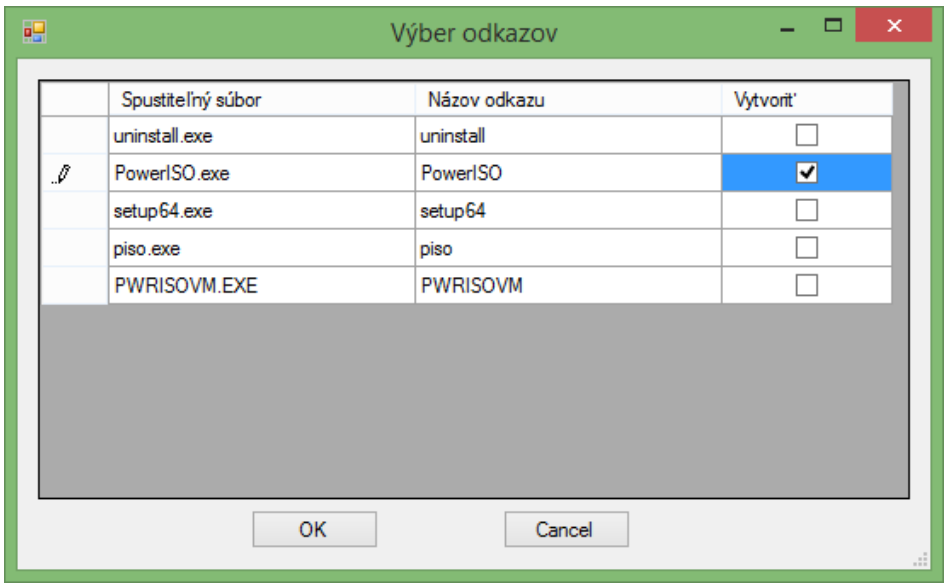


Obr. 4.3: Hlavné okno

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 používateľovi, výberom balíkov na ktorých je práve vytváraný balík závislý a zvolením kedy má byť balík inštalovaný.



Obr. 4.4: Výber odkazov na vytvorenie

4.4.2 Zoznam balíkov

Zoznam balíkov je zapísaný v súbore na serveri. Lokálne sa tento zoznam aktualizuje pri každom spustení služby a časti aplikácie Administration.exe. Administrátor môže manuálne meniť nastavenia balíkov, zmenou textových súborov v zložke s balíkmi. V týchto konfiguračných súboroch je zapísaný zoznam balíkov, odkazov, ktoré sa majú vytvoriť a zoznam zakázaných slov pri počúvaní. Takisto v zložke s balíkom nájdeme súbory so zoznamom súborov daného balíku a závislosťami, ktoré daný balík má.

4.4.3 Inštalácia balíku

Používateľ 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 informuje službu, ktorá nakopíruje potrebné súbory na určené miesto. V prípade, že balík bol nainštalovaný, aplikácia spustí požadovaný program.

KAPITOLA 5

Diskusia

5.1 Možné zlepšenia v budúcnosti

Automatizácia procesu nahrávania súborov na server Po vytvorení balíku je potrebné aby administrátor nahral daný balík a súbor packages.txt na server. Tento proces by sa dal taktiež automatizovať, avšak vyžadoval by aplikáciu na strane serveru, ktorá by požiadavky na nahranie balíku spracovávala.

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 bolo treba do administrátorského programu pridať možnosť aktualizovať určitý balík, čím by aplikácia automaticky 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 chybách v programoch závislých od daného balíčku.

Sériové kódy Pridať do administrátorskej 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 licencií. Toto riešenie by avšak vyžadovalo zistenie umiestnenia kľúča po inštalácii a jeho zmenu podľa potreby.

Lokalizácia Zatiaľ všetky výpisy a názvy v aplikácií sú napísané v slovenčine. Do budúcnosti by sa dalo všetky napevno zadefinované reťazce prepísať do konštánt, ktoré by umožnili jednoduchú úpravu do jazyka podľa vyžiadania. V jazyku C# na to slúžia zdroje, sú to súbory s koncovkou .resx v ktorých sú uložené konštanty ku ktorým sa dá

pristupovať z ostatných častí programu. Do názvu týchto súborov sa pridáva aj skratka jazyku ku ktorému patria (sk, en, fr), pomocou ktorej program rozlišuje, ktoré reťazce sa použijú.

Ikony programov V aktuálnej verzií aplikácie, všetky nami vytvorené odkazy majú ikonu preddefinovanú systémom Windows. Jeden z problémov pri implementácii tejto vlastnosti je nájdenie cesty k pôvodnej ikone programu a odkopírovanie tejto ikony spolu s programom.

Použitie MSI balíkov Tak ako bolo spomenuté v sekcií 1.2.6 MSI balíky majú mnoho vlastností, ktoré by sa dali použiť v našej aplikácii. Preto v budúcnosti by sme mohli vytvárať balíky typu MSI vďaka čomu namiesto funkcií, ktoré sme implementovali my, budeme používať funkcie implementované spoločnosťou Microsoft.

5.2 Experiment

V rámci finalizácie aplikácie sme sa rozhodli otestovať aplikáciu na vzorke programov, ktoré sú bežnou súčasťou školského počítača. Inštalovali sme následovné programy:

- Google Chrome
- Mozilla Firefox
- Opera
- Microsoft Office 2010
- OpenOffice 4
- Gimp
- Blender
- Eclipse

Balíčky sme vytvárali na stanici s operačným systémom Windows 8.1 a inštalovali sme ich na stanicu s operačným systémom Windows 7. Oba počítače boli pred experimentom plne funkčné a niektoré vyššie spomenuté programy na nich boli odinštalované až pred začatím experimentu. Táto skutočnosť mohla ovplyvniť výsledky experimentu, v podobe registrov a súborov, ktoré dané programy zanechali po odinštalovaní. Podarilo sa nám úspešne vytvoriť inštalačné balíky pre všetky programy okrem Microsoft Office 2010. Pri tomto programe sa zapisovanie balíku zasekáva pri vytváraní rozdielu registrov, príčina chyby sa nám nepodarila odhaliť ani po niekoľkých pokusoch ladenia programu. Okrem tohto problému nám táto časť experimentu ukázala zopár chýb v

aplikácie, ktoré sme okamžite odstránili. Takisto sme zistili, že ani jeden z týchto programov nezapisuje žiadne dôležité súbory do zložky s používateľskými nastaveniami.

Vytvorené inštalácie sme preniesli na druhú stanicu, kde sme spustili lokálny server. Služba našej aplikácie korektne vytvorila všetky požadované odkazy na balíky. Pri inštalovaní sa opäť vyskytlo zopár chýb, ktoré sme okamžite ošetrili, ako spúšťanie programu regedit so zlým parametrom a zle zadaná cesta na rozbalenie archívu s balíčkom (o stupeň hlbšie). Na koniec sa nám podarilo úspešne nainštalovať všetky vyššie spomenuté programy okrem webového prehliadaču Google Chrome. Ten po inštalácií hlási pri spustení chybu s textom *The application has failed to start because its side-by-side configuration is incorrect*. Pokúšali sme sa túto chybu najskôr manuálne odstrániť, avšak ani po dôkladnejšom hľadaní na internete sme nenašli funkčné riešenie. Mnohé fóre doporučovali nainštalovanie balíkov **Microsoft Visual C++ 200X Redistributable**, avšak žiaden z nich náš problém nevyriešil. V zhrnutí nám experiment pomohlo nájsť mnoho malých chýb, ktoré mohli spôsobiť nefunkčnosť aplikácie v praxi. Zároveň nás tento experiment priviedol na mnoho vylepšení ako je exportovanie registrov po jednotlivých kľúčoch aby sme znížili časovú náročnosť diferenčného algoritmu.

Záver

Úspešne sa nám podarilo implementovať aplikáciu, ktorá je zverejnená ako open-source na stránke https://github.com/Wryxo/bakalarka. Táto aplikácia funguje ako jednoduchý systém na distribuovanie programov na väčšie množstvo staníc s operačným systémom Windows. Otestovali sme ju na vybranej vzorke program, experiment sme popisali v sekcii 5.2

V budúcnosti by sme chceli do aplikácie implementovať zlepšenia, ktoré sme spomínali v sekcii 5.1 a zároveň vylepšovať implementáciu podľa skúseností (z umiestňovania aplikácie do praxe).

Literatúra