Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Administrace pracovních stanic pomocí Powershell

Jakub Bureš

! ! !

Místo této strany vložíte zadání bakalářské práce.

(Do jedné vazby originál a do druhé kopii)

! !

Název práce:	**Administrace pracovních stanic pomocí Powershell**
Název anglicky:	**Workstation administration using Powershell**
Cíle práce:	Cílem diplomové práce je analýza a vytvoření nástroje pro administraci pracovních stanic za pomocí skriptovacícho jazyka Powershell. Výstupem práce bude nástroj využivající skriptovací jazyk Powershell za pomocí GUI. Nástroj bude sloužit jak pro uživatele, tak pro administrátora. V případě uživatele, bude nástroj zpřístupňovat veškeré potřebné informace o počítači, které by byly vyžadované v případě řešení problému s PC IT Helpdeskem. Nástroj pro administrátora bude zpřístupňovat stejné informace o PC jako v případě uživatele s tím rozdílem, že bude spouštěn dálkově a obsahovat podrobnější informace o počítači.
Metodika:	V první polovině práce se autor bude zabývat teoretickými východisky a seznámením s funkcemi skriptovacího jazyka Powershell. Získání teoretických východisek je založeno na studiu a analýze odborných informačních zdrojů a jejich vzájemnou komparací. V další části teoretických východisek autor analyzuje různé možnosti vzdálené administrace. Druhá polovina práce bude zaměřena na praktické využití znalostí. V prvním kroku této části budou analyzovány potřeby na informace pro řešení IT problému. Na základě získaných poznatků budou vybrány ty nejčastěji potřebné a bude realizován návrh na funkcionalitu nástroje. Dle návrhu bude vyvinut prvotní nástroj, který bude implementován do firemního prostředí, kde bude dále testován.

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci " *Administrace pracovních stanic pomocí Powershell* " jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne: _____

Poděkování

Rád bych touto cestou poděkoval Ing. Marek Pícka, Ph.D. , za jeho cenné a věcné připomínky k této práci

Administrace pracovních stanic pomocí Powershell

Workstation administration using Powershell

Souhrn

Diplomová práce se zabývá skriptovacím jazykem Powershell a jeho implementováním do firemního prostředí ve formě nástroje pro administraci. V teoretické části se autor zabývá popisem skriptovacího jazyka Powershell a jeho vývojového prostředí. Dále pak autor uvádí funkcionalitu Windows Presentation Framework ve spojení s XAML a jeho využitelnosti v kombinaci s Powershellem. V praktické části se seznamujeme s firemním prostředím a vstupujícími aktory. Dále pak autor zpracovává jednotlivé požadavky od oddělení IT helpdesk a IT hardware a společně zadané požadavky porovnává, aby mohl stanovit podmínky na vývoj nástroje. Poté autor volí správnou metodiku vývoje softwaru s ohledem na to, že se jedná o vývoj za pomocí skriptovacího jazyka. Metodu těmto požadavkům upravuje. Následuje autorův vlastní vývoj za pomocí zvolené metodiky a výstupem práce se stávají dva moduly zpracované za pomocí skriptovacího jazyka Powershell a WPF. První z modulů slouží pro méně technicky zdatné uživatele počítače a zpřístupňuje jim veškeré potřebné informace o počítači, které slouží jako základ pro správnou identifikaci při potřebě servisního zásahu. Dalším modulem je administrační nástroj, který zpřístupňuje vzdáleně detailnější informace o počítači. Tyto informace mají sloužit pro usnadnění identifikace zdroje problému s počítačem, když je potřeba servisního zásahu ze strany administrátora.

Summary

The Master thesis deals with the Powershell scripting language and its implementation into the corporate environment in the form of an administration tool. In the first part, the author deals with the description of the Powershell scripting language and his development environment. Further, the author introduces the functionality of the Windows Presentation Framework in connection with XAML and their usability in combination with Powershell. In the second part, we familiarize with the corporate environment and the agents. Further, the author processes individual

requests from the IT helpdesk and IT hardware departments and compares the assigned requirements in order to determine the conditions for the development of the tool. Then, the author chooses the right software development method, considering that it is development using the scripting language. The method is modified for these requirements. Following is the author's own development using the chosen method and the output of the work become two modules processed using the scripting language Powershell and WPF. The first module is for users without technical knowleadge and provides them with all the necessary information about the computer, Information serves as the basis for proper identification when the computer was broke down. Second module is an administration tool that provides remote access to more detailed computer information. This information is intended to make it easier to identify the source of the problem with the computer when the administrator needed

Klíčová slova: Powershell, Scriptovací jazyk, WMI, WPF, XAML

Keywords: Powershell, Scripting language, WMI, WPF, XAML

Obsah

1 Úvod ... 1

2 Cíl a metodika práce ... 3

3 Skriptovací jazyk ... 4

4 .NET Framework ... 5

4.1 Common Language Runtime ... 5

4.2 .NET Framework Class Library ... 5

5 Powershell ... 6

5.1 Příkazy CmdLets ... 7

5.2 Funkce ... 8

5.3 Podmínky ... 8

5.3.1 IF ... 8

5.3.2 Switch ... 9

5.4 Cykly ... 9

5.4.1 For ... 9

5.4.2 Foreach ... 9

5.4.3 While ... 10

5.4.4 Do ... 10

5.5 Chyby a vyjímky ... 10

5.6 Roura (Pipeline) ... 11

5.6.1 Textová roura ... 11

5.6.2 Objektová roura ... 11

5.7 Proměnné ... 12

5.8 Datové typy ... 12

5.9 Pole ... 13

5.10 Bezpečnost ... 13

5.11 Nápověda ... 14

5.12 Formátování výstupů v příkazovém řádku. ... 14

5.13 Formátování výstupů do souboru ... 15

5.14 Vstup dat ze souboru ... 15

5.15 Vývojové prostředí a doplňky ... 15

5.15.1 Powershell ISE ... 15

5.16 Rozhraní WMI ... 16

5.16.1 Využití WMI prostřednictvím PowerShell ... 17

5.16.2 Využití WMI v ActiveDirectory ... 17
5.16.3 CIM ... 17

6 Microsoft Visual studio ... 18

6.1 Windows Presentation Foundation (WPF) ... 19

6.1.1 WPF ... 19
6.1.2 Windows Forms ... 19
6.1.3 Extansible Application Markup Language (XAML) ... 19

7 Nástroje pro vzdálenou správu ... 20

7.1 Prostředí Cmd.exe a PStools ... 20

7.2 VBScript ... 21

7.3 Porovnání nástrojů pro vzdálenou správu ... 22

7.3.1 Instalace ... 22
7.3.2 Funkcionalita ... 22
7.3.3 Přenositelnost ... 23
7.3.4 Bezpečnost ... 23
7.3.5 Podpora a údržba ... 23
7.3.6 Vzdálená správa ... 24
7.3.7 Práce s Active Directory ... 25
7.3.8 Práce s WMI ... 25

8 Analýza ... 26

8.1 Popis prostředí společnosti ... 26

8.1.1 IT Helpdesk oddělení ... 27
8.1.2 Aktoři ... 28

8.2 Analýza požadavků na software ... 30

8.3 Analýza krabicového softwaru ... 34

8.3.1 Správce IT ... 34
8.3.2 Hp OpenView ... 34
8.3.3 Nagios ... 35
8.3.4 Vyhodnocení externích programů. ... 36

9 Druhy metodik ... 37

9.1 Tradiční metodiky ... 37

9.1.1 Vodopádový model ... 37
9.1.2 Spirálový model ... 38
9.1.3 Rational Unified Process ... 39

9.2 Agilní metodiky 39
9.2.1 Extrémní programování (XP) 40
9.2.2 Scrum 41
9.2.3 Lean development 41
9.3 Volba metodiky 42
10 Návrh řešení 44
10.1 Fáze zahájení 44
10.1.1 Požadavky na základě analýz 44
10.1.2 Výsledek analýzy nejčastějších činností 46
10.1.3 Vyhodnocení požadavků 47
10.1.4 Požadavky na HW a SW 49
10.1.5 Návrh architektury sytému 49
10.1.6 Potvrzení proveditelnosti 50
10.1.7 Uživatelské rozhraní 50
10.1.8 Vyhodnocení fáze „Zahájení“ 51
10.2 Fáze rozpracování 52
10.2.1 Uživatelský nástroj 52
10.2.2 Administrátorský nástroj 54
10.2.1 Vyhodnocení fáze „Rozpracování“ 56
10.3 Fáze nasazení 57
10.3.1 Budoucí rozšíření 57
11 Zhodnocení výsledků 58
12 Závěr 59
Seznam použitých zdrojů 62
Seznam obrázků a tabulek 63
Seznam zkratek: 65
Přílohy 66

1 Úvod

Skriptovací jazyky jsou interpretované programovací jazyky vyvíjené s ohledem na uživatelskou přívětivost. Pro rychlejší a intuitivnější práci na skriptu by měl jazyk používat srozumitelnou syntaxi zápisu kódu. Skriptovací jazyky jsou v dnešní době využívány systémovými administrátory a slouží k automatizaci práce. Vytvořené automatizované skripty přináší výhody ve formě ušetřeného času, který administrátoři mohou věnovat jiným činnostem. Mezi další výhody automatizovaného skriptu do jisté míry patří zkvalitnění procesu práce. V případě skriptu, který má předem jasně definované vstupní podmínky je možné očekávat stále stejné výsledné zpracování a normalizované přehledy. V případě automatizace je potlačeno možné lidské selhání, a tudíž nedochází k chybovosti způsobené lidským faktorem. Mezi další výhody patří, že skriptovací jazyky ve většině případů nepotřebují kompilátor. Z toho vyplívá, že automatizované skripty jsou snadné na správu a lehce udržovatelné. Využití automatizovaného skriptu je velmi využívané v oblasti informačních technologií, kdy spuštění skriptu je vyvoláno ručně nebo za pomocí plánovaných událostí. Z praxe se může jednat o skripty velmi jednoduché až základní nebo o komplexní skripty. Převážně u komplexně řešených skriptů není dobré plně spoléhat na danou automatizaci a doporučuje se výsledky zpracování kontrolovat. Nejlepší volbou k využití skriptovacích jazyků se jeví oblast rutinních prací. Při vykonávání rutinních prací jsou pravidelně prováděny stejné operace. U veškeré automatizace musíme naleznout správný poměr mezi lidskou a automatizovanou prací.

Autor si pro svou diplomovou práci zvolil téma *„Administrace pracovních stanic pomocí Powershell“* Mezi hlavní autorův důvod k volbě tohoto tématu je snaha pochopení a následné naučení se skriptovací jazyk. Autor si vybral jako zástupce ze skriptovacích jazyků Powershell jelikož se jedná o jeden z nejrozšířenějších skriptovacích jazyků. Mezi další důvody autorovi volby přispěl fakt, že v současné době je zaměstnán jako IT hardware suport pracovník, kde má na starost udržování pracovních stanic po hardwarové a softwarové stránce.

Diplomová práce je zaměřena na seznámení čtenářů s vývojem nástroje vyvíjeného za pomocí skriptovacího jazyka Powershell. Práce blíže seznamuje čtenáře o vlastnostech a syntaxích daného skriptovacího jazyka. Dále se soustřeďuje na analýzu dalších nástrojů, které jsou použitelné pro vzdálenou správu a jejich vzájemným porovnáním. V neposlední řadě se autor snaží analyzovat vnější i vnitřní prostředí společnosti, kde by byly nástroje využívány. Na

základě analýzy externího softwaru, historických požadavků a lokálního šetření na oddělení se autor snaží o vyhodnocení podmínek a návrh řešení za pomocí Powershell. Snaha autora o automatizaci požadavků je z důvodu usnadnění práce dalším administrátorům a časové úspory. Na závěr práce autor rozepisuje jednotlivé kusy kódu a tím dává čtenáři jasná vodítka k jeho pochopení.

2 Cíl a metodika práce

Cílem diplomové práce je vytvoření nástroje pro administraci pracovních stanic za pomocí skriptovacího jazyka Powershell. Nástroj bude výstupy ze zpracovávaných skriptů zobrazovat za pomocí Windows Presentation Framework.

Autor se výběrem tohoto tématu chtěl blíže seznámit se skriptovacím jazykem Powershell a vyzkoušet zdali i se základními znalostmi skriptovacího jazyka dokáže vyvinout administrační nástroj, za pomocí kterého se pokusí vylepšit neuspokojivý stav identifikace a správy koncových stanic. Mezi další důvody autorova výběru je také fakt, že se v dnešní době ovládání takovéhoto skriptovacího jazyka těší velké oblibě a využitelnosti ve firemním prostředí. Vývoj současného trendu u Microsoftu je takový, že Powershell je hlavním skriptovacím jazykem na platformě Windows.

V teoretické části práce autor seznámí čtenáře se základními konstrukčními vlastnostmi a vývojovým prostředím skriptovacího jazyka Powershell. V další část autor přiblíží možnosti využití Windows Presentation Framework ve spojení s XAML. Dále autor bude porovnávat další přípustné nástroje pro vzdálenou správu a porovná jejich funkcionalitu vůči Powershell. Autor při zpracování teoretické části práce bude vycházet z odborných literárních pramenů a bude je prezentovat formou rešerše.

V praktické části diplomové práce autor představí firemní prostředí společnosti a uvede vzorek vstupujících aktorů. V další části práce bude autor analyzovat kladené požadavky jednotlivými odděleními na nástroj. Z těchto požadavků stanoví podmínky a druhy dotazů, které bude nástroj zpracovávat. Poté se autor zamyslí nad zvolením správné metodiky vývoje softwaru a její aplikování v případě vývoje za pomocí skriptovacího jazyka. Po výběru správné metodiky a případné její úpravě autor bude postupovat na základě doporučení metodiky s vývojem nástroje.

V závěru práce autor uvede nástroj do testovacího provozu a zhodnotí celkový přínos zpracování daného nástroje.

Při tvorbě této diplomové práce byla uplatněna literární rešerše, analytická metoda, metoda pozorování, analýza a její následná syntéza a v neposlední řadě metoda brainstorming.

3 Skriptovací jazyk

U skriptovacích jazyků se ve velké míře jedná o interpretované programovací jazyky. Interpretované skriptovací jazyky jsou navrženy s ohledem na jejich snadné zvládnutí, rychlý a pohodlný vývoj. U těchto jazyků není potřeba mít kompilátor, tudíž se zdrojový kod nemusí složitě překládat a je umožněna rychlá změna částí programů. Program napsaný ve skriptovacím jazyce, dovoluje práci se složitějšími datovými typy, jakými jsou například asociativní tabulka. K vytvoření skriptů není potřeba složité instalace a ve velké míře postačuje přenést zdrojový kód.

Obecné definice jakých vlastností by měl skriptovací jazyk nabývat nejsou, ale obvykle se jazyky považují skriptovacími s těmito vlastnostmi:

- Musí se jednat o interpretovaný jazyk
- Deklarování proměnných není potřeba
- Využívá dynamickou typovou kontrolu
- Práce se složitými datovými typy
- Zpracování regulárních výrazů
- Chybová hlášení

Mezi hlavní výhody skriptovacího jazyka se řadí:

- Rychlý vývoj
- Široké využití
- Kód není potřeba překládat
- Údržba

Mezi nevýhody se řadí:

- Vyšší nároky na paměť
- Potřeba spolupráce s vyššími jazyky
- Doba zpracování je delší než u kompilovaných jazyků
- Chybovost při automatické zahajování nebo dynamickém typování.

4 .NET Framework

.NET Framework je technologie, která podporuje budování a provoz nové generace aplikací a webových služeb XML. Framework se skládá z „common language runtime“ a knihovnou „.NET Framework class“.(14)

4.1 Common Language Runtime

Common language runtime řídí paměť, spouštění vláken, spuštění kódu, ověření bezpečnostního kódu, přeložení, a další systémové služby. Tyto vlastnosti jsou charakteristické pro spravovaný kód, běžícím pod Common Language Runtime.

4.2 .NET Framework Class Library

Knihovna tříd „.NET Framework“ je sbírka opakovaně použitelných typů, které jsou integrovány v „Common Language Runtime“. Knihovna tříd je objektově orientovaná a poskytuje typy kódu, ke kterým je možné doplnit funkcionalitu. Tímto činí .NET Framework snadno ovladatelným a snižuje čas potřebný k učení nových funkcí „.NET Framework.“ Další součásti třetích stran mohou být lehce integrovány s třídami v „.NET Framework“.

Služby a aplikace „.Net Framework“

- Konzolové aplikace
- Windows Form
- Winows Presentation Foundation
- ASP.NET
- Windows Service Application
- Windows Communication Foundation – servisově orientovaná aplikace.
- WorkFlow-enabled aplikace(15)

5 Powershell

Powershell je objektově orientovaným skriptovacím jazykem vyvinutým společností Microsoft. Skriptovací jazyk je postaven na platformě „.NET Frameworku.“ S použitím tohoto Frameworku vyplívají jisté odlišnosti od ostatních skriptovacích jazyků. Jeden z největších rozdílů od ostatních shellu můžeme naleznout v tom, že Powershell používá objektovou rouru (pipeline) místo textové. Díky integraci do „.NET Frameworku“ poskytuje Powershell množství funkcí pro administraci tříd s pomocí tzv. „Cmdlets“.

V nejnovějším operačním systému společnosti Microsoft se již instaluje ve verzi 5.0. Spustitelné skripty jsou označené příponou „.PS1“.

Powershell byl vyvinut společností Microsoft z důvodu mnohaleté absence jakéhokoliv skriptovacího jazyka. Poprvé byl představen v roce 2006, ale jeho vývoj se datuje od roku 2002 pod označením Monad. Microsoft začal vyvíjet toto scriptovací prostředí, aby zlepšil administraci a konfiguraci operačních systémů Windows. Před příchodem tohoto jazyka byly pouhé dvě varianty příkazového rozhraní tzv. „Shellu“.

Příkazové prostředí MS-DOS – tento shell byl zpočátku vlastně jediným rozhraním pro ovládání poměrně jednoduchého operačního systému z dílny Microsoftu. Vyznačoval se velmi omezenými možnostmi a téměř neexistujícím skriptovacím jazykem.

Příkazové prostředí Cmd.exe – tato modernější a dodnes používaná varianta vychází koncepčně z předchozí varianty. Nastoupila v době, kdy trh opanovaly plně 32bitové operační systémy. V dnešní době jej můžeme naleznout prakticky ve všech systémech Windows. Možnosti jsou ve srovnání s MS DOSem poněkud vylepšeny, avšak ani tak není výbava skriptovacího jazyka jako takového nijak ohromující a potenciál je dosti omezený. Schopnost správy byla výrazně posílena díky uvedení rozsáhlé sady dodatečných konzolových aplikací, jež jsou volány z příkazového řádku.(9)

Jazyk je postaven na „.NET platformě“. Díky provázanosti s „.NET“ poskytuje všechny možnosti této platformy a také specializované třídy zvané „Cmdlets“

Obrázek 1- logo Powershell

zdroj: httpwww.techrepublic.comarticlepowershell-the-smart-persons-guide

5.1 Příkazy CmdLets

Cmdlety jsou specializované interní příkazy pro prostředí Powershell, které realizují specifické funkce. Jedná se o specializované „.NET“ třídy, která Powershell vyvolává a konkretizuje za běhu. Výstupní hodnoty CMDletů nejsou pouze řetězce, ale mohou jím být i objekty, které je možné dále zpracovávat a mohou sloužit jako vstupní hodnotou pro další příkazy v nezměněném stavu.

Syntaxe:

Příkaz se skládá z několika částí. První část je vlastní jméno příkazu, stojí vždy na počátku řádku a nemusí být blíže specifikován. Druhou částí je přepínač daného příkazu (switch) vyjadřuje se znakem (-) následován pojmenováním, nejčastěji anglickým a slouží ke zpřesnění funkcionality. Poslední částí je parametr (argument) není nikterak označován a PowerShell jej rozeznává automaticky. Parametru můžeme mít v příkazu mnoho a vždy rozhoduje jeho umístění v příkazu.

Ukázka syntaxe:

*Get-Process –Name svc**

Obrázek 2 - Get-Process výstup

```
PS C:\> Get-Process -Name svc*

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
   6406     129    56864      81200                316   0 svchost
    941      51    14508      26392                392   0 svchost
   1341      53    13960      37628                472   0 svchost
    785      32    15884      26704                796   0 svchost
    810      30     9916      25060                836   0 svchost
    871      21     8084      15304                904   0 svchost
    863      39    15036      28896               1056   0 svchost
    631      46    26368      36068               1272   0 svchost
    341      12     3368      11228               1836   0 svchost
    174      12     1920       6564               1868   0 svchost
    471      25     9432      25832               2216   0 svchost
    246      18     7684      21184               2280   0 svchost
    382      22     4028      10572               3532   0 svchost
    129      10     1632       6296               3560   0 svchost
    439      28     6488      26384       0,17    7112   9 svchost
    148      10     1828       5908              11076   0 svchost
```

zdroj: vlastní

5.2 Funkce

Funkce je pojmenovaný blok příkazů, jejž můžeme opakovaně spouštět právě oním jménem, jež jsme tomuto bloku příkazů přiřadili. Hlavní výhodou funkce je předávání hodnot, které budou zpracovávány pomocí příkazu v bloku uvnitř funkce. (9)

5.3 Podmínky

Podmíněné příkazy v prostředí PowerShell umožňují změnit tok vykonávání skriptu. (6)

5.3.1 IF

Pokud je podmínka vyhodnocena jako $true, a pokud má hodnotu „False“ zpracuje se kód, který je zapsán pro blok ELSEIF, nebo ELSE. ELSEIF obsahuje vlastní tvrzení, které je nutné splnit. Pokud toto tvrzení není splněno, bude vykonán blok příkazů pro ELSE. Bloky ELSEIF a ELSE jsou volitelné. PowerShell vyžaduje závorky kolem bloku příkazů, a to i v případě, že blok obsahuje pouze jeden příkaz. Blok kodu se vkládá mezi složené závorky (6)

Operátory porovnání(16)

- -eq : rovná se
- -ne : nerovná se
- -gt : větší než
- -ge : větší než nebo rovno
- -lt : menší než
- -le : menší než nebo rovno
- -like : operátor zástupného znaku
- -notlike : operátor zástupného znaku
- -match : porovnání regulárního výrazu
- -notmatch : porovnání regulárního výrazu

Obrázek 3 - Příklad If

Zdroj: vlastní

5.3.2 Switch

Příkaz začíná klíčovým slovem switch a vyhodnocovaná podmínka je uvedena v kulatých závorkách. Následuje dvojice složených závorek, které vyznačují blok skriptu příkazu switch. V rámci bloku skriptu začíná každá vyhodnocovaná podmínka hodnotou, po které následuje blok skriptu. Tento blok je proveden v případě, že hodnota odpovídá podmínce. (16)

Pokud je jako vstup zvolena proměnná typu pole, vyhodnocuje se switch podmínka pro každý prvek. Stejně tak při využití parametru „–file“ je specifikovaný soubor rozdělen na řádky a jednotlivě porovnán. (6)

5.4 Cykly

Cyklus obecně spustí blok příkazu několikrát dle předem udané definice cyklu. Syntaxe cyklu v Powershellu je podobná ostatním programovacím jazykům.

5.4.1 For

PowerShell jako první spustí inicializaci FOR cyklu, která definuje vstupní proměnou a posléze vyhodnocuje podmínky. Pokud je podmínka vyhodnocena jako $true, proběhne daný příkaz bloku. Ten pak provede vyjádření daného přírůstku. PowerShell bude nadále vykonávat daný blok přírůstku tak dlouho, dokud podmínka nebude vyhodnocena jako $true(6)

5.4.2 Foreach

Cyklus FOREACH prochází pole nebo kolekci objektů. Počet cyklů závisí na počtu objektů v procházené proměnné. Aktuálně procházený objekt je uložen v proměnné, které se definuje při zápisu cyklu. (6)

Obrázek 4- Příklad ForEach

Zdroj: vlastní

5.4.3 While

WHILE cyklus zpracovává nejprve definovanou podmínku. Pokud je podmínka vyhodnocena jako „True“, provede se blok příkazů. Podmínka je následně znovu vyhodnocena a cyklus se opakuje, dokud není podmínka rovna hodnotě „False“. V tomto případě již není blok příkazů proveden a cyklus končí. (6)

5.4.4 Do

Cyklus „DO“ je velmi podobný cyklu „WHILE“ s tím rozdílem, že při zpracování je jako první zpracován blok příkazů. Po dokončení dle výsledku určuje podmínka, zda se cyklus bude opakovat. (6)

Dvojí způsob využití DO cyklu

- DO … WHILE: Cyklus je opakován do té doby, dokud podmínka vložená za sekvenci WHILE je vyhodnocena jako „True“.
- DO … UNTIL: Cyklus je opakován do té doby, dokud podmínka vložená za sekvenci UNTIL je vyhodnocena jako „False“ (6)

5.5 Chyby a vyjímky

Příkazové či skriptovací rozhraní dovoluje vytvořit celou škálu chyb, jež se od sebe liší svou závažností, obtížností odhalení či původem.

Dělení chyb:

- Syntaktické - jež vznikají chybným použitím předepsaných příkazů a jejich variant, případně složitějších konstrukcí jazyka PowerShell samotného.
- Běhové - (runtime errors), které mohou a nemusí nastat
- Logické - ty vznikají hnutím mysli administrátora či tvůrce skriptu(9)

Možnosti přístupu ke zpracování chyb

- Přesměrování chybových hlášení do externího souboru.
- Proměnná „$error" – obsahuje veškeré chybové stavy
- Parametr „-ErrorAction" – nastavení chování všech chyb
- Konstrukce „Try-Catch" – odchycená a reagující na vzniklou chybu

5.6 Roura (Pipeline)

Princip roury spočívá v tom, že každý spuštěný příkaz (program) dokáže při své činnosti odesílat výstupní hodnoty na určité místo, kde je možné data dále zpracovat

za použití jiného příkazu (programu). V podstatě pipeline je vložení výstupu z jednoho příkazu do inicializace druhého.

Pipeline v Powershellu se na rozdíl od Shellu v operačních systémech Linux liší v použití objektové roury na rozdíl od té textové, která je naprosto běžná v linuxových systémech.

Znakem pro označení roury je: |

5.6.1 Textová roura

Vycházíme z předpokladu, že data jsou z ní a do ní zasílána ve formě textových řetězců. Jedná se o sekvenci dat, která je interpretována jako zakódované znaky (písmena, číslice, jiné znaky) a takto je chápána na obou stranách roury – příkazy (programy) s tím počítají a respektují to, aby si navzájem rozuměly. (9)

5.6.2 Objektová roura

Data jsou v Powershellu posílána do roury v podobě objektů. Skutečnost, že s nimi můžeme pracovat jako s textovými řetězci, je dána schopností Powershellu provést konverzi „na požádání", ve skutečnosti za běhu. Pokud náš program (externí příkaz) zasílá či přijímá data do/z roury jako proud znaků, Powershell to respektuje a přizpůsobí se. Pokud je však zdrojem dat v rouře třeba „pravý" příkaz Powershellu (jako kupříkladu Get-Process), data vstupují do roury jako objekty a záleží jen a jen na dalších aplikacích, jak si je z roury vyzvednou. Je-li dále v řetězci zařazena aplikace, jež načítá textové řetězce, jsou objekty „zredukovány" do této podoby a my pracujeme jakoby s textovou rourou. (9)

Obrázek 5 - Příklad objektové roury

```
Get-ChildItem C:\Scripts | Where-Object {$_.Length -gt 100KB} | Sort-
Object Length
#příkaz Get-ChilItem vrátí seznam souboru v C:\Scripts předá je Where-
Object který vybere pouze ty co jsou větší než 100Kb a předá je Sort-
Object který je seřadí podle velikosti
```

zdroj 1: vlastní

5.7 Proměnné

Powershell využívá jako základní uložiště pro data proměnnou, pojmenované paměťové místo. Jako označení proměnné slouží znak „$“ následovaný řetězcem znaků. V Powershellu můžeme typ proměnné přiřadit samy nebo deklaraci plně ponechat na skriptovacím jazyce, který typ proměnné dokáže rozeznat automaticky.

5.8 Datové typy

Powershell přebírá datové typy od svého mateřského prostředí, běhové platformy .NET Gramework, a proto jeho datové typy odpovídají této platformě.(9) V tabulce máme uvedené základní druhy datových typů a jejich příslušné zkratky, které využijeme při deklaraci.

Tabulka 1- Datové typy

Označení	Datový typ
[datetime]	Datum nebo čas
[string]	Řetězec znaků
[char]	Jednotlivý znak
[double]	Desetinné číslo s dvojnásobnou přesností
[single]	Desetinné číslo se základní přesností
[int]	Celé číslo reprezentováno 32 bity
[array]	Pole hodnot
[xml]	Xml objekt
[hashtable]	Hashtable objekt
[wmi]	Instance nebo kolekce objektů WMI
[adsi]	Objekt získávaný rozhraním Active Directory Services (ADSI)
[wmiclass]	Třída objektového modelu WMI
[boolean]	Logická hodnota dle Booleovy algebry; jeden bit: ano/ne;

5.9 Pole

V případě potřeby zpracování většího objemu dat má v sobě Powershell již uskutečněnou funkci pro práci s poli. Deklarace pole se neprovádí žádným operátorem. Při přiřazení více hodnot jedné proměnné Powershell automaticky usoudí, že se žádá o vytvoření struktury typu pole.

Asociativní pole

Asociativní pole umožnuje přiřadit daným klíčovým slovům hodnoty. K zápisu se využívá syntaxe „@{}“. Powershell předpokládá, že klíčová slova jsou typu string, ale hodnoty mohou obsahovat různé datové typy. S asociativním polem lze dále pracovat a to přidáváním dalších prvků, vyhledáváním prvků podle klíčových slov, nebo řazení tohoto pole. (6)

5.10 Bezpečnost

Ve skriptovacím jazyce Powershell je možné spouštět jak konzolové aplikace, tak i jiné externí programy nebo vlastní skripty. Scripty napsané v Powershell jsou označeny příponou *.ps1 a jsou po nainstalování asociovány s poznámkovým blokem. Defaultní asociace byla zavedena, aby nedocházelo k náhodnému spuštění scriptu, který by mohl mít i vážné následky. Microsoft zavedl několik dalších bezpečnostních prvků proti nechtěnému nebo neoprávněnému spuštění scriptu. Jedním z prvků je, že vám Powershell neumožní spustit skripty z aktuálního adresáře, pokud napíšete pouze jeho jméno. Příkaz musí být ve tvaru:

```
PS C:\skript\PS> ./skript.ps1
```

Při tomto zápisu narazíme na další bezpečnostní prvek a to bezpečnostní politiky. Politiky se starají, jestli jsou spouštěné skripty digitálně podepsány. Po nainstalování je bezpečnostní politiky nastavena na „Restricted“. V tomto módu je umožňováno zapisování příkazů přímo v okně Powershellu nikoliv spouštění celých scriptů. Máme čtyři bezpečností politiky:

Tabulka 2 - Bezpečnostní politiky

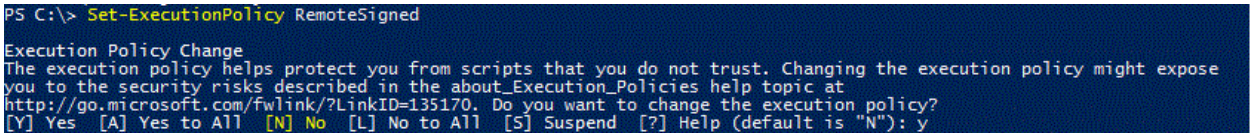
Restricted	*Prostředí nenačítá konfigurační soubory ani nespouští skripty – úroveň restricted je výchozí*
AllSigned	*Vyžaduje, aby všechny skripty a konfigurační soubory podepsal důvěryhodný vydavatel. Vztahuje se to i na skripty vytvořené v místním počítači.*
RemoteSigned	*Vyžaduje, aby důvěryhodný vydavatel podepsal všechny skripty a konfigurační soubory stažené z internetu.*

Unrestricted	*Načítá všechny konfigurační soubory a spouští všechny skripty.*
Bypass	*Nic není blokováno a nezobrazují se žádná upozornění ani výzvy.*
Undefined	*Odebere aktuálně přiřazené pravidlo spouštění z aktuálního oboru.*

Pro změnu bezpečnostní politiky pro náš script je nutné použít příkaz „Set-ExecutionPolicy“ a jako parametr použít jednu ze jmenovaných možností.

Příklad změny bezpečnostní politiky:

Obrázek 6- Příklad bezpečnostní politiky



zdroj: vlastní

5.11 Nápověda

Příkaz Powershellu poskytující podporu a užitečné informace o příkazech se nazývá „Get-Help“. Nápověda podporuje několik různých informativních pohledů v závislosti na vašich potřebách

Chcete-li získat přehlednou nápovědu pro konkrétní příkaz, musíte uvést jméno příkazu jako argument do rutiny „Get-Help“ Zobrazí se vám přehled, syntaxe daného příkazu a jeho podrobný popis. Stejnou funkcionalitu můžeme, naleznouz u Unixových systémů.

Pro výpis nápovědy existují tři parametry

- „-Detailed“ – Zobrazení detailního výpisu daného příkazu i s příklady využití
- „-Full“ – Zobrazení veškerých dostupných informací
- „-Examples“ – Výpis příkladu použití daného příkazu

5.12 Formátování výstupů v příkazovém řádku.

Jako výstupní cestu pro zobrazování dat používá Powershell výpis do okna programu. Data obsahují vlastní defaultní formát, ve kterém jsou zobrazena. Pro vlastní přizpůsobení

formátování těchto dat slouží skupina Cmdletu –FORMAT. Pomocí těchto Cmdletů můžeme editovat vlastnosti, které chceme zobrazit na výstupu.

5.13 Formátování výstupů do souboru

Powershell jako takový nabízí své vlastní možnosti jak data nasměrovat do souboru. Jednou z nich je CMDlet „Set-Content“. Důležitým rysem jeho práce je absence formátování – data jež budou zaslána do výstupního souboru jsou prostě převedena na formát řetězce a poté nasměrována do souboru bez jakýchkoliv formátovacích zásahů.(9)

5.14 Vstup dat ze souboru

Načítání dat ze souboru je pro scriptovací jazyk důležitou funkcí. Powershell pro tuto funkci využívá Cmdletu Get-Content, který vrací obsah zvoleného souboru. Díku tomuto Cmdletu je Powershell schopen načíst i více souborů dohromady. Takto načtená data ze souboru jsou následně převedena do objektové proměnné. Pro import dat existuje celá řada Cmdletů např. Cmdlet Import-Csv, který dokáže naimportovat .csv soubor.

5.15 Vývojové prostředí a doplňky

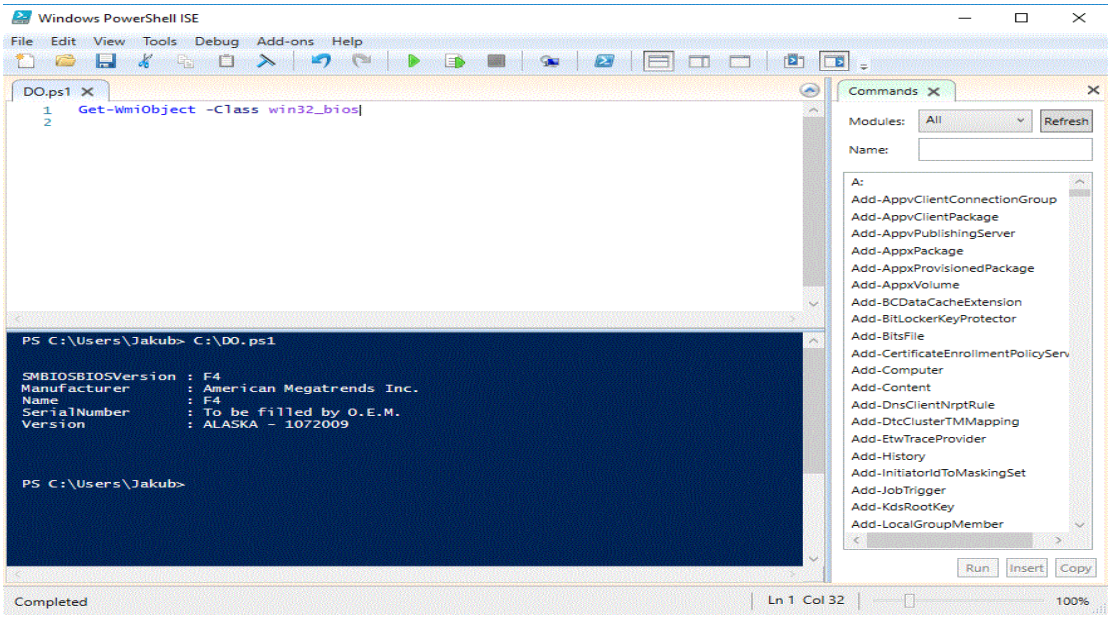
Základním rysem pro Powershell je jeho modularita, která zajišťuje, že užívání i vývoj scriptů je možný v různých vývojových prostředí za použití rozšiřujících přídavných modulů třetích stran.

5.15.1 Powershell ISE

Windows Powershell Integrated Scripting Environment slouží pro spouštění, psaní, testování a ladění scriptů. Grafické prostředí na bázi Windows umožňuje podobně jako jiná pokročilá vývojová prostředí mnoho užitečných funkcí. Mezi hlavní funkce můžeme uvést víceřádkové úpravy, našeptávání, syntaxe zbarvení, selektivní provedení, které slouží pro spuštění jen označené části. Pro potřeby bližších informací je možné použít Kontextovou nápovědu, která je nedílnou součástí vývojového prostředí. Použitím položky menu a klávesových zkratek můžeme provádět mnoho úloh, jako bychom prováděli v konzoli Windows Powershell.

Windows Powershell umožňuje přizpůsobení některých aspektů jeho vzhledu a rozšíření.

Obrázek 7 - Powershell ISE



zdroj: vlastní

5.16 Rozhraní WMI

Samotné WMI je jakýmsi univerzálním prostředkem pro pokročilou administraci Windows a jako takové je jeho využití součástí řady „velkých“ nástrojů, jako jsou Microsoft Operations Manager či System Manager Server. Jeho zpřístupnění ve skriptovacím rozhraní pak znamenalo velký krok vpřed díky otevření nových monitorovacích a ovládacích rozhraní bez nutnosti psát složité aplikace či zakupovat náročný a drahý software.(9) Mezi hlavní přednosti rozhraní WMI je vynikající zpřístupnění eventlogů, které dříve bylo dostupné pouze za použití kompilovaných nástrojů externích dodavatelů. Další z hlavních vlastností platformy WMI je především jednoduché hromadné dotazování a následná inventarizace.

Model WMI se skládá ze tří částí: (16)

- **Prostředky WMI** – k prostředkům patří vše, k čemu lze přistupovat pomocí rozhraní WMI: systém souborů, siťové komponenty, protokoly událostí.
- **Infrastruktura WMI** – infrastruktura zahrnuje tři složky: službu WMI, uložiště služby WMI a zprostředkovatele rozhraní WMI. Nejdůležitější z nich jsou zprostředkovatelé rozhraní WMI, protože díky nim může rozhraní WMI získat potřebné informace.
- **Příjemnci WMI** – příjemce konzumuje date z rozhraní WMI. Příjemce může být rutina prostředí Windows Powershell.
-

5.16.1 Využití WMI prostřednictvím PowerShell

Výhodou Powershellu je skutečnost, že Windows Management Instrumentation (WMI) je již součástí .NET Frameworku na kterém je PowerShell postaven. Pro použití slouží cmdletů Get-WmiObject. Přes tento příkaz lze zavolat jakoukoliv třídu či funkci z WMI a tím číst nebo měnit konfiguraci operačního systému jak na lokálním klientovi tak i na vzdáleném. Syntaxe WMI je velmi intuitivní.

Obrázek 8 - WMI

```
Get-WmiObject -Class Win32_OperatingSystem #Zobrazí informace o systému
```

zdroj: vlastní

5.16.2 Využití WMI v ActiveDirectory

Modul ActiveDirectory rozšiřuje možnosti práce s doménou Active Directory. Tento modul není součástí Powershellu a je nutné ho inicializovat před použitím. Pomocí tohoto modulu je možné plně pracovat s Active Directory jen pomocí prostředí Powershell. ActiveDirectory obsahuje velké množství cmdletů, které slouží k rozsáhlé administraci a konfiguraci v počítačů a uživatelů v Active Directory.

Příkazy označovány jako GPO (Group Policy Object) slouží pro editaci Group Policy v Active Directory. Pomocí těchto příkazů můžeme konfigurovat veškerá nastavení Group Policy a tím naprosto nahradit klasickou Group Policy Management Console, která se používá v serverových verzích operačních systémů Windows.

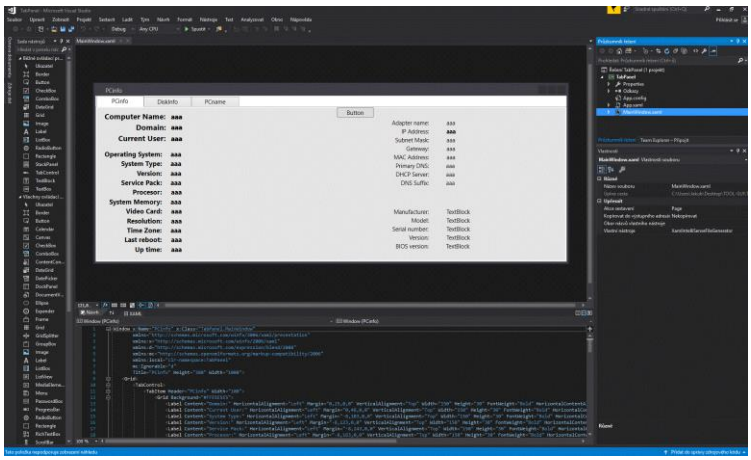
5.16.3 CIM

Mezi nové a atraktivní změny funkcí rozhraní WMI v prostředí Windows Powershell patří zavedení modelu CIM. Rutiny modelu CIM poskytují nový způsob načítání existujících instancí rozhraní WMI.(16) Mezi hlavní výhody rutiny CIM je její rychlost ve vracení informace oproti „Get-WMIObject „ Jako další hlavní výhodou je, že rutina CIM používá k síťovým přenosům službu WinRM. Službu WinRM lze velmi snadno konfigurovat a tak se dá jednoduše využít ke správě vzdálených zařízení. Rutina WMI používá, dosuď službu DCOM, která nemusí být vhodná v případě zohlednění Firewall pravidel.

6 Microsoft Visual studio

Jedná se o vývojové prostředí (IDE) od společnosti Microsoft. Je využíváno pro vývoj konzolových aplikací a aplikací s grafickým rozhraním. Vestavěné a přídavné moduly od doplnění podpory pro verzovací systémy po nové nástroje jako editory a vizuální designery pro doménově specifické jazyky z Visual Studia dělají univerzální vývojové prostředí téměř na všech úrovních. Visual Studio nepodporuje žádný programovací jazyk nebo nástroj samo o sobě. Místo toho je mu možno přidat různá rozšíření funkčnosti.

Obrázek 9: Visual Studio

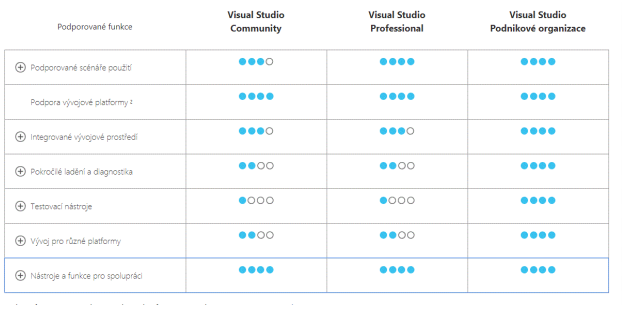


zdroj: vlastní

Porovnání verzí

- Visual Studio Community 2017 : Bezplatné plně vybavené integrované vývojové prostředí (IDE) pro studenty, vývojáře pro open-source a individuální vývojáře
- Visual Studio Professional 2017: Profesionální vývojářské nástroje, služby a výhody předplatného pro malé týmy
- Visual Studio Enterprise 2017: Komplexní řešení splňující náročné požadavky týmů všech velikostí na kvalitu a škálování

Obrázek 10: Porovnavací tabulka VS



Podporované funkce	Visual Studio Community	Visual Studio Professional	Visual Studio Podnikové organizace
⊕ Podporované scénáře použití	●●●○	●●●●	●●●●
Podpora vývojové platformy ‡	●●●●	●●●●	●●●●
⊕ Integrované vývojové prostředí	●●●○	●●●○	●●●●
⊕ Pokročilé ladění a diagnostika	●●○○	●●○○	●●●●
⊕ Testovací nástroje	●○○○	●○○○	●●●●
⊕ Vývoj pro různé platformy	●●○○	●●○○	●●●●
⊕ Nástroje a funkce pro spolupráci	●●●●	●●●●	●●●●

zdroj:https://www.visualstudio.com/cs/vs/compare/?rr=https%3A%2F%2Fwww.c-sharpcorner.com%2Farticle%2Fbasic-information-and-overview-of-visual-studio-2017%2F

6.1 Windows Presentation Foundation (WPF)

6.1.1 WPF

Jedná se o knihovnu tříd pro tvorbu grafického rozhraní, která je součástí .NET Frameworku od verze 3.0. Pro vytvoření uživatelského rozhraní využívá značkovací jazyk XAML, který dokáže oddělit funkčnost a vzhled od aplikace. WPF chce docílit jednotného uživatelského rozhraní s 2D a 3D grafikou a je plnohodnotným nástupcem zastaralých Windows Forms.

WPF je již součástí v systémech Windows Vista, Windows 7 a Windows Server 2008 u starších systému je možné doinstalovat ve formě balíčku. Historické kódové označení WPF bylo Avalon.

6.1.2 Windows Forms

Windows Forms je knihovna tříd pro tvorbu grafického rozhraní a je i nadále součástí .NET frameworku. Prozatím nebyl oficiálně prohlášen za zastaralý a tak se nadále používají paralelně Windows Forms a Windows Presentation Foundation frameworky.

6.1.3 Extansible Application Markup Language (XAML)

Jedná se o značkovací jazyk založený na jazyce XML, který je využívaný k popisu grafického rozhraní v aplikacích od společnosti Microsoft. Je převážně využíván ve spolupráci s technologií Windows Presentation Foundation. Úprava kódu je možná jak ve Visual Studiu od Microsoftu tak i v poznámkovém bloku. Výhodou tohoto značkovacího jazyka je jeho velká jednoduchost, kdy je možné popsat vše i za pomoci .NET jazyků C# nebo VB.NET

Obrázek 11: Příklad XAML

```
<TabItem Header="DiskInfo" Margin="0" Width="100">
                <Grid Background="#FFE5E5E5">
                    <ListView HorizontalAlignment="Left" Height="222" Margin="10,89,0,0"
VerticalAlignment="Top" Width="966">
                        <ListView.View>
                            <GridView>
                                <GridViewColumn Header="Drive Letter" Width="140"/>
                                <GridViewColumn Header="Drive label" Width="140"/>
                                <GridViewColumn Header="Size" Width="140"/>
                                <GridViewColumn Header="Free Space" Width="140"/>
                            </GridView>
                        </ListView.View>
                    </ListView>
                    <Button x:Name="Load_diskinfo_button" Content="Button" HorizontalAlignment="Left"
Margin="317,24,0,0" VerticalAlignment="Top" Width="75"/>
                    <Label x:Name="WPFDiskLabel" Content="Label" HorizontalAlignment="Left"
Margin="22,47,0,0" VerticalAlignment="Top" Width="242"/>
                </Grid>
</TabItem>
```

zdroj: vlastní

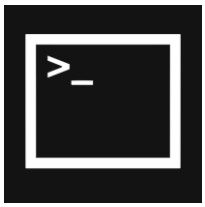
7 Nástroje pro vzdálenou správu

V následující kapitole si představíme některé další vhodné skriptovací jazyky nebo nástroje, které jsou použitelné ke splněním cílů této práce. V nabídce existují i nástroje na bázi grafického rozhraní, tyto nástroje jsem nebral v úvahu.

7.1 Prostředí Cmd.exe a PStools

Command prompt je standardně nasazen ve všech operačních systémech Windows od verze Windows XP a Windows server 2003. Přes toto rozhraní je možno pracovat s mnoha externími nástroji. Pro uvedení příkladu externího nástroje a vzdálené správy pro konfiguraci koncových stanic jsem zvolil doplněk PStools. Tento nástroj obsahuje 13 programů, které rozšiřují funkcionalitu příkazového řádku.

Obrázek 12 - logo CMD



zdroj: http://cybernetworkfirewall.blogspot.cz/2015/05/palo-alto-command-line-interface.html

PSFile	Zobrazí soubory, které mají uživatelé otevřené z jiných počítačů.
PSGetSid	Zobrazí jméno uživatele a počítače.

PSInfo	Výpis informací o systému.
PSPing	Slouží pro měření výkonu sítě.
PSKill	Ukončí procesy v systému.
PSList	Seznam a detailní informace o běžících procesech systému.
PSLoggedOn	Zobrazí uživatele, kteří jsou přihlášeni na určitý počítač.
PSLogList	Smaže heslo pro uživatele.
PSPasswd	Změni heslo pro uživatele.
PSService	Slouží pro práci se službami systému.
PSShutdown	Umožňuje vzdálené vypnutí či restart systému.
PSSunspend	Pozastaví systémový proces
PSUptime	Zobrazí dobu, po kterou je operační systém v provozu.

7.2 VBScript

Skriptovací jazyk Visual Basic Script byl poprvé představen společností Microsoft v roce 1996, kde se jednalo o verzi jazyka Visual Basic. Současná verze VBS je 5.8, která byla představena společně s vydáním operačního systému Windows 7. Od vydání této verze již nepokračuje jeho vývoj a pozornost se přesměrovala na vývoj Powershellu. Starší skripty VBS jsou nadále podporovány i v nejnovějších operačních systémech od společnosti Microsoft. VBS při své práci používá COM – Component Object Model, který se používá mezi procesy komunikace a k vytváření dynamických objektů.

Obrázek 13 - logo VBS

zdroj: httpslatesthackingnews.comvbscript-icon

7.3 Porovnání nástrojů pro vzdálenou správu

V následující kapitole porovnáme výše uvedené možné nástroje pro vzdálenou správu koncových stanic a porovnáme jejich vzájemnou funkcionalitu oproti skriptovacímu jazyku Powershell. Tímto chceme vyloučit možnost jiného lepšího nástroje pro vzdálenou správu počítače.

7.3.1 Instalace

- **CMD+PStools**

  Příkazový řádek CMD je také součástí všech operačních systémů Windows, tudíž není nutná žádná dodatečná instalace. V případě přídavného modulu, který je využit v této práci je nutné modul stáhnout z internetových stránek a nakopírovat do adresáře „\windows\system32“ nebo jej spouštět pomocí příkazového řádku rovnou ze složky do které se nejdříve musíme příkazy cd přepnout.
- **VBScript**

  VBScript je součástí všech operačních systémů Windows. Nevyžaduje další dodatečnou instalaci. Spouštění skriptu probíhá přes příkazovou řádku nebo spouštěním souboru označeného koncovkou „.vbs“
- **Powershell**

  Powershell od verze 2.0 je automaticky implementován do operačních systémů Windows 7 a Windows server 2008. Pro starší operační systémy Windows XP a Windows Server 2003 je distribuován ve formě service packu.

7.3.2 Funkcionalita

- **CMD+PStools**

  Příkazový řádek je velmi zastaralý a již se počítá s tím, že jej v nejbližší době nahradí Powershell.
- **VBScript**

  VBScript používá komponenty COM pro přístup k funkcím. VBScript je již zastaralý nástroj a již nové funkcionality nepřibývají. Například správa .NET aplikace je velmi obtížná až nemožná.

- **Powershell**

  Powershell je postaven nad .NET, což mu umožňuje využívat základních funkcí, které dokáže dále spojovat s aplikacemi od Microsoftu.

7.3.3 Přenositelnost

Všechny nástroje jsou do dnešního dne podporovány všemi operačními systémy od společnosti Microsoft.

7.3.4 Bezpečnost

- **CMD+PStools**

  Příkazová řádka také nepodporuje digitální podpis.
- **VBScript**

  Nepodporuje digitální podpis, je závislý na credentials operačního systému.
- **Powershell**

  Powershell podporuje digitální podpis spuštěného programu. V prostředí Powershell se musí nastavit bezpečnostní politika pomocí příkazu Set-ExecutionPolicy. Bezpečnostní politiky a jejích nastavování jsou uvedena v kapitole o Powershellu.

7.3.5 Podpora a údržba

- **CMD+PStools**

  Příkazová řádka je stále součásti veškerých operačních systémů od Microsoftu. Doplňkový modul PStools, který rozšiřuje možnosti klasické příkazové řádky je aktualizován s každou novou verzí operačního systému, V současné době nejnovější update je z července roku 2016. Pstools se nadále musí instalovat ručně jako doplňkový modul.
- **VBScript**

  Poslední verzí VBScript je verze 5.8 která byla představena společně s operačním systémem Windows 7. Od této doby se již neaktualizuje, avšak VBScript je stále podporován. Historicky vytvořené skripty je možné spustit i na nejnovějších operačních systémech. Podpora od výrobce již není dostupná, ale existuje velká komunita, která tento nedostatek vynahrazuje.

- **Powershell**

  Microsoft se v současné době zaměřuje hlavně na Powershell a jeho rozšiřování CMDletů. Nejnovější verzí je nyní Powershell 6.0 která se nachází ve verzi preview. Výrobce si velmi zakládá na kvalitní dokumentaci a vydávání opravných patchů jako součást systémových aktualizací operačního systému.

7.3.6 Vzdálená správa

- **CMD+PStools**

  Samostatná příkazová řádka neumožňuje pracovat se vzdáleným klientem. Ve spojení s modulem PStools a programem v něm obsaženým PSExec je možné se vzdáleně připojit**.** PSexec umožňuje spuštění CMD na vzdáleném počítači v lokálním okně.

- **VBScript**

  Pro vzdálené spuštění programů přes VBScript je možné za použití dvou tříd Win32_Process a Win32_ScheduleJob. Třídy využívají WMI.

Obrázek 14 - Příklad VBScript

```
strComputer = "." # "." znamena lokalni pocitac
strCommand = "notepad.exe"

Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set objProcess = objWMIService.Get("Win32_Process")
# propojeni ke vzdalenemu PC
errReturn = objProcess.Create(strCommand, null, null, intProcessID)
# process
If errReturn = 0 Then
Wscript.Echo "notepad.exe was started with a process ID: " & intProcessID
Else
Wscript.Echo "notepad.exe could not be started due to error: " & errReturn
End If
```

zdroj 2: vlastní

- **Powershell**

  Vzdálenou správu ve skriptovacím jazyce Powershell lze provést dvojím způsobem. První způsob je do příkazu zadat parametr –ComputerName, který spustí příkaz na definovaném počítači. Tato funkcionalita je hojně využívaná administrátory v případě nutnosti provedení stejné operace na mnoha počítačích.

```
Invoke-Command -ComputerName Server01 {Get/UICulture}
```

Druhým způsobem je použití vzdálené instance programu tzv. session. Tato funkcionalita spustí na vzdáleném počítači instanci, která se administrátorovi otevře v okně Powershellu. Druhá varianta vzdáleného klienta se používá spíše u konfigurace serverů, kdy je potřeba aby administrátor viděl výsledky ihned a mohl využít možnosti uživatelského rozhraní.

```
Enter-PSSession Server01
```

7.3.7 Práce s Active Directory

- **CMD+PStools**

  Příkazový řádek ani jeho přídavný modul PStools práci s Active Directory neumožňuje.
- **VBScript**

  Podporu práce s Active Directory v sobě VBScript v základu nemá obsaženou. Tuto funkcionalitu lze doplnit přídavným modulem. Při využití LDAP nabízí široké možnosti pro získávání informací z domény.
- **Powershell**

  Powershell jako jediný nástroj má již v sobě implementovány příkazy vyloženě pro práci s Active Directory. V případě potřeby je možné dále funkcionalitu rozšiřovat přídavnými moduly. Powershell plně podporuje správu a administraci Active Directory v doméně.

7.3.8 Práce s WMI

- **CMD+PStools**

  Příkazová řádka sama o sobě nedokáže pracovat s WMI objekty. Funkcionalitu jí zaručují přídavné moduly, kterými je možné CDM rozšířit.
- **VBScript**

  VBScript také umožňuje práci s WMI. V prvotní části kódu je nutné vytvořit objekt nacházející se ve jmenném prostoru „winmgmts:\\.\root\cimv2“. Informace jsou vypsány po načtení objektu a pomocí určitého dotazu.
- **Powershell**

  V Powershellu je možné s WMI objekty pracovat pomocí interního příkazu „Get-WmiObject“.

8 Analýza

V následující kapitole se budu zabývat analýzou testovacího prostředí, analýzou softwarů nabízených pro správu a analýzou nejčastějších požadavků řešených na daném oddělení. Na konci této kapitoly bude uvedeno doporučené řešení.

8.1 Popis prostředí společnosti

Společnost ve své správě eviduje na 2500 koncových zařízení. V takovémto počtu je nutné brát velký zřetel na bezpečnost dat a jejich vynášení ze společnosti. Z důvodu vysoké bezpečnostní politiky je veškerá síťová komunikace monitorována, uživatelé mají omezený přístup k jistým webovým stránkám a mají zakázáno jakékoliv stahování dat z internetu. Dalším bezpečnostním opatřením je také zákaz připojování externích datových disků a využívání cloudových služeb. Uživatelé dále nemají oprávnění instalace programů. V případě nutnosti instalace jakéhokoliv programu je nutné kontaktovat Oddělení IT supportu.

Síťové prostředí

Veškeré síťové a doménové komponenty si společnost obstarává sama. Celá síť je postavena na 1Gb ethernetové síti a WIFI síti používající N třídu. Na síťových portech je zapnuté ověřování přes protokol IEEE 802.1X. Slouží k ověřování zařízení pokoušejícího se připojit do doménové sítě. Pro vzdálené připojení k doménové síti je využíván software Junos Pulse, který zprostředkovává VPN připojení do sítě. V celé doméně jsou nastaveny vysoké bezpečnostní politiky.

Koncová zařízení

Koncová uživatelská zařízení pracují na operačních systémech společnosti Microsoft. Jedná se o operační systémy Windows 7 (x86), Windows 7 (x64) a v současné době probíhá migrace na Windows 10 s 64bitovou architekturou. Na zařízení jsou dále nainstalovány kancelářské balíčky Microsoft Office ve verzích od 2010 do 365. Veškeré aktualizace jsou distribuovány automaticky a nastaveny jako Group policy. Jako antivirová ochrana slouží software Microsoft Endpoint Protection. Uživatelé mají omezená práva na instalaci softwaru. Hardwarové konfigurace osobních počítačů a notebooků jsou od různých dodavatelů. Společnost se snaží obnovovat koncové zařízení a vyřazuje ze své evidence zařízení starší 6ti let.

Tabulka 3: Konfigurace základního notebooku

Procesor	Intel Core i5-2520M (2×2.50/3.20 GHz)
Operační paměť	4GB
Pevný disk	320GB (7200 ot/min)
Grafická karta	Intel HD 3000
Síťové prvky	Wi-Fi, Gigabit Ethernet, WWAN příprava
Operační systém	Windows 7 Pro 64bit

Tabulka 4: Konfigurace základního desktopu

Procesor	Intel Core i5-2400S (4×2.50/3.30 GHz)
Operační paměť	8GB
Pevný disk	500GB (7200 ot/min)
Grafická karta	Intel HD 2000
Síťové prvky	Gigabit Ethernet
Operační systém	Windows 7 Pro 64bit

8.1.1 IT Helpdesk oddělení

IT oddělení je koncipováno do několika úrovní. Helpdesk je první kontaktní místo, které uživatelé kontaktují v případě jakékoliv poruchy na PC. Ve všech případech pracovníci Helpdesku se snaží daný požadavek vyřešit okamžitě a až v případě, kdy nenaleznou řešení, kontaktují odpovědné oddělení dle obsahu požadavku. V případě předání požadavku je pracovník Helpdesku povinen zjistit informační minimum ohledně problémového PC. Jako informační minimum je považován název počítače. Helpdesk oddělení je možné kontaktovat telefonicky nebo mailem 24/7.

8.1.2 Aktoři

8.1.2.1 Uživatelé

- **Uživatel bez znalostí práce s PC**

  Lenka Malá

  Ve firmě působí jako uklízečka, počítač ke své práci potřebuje jen z důvodu kontroly jídelního lístku a jednou měsíčně pro kontrolu výplatní pásky.

  Je ji 50let a zkušenosti s PC má minimální. Vlastní starší typ mobilu, se kterým umí zručně zacházet. Nedokáže odpovědět na základní otázky Helpdesku a ani není svolná ke spolupráci v případě navádění Helpdesk technikem.

  V případě jakékoliv poruchy PC je instruována svou nadřízenou neprodleně kontaktovat Helpdesk oddělení.

- **Uživatel se znalostí práce s PC**

  Pavel Novák

  Je mu 30let a ve firmě působí jako účetní.

  Počítač ke své práci potřebuje každodenně. Zároveň je zapálený do nových technologií a nedělá mu problém zodpovědět i složitější otázky kladené od Helpdesku v případě poruch PC.

8.1.2.2 Administrátoři

Scénář 1: Administrátor má veškeré potřebné informace

Administrátor může ihned přejít k řešení požadavku nebo začít získávat podrobnější informace o počátku problému s počítačem.

Scénář 2: Administrátor nemá potřebné informace

V případě pokud administrátor nemá potřebné informace, má na výběr dvě možnosti řešení. V prvním případě administrátor může vzít požadavek a navrátit ho zpět na helpdesk oddělení pro doplnění informací (název počítače). Ve druhém případě je administrátor nucen opětovně zavolat

uživateli. V případě kancelářských pracovníků nenastává žádný problém. Ztížení podmínek nastává, až v případě, kdy volající je zaměstnán ve směnném provozu a není přítomen na pracovišti v danou chvíli potřeby. Tímto se může zkomplikovat a prodloužit doba řešení požadavku.

8.2 Analýza požadavků na software

Všeobecné nároky na software pro administraci vzdálených počítačů se neliší od nároků na kterýkoliv jiný software, avšak jistá jedinečná specifika systému zde musíme zohlednit.

V další části jsou uvedené vlastností, kterými je potřeba se při výběru softwaru pro správu zabývat.

Funkcionalita

Jednou z hlavních vlastností pro software nebo nástroj na správu stanic je jeho funkcionalita. Od této vlastnosti se očekává především plnění úloh, které administrátorovi usnadní práci a zredukují časovou náročnost operace. Požadavky administrátorů na funkcionalitu se mohou výrazně lišit a z tohoto důvodu na trhu existuje velké množství správcovského softwaru, jehož řešení se snaží co nejvíce přiblížit požadavkům na něj kladené. Následkem mnoha požadavků na funkcionalitu dochází ke složitosti daného softwaru. Nevyužité funkcionality softwaru mohou nepříznivě ovlivnit ovladatelnost, náročnost instalace, údržbu ale i potřebu systémových zdrojů. S těmito funkcemi se podstatně zvyšuje i cena daného softwaru.

Mezi podstatné vlastnosti administračního softwaru je především provádění hromadných operací a automatické řešení určitých situací. Software může provádět operace zajišťující instalaci, úpravu a update systému. Dále pak umožňuje nastavování chování jak systému, tak i softwaru. Tyto operace typu získávání informací a dat z počítačů jsou následně zpracovány do analýz, grafů nebo mohou vytvářet spojení pro další systémy. Část z nabízených softwaru nabízejí funkcionalitu zasílání zpráv o problémech operátorovi nebo reagují podle předem stanoveného scénáře.

Existují případy, kdy místo komplexního nástroje bohatě postačí soubor několika drobných programů, které dokáží samostatně řešit úlohy. Pokud tyto nástroje dokáží nahradit komplexní řešení, mohou ušetřit společnosti nemalé výdaje za software. Tuto skutečnost je potřeba zohlednit a pečlivě zvážit při výběru softwaru pro podnik.

Spolehlivost

Software obsahuje dohledové moduly, které po většinu času běží na pozadí a permanentně zatěžují systémové zdroje počítačů. Proto je nutné na klientské i serverové části softwaru klást vysoké nároky, aby přílišně nezatěžovali běžný provoz zařízení ani sítě. U takto neustále zatíženého počítače může docházet k neuvolnění paměti nebo k jejímu přetečení, které mohou skončit až zahlcením systému a dojít k jeho pádu. V klientské části softwaru nesmí docházet k přílišnému zatěžovaní systémových zdrojů koncové stanice a také by mělo být redukováno množství dat přenášené po síti. Těmto okolnostem by mělo být předcházeno správným testováním a moduly by se měly po celou dobu a za všech okolností chovat stejně.

Přenositelnost

V dnešní době podpora více druhů zařízení a systémů je nedílnou součástí všech softwaru pro správu. Software pro správu by měl zohledňovat historicky nasazené systémy i systémy do budoucna uvažované. Tímto zohledněním můžeme předejít budoucím problémům s migrací na jiný systém. Migrace by totiž představovala vynaložení velkých finančních i lidských zdrojů. Finančními zdroji je myšleno, že některé systémy jsou vázány na daný hardware a ten je součástí dodávky systému. Systém například nemůže být implementován na běžný hardware jelikož potřebuje zvláštní nadstandardní konfiguraci.

Pro administraci nehomogenních systémů jsou často využívány menší nástroje, které dokáží nahrazovat velký komplexní systém.

Ovladatelnost

Kvalitní ovládací rozhraní výrazně ovlivňuje dobu potřebnou k dosažení daného úkonu. Pro administrátory bude výhodnější rychlá textová konzole, méně zkušení uživatelé přivítají použití intuitivního grafického rozhraní. Rozhraní by mělo umožňovat provádění opakujících se operací na různých zařízeních a v různých časech. Při použití daného softwaru by měl administrátor dostat jasný přehled o daných systémových prostředcích koncových stanic nacházejících se v dané síti. Při nenadále situaci by měl mít možnost rychlé reakce na vzniklou závadu.

Modularita

Funkcionalitu na všech úrovních nedokáže pokrýt žádný systém, proto je nutné, aby systém umožňoval rozšiřitelnost za pomoci dodatečných modulů. Tyto moduly poté zajistí možnost provázanosti systému s dalšími komponentami. Provázanosti lze docílit prostřednictvím specializovaných prvků, které zajistí komunikaci s dalším softwarem, nebo dodržováním průmyslových standardů, které umožní propojení.

Představa zákazníka o funkcionalitě daného softwaru je rozdílná než jaké funkcionality aplikace mají v sobě implementovány. Proto je potřeba danou funkci doplnit přídavným modulem, kdy zákazník může požádat autora daného softwaru o úpravu. Takové to individuální úpravy často bývají velmi finančně náročné. Další možností přidáním funkcionality je možnost vlastního naprogramování. V tomto případě je nutná znalost programovacího jazyka a také zdrojového kódu. Zdrojový kód musí autor softwaru zpřístupnit pro modifikaci, což z praxe bývá velmi náročné až nemožné.

Údržba

Údržba je nedílnou součástí veškerých systémů. Nedostatky ale i chyby se projeví až s odstupem času a tak vznikají nové verze systému nebo opravy vzniklých chyb. Tudíž by stávající části systému měly být lehce nahraditelné verzemi novými.

Podpora

Dlouhodobá podpora pro zákazníka je velmi důležitá u každého softwaru. Systémy pro správu stanic bývají v provozu v řádu několika let. Tyto systémy se řadí mezi kritické a jejich migrace nebo nahrazení bývají velmi náročné. Dlouhodobá podpora zahrnuje vydávání průběžných updatu a novějších verzí systému. Dále pak poskytuje poradenství při problémech s nasazením i provozem daného systému. Podpora na takovéto úrovni bývá většinou zpoplatněna nebo je zohledněna již při objednání systému jako celku.

Komerční technologie od velkých společností mají již kvalitní podporu zajištěnou. U nekomerční technologie je nutné, aby se se vzniklými potížemi administrátoři vypořádali sami. Toto je také nutné zohlednit při výběru technologie.

Cena

Na prvním místě při výběru softwaru je jistě cena daného softwaru. Podle ceny se může rozlišovat kvalita i funkcionalita softwaru. V ceně softwaru je zahrnut software samotný případně dodávaný hardware jako technologie. Do ceny se promítne také její dlouhodobá podpora, vývoj doplňku, náklady na administraci a zaškolení uživatelů. Za dodaný software je možné zaplatit jednorázově i s podporou dle uzavřené smlouvy nebo lze platit pravidelné poplatky a software mít jako jakousi službu.

8.3 Analýza krabicového softwaru

V této kapitole si představíme pár zástupců v řešení vzdálené správy a monitoringu koncových stanic.

8.3.1 Správce IT

Software Správce IT je vyvíjen společností Micos od roku 1993, jedná se o software pro evidenci hardwarového a softwarového vybavení koncových stanic a jiného majetku. Instalace softwaru probíhá pouze pod systémem Windows a funguje jako desktopová aplikace. Sběr dat probíhá automaticky pomocí skenovacích nástrojů nebo ručně na jednotlivých stanicích. Automatický sběr dat lze nastavit v různých intervalech a tak jsou data stále aktuální. Cena softwaru od Micosu je příznivější než od ostatních dodavatelů.

8.3.2 Hp OpenView

Sada nástrojů od společnosti HP známá také jako HP OpenView je určena pro komplexní správu infrastruktury IT. HP OpenView tvoří více než 30 nástrojů specializujících se na různá odvětví IT, můžeme je rozdělit na správu sítí a internetu, nástroje na zálohování, správu systému, aplikací a procesů. Celé portfolio nástrojů dokáže spolu komunikovat a jsou velmi provázané mezi sebou. Dále pak umožňuje doplňování komponentů třetích stran. Snadnou implementací nových funkcí je schopna velice pružně reagovat na změny požadavků systému.

Systém HP OpenView disponuje robustním a komplexním řešením, které je možné přizpůsobit dle požadavků zákazníka. Vzhledem k velkým pořizovacím nákladům se takto komplexní řešení doporučuje převážně do velkých společností než do malých a středních podniků.

Níže v tabulce si uvedeme některé z nástrojů.

Tabulka 5 - Funkcionality HP OpenView

Pro správu sítí a internetu	• OV Network Node • Manager NNM, • OV Problem Diagnostic • OV Performance Insight for Networks • OV Topology Server

	• OV Internet Services.
Správa zálohování	• OV OmniBack • OV SAM,
správa systémů a aplikací	• OV Operation for UNIX • OV Operations for Windows • OV Performance • OV Smart Plug-Ins

Zdroj: Hp.com

8.3.3 Nagios

Software Nagios jako jediný z uvedených příkladů veden pod licencí GPL, tudíž je volně šířitelný. Nagios je open source software vyvíjený pro systém Linux, ale lze jej pomocí různých pluginů provozovat i na jiných systémech. Pomocí Nagiosu lze sledovat vybrané stanice v počítačové síti a jimi poskytované služby. V případě výskytu závady dokáže ihned upozornit administrátora, který může urychleně zjednat nápravu. Monitorovací služba je spouštěna dle předem stanoveného scénáře a kontroluje specifikované uzly a služby. K této funkcionalitě Nagios využívá externí moduly, které oznamují výsledky kontrol hlavnímu modulu.

Nagios se zdá jako vhodný nástroj pro menší a střední podniky, avšak i jeho implementace se neobejde bez investicí. Mezi jeho hlavní nevýhody patří složitá konfigurace, která probíhá manuálně a zároveň vyžaduje server s velkým množstvím paměti.

Monitorovací možnosti

- Monitorování síťových služeb
- Monitorování systémových prostředků operačních systémů
- Vzdálené monitorování přes protokol SSH nebo přes zašifrovaný SSL tunel.
- Tvorba vlastních pluginů pro monitorování jejich služeb
- Hierarchická struktura, která umožňuje vytvářet stromovou strukturu mapy sítě.
- Notifikace pomocí pageru, e-mailu, SMS, VoIP.
- Proaktivní řešení problémů (restart služby)
- Webové rozhraní pro vizuální kontrolu stavu sítě.

8.3.4 Vyhodnocení externích programů.

Výše uvedené softwary byly použity jako analýza vnějšího prostředí a mají sloužit jako přehled o nabízených softwarech pro správu. Na trhu existuje velké množství podobných softwaru, které nabízejí mnoho rozdílných funkcí. Tyto softwary jsem si schválně vybral, jelikož každý zvlášť reprezentuje jinou oblast. Máme zde uvedené zástupce z nižších, středních a vyšších úrovní. Z přehledu lze jednoduše odvodit, že mezi softwary existují velké rozdíly. Tyto rozdíly můžeme, naleznout jak ve funkcionalitě, modularitě, ovladatelnosti, spolehlivost, ale také hlavně v ceně. Cena vždy bude nezanedbatelnou položkou při výběru takového softwaru pro správu a je na místě jí neopomíjet. Například software od HP představuje komplexní řešení, které je možné libovolně měnit dle požadavků zákazníka, ale také od toho je odvíjena cena. Na druhou stranu software Nagios je volně šiřitelný a jeho funkcionalita se dá dále rozvíjet za pomocí přídavných modulů. U tohoto systému není zaručena podpora a i nasazení samostatných externích modulů může být bezpečnostním rizikem. Zástupce střední úrovně Správce IT nabízí kompromis mezi dvěma extrémy výše uvedenými, ale stále je tu náklad ve formě implementace.

U každého softwaru pro správu jsem vytvořil mé subjektivní hodnocení s přihlédnutím k analýze požadavků na software. Hodnocení jsem rozlišil za pomocí znaménka plus, kde jedno plus znamená méně dobrý a tři plus velmi dobrý. Jednoduché grafické zobrazení může být nápomocno při rychlém rozhodování o pořízení softwaru.

Pokud vezmeme v potaz skutečnost, že cílem práce je usnadnění administrace pouze na daném oddělení. Není potřeba využívat služeb externího softwaru a vynakládat finanční ani prostředkové zdroje. Bohatě postačí soubor několika jednoduchých nástrojů pro usnadnění a urychlení práce administrátora.

9 Druhy metodik

Metodika jako taková znamená strukturované rozdělení vývoje aplikace do oddělených fází, kde jednotlivé fáze jsou dále děleny na činnosti se zaměřením na možnosti lepšího plánování, kontroly nad procesem a managementem vývoje.

Většina metod sdílí kombinaci několika fází vývoje.

- analýza problému,
- shromažďování požadavků pro navrhované řešení,
- návrh plánu vývoje aplikace,
- vývoj aplikace,
- testování aplikace,
- nasazení aplikace,
- údržba a opravy chyb aplikace.

Všechny stadia vývoje aplikace označujeme souhrnně jako životní cyklus vývoje systému (SDLC) a různé metodiky mohou jednotlivé fáze vývoje vykonávat v různém pořadí a v různé časové dotaci. I podrobnost dokumentace v jednotlivých fázích se může lišit v závislosti na použité metodice. Další neméně důležitou odlišností v jednotlivých metodikách může být, zdali se k jednotlivým fázím vývoje v rámci metodiky přistupuje sekvenčně, nebo v cyklech či iteracích (Schwalbe, 2011).

9.1 Tradiční metodiky

Jako zástupce tradičních metodik jsem vybral tyto tři modely: vodopádový model, spirálový model a v neposlední řadě Rational Unified Process.

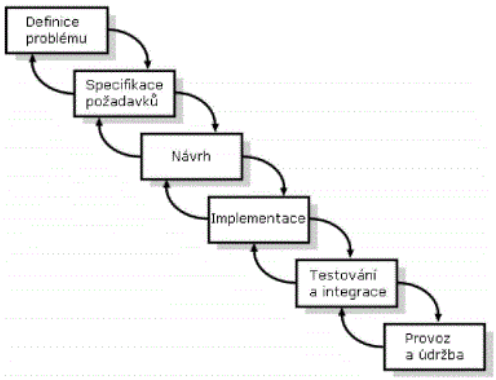
9.1.1 Vodopádový model

Je považován za prvotní ucelenou metodiku vývoje softwaru. Využívá sekvenčně seřazené fáze bez iterací, kde mezi jednotlivými fázemi je schvalovací proces, přes který musí veškeré dokumenty projít. Bez této kontroly dokumentace nemůže vývoj pokročit do další fáze.

Základní fáze vodopádového modelu.

1. Definice problému, poznání zákazníka a cílové oblasti
2. Analýza a specifikace požadavků
3. Návrh
4. Implementace
5. Testování a integrace
6. Provoz a údržba

Obrázek 15: Vodopádový model



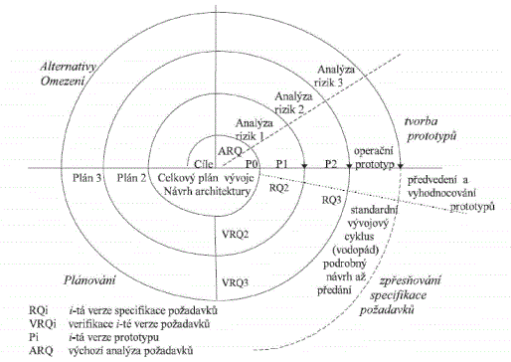
Zdroj: http://slideplayer.cz/slide/3122924/

9.1.2 Spirálový model

Spirálový model vychází z modelu vodopádového, ale jsou k němu přidány další vlastnosti, mezi které patří iterativní přístup a podrobná analýza rizik. Z tohoto důvodu je občas zařazován do skupiny přístupu řízených rizik (risk-driven approach)

Základní myšlenkou spirálovitého modelu je stanovit na začátku jen rámec architektury a funkčnosti celého systému a ten postupně rozpracovávat do detailů. Dále pak v každém cyklu je prováděná analýza rizik, která určuje další směr vývoje projektu.

Obrázek 16: Spirálový model



Zdroj: http://slideplayer.cz/slide/2913967/

9.1.3 Rational Unified Process

Celá tato metodika je objektově orientovaná a spadá do skupiny přístupů řízených případy užití. (use-case-driven approach). Vývoj dle Rational Unified Process probíhá v iterativních cyklech. *Úvodní vývojový cyklus* je první cyklus, kde je jeho výsledkem funkční softwarový produkt implementující nejzákladnější funkcionalitu. Po úvodním cyklu následují další rozvíjející cykly, jejichž počet závisí na rozsahu projektu.

Cykly většinou obsahují 4 fáze.

- Zahájení (10%)
- Projektování (30%)
- Realizace (50%)
- Předání (10%)

9.2 Agilní metodiky

Jsou to takové metodiky, které umožňují rychlý vývoj softwaru a zároveň dokáží reagovat na změnu požadavků v průběhu vývojového cyklu. Správnost daného systému je ověřována pomocí metodiky rychlého vývoje, předložení výsledků zákazníkovi a následné zapracování úprav dle zpětné vazby. Agilní metodiky našli uplatnění i v jiných odvětvích než jen pro vývoj softwaru, ve velkém se využívají v marketingovém plánování nebo BI.

Základní priority a principu agilního programování:

Priority agilního programování

- Lidé a jejich spolupráce před procesy a nástroji
- Fungující software před obsáhlou dokumentací
- Spolupráce se zákazníkem před sjednáváním smluv
- Reakce na změnu před dodržováním plánu

Principy agilního programování

- Nejvyšší prioritou je uspokojit zákazníka skrz rychlé a průběžné dodávání kvalitního software.
- Změnové požadavky jsou vítány, dokonce i v průběhu vývoje. Agilní procesy je zpracují tak, aby zákazníkovi přinášely konkurenční výhody.
- Dodávejte fungující software často, v intervalech týdnů až měsíců. Upřednostňujte kratší intervaly dodání.
- Lidé z businessu a vývojáři musí spolupracovat každý den během celého projektu.
- Pro práci na projektu vybírejte motivované jedince. Dejte jim prostředí a podporu, kterou potřebují, a důvěřujte jim, že práci dokončí.
- Nejúčinnější metoda sdílení informací vývojářskému týmu (i uvnitř tohoto týmu) je osobní setkání.
- Fungující software je hlavním měřítkem postupu vývoje.
- Agilní procesy podporují udržitelný vývoj. Sponzoři, vývojáři i uživatelé by měli být schopní dodržovat stálý výkon dokud je třeba.
- Průběžná pozornost věnovaná technické dokonalosti a dobrému návrhu posiluje agilní přístup.
- Základem je jednoduchost – umění co nejvíce práce vůbec nedělat.
- Nejlepší architektury, požadavky a návrhy vznikají v týmech, které se samy organizují.
- Tým v pravidelných intervalech vyhodnocuje svou práci a upravuje své postupy tak, aby byl co nejefektivnější.

9.2.1 Extrémní programování (XP)

Extrémní programování je považováno za nejrozšířenějšího zástupce agilních metodik. Tato metodika se hodí pro menší projekty a malé týmy vyvíjející software podle zadání, které je nejasné nebo rychle se měnící dle přání zákazníka. Extrémní programování využívá jednoduché

principy, které ale přivádí do extrémů. Upřednostňuje časté dodávky softwaru v krátkých vývojových cyklech. Tato metoda dále navrhuje jednotkové testování, párové programování. Dále pak zastává názor programovat pouze co je v danou chvíli nezbytné a tvorba jednoduchého a jasného kódu.

9.2.2 Scrum

Hlavní částí metodiky je každodenní setkání týmu, nazývaná daily standups, kde každý člen zde referuje o své činnosti z předešlého dne a o tom co je v plánu dnes a s jakými problémy se potýkal nebo stále potýká. Metodika využívá iterativní vývoj, kde období iterace se nazývá Sprint a trvá 1-4 týdny. Výstupem z této iterace je demo vzniklých úprav, které je prezentováno zákazníkovi (stakeholderům). Stakeholdeři dále poskytují zpětnou vazbu a tím je možné rychle reagovat na změny v požadavcích.

Ve SRUM metodice jsou rozeznávány tři role:

- Product Owner: mezi jeho úkoly patří komunikace se zákazníkem a zpracovávat definici co nejlepšího produktu.
- Srcum Master: zajišťuje správné fungování vývojového týmu.
- Scrum Alliance: má na starosti školení a vývoj metodiky.

9.2.3 Lean development

Lean devolopmetn není ani tak agilní metodikou jako souborem pravidel, kde díky dodržování těchto pravidel by mělo zefektivnit a zrychlit vývojový proces.

Součástí metodiky jsou také principy a nástroje, které tyto pravidla umožní realizovat

Pravidla:

- eliminovat zbytečné
- zdůraznit proces učení
- rozhodovat se tak rychle a pozdě, jak je možné
- posílit odpovědnost týmu
- zabudovat integritu a vidět systém jako celek

9.3 Volba metodiky

Vlastní zadání projektu je velmi volné a není zde kladen žádný důraz na čas ani zdroje. Jedná se hlavně o zjednodušení práce daných oddělení uvnitř společnosti.

I když tento nástroj má být jakýmsi pilotním projektem až čas ukáže, zdali se s tímto nástrojem uživatelé ztotožní či nikoliv. V obou případech jej podstatné předpokládat do budoucna další požadavky na rozšíření tohoto nástroje. Je žádoucí udělat takovým způsobem, aby byl přehledný a jeho veškeré funkce lehce identifikovatelné. Pro dodržení těchto podmínek je nutné zvolit správnou metodiku, která pomůže s pochopením, vytvořením a implementováním jednotlivých požadavku na daný nástroj. V neposlední řadě je nutné, aby bylo umožněno provádět jednoduše a přehledně změny i nezainteresovaným osobám.

V kapitole druhy metodik jsem se zabýval výčtem různých přístupů k vývoji softwaru. Po prostudování jednotlivých metodik jsem došel k názoru, že žádnou z daných metodik nemohu použít pro potřeby mého nástroje využívajícího interpretovaný jazyk Powershell, jelikož tyto metodiky jsou spíše určeny pro potřeby vyšších programovacích jazyků. Funkce, které mají vyšší programovací jazyky Powershell postrádá nebo jsou v jeho případě velmi omezené. Tvorba nástroje za použití skriptovacího jazyka je jednodušší v tom, že se nemusíme tolik zabývat analýzami a samostatnými návrhy, ale jdeme přímou cestou, kde se orientujeme na konečný výsledek.

V praxi u jednoduchých skriptů není nutné dodržovat doporučené postupy nebo sofistikované způsoby vývoje, ale je žádoucí reagovat promptně na dané potřeby při vzniku kolize nebo problému. V určitých případech se do skriptu pouze implementují podmínky a vyzkouší se funkcionalita daného skriptu.

Z výše uvedených skutečností jsem se rozhodl využít metodiky Unified Process (UP), kterou si budu muset přizpůsobit potřebám mého vývoje. Jako další důvod pro zvolení této metodiky je tvrzení Jima Arlowa. *Jako rámec je ale metodika UP dostatečně přizpůsobivá a lze ji poměrně dobře upravit každému projektu a přizpůsobit jeho specifickým vlastnostem a požadavkům.*

Ke správnému porozumění metodiky UP je potřeba porozumět iteracím. Základní myšlenka je velice prostá – historie ukazuje, že člověku se obecně vzato řeší lépe menší problémy než větší. Snažíme se tedy o rozložení velkého softwarového projektu na řadu menších „miniprojektů“

Fáze Unified Process

- *zahájení - plánování*
- *rozpracování - období architektury*
- *konstrukce - počátky schopností provozu*
- *zavedení - nasazení produktu*

Dobu přechodu mezi jednotlivými fázemi nám určují tzv. milníky, které jsou v této metodice zásadní. Pro postup na další fázi musejí být milníky splněny.

V rámci Unified Process je rozlišováno pět základních postupů.

- Požadavky
- Analýzy
- Návrh
- Implementace
- Testování

Metodika Unified Process (UP) je velice orientovaná na detailní rozpracování všech požadavků. Také tato metodika počítá s mnohačetným vývojovým teamem, což se neztotožňuje s mým vlastním vývojem, proto jsem se rozhodl aplikovat jednotlivé metodické pokyny jen v nezbytném rozsahu.

10 Návrh řešení

V této kapitole se budeme věnovat vlastnímu vývoji nástroje za pomocí vybraných doporučených částí metodiky Unified Process (UP).

10.1 Fáze zahájení

Naplněním uvedených podmínek můžeme postoupit do další fáze projektu.

1. Dohodnout a zachytit klíčové požadavky uživatelů a zainteresovaných osob formou UML diagramu.
2. Potvrzení proveditelnosti obsažené v předloženém prototypu.
3. Načrtnout architekturu systému

10.1.1 Požadavky na základě analýz

Zpracování analýzy bylo uskutečněno pomocí průzkumu na oddělení HW a zároveň porovnáním statistiky dle tiket systému.

Analýza na HW oddělení.

Analýza nejčastějších činností proběhla za použití Brainstorming metody. Tato metoda je zaměřena na skupinové generování co nejvíce nápadů na dané téma, kdy je prvotní předpoklad, že lidé dokáží ve skupině na základě ostatních podmětů vymyslet více než jednotlivec. Skupina se skládala ze sedmi techniků pracujících na oddělení HW. Každý z těchto techniků měl prostor se vyjádřit k dané problematice a přijít s nápadem, který by oddělení jako celku zjednodušil práci a pomohl při identifikaci a následném řešení požadavků. Mezi techniky byla nejvíce diskutovanou množinou přesná identifikace koncové stanice. Jednalo se jak o její fyzické umístění, tak i ohledně informace o softwarových úpravách, které byly distribuovány odpovědným oddělením nebo uživatelem samotným.

Níže uvedený přehled zobrazuje přehled nejvíce diskutovaných problémů.

- Informace o PC (typ, bios, disk, SMART)
- Informace o síťovém adapteru a porovnání s DHCP
- „nový profil ve windows“
- Event log.
- Certifikáty
- Aktualizace
- Inventarizace (disk, RAM, bios,)

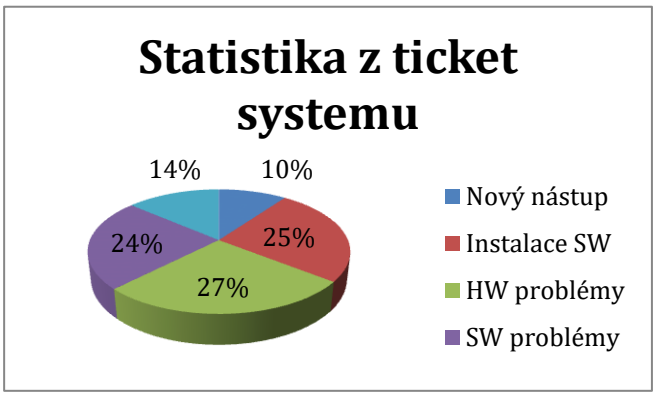
Statistika nejvíce požadavků dle Helpdesku.

Statistika byla získána z ticket systému, který společnost využívá pro evidenci veškerých požadavků od koncových uživatelů. Tiket obsahuje informace o volajícím uživateli, popis závady a na základě vyhodnocení Helpdesku i uvedení jaké oddělení se má danému problému věnovat. Technik dále může do přiřazeného tiketu vpisovat jeho postup řešení vzniklé závady. Informace o postupu řešení velmi pomáhá celému oddělení k získání přehledu, zdali tato závada nebyla již historicky řešena a případně jak byla vyřešena. Pokud požadavek připadá na oddělení HW, dle směrnic by měl záznam obsahovat veškeré informace o koncovém PC. Jako informační minimum je považován název počítače nebo jeho IP adresa.

V loňském roce bylo HW oddělením zpracováno přes 610 požadavků. Z těchto požadavků byly nejčastěji zpracovávané tyto:

- Nový nástup – Skládající se z vytvoření uživatelského účtu v Active Directory a následná příprava techniky dle požadavků uživatele.
- Instalace softwaru – Jedná se o instalace software, který není ve standartní OS.
- Hardwarové problémy – výměna HDD a RAM.
- Softwarové problémy – kolize mezi programy.
- Jiné – dodání techniky atp.

Graf 1 - Rozložení servisních požadavků



zdroj: vlastní

10.1.2 Výsledek analýzy nejčastějších činností

Dle šetření na oddělení bylo zjištěno, že je žádoucí vytvoření skriptu na zobrazování podrobných informací o koncové stanici a případné jejich uložení do souboru se specifickým názvem pro případnou evidenci. Mezi dalšími požadavky bylo rychlé zjištění vytíženosti koncového zařízení s případným výpisem Event logu.

Jako další požadavek byla možnost instalace nestandartního softwaru na více stanic najednou.

Ve všech případech se jedná o rutinní práci administrátora. V případě nestandardního chování koncového zařízení administrátor obvykle zjišťuje chybové hlášení přes Event log. Informace o zařízeních slouží případně k identifikaci správných ovladačů, které musejí být na danou stanici nainstalovány. Také díky těmto informacím může administrátor získat přehled v jaké konfiguraci je stanice osazena a případně si dopředu připravit její náhradní součástku. Automatické instalace bez nutnosti obtěžování uživatele je vždy vítanou funkcionalitou.

Ověření dle ticket systému

Dle ticket systému bylo zjištěno, že více jak polovina požadavků má charakter softwarového nebo hardwarové selhání. Tímto jsme si ověřili, že získávání informací z koncových stanic je důležitou součástí a byly potvrzeny výsledky šetření, které proběhlo na IT oddělení.

Automatizací zjišťování informací ze zařízení dojde k časové úspoře.

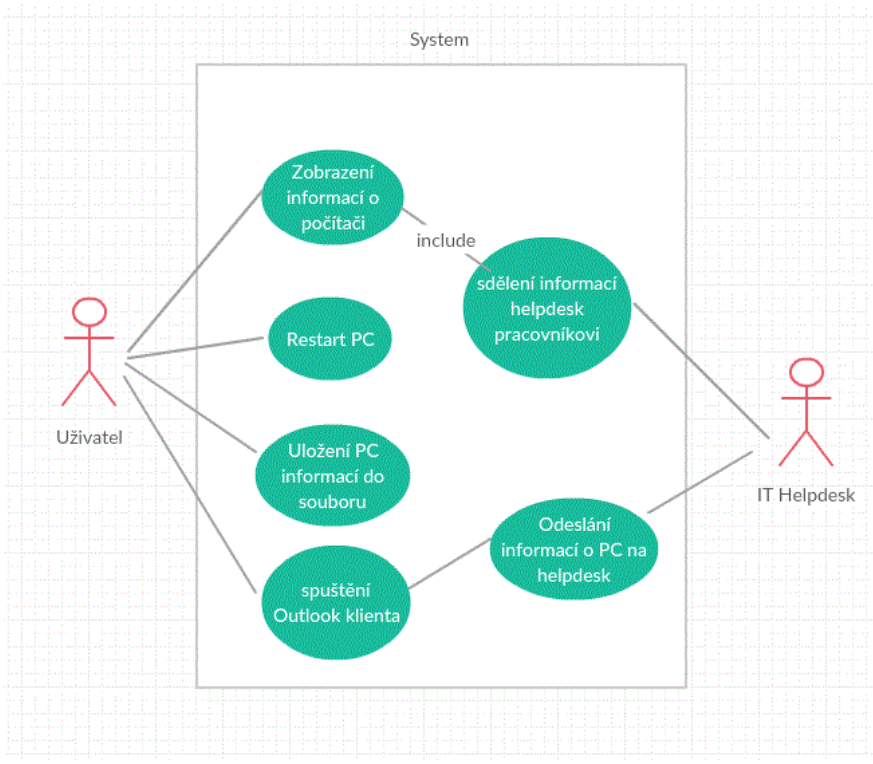
10.1.3 Vyhodnocení požadavků

Na základě těchto analýz provedených na Hardware oddělení a analýz požadavků zpracovaných Helpdesk oddělením jsme dospěli k závěru ohledně vytvoření jednoduchého nástroje, který by byl určen pro méně technicky zdatné uživatele. Hlavní funkčnost tohoto nástroje je zkrátit a usnadnit Helpdesk oddělení identifikaci daného počítače. Tento nástroj bude zobrazovat základní informace o počítači a bude přístupný jako zástupce uložený ve složce veřejné plochy, kde je zajištěný přístup všem uživatelům daného počítače. Informace bude možné sdělit Helpdesk pracovníkovi jak po telefonu, tak i za pomocí mailu. Mail bude možné odeslat přímo z nástroje s informacemi o daném počítači.

Administrátorské rozšíření tohoto nástroje bude spočívat v zobrazení detailnějších informací ohledně běžících procesů, servis a výpis systémových chyb v eventlogu. Tyto informace budou přístupné vzdáleně za předpokladu, že počítač je dostupný na síti.

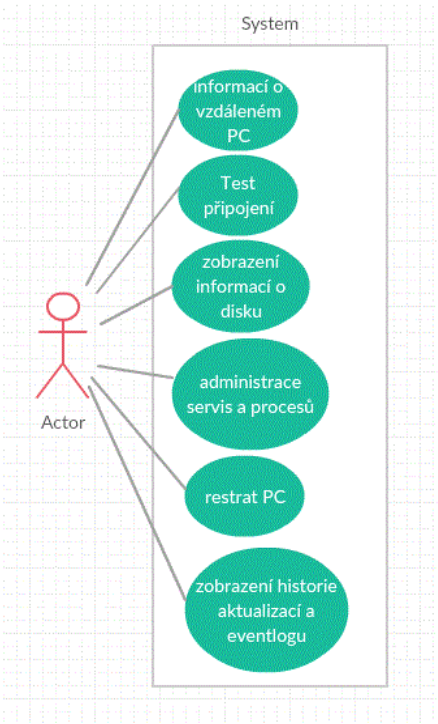
Na základě analýzy v předchozí kapitole jsem navrhl několik řešení, která by mohla ulehčit a automatizovat práci administrátorům. Navrhl jsem relativně jednoduché skripty, které jsou dostačující na dané problémy.

Obrázek 17: Use case uživatelského modulu



zdroj: vlastní

Obrázek 18: USE CASE administrátorského modulu



zdroj: vlastní

10.1.4 Požadavky na HW a SW

Pro potřeby spouštění skriptu nejsou hardwarové ani softwarové požadavky nikterak vysoké. Postačující bude standardní kancelářský počítač se standardním systémem distribuovaným IT oddělením.

Minimální hardware požadavky na počítač.

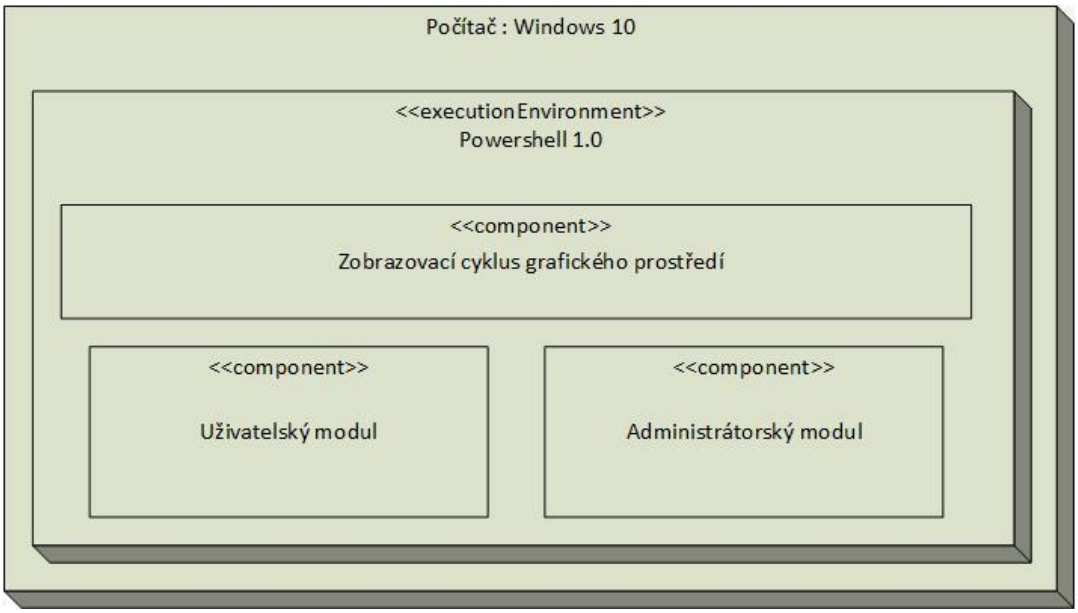
- Frekvence procesoru: 2 500 MHz
- Operační paměť RAM: 4 GB
- HDD disk minimálně: 60 GB
- Integrovaná grafická karta a vyšší

Minimální software požadavky na počítač.

- Microsoft Windows 7 (x64)
- Microsoft Office 2010+
- Internet Explorer 11

10.1.5 Návrh architektury sytému

Obrázek 19: architektura modulů



zdroj: vlastní

10.1.6 Potvrzení proveditelnosti

Metodika UP doporučuje zhotovení prototypů daného nástroje. Zhotovením tohoto prototypu, můžeme otestovat dané funkce nástroje ale, také nám dokáže posloužit k prezentování jednotlivých funkcí zákazníkovi.

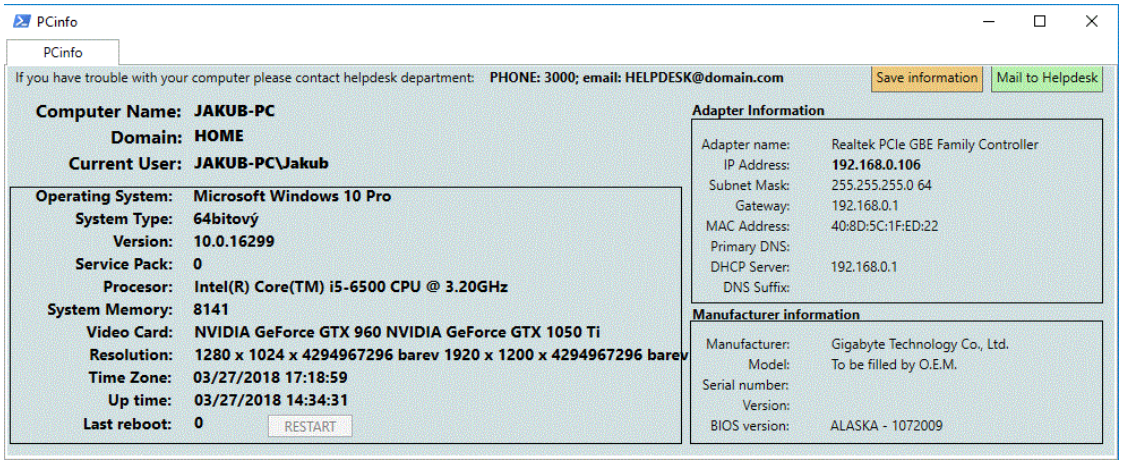
Jako prototyp k otestování a případné prezentaci zákazníkovi byl napsán skript s informačním výstupem do konzole a otestován jako celek i jeho jednotlivé funkce.

10.1.7 Uživatelské rozhraní

V této kapitole si ukážeme jak by měl grafický interface vypadat a to jak v uživatelské tak i v administrátorské verzi

Grafické prostředí uživatelského nástroje

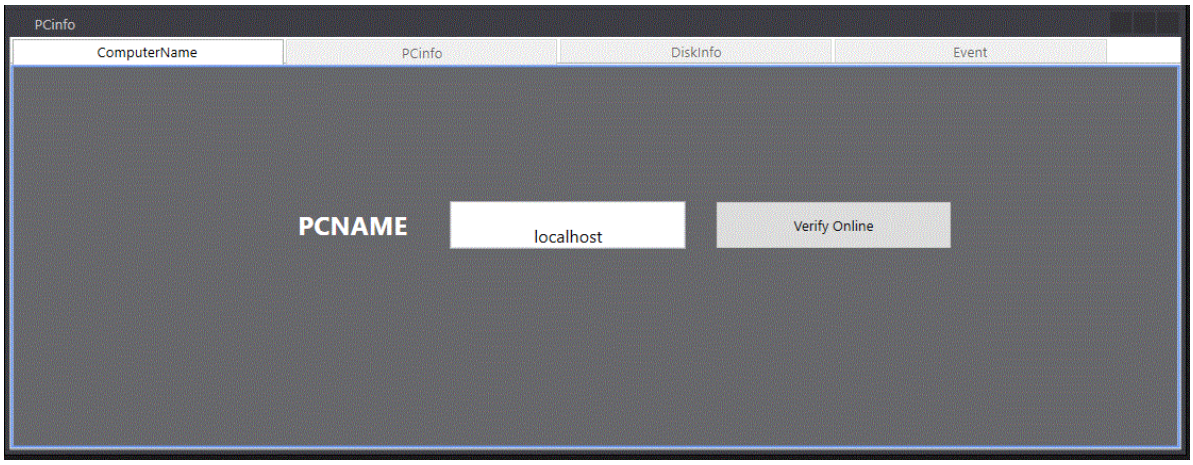
Obrázek 20: GUI uživatele



zdroj: vlastní

Grafické prostředí administrátorského nástroje

Obrázek 21: GUI administratora



zdroj: vlastní

10.1.8 Vyhodnocení fáze „Zahájení“

Z výše uvedeného je zřejmé, že milník pro přechod do další fáze byl naplněn. Nejdříve jsme analyzovali ticket systém a jeho historické požadavky, poté jsme provedli dotazování na Hardware oddělení. A z toho nám vyplynula jasná představa, co by jednotlivé nástroje měli zobrazovat. Tyto představy jsme znázornili na UML diagramech. V další části fáze jsme si potvrdili proveditelnost daného projektu a ke konci jsme si načrtli architekturu daného nástroje a ukázali jsme si předpokládanou podobu grafického rozhraní.

10.2 Fáze rozpracování

V předchozí kapitole jsme se zabývali analýzou jednotlivých částí a teď se budeme zabývat praktickému využití Windows Powershellu. Jelikož se zde jedná o vývoj ve skriptovacím jazyce ve kterém je obtížné dodržovat veškeré doporučené podmínky dané metodiky, přiřadím do této fáze i podmínky připadající do fáze konstrukce.

Milníky fáze rozpracování:

- Tvorba spustitelného architektonického základu
- Vytvoření nástroje, který je možné distribuovat na počítače uživatelů.
- Vytvoření nástroje, který je možné distribuovat na počítače administrátorů

10.2.1 Uživatelský nástroj

Uživatelský nástroj je pouhé zobrazení všech dostupných informací za pomocí WMI tříd. Tyto informace jsou zobrazovány v uživatelském rozhraní, které je vykreslováno pomocí WPF. Aby Powershell byl schopen vykreslit veškeré objekty správně je potřeba vymazat některé vlastnosti deklarací okna a také vymazat veškerá x: před každým názvem objektu. Tuto úpravu pro potřeby Powershell jsem převzal od Jima Moyla, kterého tímto uvádím ve skriptu jako autora.

```
<#
.NOTES
      Author: Jim Moyle@jimmoyle
      GitHub: https://github.com/JimMoyle/GUIdemo
#>

#Add Framework for WPF GUI
Add-Type -AssemblyName presentationframework, presentationcore
#empty hashtable
$wpf = @{}
#path where is stored XAML file
$inputXML = Get-Content -Path
"C:\Users\Jakub\Desktop\TabPanel_User\TabPanel\MainWindow.xaml"
#Clean up all sysntex form VS which PS can't understand
$inputXMLClean = $inputXML -replace 'mc:Ignorable="d"','' -replace "x:N",'N' -replace
'x:Class=".*?"','' -replace 'd:DesignHeight="\d*?"','' -replace
'd:DesignWidth="\d*?"',''
#change string variable into xml
[xml]$xaml = $inputXMLClean
#read xml data
$reader = New-Object System.Xml.XmlNodeReader $xaml
#create System.Window.object
$tempform = [Windows.Markup.XamlReader]::Load($reader)
#select each named node using an Xpath expression
$namedNodes =
$xaml.SelectNodes("//*[@*[contains(translate(name(.),'n','N'),'Name')]]")
#add all the named nodes as members to the $wpf variable, this also adds in the
correct type for the objects.
$namedNodes | ForEach-Object {$wpf.Add($_.Name, $tempform.FindName($_.Name))}
#--------------------------------------------------------------------------------
```

Skript je spouštěn poté co jsou načteny veškeré objekty a zobrazí se GUI. Poté jsou volány veškeré WMI třídy, ze kterých dále získáváme informace a zobrazujeme je za pomocí TextBlocku v GUI.

```
    $comp = Get-WmiObject Win32_ComputerSystem
    $bios = Get-WmiObject Win32_BIOS

          $wpf.PCtb.text = $comp.Manufacturer
          $wpf.PC_tb1.text = $comp.Model
          $wpf.PC_tb2.text = $comp.serialnumber
          $wpf.PC_tb3.text = $comp.Version
          $wpf.PC_tb4.text = $bios.Version
```

Mezi další funkcionalitu uživatelského rozhraní patří restart tlačítko. Toto tlačítko provede restart daného PC za předpokladu, že je splněna podmínka a PC nebylo restartováno minimálně 2 dny. Pokud je podmínka splněna a PC nebylo restartováno 2 a více dní restart tlačítko je zpřístupněno a doprovázeno hlášením, že restart PC je doporučené. Touto funkcionalitou jsem se snažil předejít situacím, kdy PC nebylo restartováno dlouhou dobu a jeví známky běžících zombie procesů.

```
#Enabled restart button, depending on condition

    if ($uptime_days -ge "1") {
    $wpf.restartBTN.IsEnabled = $true
    $wpf.TBevent1.text = "Restart recomended"
        $wpf.restartBTN.Add_Click({Restart-Computer -computername $computername})
       }
   else{$wpf.restartBTN.IsEnabled = $false}
```

Další funkcionalitou je odeslání emailu na Helpdesk oddělení, kde v příloze mailu bude přiložena příloha obsahující veškeré potřebné informace o PC. Okno mailu je otevřeno v programu Microsoft Outlook, který je firmou doporučován.

```
$wpf.mailBTN.Add_Click({

          $Outlook = New-Object -ComObject Outlook.Application
          $path = "C:\PCinfo.txt"
          $Mail = $Outlook.CreateItem(0)
          $Mail.To = "helpdesk@domain.com"
          $Mail.Subject = "Servisní požadavek"
          $Mail.Body ="Example of body..."
          $Mail.Attachments.Add($path)
          $mail.save()
          $inspector = $mail.GetInspector
          $inspector.Display()

   })
```

Tento nástroj bude distribuován ve formě zástupce, který se bude nacházet ve složce veřejné plochy, aby k němu měli přístup všichni uživatelé.

10.2.2 Administrátorský nástroj

Jako v případě u uživatelského grafického rozhraní dochází ze strany Powershell k úpravám XAML souboru a jeho následném zobrazení. Administrátorský nástroj je koncipován do čtyř záložek, kde každá záložka slouží k jinému účelu.

Na první záložce nástroje dochází k ověření, zdali vzdálený počítač je skutečně připojen k síti. V případě, že podmínka je splněna, stanou se přístupnými i další záložky nástroje a je umožněno dále s nástrojem pracovat. Pokud podmínka není splněna a počítač není na síti, jsou ostatní záložky nadále uzamčeny.

```
#Verifikace, že PC je připojeno do sítě. //rozšíření nabídky TAB
$wpf.Verify.Add_Click({if (Test-Connection $wpf.ComputerName.Text -Count 1 -Quiet){
        $wpf.PingLBL.text = "$($wpf.ComputerName.Text ) is online, unlocking"
        $wpf.tabControl.Items[1..3] | % {$_.IsEnabled = $true}
        }
        else{
        $wpf.PingLBL.text = "$($wpf.ComputerName.Text ) is offline"
        $wpf.tabControl.Items[1..3] | % {$_.IsEnabled = $false}
        }
})
```

Na druhé kartě záložek je implementována stejná funkcionalita jako v případě uživatelského rozhraní s tím rozdílem, že se jedná o vzdálený přístup.

Záložka Disk INFO zobrazuje v seznamu veškeré informace o discích a jejich zaplnění.

```
Function Get-DiskInfo {
param($computername = $env:COMPUTERNAME)

    Get-WMIObject Win32_logicaldisk -ComputerName $wpf.ComputerName.Text
| Select-Object @{Name='ComputerName';Ex={$computername}},
@{Name='Drive Letter';Expression={$_.DeviceID}},`
@{Name='Drive Label';Expression={$_.VolumeName}},`
@{Name='Size(MB)';Expression={[int]($_.Size / 1MB)}},`
@{Name='FreeSpace%';Expression={[math]::Round($_.FreeSpace / $_.Size,2)*100}}
}


Get-DiskInfo -computername $wpf.ComputerName.Text | %
{$wpf.disk_listView.AddChild($_)}

$wpf.DiskLabel.Content = "Disk info for system $($wpf.ComputerName.Text)"
```

Dále pak zobrazuje, jaký proces využívá větší množství systémových prostředku, toto slouží k lepší identifikaci procesu a jeho případném ukončení.

```
$TotalRAM = [Math]::Truncate((get-WMIObject win32_operatingsystem -computername
$wpf.ComputerName.Text | Measure-Object TotalVisibleMemorySize -sum).sum / 1024)
$FreeRAM = ((get-WMIObject -computername $wpf.ComputerName.Text -class
win32_operatingsystem).freephysicalmemory) / 1024
$UsedRAM = (($TotalRAM) - ($FreeRAM))
$RAMPercentUsed = ([math]::truncate(($UsedRAM) / ($TotalRAM) * 100))
```

```
$memhog = (get-process -computername $wpf.ComputerName.Text | sort-object WS -
descending | select-object -first 1)
$memhogname = ($memhog).processname
$memhogram = ([math]::truncate(($memhog).WS / 1MB))
$memhogid = ($memhog).Id

$CPUpercent = (Get-WmiObject -computername $wpf.ComputerName.Text win32_processor |
Measure-Object -property LoadPercentage -Average).Average
$cpuhog = (get-process -computername $wpf.ComputerName.Text | sort-object CPU -
descending | select-object -first 1)
$cpuhogname = ($cpuhog).processname

    $wpf.ramTB.text = $TotalRAM
    $wpf.freeTB.text = $FreeRAM
    $wpf.usedTB.text = $UsedRAM

    $wpf.processNameTB.text = $memhogname
    $wpf.processRAM.text = $memhogram

    $wpf.cpuTB.text = $cpuhogname
    $wpf.cpuPercentTB.  text = $CPUpercent
```

Dále pak jsou v tabulkách zobrazeny veškeré běžící servisy a procesy na daném počítači. Funkcionalita ukončení a opětovné spuštění procesu je také implementována, stejně tak i v případě servis.

```
$WPF.service_listView.Items.Clear()
get-service -ComputerName $WPF.ComputerName.Text |
Sort-Object Status | % {$WPF.service_listView.AddChild($_)}
#Start Service BTN
$wpf.StartBTN.Add_Click({
    $wpf.service_listView.SelectedItems | % {
        ($stat = GET-WMIObject Win32_Service -ComputerName $wpf.ComputerName.Text |
                ? Name -like $_.Name).StartService()
            if ($stat -eq 0){}
            elseif ($stat -eq 2){}
            else{}
        }
    $wpf.service_listView.Items.Clear()
    get-service -ComputerName $wpf.ComputerName.Text |
        % {$wpf.service_listView.AddChild($_)}
})

#Stop service BTN
$wpf.stopBTN.Add_Click({
    $wpf.service_listView.SelectedItems | % {
        ($stat = GET-WMIObject Win32_Service -ComputerName $WPF.ComputerName.Text |
                ? Name -like $_.Name).StopService()
            if ($stat -eq 0){}
            elseif ($stat -eq 2){
            }
            else{}
        }

    $WPF.service_listView.Items.Clear()
    get-service -ComputerName $WPF.ComputerName.Text |
           % {$WPF.service_listView.AddChild($_)}
})

#Show all running proceses

$wpf.process_listView.Items.Clear()
get-process -ComputerName $wpf.computerName.text | sort-Object Caption |
        % {$WPF.process_listView.AddChild($_)}

#stop process button
$wpf.ProcessStopBTN.Add_Click({
    $wpf.process_listView.SelectedItems | % {
        ($stat = GET-WMIObject win32_process -ComputerName $wpf.ComputerName.Text |
                ? Name -like $_.ProcessName).StopProcess()
            if ($stat -eq 0){}
            elseif ($stat -eq 2){}
```

```
            else{}
        }
    $wpf.process_listView.Items.Clear()
    Get-process -ComputerName $wpf.ComputerName.Text |
        sort-Object Caption | % {$wpf.process_listView.AddChild($_)}
})
})
```

Jako poslední záložka Administrátorského nástroje jsou veškeré události na daném PC. V první tabulce jsou uvedeny nejnovější systémové chyby. V další tabulce se můžeme dozvědět o nainstalovaných aktualizacích. Tyto informace jsou velice podstatné při řešení servisních požadavků.

10.2.1 Vyhodnocení fáze „Rozpracování“

Cílem fáze rozpracování a konstrukce bylo sestavení robustního architektonického základu a vytvoření dvou provozuschopných nástrojů. Na architektonickém základu byly postaveny dva nástroje. Jeden z nich byl určen pro uživatele jako zobrazení veškerých dostupných informací o daném PC. Funkcionality tohoto nástroje jsou volány lokálně a zobrazují veškeré možné informace o počítači, které budou nápomocny při řešení servisního požadavku. Jako další byl vytvořen administrátorský nástroj, který zobrazuje jak základní tak i rozšířené informace o vzdáleném PC. Fázi rozpracován jsme spojili dohromady s fází konstrukce a to z důvodu, že vytváříme nástroje za pomocí skriptovacího jazyka a metodika je spíše brána na vývoj softwaru pro vyšší programovací jazyky a větší vývojové týmy. Z tohoto důvodu jsme ve fázi rozpracování zobrazili a popsali jednotlivé funkcionality daných nástrojů.

10.3 Fáze nasazení

Dle metodiky UP milníkem této fáze je předání produktu zákazníkovi s tím, že jsou veškeré požadavky splněny na produkt kladené. Produkt je předán zákazníkovi a spouštěn v reálném provozu.

V současné době je dedikována malá skupina lidí, která má za úkol otestovat funkcionalitu a použitelnost daného produktu. Skupina má za úkol otestovat veškerou funkcionalitu produktu a z pohledu Helpdesk oddělení také sledovat četnost výskytu daných emailů, nebo nasměrování těchto volajících uživatelů na tento nástroj. Posléze bude produkt nasazen celoplošně v rámci společnosti. Prvotní testování je za námi, ale bylo vyžádáno managementem další samostatné testováním které bude trvat 3-4 měsíce, což je dostatečný čas na opravu případných připomínek i rozšíření funkcionality. Z tohoto důvodu nemůže být dokončena fáze zavedení do data odevzdání této diplomové práce.

10.3.1 Budoucí rozšíření

V uživatelském nástroji bych rád implementoval funkcionalitu ovládání virtualizovaných aplikací přes Active Directory, kdy by si uživatel sám mohl navázat aplikaci na svůj login, tím nainstalovat aplikaci a vypublikovat ikonu na plochu.

Podobnou funkcionalitu bych implementoval i do administrátorského nástroje, kde by byla vidět celková struktura Active Directory, dále pak zobrazení veškerých skupin ve kterých je daný uživatel veden. Jako další funkcionalitu bych rád navázal nástroj na získávání informací z DHCP serveru. Tímto by se získávaly informace ohledně Firewallových pravidel aplikovaných na PC.

11 Zhodnocení výsledků

Dané nástroje slouží k automatizaci předem specifikovaných požadavků. Jedná se o poměrně jednoduché nástroje, u kterých jejich funkcionalita není potřeba nahrazovat nákladným již vyvinutým softwarem od různých společností. Mezi benefity interního vývoje nástroje patří jeho modularita a přizpůsobení procesním, provozním podmínkám dané společnosti.

Administrátorský nástroj

Nástroj umožňuje administrátorovi získání přehledu o hardwarové konfiguraci a stavu koncové stanice. Tyto informace přístupné přes grafické prostředí nástroje by měli urychlovat administrátorovu analýzu ohledně závady na zařízení, zjednodušovat její identifikaci a případně zrychlit vyřízení servisního požadavku. Spouštění a zastavování procesu je nedílnou součástí nástroje z důvodu, kdy může dojít, že určitý proces naplno vytěžuje systémové prostředky a tím dochází ke zpoždění požadavků uživatele. Takovéto chování má za následek pomalý chod zařízení a může i způsobovat pády systému samotného.

Uživatelský nástroj

Uživatelský nástroj umožňuje získat veškeré informace o lokálním počítači v jednom systémovém okně. Tyto informace jsou převážně určeny méně technicky zdatným uživatelům využívající počítač k pracovním povinnostem. Pro tyto uživatele je daleko jednodušší kliknout na ikonu, která se nachází na ploše počítače, než složitě hledat název počítače jinými způsoby. Tato funkcionalita také napomáhá administrátorům z Helpdesk oddělení v rychlejší a pohodlnější navigaci samotného uživatele v případě řešení servisního požadavku.

Nástroje byly sestrojeny k další modifikaci a možnostem rozšíření.

12 Závěr

Předkládaná diplomová práce „Administrace pracovních stanic pomocí Powershell“ přináší na stanovené cíle tyto závěry.

Volbou tohoto tématu si chtěl autor vyzkoušet na skutečném projektu jak vývoj ve skriptovacím jazyce Powershell probíhá a jaké jsou jeho funkční výhody oproti jiným nástrojům. Mezi další cíle autorovi volby daného tématu patří snaha o zlepšení stávající situace vzdálené administrace a identifikace pracovních stanic na IT hardware, helpdesk oddělení.

V teoretické části jsme se seznámili s jednotlivými funkcionalitami skriptovacího jazyka Powershell a zároveň jsme si uvedli syntaxe jednotlivých komponent. Vývoj skriptovacího jazyka Powershell jde velmi rychle a v současné době je již verze Powershell 6.0 to znamená, že funkcionalita Powershellu každou vydanou verzí nadále narůstá. To má za následek, že v současné době se již nevyvíjí skripty za pomocí VBS či jiných nástrojů a plnohodnotně tyto starší nástroje vytlačuje již Powershell. Tomuto trendu také přispívá to, že Powershell je ze strany Microsoftu plně podporován. Naopak starší nástroje upadají do zapomnění a budou postupně vytlačeny. Jedná se o rychlý a efektivní nástroj, díky jeho provázanosti celého prostředí. Tímto provázáním může Powershell ve spojení s XAML a Windows Presentation Form dokázat vytvářet i softwary většího charakteru.

V praktické části diplomové práce autor představil firemní prostředí společnosti, kdy se zabýval jeho síťovým prostředím a zabezpečením. Uvedl, že uživatele ve firemním prostředí nemají v rámci bezpečnosti žádná oprávněním a z tohoto důvodu pro každý servisní požadavek musí volat helpdesk. Na helpdesk jsou kladeny minimální požadavky při zakládání servisního požadavku a tím je, že v požadavku musí být typ poruchy a název počítače. Sdělení názvu počítače od uživatele pro vzdálenou správu je zásadní problém. Ve společnosti existují různé druhy uživatelů a jejich počítačové znalosti se také liší. Pro usnadnění práce s navigováním uživatelů pro helpdesk oddělení byl vytvořen návrh nástroje PCinfo, který bude vypublikován ve veřejné složce na každém PC. Tento nástroj má za cíl, zobrazit veškeré informace o PC ve spolupráci s uživatelem a tím urychlení správné identifikace daného PC. Jedná se o jakýsi komfort uživatele a pracovníka helpdesku.

Z dalšího šetření, které probíhalo na IT hardware oddělení ve formě metody brainstorming, vyšlo najevo, že i po zjištění názvu počítače je potřeba zajistit i další informace o daném PC. Byly

stanoveny podmínky na nástroj, které byly porovnány se skutečnosti z ticket systému a zapracovány do návrhu. Detailnějšími informacemi administrátor získá větší povědomí o tom co se na daném počítači za několik dní stalo a urychlí to jeho případné řešení servisního požadavku. Hlavními funkcionalitami bylo vzdálené spouštění a ovládání servis, procesů daného počítače. Zobrazení eventlogu a nejaktuálnějších nainstalovaných aktualizací má být nedílnou součástí.

V další praktické části jsme analyzovali a porovnávali možnosti zakoupení podobného již hotového softwaru. Z tohoto šetření nám vyšlo najevo, že by se buď jednalo o velmi drahé řešení, nebo řešení komplikované. S ohledem na to, že vývoj takového nástroje není finančně podporován rozhodli jsme se jít cestou vlastního vývoje.

Výběr správné metodiky byl pro mne velmi obtížný, jelikož se jedná o vývoj ve skriptovacím jazyce a veškeré mnou uváděné metodiky jsou určeny spíše pro vícečlenné týmy a větší projekty. Vývoj jsem prováděl sám a o veliký projekt se také nejedná, z toho důvodu jsem si metodiku vývoje poupravil tak, aby vyhovovali mým potřebám. Na druhou stranu s metodikou Unified Process jsem velice spokojen. Na konci každé fáze jsme zkontrolovali milníky, co vše se zvládlo v rámci fáze dokončit. Z toho vyplívá, že zvolená metodika splnila má očekávání a dovedla celý vývoj ke zdárnému konci.

Na základě návrhů a za pomocí dané metodiky, výsledný produkt se skládá ze dvou modulů. První z modulů je určen pro konečného uživatele se znalostmi počítače i bez nich a je určen pro rychlou a snadnou identifikaci počítače uživatelem. Do nástroje dále byla zakomponována funkcionalita odeslání emailu rovnou z aplikace na helpdesk, kde v příloze je již obsažený soubor se všemi potřebnými informacemi. Uživatel může tuto funkcionalitu využít, když potřebuje přesně popsat helpdesku svůj problém. Do odesílaného mailu je možné připojit i další přílohy, jedná se o mailového klienta Microsoft Outlook, který ve firemním prostředí je nainstalován na každém počítači. Toto opět může usnadnit helpdesku práci v tom, že není potřeba z uživatele dostávat potřebné informace, ale sám uživatel je na helpdesk zašle ve formě .CSV souboru doprovázeným stručným popisem od uživatele. Druhým výstupem práce je administrátorský nástroj, který vychází z konceptu PCinfo, ale má rozšířenější funkcionalitu a počítač vyžadující servisní zásah lze administrovat vzdáleně. Mezi detailnější informace patří informace o discích, napomůže při zjištění plného disku a systém například nemůže ukládat do virtualizované paměti na disk, nebo díky zaplněnému disku může být znemožněna správná funkčnost programového vybavení počítače. Mezi další funkcionality patří spouštění a zastavování servis a procesu počítače, kde běžící zombie proces dokáže celkové zamrznutí

systému i jeho pád. V neposlední řadě je funkcionalita výpis eventlogu a nejnovějších aktualizací, kdy i nejnovější aktualizace mohou způsobit neshodu s nainstalovaným softwarem a způsobit tak jeho pády.

Ačkoliv se oba nástroje stále nacházejí ve stavu testování a zpracování případných výtek, Můžeme prohlásit, že otestování modulů na několika počítačích dokázalo, že výše zmiňovaná tvrzení jsou uskutečněna. Budoucnost daných modulů závisí na tom, zdali se s nimi uživatelé a administrátoři sžijí. Pokud ano, můžeme pokračovat v dalším nasazování do firemního prostředí a také v jeho vývoji.

V závěru práce lze říci, že vytyčené cíle práce byly naplněny. Práce blíže seznamuje s problematikou „*Administrace pracovních stanic pomocí Powershel*“. Uvádí vývoj Powershellu v čase jeho hlavní předností a funkce. Dále pak práce zpracovává analýzu a v neposlední řadě uvádí příklady možného interního řešení zobrazování informací za pomocí skriptů a GUI.

Seznam použitých zdrojů

1. Bruce Payette, *Powershell in action*, Manning publication co., 2007, 532 s., ISBN: 1932394-90-7
2. Don Jones, *Learn windows Powershell in a month of Lunches*, Manning publication co., 2011, 299 s., ISBN: 9781617290213
3. Don Jones, *Learn windows Powershell 3 in a month of Lunches*, Manning publication co., 2013, 333 s., ISBN: 9781617291081
4. Don Jones, Jeffery Hicks, *Learn Powershell Toolmaking in Month of Lunches*, Manning publication co., 2013, 281 s., ISBN: 9781617291166
5. Ed Wilson, *Windows Powershell scripting guide*, Microsoft press, 2008, 643 s., ISBN: 2007941089
6. Lee Holmes. *Windows PowerShell Pocket Reference.* Sebastopol : O'Reilly Media, 2013. ISBN: 978-1-449-32096-6.
7. Ed Wilson, *Windows Powershell step by step*, Microsoft press, 2007, 289 s., ISBN: 2007924649
8. Lee Holmes, *Windows Powershell cookbook*, O'Realy, 2010, 821 s., ISBN: 978-0-596-80150-2
9. Patrik Malina, *Powershell podrobný průvodce scriptování*, Computer press, 2007, 335 s. , ISBN: 978-80-251-1816-0
10. Richard Siddaway, *Powershell and WMI*, Meaning publication co., 2012, 505 s., ISBN: 9781617290114
11. LEE, Thomas L. *Windows powershell 2.0 bible*. 1st ed. Indianapolis, IN: Wiley Pubishing, Inc., 2011, p. cm. ISBN 11-180-2198-3.
12. WILSON, Ed. *Windows PowerShell 2.0 best practices*. Redmond, Wash.: Microsoft Press, c2010, xxv, 715 p. ISBN 07-356-2646-4.
13. Wilson, Ed**.** *Windows PowerShell 3.0 Step by Step.* Sebastopol : O'Reilly Media, Inc., 2013. ISBN: 978-0-735-66339-8.
14. *Overview of the .NET Framework*. Msdn.Microsoft.com [online] [Citováno: 26. 3. 2017] https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx
15. Ed Wilson, *Windows Powershell průvodce skriptováním*, Computer press, 2015, 255 s., ISBN: 9788025143865
16. *Powershell vs. VBScript*. Msdn.Microsoft.com [online] [Citováno: 26. 3. 2017] http://www.rlmueller.net/PSVBScript.htm

Seznam obrázků a tabulek

Obrázky

Obrázek 1- logo Powershell 7

Obrázek 2 - Get-Process výstup 7

Obrázek 3 - Příklad If 9

Obrázek 4- Příklad ForEach 10

Obrázek 5 - Příklad objektové roury 12

Obrázek 6- Příklad bezpečnostní politiky 14

Obrázek 7 - Powershell ISE 16

Obrázek 8 - WMI 17

Obrázek 9: Visual Studio 18

Obrázek 10: Porovnavací tabulka VS 19

Obrázek 11: Příklad XAML 20

Obrázek 12 - logo CMD 20

Obrázek 13 - logo VBS 21

Obrázek 14 - Příklad VBScript 24

Obrázek 15: Vodopádový model 38

Obrázek 16: Spirálový model 39

Obrázek 17: Use case uživatelského modulu 47

Obrázek 18: USE CASE administrátorského modulu 48

Obrázek 19: architektura modulů 49

Obrázek 20: GUI uživatele 50

Obrázek 21: GUI administratora 51

Tabulky

Tabulka 1- Datové typy 12

Tabulka 2 - Bezpečnostní politiky 13

Tabulka 3: Konfigurace základního notebooku 27

Tabulka 4: Konfigurace základního desktopu 27

Tabulka 5 - Funkcionality HP OpenView 34

Seznam zkratek:

IT	Informační technologie
WMI	Windows Management Instrumetation
OS	Operační systém
AD	Active Directory (Služba pro správu uživatelů v serverových Windows)
GP	Group Policy (Zásady skupiny v Active directory)
IE	Internet Explorer (Internetový prohlížeč od společnosti Microsoft)
SQL	Structured Query Language (Strukturovaný dotazovací jazyk)
RDP	Remote Desktop Protokol (Protokol pro vzdálené připojení na PS)
U. I.	User Interface (uživatelské prostředí)
CMD	Command Prompt (Příkazový řádek)
WMI	Windows Management Instrumentation (Nástroj pro obsluhu Windows pomocí příkazu)
LDAP	Lightweight Directory Access Protocol (Protokol pro správu Active Directory)

Přílohy

XAML pro uživatele

```
<Window x:Name="PCinfo" x:Class="TabPanel.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:TabPanel"
      mc:Ignorable="d"
            Title="PCinfo" Height="410" Width="1000">
  <Grid>
    <TabControl x:Name="tabControl">
      <TabItem Header="PCinfo" Width="100" IsEnabled="True">
        <TabItem.Background>
          <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
            <GradientStop Color="#FFF0F0F0" Offset="0"/>
            <GradientStop Color="#FF198995" Offset="1"/>
          </LinearGradientBrush>
        </TabItem.Background>
        <Grid Background="#FFD2E1E2" Margin="0,-1,0,1" HorizontalAlignment="Right" Width="986" Height="351" VerticalAlignment="Bottom">
          <Label Content="Domain:" HorizontalAlignment="Left" Margin="17,54,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="16" Padding="5,5,5,0" UseLayoutRounding="True"/>
          <Label Content="Current User:" HorizontalAlignment="Left" Margin="17,77,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="16" Padding="5,5,5,0" UseLayoutRounding="True"/>
          <Label Content="System Type:" HorizontalAlignment="Left" Margin="9,127,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="Version:" HorizontalAlignment="Left" Margin="9,147,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14" Padding="5,5,5,0"/>
          <Label Content="Service Pack:" HorizontalAlignment="Left" Margin="9,167,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="Procesor:" HorizontalAlignment="Left" Margin="9,187,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="System Memory:" HorizontalAlignment="Left" Margin="9,207,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="Video Card:" HorizontalAlignment="Left" Margin="9,227,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="Resolution:" HorizontalAlignment="Left" Margin="9,247,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="Time Zone:" HorizontalAlignment="Left" Margin="8,267,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <Label Content="Last reboot:" HorizontalAlignment="Left" Margin="8,308,0,13" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14" Width="150" Height="30"/>
          <Label Content="Up time:" HorizontalAlignment="Left" Margin="8,287,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
          <TextBlock x:Name="TB1" Margin="173,37,242,290" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" LineHeight="16" Text="aaa" Height="24" FontFamily="Segoe UI Black"/>
          <TextBlock x:Name="TB2" Margin="173,59,269,268" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="24" FontFamily="Segoe UI Black"/>
          <TextBlock x:Name="TB3" Margin="173,83,269,244" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="24" FontFamily="Segoe UI Black"/>
          <Label Content="Operating System:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14" Margin="9,107,0,0"/>
          <TextBlock x:Name="TB4" Margin="172,112,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Height="20" Padding="0" LineHeight="9" Text="aaa" HorizontalAlignment="Left" VerticalAlignment="Top"/>
          <TextBlock x:Name="TB5" Margin="172,132,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0" HorizontalAlignment="Left" VerticalAlignment="Top"/>
          <TextBlock x:Name="TB6" Margin="172,152,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Padding="0" HorizontalAlignment="Left" VerticalAlignment="Top"/>
          <TextBlock x:Name="TB7" Margin="172,170,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
          <TextBlock x:Name="TB8" Margin="173,190,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
```

```
        <TextBlock x:Name="TB9" Margin="172,210,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        <TextBlock x:Name="TB10" Margin="173,230,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        <TextBlock x:Name="TB11" Margin="173,250,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        <TextBlock x:Name="TB12" Margin="172,270,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        <TextBlock x:Name="TB13" Margin="173,290,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        <TextBlock x:Name="TB14" Margin="173,310,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        <TextBlock x:Name="TBevent" Margin="593,38,210,561" TextWrapping="Wrap" Grid.RowSpan="2"/>
        <Label Content="IP Address:" HorizontalAlignment="Right" Margin="0,81,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Subnet Mask:" HorizontalAlignment="Right" Margin="0,99,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Gateway:" HorizontalAlignment="Right" Margin="0,117,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="MAC Address:" HorizontalAlignment="Right" Margin="0,135,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Primary DNS:" Margin="226,153,282,0" VerticalAlignment="Top" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Adapter name:" HorizontalAlignment="Right" Margin="0,63,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="DHCP Server:" HorizontalAlignment="Right" Margin="0,171,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="DNS Suffix:" HorizontalAlignment="Right" Margin="0,189,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
        <TextBlock x:Name="aTB1" HorizontalAlignment="Left" Margin="736,86,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" FontWeight="Bold" Text="aaa"/>
        <TextBlock x:Name="aTB2" HorizontalAlignment="Left" Margin="736,104,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
        <TextBlock x:Name="aTB3" HorizontalAlignment="Left" Margin="736,122,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
        <TextBlock x:Name="aTB4" HorizontalAlignment="Left" Margin="736,140,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
        <TextBlock x:Name="aTB5" HorizontalAlignment="Left" Margin="735,158,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="139" Text="aaa"/>
        <TextBlock x:Name="aTB6" HorizontalAlignment="Left" Margin="736,68,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="200" Text="aaa"/>
        <TextBlock x:Name="aTB7" HorizontalAlignment="Left" Margin="735,176,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
        <TextBlock x:Name="aTB8" HorizontalAlignment="Left" Margin="735,194,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
        <Label Content="Manufacturer:" HorizontalAlignment="Left" Margin="613,239,0,0" VerticalAlignment="Top" Width="91" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Model:" HorizontalAlignment="Left" Margin="613,257,0,0" VerticalAlignment="Top" Width="91" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Serial number:" HorizontalAlignment="Left" Margin="611,275,0,0" VerticalAlignment="Top" Width="92" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="Version:" HorizontalAlignment="Left" Margin="612,293,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.421,0.564" Width="92" HorizontalContentAlignment="Right" Height="28"/>
        <Label Content="BIOS version:" HorizontalAlignment="Left" Margin="624,311,0,0" VerticalAlignment="Top" Width="100"/>
        <TextBlock x:Name="PCtb" HorizontalAlignment="Left" Margin="736,244,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="200"/>
        <TextBlock x:Name="PC_tb1" HorizontalAlignment="Left" Margin="736,262,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="200"/>
        <TextBlock x:Name="PC_tb2" HorizontalAlignment="Left" Margin="736,280,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" RenderTransformOrigin="0.5,0.5" Width="200"/>
        <TextBlock x:Name="PC_tb3" HorizontalAlignment="Left" Margin="736,298,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="200"/>
        <TextBlock x:Name="PC_tb4" Margin="735,316,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" HorizontalAlignment="Left" Width="200"/>
        <Label Content="Computer Name:" HorizontalAlignment="Left" Margin="17,31,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="16" Padding="5,5,5,0" UseLayoutRounding="True"/>
        <Button x:Name="restartBTN" Content="RESTART" HorizontalAlignment="Left" Margin="238,315,0,0" VerticalAlignment="Top" Width="75" IsEnabled="false" Background="#FFF3931F"/>
        <TextBlock x:Name="TBevent1" HorizontalAlignment="Left" Margin="318,315,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="20" Width="127"/>
        <Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left" Height="163" Margin="612,54,0,0" VerticalAlignment="Top" Width="364"/>
        <Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left" Height="109" Margin="611,232,0,0" VerticalAlignment="Top" Width="365"/>
        <Label Content="If you have trouble with your computer please contact helpdesk department: " HorizontalAlignment="Left" Margin="10,4,0,0" VerticalAlignment="Top" Width="419"/>
        <Label Content="PHONE: 3000; email: HELPDESK@domain.com" HorizontalAlignment="Left" Margin="429,4,0,0" VerticalAlignment="Top" Width="337" FontWeight="Bold"/>
        <Button x:Name="mailBTN" Content="Mail to Helpdesk" HorizontalAlignment="Left" Margin="877,4,0,0" VerticalAlignment="Top" Width="99" Height="27" Background="#FFB8F0AF"/>
        <Label Content="Adapter Information" HorizontalAlignment="Left" Margin="608,33,0,0" VerticalAlignment="Top" FontWeight="Bold"/>
        <Label Content="Manufacturer information" HorizontalAlignment="Left" Margin="608,213,0,0" VerticalAlignment="Top" FontWeight="Bold"/>
        <Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left" Height="227" Margin="10,114,0,0" VerticalAlignment="Top" Width="594"/>
        <Button x:Name="saveBTN" Content="Save information" HorizontalAlignment="Left" Margin="771,4,0,0" VerticalAlignment="Top" Width="99" Height="27" Background="#FFEECA7E"/>
    </Grid>
```

```
        </TabItem>
    </TabControl>
  </Grid>
</Window>
```

Uživatelský skript

```
Add-Type -AssemblyName presentationframework, presentationcore
$wpf = @{}
$inputXML = Get-Content -Path "F:\DIPLOMOVÁ PRÁCE (kopie)\TOOL-
GUI_27.3\TabPanel_User\TabPanel\MainWindow.xaml"
$inputXMLClean = $inputXML -replace 'mc:Ignorable="d"','' -replace "x:N",'N' -replace
'x:Class=".*?"','' -replace 'd:DesignHeight="\d*?"','' -replace
'd:DesignWidth="\d*?"',''
[xml]$xaml = $inputXMLClean
$reader = New-Object System.Xml.XmlNodeReader $xaml
$tempform = [Windows.Markup.XamlReader]::Load($reader)
$namedNodes =
$xaml.SelectNodes("//*[@*[contains(translate(name(.),'n','N'),'Name')]]")
$namedNodes | ForEach-Object {$wpf.Add($_.Name, $tempform.FindName($_.Name))}
#---------------------------------------------------------------------------
$Wpf.PCinfo.Add_Loaded({

})

$computername = $env:COMPUTERNAME

# PC information

        $User = Get-WmiObject win32_computersystem -ComputerName $computername -
Credential $Credential
        $os = Get-WmiObject Win32_OperatingSystem -computerName $computername -
Credential $Credential


#Lastboot/time zone/ uptime days

        $boottime = $OS.converttodatetime($OS.LastBootUpTime)
        $uptime = New-TimeSpan (get-date $boottime)
        $uptime_days = [int]$uptime.days


# Ram information

       $totalRAM = [Math]::Truncate((get-WMIObject win32_operatingsystem | Measure-
Object TotalVisibleMemorySize -sum).sum / 1024)
       $cpu = Get-WmiObject Win32_Processor
       $computerVideo = Get-WmiObject Win32_VideoController


            $wpf.TB1.text = $user.name
            $wpf.TB2.text = $User.domain
            $wpf.TB3.text = $User.userName

            $wpf.TB4.text = $os.caption
            $wpf.TB5.text = $os.OSArchitecture
            $wpf.TB6.text = $os.version
            $wpf.TB7.text = $os.ServicePackMajorVersion
            $wpf.TB8.text = $cpu.Name
            $wpf.TB9.text = $totalRAM
            $wpf.TB10.text = $computerVideo.description
            $wpf.TB11.text = $computervideo.VideoModeDescription
            $wpf.TB12.text = (get-date)
            $wpf.TB13.text = $boottime
            $wpf.TB14.text = $uptime_days

#Adapter INFo

    $adapter = get-wmiobject win32_networkadapter -filter "NetEnabled='True'"
    $IP = Get-WmiObject Win32_NetworkAdapterConfiguration -filter 'IPEnabled="True"'
| Select -expand IPAddress | ?{$_ -notmatch ':'}
```

```
    $network = Get-WmiObject Win32_NetworkAdapterConfiguration -filter
'IPEnabled="True"'

            $wpf.aTB1.text = $IP
            $wpf.aTB2.text = $network.IPSubnet
            $wpf.aTB3.text = $network.DefaultIPGateway
            $wpf.aTB4.text = $adapter.MACAddress
            $wpf.aTB5.text = $IP.DNSHostName
            $wpf.aTB6.text = $adapter.name
            $wpf.aTB7.text = $network.DHCPServer
            $wpf.aTB8.text = $network.DNSDomain


# Computer info

    $comp = Get-WmiObject Win32_ComputerSystem
    $bios = Get-WmiObject Win32_BIOS

         $wpf.PCtb.text = $comp.Manufacturer
         $wpf.PC_tb1.text = $comp.Model
         $wpf.PC_tb2.text = $comp.serialnumber
         $wpf.PC_tb3.text = $comp.Version
         $wpf.PC_tb4.text = $bios.Version


# Save PC information to the folder
  $wpf.saveBTN.Add_Click({
  $props = @{ 'ComputerName' =  $user.name;
                    'Domain'=$User.domain;
                    'Current User'=$User.userName;
                    'Operating system'=$os.caption
                    'System Type'=$os.OSArchitecture;
                    'Version' =$os.version;
                    'Service Pack'=$os.ServicePackMajorVersion;
                    'Procesor'=$cpu.Name;
                    'System memory'=$totalRAM;
                    'Video card' =$computerVideo.description;
                    'Resolution' =$computervideo.VideoModeDescription;
                   #'Time zone' =$MonitorInfo.Model
                    'Last reboot' =$uptime_days;
       }
            $obj = New-Object -TypeName PSObject -Property $props
            $obj | Out-File C:\PCinfo.txt
 })

#Enabled restart button, depending on condition

    if ($uptime_days -ge "1") {
    $wpf.restartBTN.IsEnabled = $true
    $wpf.TBevent1.text = "Restart recomended"
        $wpf.restartBTN.Add_Click({Restart-Computer -computername $computername})
       }
   else{$wpf.restartBTN.IsEnabled = $false}


# Send email to the helpdesk with PC information in attachment

   $wpf.mailBTN.Add_Click({

         $Outlook = New-Object -ComObject Outlook.Application
         $path = "C:\PCinfo.txt"
         $Mail = $Outlook.CreateItem(0)
         $Mail.To = "helpdesk@domain.com"
         $Mail.Subject = "Servisní požadavek"
         $Mail.Body ="Example of body..."
         $Mail.Attachments.Add($path)
         $mail.save()
         $inspector = $mail.GetInspector
         $inspector.Display()

   })


$wpf.PCinfo.ShowDialog() | Out-Null
```

XAML pro administratora

```
<Window x:Name="PCinfo" x:Class="TabPanel.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:TabPanel"
      mc:Ignorable="d"
            Title="PCinfo" Height="380" Width="1000">
   <Grid>
      <TabControl x:Name="tabControl">
         <TabItem Header="ComputerName" Margin="0" Width="230">
            <TabItem.Background>
               <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
                  <GradientStop Color="#FFF0F0F0" Offset="0"/>
                  <GradientStop Color="#FF68696C" Offset="1"/>
               </LinearGradientBrush>
            </TabItem.Background>
            <Grid Background="#FF68696C">
               <Grid.ColumnDefinitions>
                  <ColumnDefinition Width="149*"/>
                  <ColumnDefinition Width="837*"/>
               </Grid.ColumnDefinitions>
               <TextBox x:Name="ComputerName" TextWrapping="Wrap" Height="40" Margin="221,114,416,0" VerticalAlignment="Top" Width="200" FontSize="14.667" HorizontalAlignment="Left" HorizontalContentAlignment="Center" VerticalContentAlignment="Bottom" Text="localhost" Grid.Column="1"/>
               <Button x:Name="Verify" Content="Verify Online" HorizontalAlignment="Left" Margin="446,114,0,0" VerticalAlignment="Top" Width="200" Height="40" Grid.Column="1" HorizontalContentAlignment="Center"/>
               <Label Content="PCNAME" Margin="62,114,0,0" VerticalAlignment="Top" Width="154" Height="40" FontSize="22" FontWeight="Bold" HorizontalAlignment="Left" Foreground="White" Grid.Column="1" HorizontalContentAlignment="Center"/>
               <TextBlock x:Name="PingLBL" HorizontalAlignment="Left" Margin="221,159,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="425" Height="26" Foreground="White" Grid.Column="1"/>
            </Grid>
         </TabItem>
         <TabItem Header="PCinfo" IsEnabled="False" Margin="2,0,-2,-2" Width="230">
            <TabItem.Background>
               <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
                  <GradientStop Color="#FFF0F0F0" Offset="0"/>
                  <GradientStop Color="#FF68696C" Offset="1"/>
               </LinearGradientBrush>
            </TabItem.Background>
            <Grid Background="#FF68696C">
               <Label Content="Domain:" HorizontalAlignment="Left" Margin="0,25,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="16" Padding="5,5,5,0" UseLayoutRounding="True"/>
               <Label Content="Current User:" HorizontalAlignment="Left" Margin="0,48,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="16" Padding="5,5,5,0" UseLayoutRounding="True"/>
               <Label Content="System Type:" HorizontalAlignment="Left" Margin="-8,103,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="Version:" HorizontalAlignment="Left" Margin="-8,123,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14" Padding="5,5,5,0"/>
               <Label Content="Service Pack:" HorizontalAlignment="Left" Margin="-8,143,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="Procesor:" HorizontalAlignment="Left" Margin="-8,163,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="System Memory:" HorizontalAlignment="Left" Margin="-8,183,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="Video Card:" HorizontalAlignment="Left" Margin="-8,203,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="Resolution:" HorizontalAlignment="Left" Margin="-8,223,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="Time Zone:" HorizontalAlignment="Left" Margin="-9,243,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
               <Label Content="Last reboot:" HorizontalAlignment="Left" Margin="-9,263,0,28" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14" Width="150" Height="30"/>
               <Label Content="Up time:" HorizontalAlignment="Left" Margin="-9,283,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14"/>
```

```
<TextBlock x:Name="TB1" Margin="156,9,259,288" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" LineHeight="16" Text="aaa" Height="24" FontFamily="Segoe UI Black"/>
<TextBlock x:Name="TB2" Margin="156,31,286,266" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="24" FontFamily="Segoe UI Black"/>
<TextBlock x:Name="TB3" Margin="156,55,286,242" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="24" FontFamily="Segoe UI Black"/>
<Label Content="Operating System:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="14" Margin="-8,83,0,0"/>
<TextBlock x:Name="TB4" Margin="155,88,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Height="20" Padding="0" LineHeight="9" Text="aaa" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB5" Margin="155,108,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB6" Margin="155,128,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Padding="0" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB7" Margin="155,146,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB8" Margin="156,166,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB9" Margin="155,186,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB10" Margin="156,206,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB11" Margin="156,226,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB12" Margin="155,246,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB13" Margin="156,266,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TB14" Margin="156,286,0,0" TextWrapping="Wrap" RenderTransformOrigin="0.486,0.478" FontWeight="Bold" FontSize="14" Text="aaa" Height="20" Padding="0,2" HorizontalAlignment="Left" VerticalAlignment="Top"/>
<TextBlock x:Name="TBevent" Margin="593,38,210,561" TextWrapping="Wrap" Grid.RowSpan="2"/>
<Label Content="IP Address:" HorizontalAlignment="Right" Margin="0,37,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Subnet Mask:" HorizontalAlignment="Right" Margin="0,55,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Gateway:" HorizontalAlignment="Right" Margin="0,73,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="MAC Address:" HorizontalAlignment="Right" Margin="0,91,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Primary DNS:" Margin="226,109,282,0" VerticalAlignment="Top" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Adapter name:" HorizontalAlignment="Right" Margin="0,19,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="DHCP Server:" HorizontalAlignment="Right" Margin="0,127,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="DNS Suffix:" HorizontalAlignment="Right" Margin="0,145,282,0" VerticalAlignment="Top" Width="100" HorizontalContentAlignment="Right" Height="28"/>
<TextBlock x:Name="aTB1" HorizontalAlignment="Left" Margin="736,42,-274,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" FontWeight="Bold" Text="aaa"/>
<TextBlock x:Name="aTB2" HorizontalAlignment="Left" Margin="736,60,-274,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
<TextBlock x:Name="aTB3" HorizontalAlignment="Left" Margin="736,78,-274,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
<TextBlock x:Name="aTB4" HorizontalAlignment="Left" Margin="736,96,-274,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
<TextBlock x:Name="aTB5" HorizontalAlignment="Left" Margin="735,114,-274,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="139" Text="aaa"/>
<TextBlock x:Name="aTB6" HorizontalAlignment="Left" Margin="736,24,-336,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="200" Text="aaa"/>
<TextBlock x:Name="aTB7" HorizontalAlignment="Left" Margin="735,132,-273,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
<TextBlock x:Name="aTB8" HorizontalAlignment="Left" Margin="735,150,-273,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="138" Text="aaa"/>
<Label Content="Manufacturer:" HorizontalAlignment="Left" Margin="613,206,-104,0" VerticalAlignment="Top" Width="91" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Model:" HorizontalAlignment="Left" Margin="613,224,-104,0" VerticalAlignment="Top" Width="91" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Serial number:" HorizontalAlignment="Left" Margin="611,242,0,0" VerticalAlignment="Top" Width="92" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="Version:" HorizontalAlignment="Left" Margin="612,260,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.421,0.564" Width="92" HorizontalContentAlignment="Right" Height="28"/>
<Label Content="BIOS version:" HorizontalAlignment="Left" Margin="624,278,-124,0" VerticalAlignment="Top" Width="100"/>
<TextBlock x:Name="PCtb" HorizontalAlignment="Left" Margin="736,211,-217,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="200"/>
<TextBlock x:Name="PC_tb1" HorizontalAlignment="Left" Margin="736,229,-202,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="200"/>
<TextBlock x:Name="PC_tb2" HorizontalAlignment="Left" Margin="736,247,-192,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" RenderTransformOrigin="0.5,0.5" Width="200"/>
<TextBlock x:Name="PC_tb3" HorizontalAlignment="Left" Margin="736,265,-198,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="200"/>
<TextBlock x:Name="PC_tb4" Margin="735,283,-203,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" HorizontalAlignment="Left" Width="200"/>
```

```
            <Label Content="Computer Name:" HorizontalAlignment="Left" Margin="0,2,0,0" VerticalAlignment="Top" Width="150" Height="30" FontWeight="Bold" HorizontalContentAlignment="Right" FontSize="16" Padding="5,5,5,0" UseLayoutRounding="True"/>
            <Button x:Name="restartBTN" Content="RESTART" HorizontalAlignment="Left" Margin="211,286,0,0" VerticalAlignment="Top" Width="75" IsEnabled="false" Background="#FFF3931F"/>
            <TextBlock x:Name="TBevent1" HorizontalAlignment="Left" Margin="291,286,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="20" Width="127"/>
            <Button x:Name="infoPCBT" Content="Get INFO" HorizontalAlignment="Left" Margin="901,4,0,0" VerticalAlignment="Top" Width="75"/>
          </Grid>
        </TabItem>
        <TabItem Header="DiskInfo" Margin="4,0,-4,0" IsEnabled="False" Width="230">
          <TabItem.Background>
            <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
              <GradientStop Color="#FFF0F0F0" Offset="0"/>
              <GradientStop Color="#FF68696C" Offset="1"/>
            </LinearGradientBrush>
          </TabItem.Background>
          <Grid Background="#FF68696C">
            <ListView x:Name="disk_listView" HorizontalAlignment="Left" Height="222" Margin="10,89,0,0" VerticalAlignment="Top" Width="325">
              <ListView.View>
                <GridView>
                  <GridViewColumn Header="Drive Letter" DisplayMemberBinding ="{Binding 'Drive Letter'}" Width="80"/>
                  <GridViewColumn Header="Drive label" DisplayMemberBinding ="{Binding 'Drive Label'}" Width="80"/>
                  <GridViewColumn Header="Size" DisplayMemberBinding ="{Binding Size(MB)}" Width="80"/>
                  <GridViewColumn Header="Free Space" DisplayMemberBinding ="{Binding FreeSpace%}" Width="80"/>
                </GridView>
              </ListView.View>
            </ListView>
            <Button x:Name="Load_diskinfo_button" Content="Get info" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="75"/>
            <Label x:Name="DiskLabel" Content="Label" HorizontalAlignment="Left" Margin="10,35,0,0" VerticalAlignment="Top" Width="242"/>
            <ListView x:Name="service_listView" HorizontalAlignment="Left" Height="222" Margin="349,89,0,0" VerticalAlignment="Top" Width="325">
              <ListView.View>
                <GridView>
                  <GridViewColumn Header="Name" DisplayMemberBinding ="{Binding ServiceName}" Width="160"/>
                  <GridViewColumn Header="State" DisplayMemberBinding ="{Binding Status}" Width="80"/>
                  <GridViewColumn Header="Display Name" DisplayMemberBinding ="{Binding DisplayName}" Width="80"/>
                </GridView>
              </ListView.View>
            </ListView>
            <ListView x:Name="process_listView" HorizontalAlignment="Left" Height="222" Margin="689,89,0,0" VerticalAlignment="Top" Width="278">
              <ListView.View>
                <GridView>
                  <GridViewColumn Header="Name" DisplayMemberBinding ="{Binding ProcessName}" Width="150"/>
                  <GridViewColumn Header="Status" DisplayMemberBinding ="{Binding PID}" Width="150"/>
                </GridView>
              </ListView.View>
            </ListView>
            <Label Content="Services" HorizontalAlignment="Left" Margin="349,58,0,0" VerticalAlignment="Top" Width="59" FontWeight="Bold"/>
            <Label Content="Process" HorizontalAlignment="Left" Margin="689,61,0,0" VerticalAlignment="Top" Width="49" FontWeight="Bold"/>
            <Label Content="Disk information" HorizontalAlignment="Left" Margin="10,58,0,0" VerticalAlignment="Top" Width="325" FontWeight="Bold"/>
            <Button x:Name="StartBTN" Content="Start" HorizontalAlignment="Left" Margin="422,61,0,0" VerticalAlignment="Top" Width="75"/>
            <Button x:Name="StopBTN" Content="Stop" HorizontalAlignment="Left" Margin="502,61,0,0" VerticalAlignment="Top" Width="75"/>
            <Button x:Name="ProcessStopBTN" Content="Stop" HorizontalAlignment="Left" Margin="749,64,0,0" VerticalAlignment="Top" Width="75"/>
            <Label x:Name="ramLBL" Content="Total RAM" HorizontalAlignment="Left" Margin="100,7,0,0" VerticalAlignment="Top" Width="66"/>
            <TextBlock x:Name="ramTB" HorizontalAlignment="Left" Margin="162,12,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" FontWeight="Bold"/>
            <Label Content="Free RAM" HorizontalAlignment="Left" Margin="214,7,0,0" VerticalAlignment="Top" Width="80"/>
            <TextBlock x:Name="FreeTB" HorizontalAlignment="Left" Margin="272,12,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="89" FontWeight="Bold"/>
            <Label Content="Used RAM" HorizontalAlignment="Left" Margin="366,7,0,0" VerticalAlignment="Top" Width="69"/>
            <TextBlock x:Name="UsedTB" HorizontalAlignment="Left" Margin="428,12,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" FontWeight="Bold"/>
            <TextBlock x:Name="processNameTB" HorizontalAlignment="Left" Margin="761,35,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" FontWeight="Bold"/>
            <TextBlock x:Name="processRAM" HorizontalAlignment="Left" Margin="761,48,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" FontWeight="Bold"/>
            <Label Content="Most using:" HorizontalAlignment="Left" Margin="689,30,0,0" VerticalAlignment="Top"/>
            <Label Content=" RAM (MB) :" HorizontalAlignment="Left" Margin="689,43,0,0" VerticalAlignment="Top"/>
            <Label Content="Most using CPU:" HorizontalAlignment="Left" Margin="689,4,0,0" VerticalAlignment="Top" Width="99"/>
            <TextBlock x:Name="cpuTB" HorizontalAlignment="Left" Margin="788,9,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="122" FontWeight="Bold"/>
```

```
            <Label Content="CPU percent (%):" HorizontalAlignment="Left" Margin="689,17,0,0" VerticalAlignment="Top"/>
            <TextBlock x:Name="cpuPercentTB" HorizontalAlignment="Left" Margin="788,22,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="83"
FontWeight="Bold"/>
          </Grid>
        </TabItem>
        <TabItem Header="Event" Margin="7,0,-7,0" IsEnabled="False" Width="230">
          <TabItem.Background>
            <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
              <GradientStop Color="#FFF0F0F0" Offset="0"/>
              <GradientStop Color="#FF68696C" Offset="1"/>
            </LinearGradientBrush>
          </TabItem.Background>
          <Grid Background="#FF68696C">
            <ListView x:Name="log_listView" HorizontalAlignment="Left" Height="134" Margin="10,35,0,0" VerticalAlignment="Top" Width="966">
              <ListView.View>
                <GridView>
                  <GridViewColumn Header="TimeGenerated" DisplayMemberBinding ="{Binding TimeGenerated}" Width="150"/>
                  <GridViewColumn Header="MSG" DisplayMemberBinding ="{Binding Message}" Width="500"/>
                  <GridViewColumn Header="Entry" DisplayMemberBinding ="{Binding EntryType}" Width="150"/>
                  <GridViewColumn Header="Source" DisplayMemberBinding ="{Binding Source}" Width="150"/>
                </GridView>
              </ListView.View>
            </ListView>
            <Button x:Name="logBTN" Content="Get-event" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="75"/>
            <ListView x:Name="hotfix_listView" HorizontalAlignment="Left" Height="137" Margin="10,174,0,0" VerticalAlignment="Top" Width="966">
              <ListView.View>
                <GridView>
                  <GridViewColumn Header="Property" DisplayMemberBinding ="{Binding Description}" Width="250"/>
                  <GridViewColumn Header="HotFixID" DisplayMemberBinding ="{Binding HotFixID}" Width="200"/>
                  <GridViewColumn Header="InstalledOn" DisplayMemberBinding ="{Binding InstalledOn}" Width="250"/>
                  <GridViewColumn Header="InstalledBy" DisplayMemberBinding ="{Binding InstalledBy}" Width="250"/>
                </GridView>
              </ListView.View>
            </ListView>

          </Grid>
        </TabItem>
      </TabControl>

    </Grid>
</Window>
```

Administrátorský skript

```
Add-Type -AssemblyName presentationframework, presentationcore
$wpf = @{}
$inputXML = Get-Content -Path "F:\DIPLOMOVÁ PRÁCE (kopie)\TOOL-
GUI_27.3\AdminModule\TabPanel\TabPanel\MainWindow.xaml"
$inputXMLClean = $inputXML -replace 'mc:Ignorable="d"','' -replace "x:N",'N' -replace
'x:Class=".*?"','' -replace 'd:DesignHeight="\d*?"','' -replace
'd:DesignWidth="\d*?"',''
[xml]$xaml = $inputXMLClean
$reader = New-Object System.Xml.XmlNodeReader $xaml
$tempform = [Windows.Markup.XamlReader]::Load($reader)
$namedNodes =
$xaml.SelectNodes("//*[@*[contains(translate(name(.),'n','N'),'Name')]]")
$namedNodes | ForEach-Object {$wpf.Add($_.Name, $tempform.FindName($_.Name))}
#-------------------------------------------------------------------------------
-------------------------------------------------------

$tempform.Add_Loaded({$wpf.ComputerName.Text = $env:COMPUTERNAME})

#Verifikace, že PC je připojeno do sítě. //rozšíření nabídky TAB
$wpf.Verify.Add_Click({if (Test-Connection $wpf.ComputerName.Text -Count 1 -Quiet){
        $wpf.PingLBL.text = "$($wpf.ComputerName.Text ) is online, unlocking"
```

```
        $wpf.tabControl.Items[1..3] | % {$_.IsEnabled = $true}
        }
        else{
        $wpf.PingLBL.text = "$($wpf.ComputerName.Text ) is offline"
        $wpf.tabControl.Items[1..3] | % {$_.IsEnabled = $false}
        }
})


$wpf.infoPCBT.Add_Click({
# PC information

        $User = Get-WmiObject win32_computersystem -ComputerName
$wpf.ComputerName.Text
        $os = Get-WmiObject Win32_OperatingSystem -computerName $wpf.ComputerName.Text

#Lastboot/time zone/ uptime days

        $boottime = $OS.converttodatetime($OS.LastBootUpTime)
        $uptime = New-TimeSpan (get-date $boottime)
        $uptime_days = [int]$uptime.days


# da se řešit i jinak

       $totalRAM = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock
{[Math]::Truncate((get-WMIObject win32_operatingsystem | Measure-Object
TotalVisibleMemorySize -sum).sum / 1024)}
       $cpu = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock {Get-
WmiObject Win32_Processor}
       $computerVideo = Invoke-Command -ComputerName $wpf.ComputerName.Text -
ScriptBlock {Get-WmiObject Win32_VideoController}


            $wpf.TB1.text = $user.name
            $wpf.TB2.text = $User.domain
            $wpf.TB3.text = $User.userName

            $wpf.TB4.text = $os.caption
            $wpf.TB5.text = $os.OSArchitecture
            $wpf.TB6.text = $os.version
            $wpf.TB7.text = $os.ServicePackMajorVersion
            $wpf.TB8.text = $cpu.Name
            $wpf.TB9.text = $totalRAM
            $wpf.TB10.text = $computerVideo.description
            $wpf.TB11.text = $computervideo.VideoModeDescription
            $wpf.TB12.text = (get-date)
            $wpf.TB13.text = $boottime
            $wpf.TB14.text = $uptime_days

#Adapter INFo

    $adapter = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock {get-
wmiobject win32_networkadapter -filter "NetEnabled='True'"}
    $IP = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock {Get-
WmiObject Win32_NetworkAdapterConfiguration -filter 'IPEnabled="True"' | Select -
expand IPAddress | ?{$_ -notmatch ':'}}
    $network = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock {Get-
WmiObject Win32_NetworkAdapterConfiguration -filter 'IPEnabled="True"'}

            $wpf.aTB1.text = $IP
            $wpf.aTB2.text = $network.IPSubnet
            $wpf.aTB3.text = $network.DefaultIPGateway
            $wpf.aTB4.text = $adapter.MACAddress
            $wpf.aTB5.text = $IP.DNSHostName
            $wpf.aTB6.text = $adapter.name
            $wpf.aTB7.text = $network.DHCPServer
            $wpf.aTB8.text = $network.DNSDomain

# Computer info

    $comp = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock {Get-
WmiObject Win32_ComputerSystem}
    $bios = Invoke-Command -ComputerName $wpf.ComputerName.Text -ScriptBlock {Get-
WmiObject Win32_BIOS}

         $wpf.PCtb.text = $comp.Manufacturer
```

```
          $wpf.PC_tb1.text = $comp.Model
          $wpf.PC_tb2.text = $comp.serialnumber
          $wpf.PC_tb3.text = $comp.Version
          $wpf.PC_tb4.text = $bios.Version

 $props = @{ 'ComputerName' =  $user.name;
               'Domain'=$User.domain;
               'Current User'=$User.userName;
               'Operating system'=$os.caption
               'System Type'=$os.OSArchitecture;
               'Version' =$os.version;
               'Service Pack'=$os.ServicePackMajorVersion;
               'Procesor'=$cpu.Name;
               'System memory'=$totalRAM;
               'Video card' =$computerVideo.description;
               'Resolution' =$computervideo.VideoModeDescription;
              #'Time zone' =$MonitorInfo.Model
               'Last reboot' =$uptime_days;
   }
        $obj = New-Object -TypeName PSObject -Property $props
        $obj | Out-File $env:USERPROFILE\Desktop\PCinfo.txt

#Na základě podmínky vyhodnotíme , zdali je vyžadován restart či nikoliv

    if ($uptime_days -ge "1") {
    $wpf.restartBTN.IsEnabled = $true
    $wpf.TBevent1.text = "Doporučujeme restart!!!"
        $wpf.restartBTN.Add_Click({Restart-Computer -computername
$wpf.computername.text})
       }
   else{$wpf.restartBTN.IsEnabled = $false}
   })

#disk inforamation
$wpf.Load_diskinfo_button.Add_Click({

$TotalRAM = [Math]::Truncate((get-WMIObject win32_operatingsystem -computername
$wpf.ComputerName.Text | Measure-Object TotalVisibleMemorySize -sum).sum / 1024)
$FreeRAM = ((get-WMIObject -computername $wpf.ComputerName.Text -class
win32_operatingsystem).freephysicalmemory) / 1024
$UsedRAM = (($TotalRAM) - ($FreeRAM))
$RAMPercentUsed = ([math]::truncate(($UsedRAM) / ($TotalRAM) * 100))

$memhog = (get-process -computername $wpf.ComputerName.Text | sort-object WS -
descending | select-object -first 1)
$memhogname = ($memhog).processname
$memhogram = ([math]::truncate(($memhog).WS / 1MB))
$memhogid = ($memhog).Id

$CPUpercent = (Get-WmiObject -computername $wpf.ComputerName.Text win32_processor |
Measure-Object -property LoadPercentage -Average).Average
$cpuhog = (get-process -computername $wpf.ComputerName.Text | sort-object CPU -
descending | select-object -first 1)
$cpuhogname = ($cpuhog).processname

    $wpf.ramTB.text = $TotalRAM
    $wpf.freeTB.text = $FreeRAM
    $wpf.usedTB.text = $UsedRAM

    $wpf.processNameTB.text = $memhogname
    $wpf.processRAM.text = $memhogram

    $wpf.cpuTB.text = $cpuhogname
    $wpf.cpuPercentTB.  text = $CPUpercent

Function Get-DiskInfo {
param($computername = $env:COMPUTERNAME)

    Get-WMIObject Win32_logicaldisk -ComputerName $wpf.ComputerName.Text | Select-
Object @{Name='ComputerName';Ex={$computername}},`

@{Name='Drive Letter';Expression={$_.DeviceID}},`

@{Name='Drive Label';Expression={$_.VolumeName}},`

@{Name='Size(MB)';Expression={[int]($_.Size / 1MB)}},`
```

```
@{Name='FreeSpace%';Expression={[math]::Round($_.FreeSpace / $_.Size,2)*100}}

}


    Get-DiskInfo -computername $wpf.ComputerName.Text | %
{$wpf.disk_listView.AddChild($_)}

    $wpf.DiskLabel.Content = "Disk info for system $($wpf.ComputerName.Text)"

#=============================================================================================

#Service information

$WPF.service_listView.Items.Clear()
get-service -ComputerName $WPF.ComputerName.Text |
Sort-Object Status | % {$WPF.service_listView.AddChild($_)}

#Start Service BTN

$wpf.StartBTN.Add_Click({
    $wpf.service_listView.SelectedItems | % {
        ($stat = GET-WMIObject Win32_Service -ComputerName $wpf.ComputerName.Text |
                ? Name -like $_.Name).StartService()
            if ($stat -eq 0){}
            elseif ($stat -eq 2){}
            else{}

        }

    $wpf.service_listView.Items.Clear()
    get-service -ComputerName $wpf.ComputerName.Text |
        % {$wpf.service_listView.AddChild($_)}
})

#Stop service BTN

$wpf.stopBTN.Add_Click({
    $wpf.service_listView.SelectedItems | % {
        ($stat = GET-WMIObject Win32_Service -ComputerName $WPF.ComputerName.Text |
                ? Name -like $_.Name).StopService()
            if ($stat -eq 0){}
            elseif ($stat -eq 2){
            }
            else{}
        }

    $WPF.service_listView.Items.Clear()
    get-service -ComputerName $WPF.ComputerName.Text |
           % {$WPF.service_listView.AddChild($_)}
})

#Show all running proceses

$wpf.process_listView.Items.Clear()
get-process -ComputerName $wpf.computerName.text | sort-Object Caption |
        % {$WPF.process_listView.AddChild($_)}

#stop process button

$wpf.ProcessStopBTN.Add_Click({
    $wpf.process_listView.SelectedItems | % {
        ($stat = GET-WMIObject win32_process -ComputerName $wpf.ComputerName.Text |
                ? Name -like $_.ProcessName).StopProcess()
            if ($stat -eq 0){}
            elseif ($stat -eq 2){}
            else{}

        }

    $wpf.process_listView.Items.Clear()
    Get-process -ComputerName $wpf.ComputerName.Text |
        sort-Object Caption | % {$wpf.process_listView.AddChild($_)}
})

})
```

```
$wpf.logBTN.Add_Click({
$wpf.log_listView.Items.Clear()
Get-EventLog -ComputerName $wpf.ComputerName.text -LogName System -newest 10 | Where-
Object {$_.TimeGenerated -gt (Get-Date).AddDays(-2)} | %
{$wpf.log_listView.AddChild($_)}


$wpf.hotfix_listView.Items.Clear()
#$startdate = (Get-date).AddDays(-30)
Get-HotFix -ComputerName $wpf.ComputerName.text | %
{$wpf.hotfix_listView.AddChild($_)}

})


$wpf.PCinfo.ShowDialog() | Out-Null
```