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

# Zoznam obrázkov

# Zoznam tabuliek

# 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 zamestnanca vo firme aby bol produktívny 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 presunúť 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. 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

## 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 neskôr 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.

### 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
- Zoznam spustiteľných súborov na ktoré sa ma vytvoriť odkaz
- Balíčky, ktoré treba mať nainštalované
- Inštalácia pri štarte systému alebo až na vyžiadanie užívateľa

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.

### 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. Úlohou aplikácie po vytvorení balíčku na vzorovom počítači bude poslať tento balíček na server a zapísať ho do zoznamu dostupných balíčkov. Prenos dát a komunikácia medzi serverom a ostatnými stanicami bude prebiehať pomocou SSL technológie.

## 1.3 Fungovanie 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

**Administrátor** Po spustení tejto časti aplikácie, bude administrátor prezentovaný aplikáciou obsahujúcou ovládanie na úpravu všetkých nastavení ako napríklad adresa serveru, adresa kam treba vyrábať odkazy na programy a podobne. Ďalej sa tu bude nachádzať ovládanie časti na vytváranie balíčkov, spustenie/vypnutie zachytávania zmien v súborovom systéme a chybový výpis. Zmena niektorých nastavení alebo odchytávanie procesu bude vyžadovať administrátorské heslo.

**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. Konfigurácia

Všetky nastavenia, ktoré aplikácia potrebuje budú uložené vo Windows registroch a bude sa k ním dať pristúpiť cez administračnú časť aplikácie. Tieto nastavenia zahŕňajú:

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

Takisto v týchto registroch budú uložené ďalšie informácie, ktoré sa nebudú dať zmeniť a budú generované automatický aplikáciou ako napríklad čas posledného obnovenia zoznamu balíčkov na serveri.

KAPITOLA 2

# Technológie

## 2.1 C#

Jazyk C# sme zvolili kvôli podpore .NET frameworku. Ten nám poskytuje mnohé funkcie na ovládanie Windowsového file systému a registrov. Monitorovanie inštalačného programu by sa dalo robiť viacerými spôsobmi. Prvý najpriamejší by bol, spraviť zoznam súborov pred inštaláciou, po inštalácii a vyrobiť si rozdiel týchto zoznamov. Tento prístup je avšak veľmi zdĺhavý. Ďalšie asi najidealnejšie riešenie čo sa týka efektivity by bolo obaliť inštaláciu našou aplikáciou a všetky príkazy, ktoré inštalácie posiela systému odchytiť a zapísať si. Pri tomto spôsobe by sa nám do výsledkov nedostali ani súbory zmenené operačným systémom, ktoré nesúvisia s danou inštaláciou. Podobné riešenie, relatívne ľahko dosiahnuteľné v .NET, odchytáva asynchronné události pri zmene v súborovom systéme alebo registrov, a vie na tieto události reagovať, toto riešenie avšak odchytáva aj súbory zmenené ostatnými processmi a to nie je vždy to čo chceme.

## 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 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. Trieda FileSystemWatcher spĺňa takmer všetky naše požiadávky, avšak nevie rozpoznať, ktorý proces stojí za vytvorením daného súboru, čo spôsobuje problém, že môžu byť odchytené udalosti zmien v súboroch od iných procesov ako je naša sledovaná inštalácia, bežne totiž v operačnom systéme beží mnoho programov na pozadí, ktoré si vytvárajú a upravujú súbory.

**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 udalosti 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.

```
DriveInfo[] allDrives = DriveInfo.GetDrives();
foreach (DriveInfo d in allDrives)
{
    if (d.DriveType == DriveType.Fixed)
    {
        FileSystemWatcher watcher = new FileSystemWatcher();
        watcher.Path = d.Name;
        watcher.IncludeSubdirectories = true;
        watcher.NotifyFilter = NotifyFilters.LastWrite
           | NotifyFilters.FileName | NotifyFilters.DirectoryName;

        watcher.Changed += new FileSystemEventHandler(OnChanged);
        watcher.Created += new FileSystemEventHandler(OnChanged);
        watcher.Deleted += new FileSystemEventHandler(OnChanged);
        watcher.Renamed += new RenamedEventHandler(OnRenamed);

        watcher.EnableRaisingEvents = true;
        watchers.Add(watcher);
    }
}
```

Zdrojový kód 3.1: Inicializácia

## 3.2 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 implementáciu **referencia**. Tento kus kódu dostane na vstup 2 textové súbory a vráti nám riadky v ktorých sa odlišujú. Ako vstup mu teda dodáme výpisy registrov v štandardnom formáte vytvorené pomocou program regedit. Ako výstup dostane výpis nových kľúčov a ich hodnôt vo formáte, ktorý program regedit dokáže prečítať a vytvoriť dané kľúče v registroch. Problém avšak vzniká ak inštalácia zmení hodnotu už vytvoreného kľúča, pretože algoritmus rozpozná zmenu len na riadku priamo s novom hodnotou a nie definíciou kľúča, ktorá bude vtedy identická s tou pred inštaláciou. Tento problém budeme musieť neskôr odstrániť, zatiaľ ho však odignorujeme, vzhľadom na to, že podľa prieskumu nie mnoho programov upravuje počas inštalácie predtým vytvorené registre.

KAPITOLA 4

# Dokumentácia

Aj keď to nevie veľa toho, to málo sa dá robiť takto.

## 4.1 Vytvorenie balíku

Po spustení aplikácie, napíšeme názov balíku (predvolené tam je Testing) vyberieme disk C a/alebo D (zatiaľ tam je podpora len pevne na tieto 2 disky) a stlačíme Start Listen. Nainštalujeme program (rozbalíme .zip a pripíšeme niečo do registrov), na obrazovke sa nám bude zobrazovať zoznam súborov, ktoré aplikácia detekovala. Po skončení inštalácie, stlačíme Stop Listen, program si prekopíruje všetky súbory do zložky Packages pri aplikácii.

## 4.2 Nainštalovanie balíku

Po spustení aplikácie v spodnej časti si vybereme názov balíku, ktorý chceme nainštalovať a klikneme Install. Aplikácia následne nakopíruje všetky odložené súbory na miesto, odkiaľ boli zaznamenané a do registrov pridá zmeny, ktoré nastali počas inštalácie.

## 4.3 Dočasné nedostatky

Keďže aplikácie je stále vo vývoji, treba si dať pozor na následovné veci:

- Na disku C, aplikácia zachytáva aj systemové a dočasné súbory, ktoré neskôr možu spôsobiť výnimku.
- Pri inštalácii prepisuje súbory
- Je tam ďalšie obrovské množstvo vecí, ktoré môžu vyvolať výnimky

# Literatúra