VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

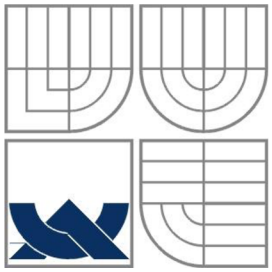
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SPOŘIČ OBRAZOVKY S ANTIVIROVOU KONTROLOU

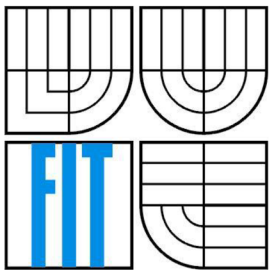
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE IVO REBENDA
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SPOŘIČ OBRAZOVKY S ANTIVIROVOU KONTROLOU

ANTI-MALWARE SCREENSAVER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

IVO REBENDA

VEDOUCÍ PRÁCE
SUPERVISOR

Dr. Ing. PETR PERINGER

BRNO 2008

Abstrakt

Obsahem této práce je vytvoření spořiče obrazovky s antivirovou kontrolou a přesunutí náročnosti antivirové kontroly na hardware, na dobu kdy uživatel s počítačem nepracuje. Použité aplikační programovací rozhraní je AVG API, programovací jazyk C++. Cílový operační systém je Microsoft Windows.

Klíčová slova

Spořič obrazovky, Windows, C++, AVG, antivirová kontrola, viry, návrhové vzory

Abstract

The aim of this thesis is creating anti-malware screen saver and move hardware demanding anti-malware scanning, on time when the user doesn't work with computer. Application programming interface is AVG API, programming language C++. Target operating system is Microsoft Windows.

Keywords

Screensaver, Windows, C++, AVG, anti-malware detection, viruses, design patterns

Citace

Rebenda Ivo: Spořič obrazovky s antivirovou kontrolou. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Spořič obrazovky s antivirovou kontrolou

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Dr. Ing. Peringera Petra.
Další informace mi poskytnul Ing. Obluk Karel Ph.D., AVG Technologies CZ.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ivo Rebenda
8.5.2008

Obsah

Úvod

Testování počítače na přítomnost škodlivého software je v této době téměř nutnost. Vzhledem k tomu, že přibývá typů souborů, které mohou být infikovány škodlivým kódem (např. i video, fotky apod.), stoupá také časová náročnost na provedení testů, zejména nárok na procesor a pevný disk. Antivirové produkty se proto snaží přesunout antivirové testy na dobu, kdy test uživatele při práci zatíží co nejméně. Většinou se to snaží řešit plánováním testů na půlnoc a večerní hodiny, kdy se předpokládá, že uživatel s počítačem nepracuje. Toto řešení má ovšem spoustu nevýhod. Další možností je využít existující funkci operačního systému, a tou je spořič obrazovky. Ačkoliv tato funkce operačního systému byla vytvořena k poněkud jinému účelu, pouští se opravdu jen v době, kdy uživatel s počítačem nepracuje.

Cílem této práce je návrh a implementace spořiče obrazovky s antivirovou kontrolou pro systémy Microsoft Windows. Bude se jednat o spořič obrazovky, který k antivirové kontrole bude používat antivirový systém AVG. Za pomocí programového aplikačního rozhraní se při spuštění připojí k jádru antiviru AVG a po připojení zpřístupní skenovací funkce. Po spořiči se požaduje, aby bylo možné nastavit parametry testu. Tím je myšleno především možnost nastavení standardního chování při nalezení virové infekce. Téměř nutností je možnost pozastavení antivirové kontroly po ukončení spořiče a automatické obnovení při příštím spuštění. Hlavním důvodem je fakt, že velké části uživatelů neběží spořič dostatečně dlouho, aby se stihli otestovat všechny soubory v počítači.

V první kapitole se budu věnovat přehledu. Spořiče obrazovky od počátku sloužily k předcházení poškozování televizních obrazovek při dlouhodobějším zobrazování a u CRT monitorů měli zabránit vypalováním pixelů. Kapitola se lehce dotýká i způsoby programování vlastních spořičů obrazovky. Další částí kapitoly jsou návrhové vzory. Návrhové vzory jsou použity v tomto programu a jsou základem každé správně napsané větší aplikace. Budu se věnovat dvěma vzorům, které jsem použil. Poslední část kapitoly se věnuje virům a antivirům. Začíná se historií virů a za ním následuje základní rozdělení do nejdůležitějších skupin.

Další kapitola se věnuje návrhu řešení. Jako první je zde popsána specifikace požadavků na spořič. Vzhledem k situaci, že při programování se bude používat objektově orientované programování, nesmíme opomenout důkladný návrh tříd. Dále v kapitole najdeme návrh řešení chování spořiče s ohledem na režim napájení počítače. Nakonec jsou zde formáty XML souborů, které se budou využívat k ukládání průběhu testu a nastavení programu.

Třetí kapitola nese název implementace. V této kapitole se zpočátku věnuji struktuře programu. Poté rozebírám jednotlivé části programu - spořič, nastavení spořiče a skenování. Ke konci kapitoly je zde popsán způsob instalace spořiče do systému a způsob odstranění ze systému.

Poslední kapitolou je závěr. V závěru je shrnutí celé práce.

1 Přehled

1.1 Spořič obrazovky

Spořič obrazovky je malý program, spuštěný přes celou obrazovku, který se pouští po nastavené době nečinnosti uživatele a při jakémkoliv náznaku, že uživatel chce opět s počítačem pracovat (pohyby myší, stisk klávesy) se program okamžitě ukončí.

1.1.1 Historie

První spořiče obrazovky se objevily v herních konzolách Atari VCS/2600 v roce 1977. Tenkrát byl ale spořič zabudovaný ve firmware konzole a nešlo jej nastavit ani jinak změnit. Byl zaveden především z důvodu možného poškození tehdejších televizních obrazovek při delším zobrazením jedné scény. Spořič pouze automaticky měnil zobrazené barvy.

Myšlenka programového spořiče obrazovky byla poprvé implementována v roce 1983 programátorem John Socha (známého především jako autora legendárního správce souborů Norton Commander), kdy program, pojmenovaný *scrnsave,* po třech minutách neaktivity ztmavla obrazovka počítače.

1.1.2 Důvody vzniku

Dříve, kdy LCD monitory nebyly standardní záležitostí, byly počítače vybaveny CRT monitory. Starší CRT monitory byly náchylné na takzvané vypálení obrazu. Při dlouhodobém zobrazení stejného obrazu, nejčastěji zobrazení přihlašovacího dialogu po dobu několika dnů, se při změně obrazu na obrazovce stále objevoval i takzvaný duch, který představoval například zmíněný přihlašovací dialog. Dělo se tak z principu zobrazení obrazu na fosforeční matici. Moderní CRT monitory se tento nežádoucí efekt snažily eliminovat silnější a kvalitnější vrstvou fosforenčního materiálu. Ovšem vypalování pixelů se týká i LCD monitorů. Pokud má určitý pixel dlouhodobě stejnou barvu, tedy na zobrazovací matici je stále stejné napětí, dochází nějakým způsobem k dočasné degradaci elektronických součástek, která se nám projevuje zobrazováním původní barvy pixelu. K tomuto efektu může oproti CRT dojít v relativně krátké době - i za několik dnů. Naštěstí tento nechtěný efekt lze odstranit problikáváním nejsvětlejší a nejtmavší hodnoty pixelu ve frekvenci, která je jen o málo (doporučuje se o polovinu) menší než zobrazovací frekvence monitoru. Pokud vypálení

není ještě moc výrazné, lze podobného efektu docílit i vypnutím a opětovným zapnutím monitoru. Oproti CRT obrazovkám ale nedochází k trvalému poškození.

1.1.3 Programové řešení

Spořič obrazovky je pod MS Windows možné psát v podstatě v každém programovacím jazyce podporujícím aplikační rozhraní Windows API. Nejčastěji se jako programovací jazyk pro spořiče obrazovky používá C++ a C#. Při programování je nejlepší řešení použít metody pro vytvoření spořiče z běžně používaných Frameworků, pro C++ například MFC. Tyto metody za nás zařídí zpracování parametrů při práci se spořičem. Ikdyž je spořič běžná aplikace, spouští se parametrem */s*. Parametr */p* vyvolává vykreslení ukázkového okna při výběru spořiče. A poslední parametr je parametr, který vyvolá okno s nastavením spořiče */c*. Při použití frameworku se o tyto parametry většinou už nemusíme starat, framework za nás podle parametru automaticky zavolá příslušné metody. Další odlišnost od ostatních aplikací je přípona programu. Používá se koncovka *.scr*, podle které systém pozná, že se jedná o spořič a podle toho reaguje na spuštění. Spořič může grafický výstup zobrazovat pomocí GDI, Direct X nebo OpenGL.

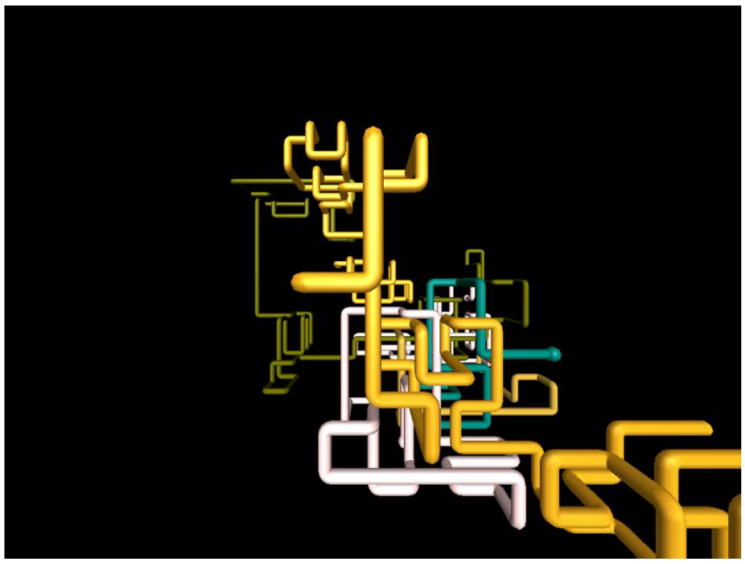
1.1.4 Současnost

V současné době se spořiče obrazovky k původnímu účelu využívají jen zřídka. Důvodů, proč se tak děje, je více.

Prvním důvodem je čím dál častější použití modernějších CRT monitorů, nebo LCD monitorů, které by trpěly vypalováním obrazu za krátkou dobu a postupné dosluhování starších CRT monitorů. Vzhledem ke klesajícím cenám kvalitních LCD monitorů a nárokům na nižší spotřebu si dnes pořizuje CRT monitory minimální počet lidí.

Dalším důvodem je také funkce, zabudovaná a lehce dostupná ve většině používaných operačních, která po době nečinnosti vyšle signál k vypnutí monitoru. Takto má monitor minimální spotřebu oproti zapnutému stavu a zároveň se ještě více šetří.

Moderní spořiče obrazovky také už pouze neztmavují obrazovku. Většinou se snaží graficky zaujmout, takže na obrazovku vykreslují například fraktály, různé grafické ornamenty a tím při šetření obrazovky ve své podstatě nešetří ostatní komponenty počítače, které jsou obvykle vytěžovány na své maximum.



Obr. 1 - Ukázka spořiče obrazovky „Potrubí“

1.1.5 Shrnutí

Dnes se spořiče obrazovky už nevyužívají v takovém množství jako v době, kdy ke každému počítači byl připojen CRT monitor. Avšak z hlediska provádění náročnějších úloh na pozadí je to stále zajímavá funkce operačního systému, která pořád najde své uplatnění. Spořiče dnes ze značné části způsobují bezpečnostní riziko, protože některé viry se kromě nakažení *.exe* souborů, pokouší nakazit i *.scr* soubory.

1.2 Návrhové vzory

Vzor popisuje problém, na který můžeme narazit při návrhu a implementaci software. Vzory můžeme dělit podle etapy vývoje, ve které je použijeme:

- **Analytické vzory** – využívá se ve fázi specifikace požadavků a při vytváření analytického modelu
- **Architektonické vzory** – strukturální prvky softwarových systémů
- **Návrhové vzory** – typické vztahy a interakce mezi třídami a objekty
- **Idiomy** – vzor jak daný problém implementovat v určitém programovacím jazyce

Návrhové vzory představují obecné řešení problémů, které se opakovaně objevují při návrhu software. Návrhové vzory nejsou knihovny nebo části zdrojového kódu, které by se daly přímo vložit do programu. Jedná se o popis či šablonu, jak řešit problém způsobem, který může být použit v různých situacích. Objektově orientované návrhové vzory typicky ukazují vztahy a interakce mezi třídami a objekty, aniž by určovaly implementaci konkrétní třídy. Algoritmy nejsou považovány za návrhové vzory, protože řeší výpočetní problémy a nikoliv návrhové.

Návrhové vzory nepocházejí ze softwarového inženýrství – jsou zcela běžné v každodenním životě. K asi nejmarkantnějším a nejstarším příkladům patří architektura. Gotickou katedrálu poznáte už zdaleka proto, že tehdejší architekti a jejich stavební hutě používaly stejných návrhových vzorů.[4]

O návrhových vzorech v oblasti objektově orientovaného programování se začalo mluvit po vydání knihy **Design Pattern** Elements of Reusable Object-Oriented Software od **E. Gamma**, **R. Helm**, **R. Johnson**, **J. Vlissides** The Gang of Four (GoF) v roce 1995. V knize vydali 23 vzorů, které jsou v dalších publikacích různě rozšiřovány podle užití. Všechny vzory byly zařazeny do tří základních kategorií - tvořivé vzory, strukturální vzory a vzory chování. Seznam základních 23 vzorů:

- Abstract factory
- Factory method
- Builder
- Prototype
- Singleton
- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy
- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template method
- Visitor

1.2.1 Knihovní třída

Knihovní třída slouží jako obálka pro soubor statických metod. Protože k tomu nepotřebuje vytvářet instance, je vhodné jejich vytváření znemožnit. Pro knihovní třídu se v anglické literatuře někdy používá termín Utility. [2]

Používá se v případě, že chceme v programu oddělit nebo zabalit určité používané funkce. Funkce zabalíme do tříd a převedeme na statické metody. To nám zajistí, že se nám nebudou plést do aktuálního jmenného prostoru a budeme k nim moci přistoupit pouze přes instanci třídy, která v tomto případě vůbec nemusí být vytvořena, takže nám stačí přistupovat pouze přes název třídy. [5]

Příklad:

```
static class Utility
{
public:
	static int Sum(int a, int b, int c)
	{
		return a + b + c;
	}
};
```

Použití:

```
	int vysledek = Utility::Sum (2, 3, 4);
```

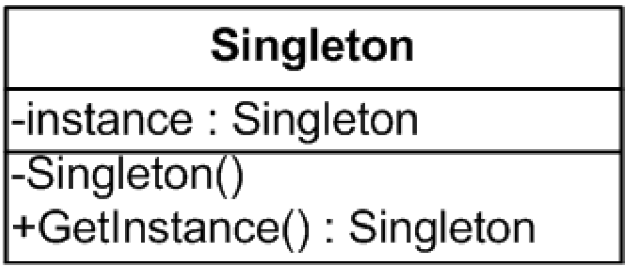
1.2.2 Singleton (Jedináček)

Singleton specifikuje, jak vytvořit třídu, která bude mít nejvýše jednu instanci. Tato instance přitom nemusí být vlastní instancí dané třídy.[2]

Nutnost existence jediné instance se v programu objevuje poměrně často. Jednak to bývá z důvodu časté práce s určitou třídou, u které vytvoření instance zabere delší dobu, a nebo s třídami, se kterými pracuje i více aplikací zároveň.

Asi nejznámějším příkladem z operačního systému je například schránka, jejímž prostřednictvím se přesouvají či kopírují data mezi aplikacemi. Standartní schránka je pro všechny aplikace společná. Když se do ní v některé spuštěné aplikace něco uloží, přemaže se původní obsah.

Jedináček se také hojně využívá při impementaci ostatních návrhových vzorů. Možností implementace totoho návrhového vzoru je velmi mnoho, avšak všechny mají společný minimálně jeden krok – skrytí konstruktoru jako privátní metodu. Pouze pokud skryjeme konstruktor, tak můžeme zajistit, aby si nikdo nemohl vytvořit další instanci třídy, která by měla mít instanci pouze jedinou.



Obr. 2 - UML diagram

Příklad:

```
class Singleton
{
private:
	static Singeton* inst;
protected:
	Singleton();
public:
	static Singleton* GetInstance();
}

Singleton* Singleton::inst = 0;  // Inicializace ukazatele
Singleton* Singleton::GetInstance()
{
	if (inst == 0)
		inst = new Singleton();

	return inst;
};
```

Použití:

```
	Singleton *instance = Singleton::GetInstance();
```

1.3 Viry a antivirová kontrola

1.3.1 Historie

První sebe-replikující se program naprogramoval Dr. Frederik Cohen už v roce 1983. Byl to první program, který můžeme označovat jako virus. Jednalo se sice o neškodný kód, ale uměl se sám

replikovat. V roce 1986 se objevil program od bratří Basid a Amjad Farooq Alvi. Jmenoval se Brain a byl to první virus, který kromě schopnosti sebe-replikace útočil na určitou část disku.

Pravděpodobné důvody vzniku virů:

- Vytváří je mladí programátoři, kteří si chtějí vyzkoušet své dovednosti a neuvědomují si, jaké může mít rozšíření viru následky
- Pomsta propuštěných zaměstnanců tím, že napíší vir a vypustí jej do firemní sítě za účelem co nejvíce poškodit firmu
- Způsob výdělku tím, že při rozšíření viru můžou mít přístup k infikovaným počítačům a mohou z nich rozesílat nevyžádanou poštu, pokusit se získat důvěrné informace za účelem dalšího prodeje

1.3.2 Viry

Název je odvozen díky jistým podobnostem od biologických virů. Oba typy virů jsou schopny sebe-replikace, tedy množení se, ale pouze za přítomnosti hostitele, k němuž jsou připojeni. Hostitelem mohou být například spustitelné soubory, systémové oblasti disku, popřípadě soubory, které nelze spustit přímo, ale při otevření specifickými aplikacemi (dokumenty Microsoft Wordu apod.). Jakmile je tento hostitel spuštěn (vykonán), provede se rovněž kód viru. Během tohoto okamžiku se obvykle virus pokouší zajistit další sebe-replikaci a to připojením k dalším vhodným vykonatelným hostitelům.

Podle typu hostitele a způsobů infekce můžeme viry rozdělovat do dalších skupin. [3]

1.3.2.1 Trojské koně

Na rozdíl od virů není tento typ škodlivého kódu schopen sebe-replikace a infekce souborů. Trojský kůň nejčastěji vystupuje pod spustitelným souborem typu Exe, který neobsahuje nic jiného (užitečného), než samotné „tělo“ trojského koně. Odtud společně se skutečností, že trojan není připojen k žádnému hostiteli, plyne, že jedinou formou dezinfekce je odmazání dotyčného souboru. Starší definice říkají, že trojan je program, vizuálně vypadající jako užitečný, ve skutečnosti však škodlivý. [7] V současnosti se můžeme setkat nejčastěji s následující formou trojanů:

- **Password-stealing trojani**

Trojské koně, kteří jsou naprogramování obvykle pro sledování stisků jednotlivých kláves, které ukládá a většinou pak rozesílá na e-mailové adresy. Tyto e-maily mohou pak obsahovat i informace o našich heslech, bankovních účtech apod.

- **Destruktivní trojani**

Klasická forma, pod kterou je pojem trojských koní obecně chápán. Pokud je takový trojský kůň spuštěn, pak likviduje soubory na disku, případně se ho pokusí celý smazat.

- **Dropper**

Program určený pouze k vypuštění jiného škodlivého programu, který má obsažen v sobě.

- **Downloader**

V podstatě stejný typ jako „Dropper“, ale nenese si s sebou žádný škodlivý software. Škodlivý software si stahuje z internetu z přednastavených míst.

- **Proxy Trojan**

Vytvoří na daném počítači proxy server a pomocí něho například odesílají nevyžádanou poštu. Je téměř nulová šance, že se při odeslání takovéto pošty podaří vypátrat původce.

1.3.2.2 Zadní vrátka (Backdoor)

Škodlivý kód, úmyslně obsažený v některé aplikaci, která se navenek tváří jako užitečná. Tento kód je většinou použit k převzetí řízení počítače (většinou přes internet) a určen například na rozesílání nevyžádané pošty a podobně. Počítačový červy často po průniku do systému instalují do počítače zadní vrátka.

1.3.2.3 Červi

Viry označené jako Červ/Worm se šíří po síti formou datových paketů. K vlastnímu šíření využívají bezpečnostní díry nebo slabé místa síťových bezpečnostních protokolů. Dominový efekt může mít za následek až zahlcení počítačové sítě. Šíří se obvykle bez účasti uživatele. Nejčastější jsou e-mailové červy. Šíří se jako soubory v přílohách elektronické pošty a napadají adresy, které naleznou v adresáři napadeného počítače. Červy využívají chyby programů pro čtení pošty, sociální inženýrství nebo kombinaci obojího ke spuštění infikovaného programu, který je v příloze e-mailu. Ostatní červy většinou využívají chyby v serverových programech a šíří se tak sítí zcela automaticky. [6]

První počítačový červ byl naprogramován v roce 1978 v Xerox PARC. Šlo o červa, který na síti nacházel nečinné procesory a přiděloval jim činnost. Tedy záměrem prvního červa bylo dosáhnutí efektivního využívání procesorů na síti. První škodlivý červ byl tzv. Morrisův červ, který v roce 1989 zahltil velkou část tehdejší sítě internet. Vzhledem k povaze počítačových červů se jako nejúčinnější obrana jeví používání alternativních programů. Červi nejčastěji útočí jen na celosvětově nejpoužívanější programy. Asi "nejslavnější" červ je ILOVEYOU, který do prvních e-mailových schránek dorazil 4. března roku 2000. Jednalo se o e-mail, který v příloze obsahoval soubor napsaný

ve Visual Basic skriptu se škodlivým kódem. Tento červ během 24 hodin nakazil 10% počítačů připojených k internetu. Napáchal škody za 5.5 miliardy dolarů.

1.3.2.4 Retroviry

Tyto viry mají za úkol detekci antivirů. Po nalezení antiviru se jej pokusí deaktivovat a v některých případech nakonec i vymazat. Pokud je tento vir úspěšný, většinou po něm nastoupí „normální“ virus.

1.3.2.5 Spyware

Jsou programy, které mimo svoji funkci odesílají informace z počítače na internet bez vědomí uživatele. Spyware nejčastěji odesílá:

- Informace uložené v registrech
- Historii navštívených stránek
- IP adresu počítače
- Seznam nejčastěji používaných programů a souborů
- Informace o nainstalovaném software

Některé spyware uživatele obtěžují náhodně vyskakujícími hláškami s doporučením na koupi určitých softwarových produktů, přednastavují domovskou stránku a také přidávají nežádoucí ikony na plochu. Jako nejúčinnější ochrana proti spyware se doporučuje neprohlížet internetové stránky s podezřelým obsahem, používat bezpečný a aktualizovaný internetový prohlížeč, a používat firewall. Standardní antivirové programy většinou spyware nenajdou, je proto nutné použít antispyware. Antispyware bývá většinou součástí programových balíků kompletních antivirových systémů nebo se dá pořídit jako samostatný software.

1.3.2.6 Adware

Pojmem Adware(**Ad**vertising-supported soft**ware)** se označují programy, které se nám snaží znepříjemnit práci na počítači reklamou. Většinou se jedná o obtěžující vyskakující okna nebo o ikony v oznamovací oblasti. Adware není přímo nebezpečný jako spyware nebo jiné typy škodlivého software, ale nabízením všemožných produktů dokáže uživateli velice znepříjemnit práci. Oproti spyware adware neodesílá žádné data na internet.

1.3.3 Antiviry

Antivirové programy jsou software, který slouží k odhalení a odstranění počítačových virů a jiného škodlivého software. K identifikaci se používají nejčastěji tyto dvě techniky:

- Procházením souborů na místních discích a vyhledávat podezřelé soubory podle identifikátorů uložených ve své databázi
- Sledování aktivit běžících programů, jejich komunikace a zápisů na disk

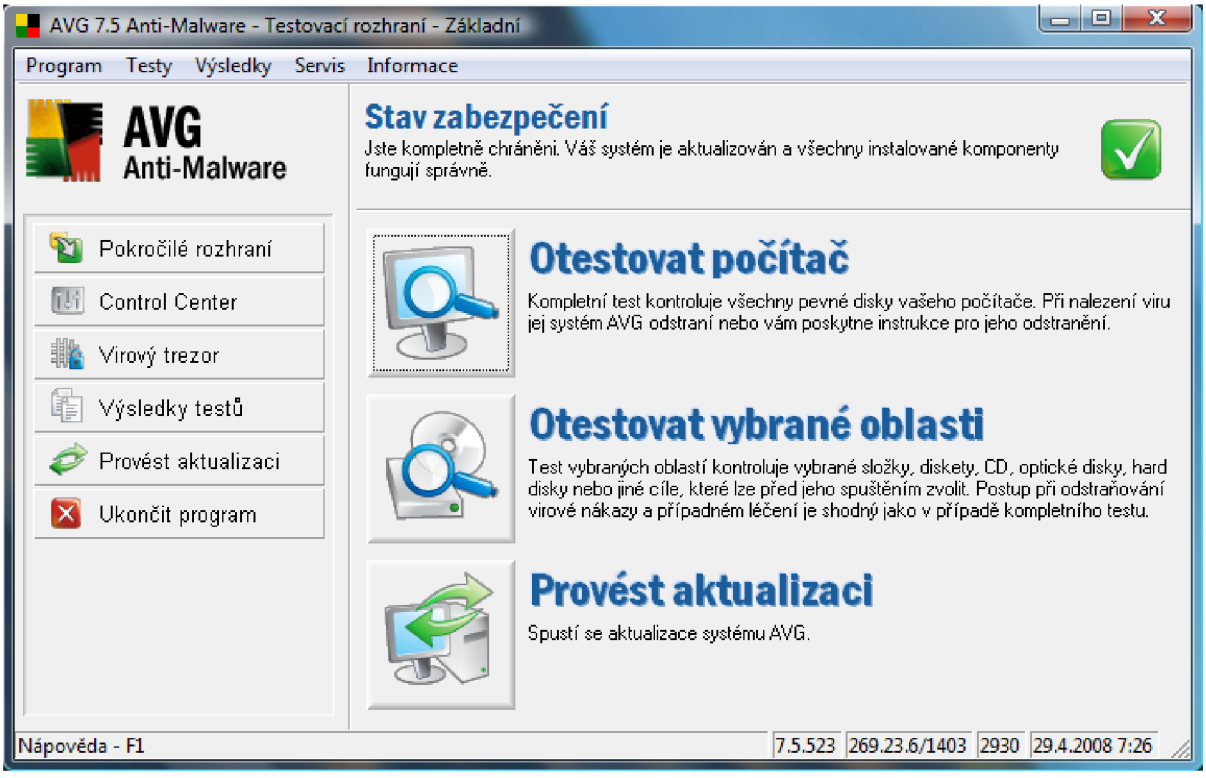
Po nálezu můžeme infikovaný soubor vyléčit nebo smazat. Při léčení se používají následující tři techniky:

- Algoritmické
- Heuristické
- Speciální

Algoritmické léčení spoléhá na funkčnost programu při odstranění té části kódu, ve které se nachází virus. Poté musí antivirový program upravit potřebné hlavičky těchto souborů, aby byl program stále spustitelný. Vzhledem k tomu, že velké množství virů při své infikaci souboru přepíše i část dalších částí kódu, ne vždy je rekonstrukce úspěšná.

Heuristické léčení funguje na principu simulovaného spuštění programu, kdy antivirový program zjišťuje, jestli právě vykonává kód viru nebo daného programu. Většina virů totiž v nějaké fázi nakonec předá řízení programu původnímu kódu, heuristické léčení tak může přesně zjistit, které části kódu jsou samotný vir.

Mezi speciální léčení patří léčení trojských koní, backdoorů apod. Tyto programy se vyléčit nedají, protože neobsahují žádný užitečný kód. Jediný způsob vyléčení je smazání souborů.



Obr. 3 - Ukázka prostředí antiviru AVG

2 Návrh řešení

Program (spořič obrazovky) po spuštění zkontroluje, kdy byl naposled spuštěn a zjistí, zda poslední test proběhl do konce. Pokud poslední antivirová kontrola nedoběhla kompletní a uplynulá doba mezi těmito spuštěními není větší než jeden den, program si načte pozici, kde minulý test skončil a pokračuje v testu. Jinak pustí celý test od začátku.

V nastavení spořiče je možné nastavit akce při nalezení škodlivého software nebo viru. Nabízeny jsou tři možnosti – vyléčit, smazat a upozornit později. Při zvolení „vyléčit“, se spořič pokusí za pomocí AVG API soubor vyléčit, při zvolení „smazat“ se jej pokusí vymazat. Pokud jakákoliv z těchto akcí nepůjde provést, po ukončení spořiče se spustí systémový dialog, který uživatele informuje o nezdaru a nabídne další možné řešení. Při možnosti „upozornit později“ si program zaznamená všechny podezřelé soubory a po ukončení se uživatele zeptá, jak se soubory zacházet.

Samotný grafický výstup spořiče je pouhé ztmavnutí obrazovky. Je to především z důvodu rychlejšího průběhu testu, než v případě, že se na obrazovku vykreslují např. grafické ornamenty. Samotný test jako další vlákno, aby byl spořič stále aktivní a mohl se kdykoliv ukončit.

Nutností je také zohlednění režimů napájení. Pokud je počítač napájen z baterií, není ve většině případů žádoucí, aby spořič, který bude výrazně zatěžovat procesor a diskové jednotky, plýtval zbytečně energií. Proto při zjištění napájení z baterií nebude spuštěn antivirový test, pouze ztmavne monitor.

2.1 Specifikace požadavků

V této kapitole se budeme věnovat specifikaci požadavků na vytvořený spořič obrazovky s antivirovou kontrolou.

Výstupem bude aplikace s příponou *.scr*, tedy formát spořiče obrazovky. Spořič by měl co nejefektivněji skenovat disk antivirovým systémem, a proto by neměl zatěžovat zbytečně systém grafickým výstupem na obrazovku.

- Schopnost pozastavit a obnovit test, pokud kontrola celého počítače neproběhne kompletně
- Zohlednit režim napájení
- Nastavit parametry testu

Pozastavení a následné obnovení testu je potřebné zejména po ty uživatele, kterým spořič obrazovky pravidelně neběží dostatečně dlouhou dobu na proskenování všech pevných disků. Díky této

schopnosti bude testování po dalším spuštění pokračovat na místě, kde minulý test skončil. Docílí se tím, že nakonec budou otestovány všechny soubory.

Zohledněním režimu napájení se má na mysli především zabránění zbytečného plýtvání energií při provozu počítače na baterie. Při zjištění provozu na baterie se nebude provádět antivirový test.

Nastavení parametrů testu musí probíhat přes klasické dialogové okno „Nastavení“ spořiče obrazovky. Parametry testu se mají na mysli akce provedené po nalezení infekce.

2.1.1 Použité prostředky

Jako programovací jazyk jsem zvolil C++, protože samotné aplikační rozhraní AVG API je napsáno v tomtéž jazyce. Jako framework jsem zvolil MFC (Microsoft Windows Foundation), které umožňuje efektivnější programování spořiče obrazovky. Vývojové prostředí jsem zvolil Microsoft Visual Studio .Net 2003, protože mi firma AVG Technologies CZ poskytla knihovnu s aplikačním rozhraním AVG API přímo pro toto vývojové prostředí. Pro testování jsem použil operační systém Microsoft Windows a to ve dvou nejpoužívanějších verzích – XP a Vista.

2.1.2 AVG API

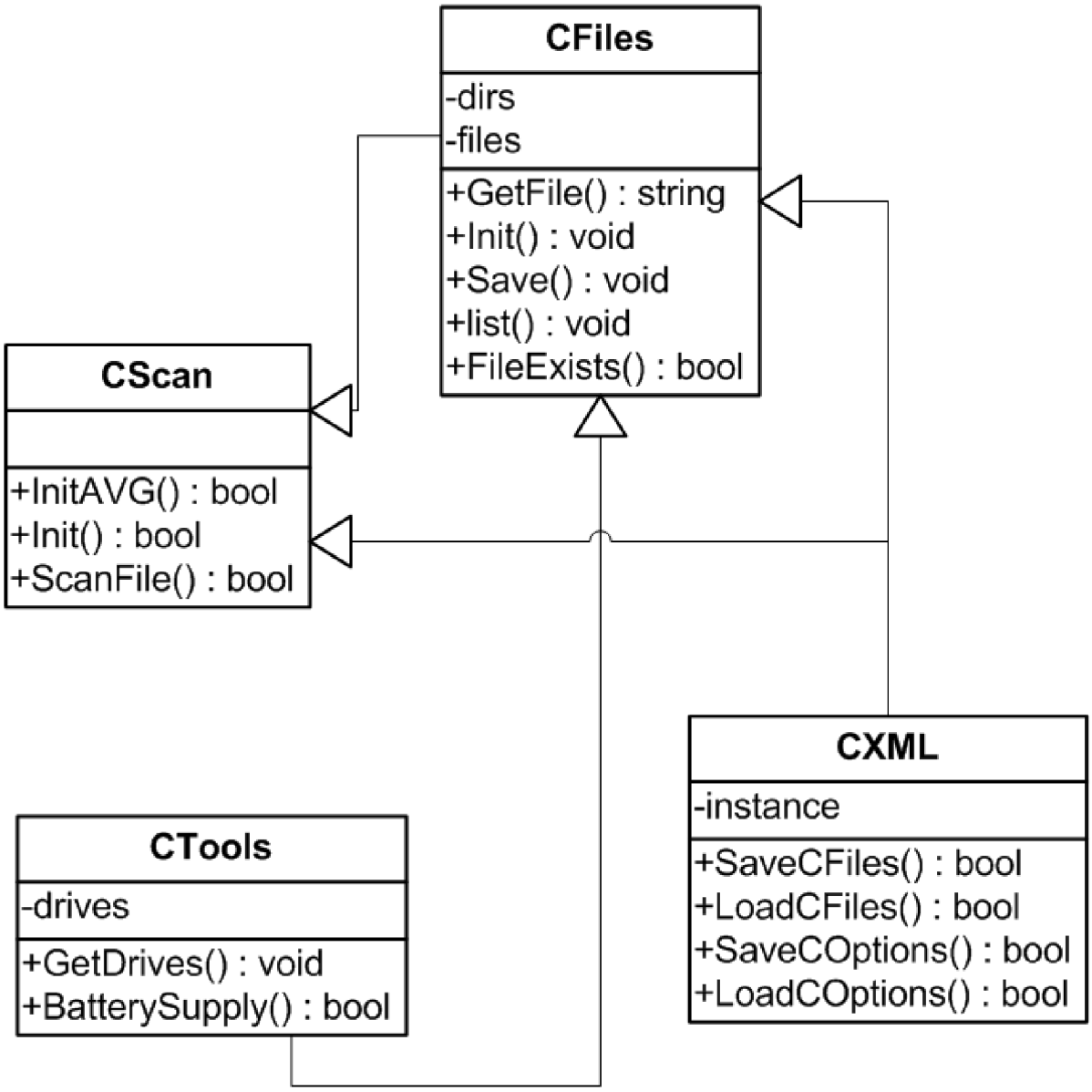
K antivirovému testu se využívá programové rozhraní AVG – AVG API. Rozhraní bylo poskytnuto firmou AVG Technologies CZ. Jedná se o knihovny napsané v programovacím jazyce C++, které slouží k zpřístupnění testovacího rozhraní antiviru AVG. Ke správné funkčnosti je nutno mít na počítači nainstalovanou funkční verzi AVG od verze 6 až po verzi 7.5.

2.1.3 Návrh tříd

Program bude programován pomocí metod objektově orientovaného programování. Objektově orientované programování, zjednodušeně OOP, klade velký důraz na návrh tříd. Program se bude dělit na tři části:

- Zobrazení spořiče obrazovky
- Dialogové okno s nastavením spořiče
- Skenování souborů

Navrhované třídy pro proces skenování:



Obr. 4 - Zjednodušený diagram tříd pro skenovací proces

CFiles

Třída, která se budu starat o systematické procházení diskové struktury. Její hlavní veřejná metoda GetFile() bude vracet název souboru ke skenování.

CScan

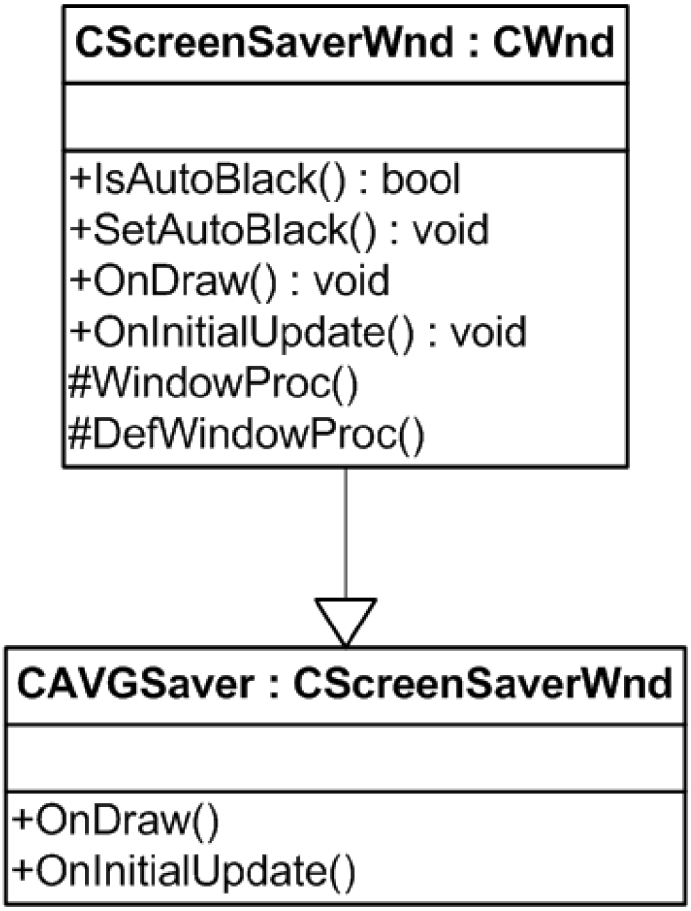
Tato třída by měla obstarávat skenování souboru. Po vytvoření by měl zajistit propojení s antivirem AVG, poté jí budou předávány pouze soubory k otestování.

CTools

V této třídě se budou nacházet funkce ke zjištění seznamu místních pevných disků a funkce zjišťující režim napájení. Tato třída se bude volat v jiných třídách.

CXML

Důležitá třída pro práci s XML. Umožňuje načítání a ukládání nastavení a obsahu zásobníků.



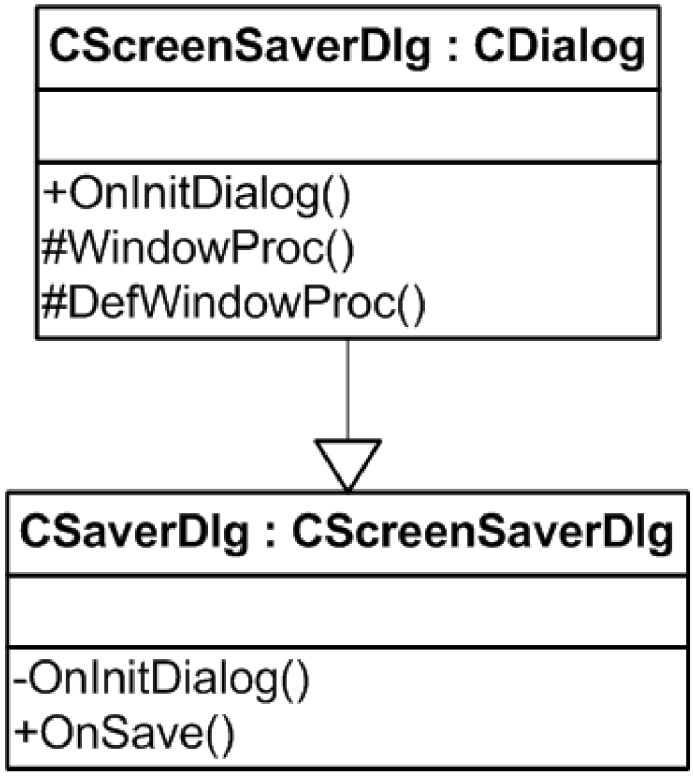
Obr. 5 - Zjednodušený diagram tříd spořiče

CScreenSaverWnd

Základní třída spořiče, zajišťuje základní funkčnost spořiče obrazovky jako je reakce na systémové signály, na ukončení po stisku klávesy nebo pohybu myši.

CAVGSaver

Tato třída je odvozená od CScreenSaverWnd, obsahuje především funkce k vykreslování na obrazovku



Obr. 6 - Zjednodušený diagram dialogu nastavení

CScreenSaverDlg

Základní třída dialogu, odvozená od třídy MFC CDialog. V této třídě se nachází standardní funkce pro obsluhu událostí na formuláři a systémové události.

CSaverDlg

Odvozená třída od CScreenSaverDlg. Obstarává vykreslení prvků na formulář, načtení a zobrazení hodnot z konfiguračního souboru a uložení konfigurace.

2.2 Režim napájení

Při spuštění programu se zkontroluje režim napájení. Pokud bude zjištěn režim napájení – baterie, nebude vůbec spuštěna antivirová kontrola. Spořič pouze způsobí zčernání obrazovky a dále bude nečinný. Kontrola je velice náročná na hardware, a proto není žádoucí, aby při provozu počítače na baterie se kontrola prováděla. Způsobovala by pouze zbytečné vybíjení baterií. Pokud režim napájení ukáže napájení z elektrické sítě, spustí se test normálně.

2.3 Formát konfiguračního souboru

Konfigurační soubor je standardní XML soubor obsahující nastavení nutné pro běh programu. Standardní umístění souboru je v adresáři s instalací spořiče (typicky to bývá v adresáři

windows/system32) a jeho název je *cdata.xml*. Díky tomu, že je to standardní XML formát, je možné soubor měnit i ručně libovolným textovým editorem.

Příklad konfiguračního souboru:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<Data>
      <default action=“heal“ />
      <newtest afterdays=“1“ />
</Data>
```

Element „default“ s atributem „action“ může nabývat tří hodnot:

- **heal** - určuje, že se po nalezení viru pokusí infikovaný soubor vyléčit
- **delete** - určuje, že soubor bude smazán
- **later** - soubor bude přesunut do schránky pro pozdější případné rozhodnutí, co se souborem udělat

Element „newtest“ a jeho atribut „afterdays“ obsahuje číselnou hodnotu, která určuje maximální dobu mezi opakovanými spuštěními testu, kdy se bude v testu pokračovat. Pokud je tato doba překročena, spustí se tet opět od začátku.

2.4 Formát souboru s pozastaveným testem

V tomto případě se jedná o soubor, obsahující poslední obsah zásobníků a čas ukončení testu. Standardní umístění souboru je v adresáři s instalací spořiče (typicky to bývá v adresáři windows/system32) a jeho název je *filesdata.xml*. Soubor je ve formátu XML a je možné jej sice měnit ručně, ale obecně to nedoporučuji, protože nesprávné nastavení hodnot, může vést k nestabilitě celého programu.

Příklad souboru s uloženými zásobníky:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<Data>
      <TestingTime unixtime=“10281281“ />

      <Files>
            <FileList Count=“3“ />
            <File Name=“C:\test.a“ />
            <File Name=“C:\test.b“ />
            <File Name=“C:\test.c“ />
      </Files>

      <Directories>
```

```
            <DirList Count=“2“ />
            <Directory Name=“C:“ />
            <Directory Name=“D:“ />
      </Directories>
</Data>
```

Element „TestingTime“ obsahuje atribut „unixtime“. V tomto atributu najdeme čas ukončení posledního testu v Unixovém časovém formátu. Tento časový formát udává počet sekund uplynulých od 1. ledna 1970.

V elementu „Files“ najdeme další dva typy elementů. První z nich je „FileList“ s atributem „Count“. Ten udává počet uložených souborů v následujícím seznamu. Další je „File“ s atributem „Name“, ten má v sobě uložený název souboru spolu s absolutní cestou.

V elementu „Directories“ se nachází obdobné elementy jako u elementu předchozího. První, „DirList“ a jeho atribut „Count“ určuje počet uložených adresářů. Následují ho elementy „Directory“ s atributem „Name“, které obsahují celé názvy adresářů.

3 Implementace

Programovací prostředí jsem použil Visual Studio .NET 2002 a překladač jazyka C++ *cl* verze 1.0.3705, který je součástí tohoto vývojového prostředí.

3.1 Struktura programu

Program se bude skládat ze tří logických částí. První část bude jednoduchý spořič obrazovky, druhá část dialog a obsluha dialogu pro nastavení vlastností spořiče a poslední část bude samotná antivirová kontrola. Po spuštění spořiče se vytvoří nové vlákno, ve kterém bude běžet antivirová kontrola. Tomuto vláknu bude nastavená druhá nejvyšší možná priorita. Kdyby se mu nastavila nejvyšší priorita, mohlo by se stát, že při pohybu myší nebo po stisknutí klávesy, by uživatel musel čekat delší dobu na vypnutí spořiče. Testování souborů by mělo vyšší prioritu než ukončení aplikace. Při ukončení programu se vlákno zruší.

3.2 Proces spořiče

Proces spořiče po spuštění ihned zahájí inicializaci AVG API, které se pokusí spojit s existující instalací Antiviru AVG. Poté vytvoří vlákno, ve kterém se bude odehrávat testování souborů. Třídy spořiče jsou zděděny od existujících tříd pro vytvoření spořiče ve frameworku MFC, proto přebírají standardní chování těchto programů. Mezi převzaté chování patří například reakce:

- systémové události
- na pohyb myší
- stlačení libovolné klávesy

Na tyto události reaguje, jako většina spořičů obrazovky, ukončením.

3.3 Proces skenování

Bude spuštěn v samostatném vlákně, takže může v nekonečné smyčce načítat soubory z disku a předávat jej antiviru k otestování. Při inicializaci se ovšem načte konfigurační soubor a podle něj se bude chovat při nalezení nákazy. Zároveň zkontroluje, zda při posledním spuštěním proběhla kontrola kompletně. Pokud proběhla kompletní, spustí se kontrola od začátku - tj. znovu zkontroluje všechny soubory na pevných discích v daném počítači. Pokud zjistí, že poslední kontrola neproběhnula celá, ověří, jestli doba uplynulá mezi minulou kontrolou a aktuální kontrolou není delší než doba nastavená konfiguračním souboru. Jestliže je má test pokračovat v předchozím umístění, načte se XML soubor

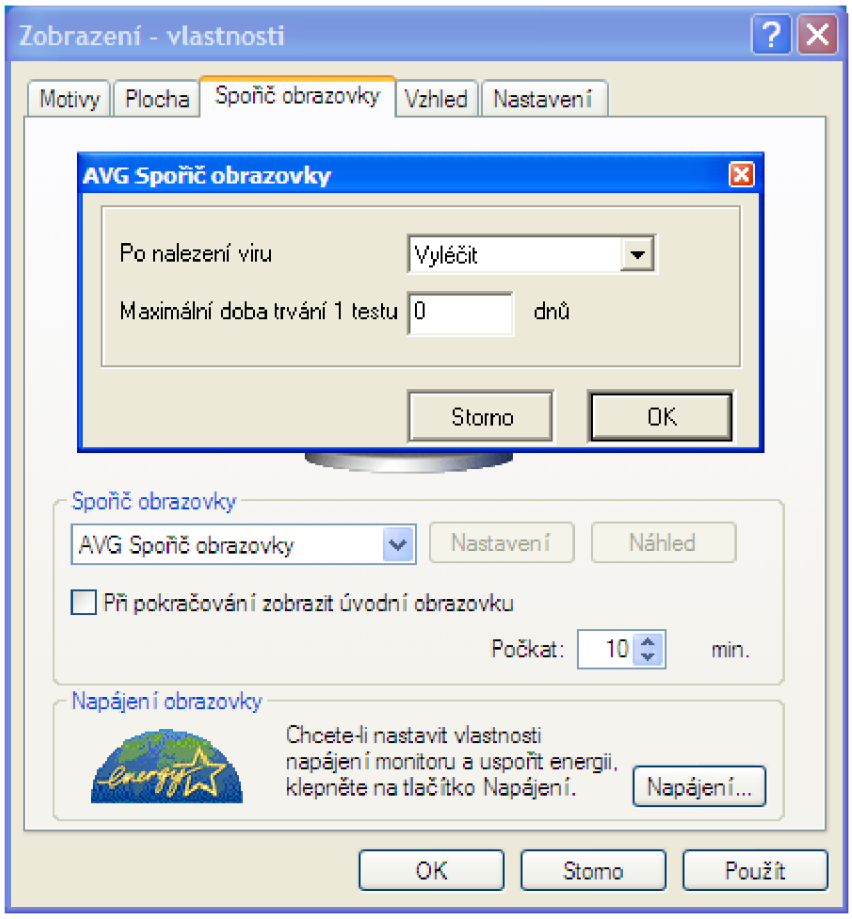
filesdata.xml a provede se ověření existence uložených souborů. Poté se načtené údaje o adresářích a souborech uloží do zásobníku v případě adresářů a do fronty v případě, že se jedná o soubory. V okamžiku, kdy dojde k otestování celého počítače, se vlákno samo ukončí.

3.4 Nastavení spořiče

Nastavení spořiče probíhá přes dialogové okno, které se vyvolá při zvolení nastavení u výběru spořiče. Přes toto okno se nastavuje, jak se program má chovat pokud nalezne virus nebo škodlivý software a maximální doba trvání jednoho testu.

U chování při nálezu viru máme na výběr tři možnosti. První možnost nám nabízí okamžité vyléčení. Při zvolení této možnosti se program po nalezení viru bude snažit nakažený soubor ihned vyléčit za pomocí všech jeho dostupných prostředků. Druhá možnost – smazat, po nalezení virové infekce, nakažený soubor smaže z disku tak, aby nemohl být obnoven. Poslední možnost nabízí využití virového trezoru antiviru AVG. Při nálezu přesune nakažený soubor do virového trezoru. Soubor se z disku odstraní a není s ním možné tedy dále pracovat. Pokud bychom se chtěli soubor okusit soubor později ručně vyléčit, obnovit nebo dokonce vymazat, musíme spustit program Virový trezor, který se dodává jako součást antivirového systému, který nám soubor k těmto akcím zpřístupní.

V dialogovém okně je také možné nastavit maximální dobu trvání jednoho testu. Tato možnost je tu zejména pro ty uživatele, kterým neběží denně spořič dostatečně dlouho dobu, aby se stihnula provést kompletní kontrola disku. U těchto uživatelů by se po každém spuštění spořiče začala skenovat stejná část disku. Zato ostatní části disku by zůstaly neotestované a tento stav není žádoucí. Proto je možné nastavit maximální dobu trvání testu například na dva dny. Po každém spuštění se porovná aktuální čas s časem minulého testu, načtou se zásobníky se skenovacími daty z minulého testu a pokračuje se v testování na místě, kde se v minulém testu skončilo.



Obr. 7 - Příklad dialogu nastavení

3.5 Instalace do systému

Jako instalátor do systému byl zvolen jednoduchý dávkový soubor BAT. Spořič se nemusí instalovat, ani ve windows nijak registrovat. Jako spořič se automaticky berou všechny soubory s příponou *.scr*, které ovšem musí splňovat ještě další požadavky, nacházející se v adresáři s instalací windows, v pod-adresáři system32. Instalační skript tedy pouze zkopíruje zkompilovanou verzi vytvořeného programu do již zmíněného adresáře.

Ukázka skriptu:

```
@echo off
copy AVGSS.scr %WINDIR%\system32 /Y
```

3.6 Odstranění ze systému

Jak je již zmíněno výše, spořič je nainstalován pouze tak, že existuje soubor v určitém umístění. Odinstalační skript tedy smaže program z umístění v adresáři s instalací windows a smaže konfigurační soubory vytvořené programem.

Ukázka skriptu:

```
@echo off
del %WINDIR%\system32\AVGSS.scr
del %WINDIR%\system32\cdata.xml
del %WINDIR%\system32\filesdata.xml
```

4 Závěr

V rámci této bakalářské práce byl vytvořen spořič obrazovky s antivirovou kontrolou, který má sloužit k menší zátěži počítače antivirovými testy v době, kdy uživatel s počítačem pracuje. Dle zadání bylo použito aplikační rozhraní AVG API, které zajistí využívání skenovacích funkcí antiviru AVG.

Při vývoji aplikace bylo dbáno zejména na efektivnost testování souborů a případné rychlosti odezvy na ukončení aplikace. Tohle bylo docíleno především přeložením testování souborů do samostatného vlákna.

Z pohledu dalšího vývoje by mohl být systém více spojen či začleněn do AVG. Mohlo by přibýt pokročilejší nastavení parametrů testů pro pokročilé uživatele. Také možnost spuštění jiného grafického spořiče a mezitím testovat soubory by se mohla líbit uživatelům. Tuto možnost momentálně (duben 2008) nabízí spořič obrazovky od antiviru Avast.

Vypracováním zadání jsem si utvrdil své znalosti při vývoji aplikací v programovacím jazyce C++ za použití MFC. Rozšířil jsem si dosavadní znalosti a získal nové poznatky z oblasti využívání objektově orientovaného programování a návrhových vzorů.

Literatura

[1] WWW stránky. Wikipedia – Screen Saver
`http://en.wikipedia.org/wiki/Screensaver`
Dostupná v dubnu 2008

[2] Pecinovský Rudolf. *Návrhové vzory*. Brno. Computer Press, a.s. 2007

[3] WWW stránky. Viry.cz
`http://viry.cz`
Dostupná v dubnu 2008

[4] WWW stránky. Wikipedia – Návrhový vzor
`http://cs.wikipedia.org/wiki/Návrhový_vzor`
Dostupná v dubnu 2008

[5] WWW stránky.
`http://objekty.vse.cz/Objekty/Vzory`
Dostupné v dubnu 2008

[6] WWW stránky.
`http://encyklopedie.seznam.cz/heslo/444639-pocitacovy-cerv`
Dostupná v dubnu 2008

[7] Szor Peter. *Počítačové viry*.ZONER software s.r.o. 2006

Seznam příloh

Příloha 1: Manuálová stránka programu

Aby bylo možné program vyzkoušet, je na počítači potřeba mít nainstalován antivirový systém AVG a to ve verzi – 6, 6.5, 7 a 7.5. Poté se musí program nainstalovat do systému. Instalace probíhá přes skript *install.bat*, který je přiložen. Skript je třeba spouštět s administrátorskými právy, protože přistupuje do systémového adresáře, kam kopíruje soubory.

Po úspěšné instalaci se již může spořič vyzkoušet. Vyzkoušíme přes menu Obrazovka v záložce „Spořič obrazovky“. Zacházet je s ním možné jako s jakýmkoliv jiným spořičem obrazovky.

Odstranění ze systému se provádí skriptem *uninstall.bat*, který je nutné rovněž spustit s administrátorskými právy.

Příloha 2: Obsah přiloženého CD

Struktura dat na přiloženém CD :

- **AVGScreenSaver** – Zkompilovaný program, spolu se všemi skripty potřebnými k instalaci/odinstalaci
- **Zdrojové kódy** – Obsahuje kompletní použité zdrojové kódy, včetně projektového souboru pro MS Visual Studio.NET 2002
- **Bakalářská práce** – Tato práce uložená ve formátu PDF, DOC a DOCX
- **Dokumentace** - Programová dokumentace vygenerovaná programem Doxygen