VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

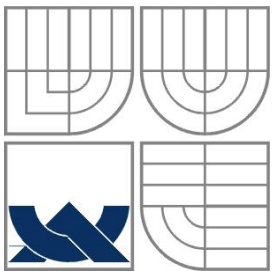
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZŠIŘUJÍCÍ MODUL PRO MS OFFICE

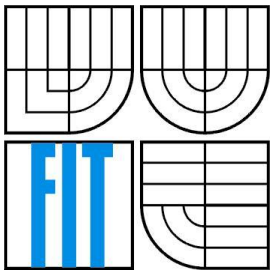
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE DAVID RUSEK
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZŠIŘUJÍCÍ MODUL PRO MS OFFICE

MS OFFICE ADD-IN MODULE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DAVID RUSEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK VAŠÍČEK, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce se zabývá problematikou přidání rozšiřujícího modulu do textového procesoru MS Word, tabulkového kalkulátoru MS Excel a nástroje pro tvorbu prezentací MS PowerPoint z kancelářské sady programů MS Office. Modul bude umožňovat komunikaci, synchronizaci a správu dokumentů se vzdáleným privátním úložištěm, které bude podporovat správu verzí. To všechno by mělo fungovat napříč verzemi sady MS Office 2003 (11), 2007 (12) a 2010 (14), případně i 2013 (15). Implementace je realizována pomocí jazyka C# z platformy .NET a PIA (Primary Interop Assemblies), které přidávají podporu pro manipulaci s aplikacemi a objekty sady MS Office.

Abstract

This work deals with the issue of addition of extension module to the text processor MS Word, table calculator MS Excel and tool for presentation creation MS PowerPoint from office set of programs MS Office. Module will support communication, synchronization and management of documents in remote and private storage, with support for version management. This all should work across different versions of MS Office application set. These are MS Office 2003 (11), 2007 (12) a 2010 (14), possibly even 2013 (15). Implementation is realized using programming language C# from .NET platform and PIA (Primary Interop Assemblies), which adds support for manipulation with applications and objects from MS Office application set.

Klíčová slova

Microsoft Office, MS Office, rozšiřující modul, add-in, podpora verzovaní, Subversion, Svn, .NET, Synchronizace dokumentů

Keywords

Microsoft Office, MS Office, extension module, add-in, versioning support, Subversion, Svn, .NET, Document synchronization

Citace

David Rusek: Rozšiřující modul pro MS Office, bakalářská práce, Brno, FIT VUT v Brně, 2013

Rozšiřující modul pro MS Office

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Zdeňka Vašíčka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
David Rusek
6.5.2013

Poděkování

Na tomto místě bych rád poděkoval Ing. Zdeňkovi Vašíčkovi, Ph.D. za odporné rady, vedení a konzultace během práce na této bakalářské práci.

Obsah

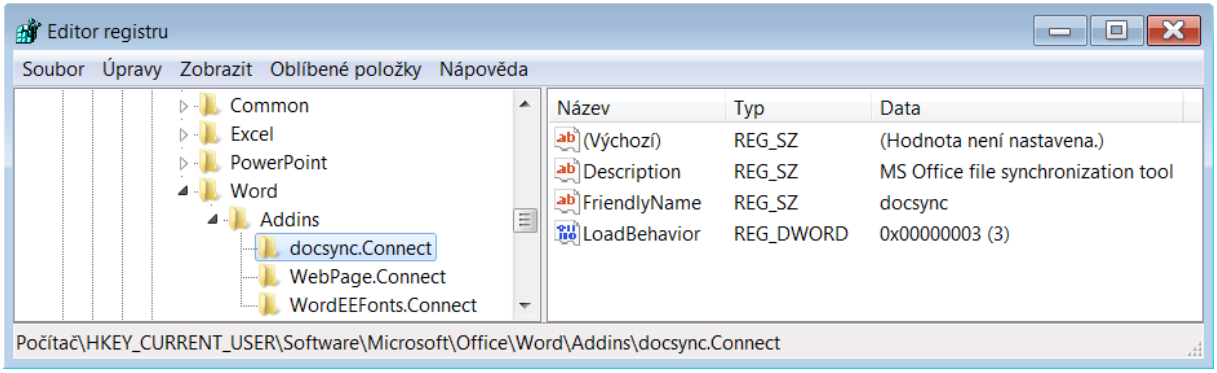
1 Úvod

Microsoft Office (dále také MS Office) je bezesporu jednou z nepoužívanějších kancelářských programových sad vůbec. Tato sada umožňuje tvorbu textových dokumentů v textovém procesoru Microsoft Word, prezentací v nástroji MS PowerPoint, tabulkových kalkulátorů v MS Excel a spoustu dalšího. Nicméně je známou skutečností, že v dnešní době nestačí pouze kvalitní nástroje pro tvorbu kvalitních dokumentů či jiných dat. Velice důležitá je také spolehlivá správa verzí a jejich efektivní šíření (distribuce). To vedlo ke vzniku například Microsoft Office modulu Google Cloud Connect[1][2], který umožňuje automatickou synchronizaci dokumentů z již zmíněných aplikací MS Word, MS Excel a MS PowerPoint a správu verzí. Tento modul však přináší také nevýhody a nejvýraznější z nich je nemožnost kontroly, kde jsou data uložena a velice častá skutečnost, že majitel úložiště má k takovým datům přístup. Google Cloud Connect, jak je již v názvu napovězeno, ukládá data do některého z datových úložišť společnosti Google připojených do Cloudu. Pokud by chtěl mít uživatel nad daty větší kontrolu, je Google Cloud Connect nepoužitelný a právě výstup tohoto bakalářského projektu by měl být alternativní modul pro MS Office, tento problém řešící.

Práce je dělena následovně. První kapitola technické zprávy se snaží přiblížit problematiku programování modulů pro MS Office a jaké přístupy a technologie je možno při jejich vývoji zohlednit. Stejně tak požadavek, aby výsledný modul pracoval s úložištěm podporujícím verzování, si zaslouží zvláštní kapitolu, zvažující jaké možnosti v této oblasti jsou k dispozici. Po úvodu do problematiky se technická zpráva zabývá již klasickými tématy jako je návrh a implementace modulu, následováno popisem způsobu testování.

2 Možnosti rozšiřování MS Office

Funkcionalitu MS Office lze rozšiřovat pomocí takzvaných modulů (add-ins), pro jejichž tvorbu MS Office od verze 2003 podporují jednotnou architekturu takzvaných COM Add-ins nebo-li Microsoft Component Object Model Add-ins. Výsledným souborem je dynamicky linkovaná knihovna, která se registruje v operačním systému Windows. Poté co se knihovna modulu registruje (nainstaluje), může ji uživatel ručně v MS Office aktivovat pomocí nabídky pro správu modulů a nebo je možné ji aktivovat již během instalace přidáním několika údajů do registru operačního systému Windows, což umožňuje například Windows MSI Instalátor, který umí Visual Studio vygenerovat (ve verzi Visual Studia 2012 už není MSI Instalátor podporován).

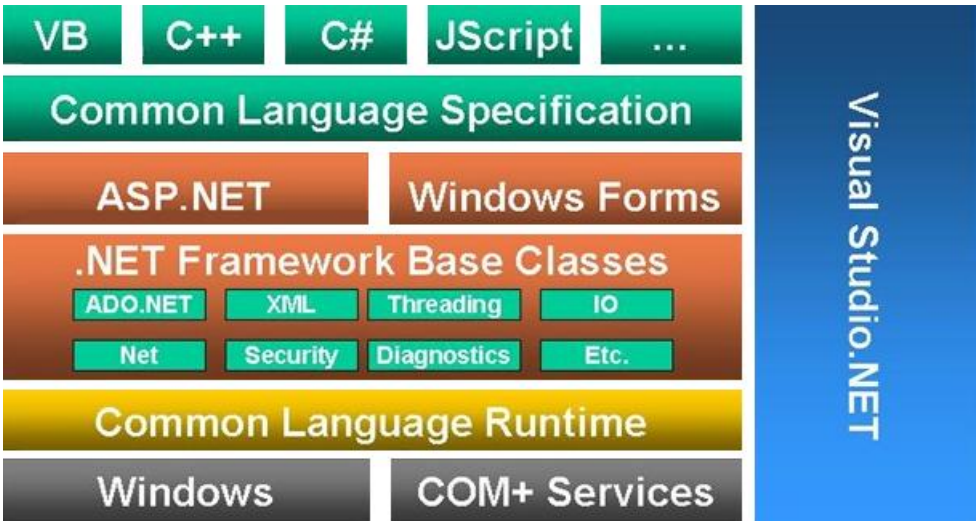


Obrázek 1: Nastavení modulu pro automatické načítání v MS Word textovém procesoru.

Položky Description a FriendlyName, jenž jsou vidět na obrázku 1, slouží k popisu a identifikaci modulu v nabídce správy modulů v MS Office aplikacích a položka LoadBehavior definuje způsob načítání modulu, kde hodnota tři znamená automatické načítání při spuštění (Load at startup). Celkově existuje sedm různých způsobů načítaní, zahrnujících kromě zřejmých možností jako "Do not load automaticaly" (hodnota 0 a 1 s menšími obměnami) tak i možnosti jako "Load on demand" nebo "Load first time, then load on demand". Poslední zmíněná možnost například znamená, že se načte uživatelské rozhraní modulu, ale samotný modul - tedy jeho funkcionalita - se načte až v okamžiku interakce uživatele s nějakou součástí uživatelského rozhraní modulu (podrobnější popis je dostupný na [18]).

Pro implementaci funkcionality modulů jsou pak k dispozici takzvané Primary Interop Assemblies (také uváděné pod zkratkou PIA), které tvoří spravovanou (managed) "obálku" nad nespravovaným (unmanaged) COM API, pomocí kterého jsou MS Office aplikace implementovány. Díky tomu umožňují implementaci MS Office modulů v CLI (Common Language Infrastructure) kompatibilních jazycích z .NET frameworku.

Microsoft .NET framework, je programová platforma vytvořená firmou Microsoft a běžící primárně nad operačním systémem Microsoft Windows. Tato platforma zahrnuje více než dvacet vysoko úrovňových programovacích jazyků, jenž je všechny možné převést do takzvaného Intermidiate Language (IL), který je proveditelný pouze pod správou CLR (Common Language Runtime) virtuálního stroje, jenž zajišťuje kontrolu typů, hranic polí a indexů, obsluhu výjimek a správu paměti. Také však umožňuje aby kód napsaný v jednom jazyce bylo možné použít v jiném.



Obrázek 2: Architektura .NET frameworku.

Microsoft .NET framework také zavádí výše zmíněné pojmy spravovaný a nespravovaný kód. Spravovaný kód je kód napsaný právě pomocí některého z jazyků .NET platformy, mezi které patří například C#, J#, .NET Visual Basic, .NET JScript a C++ (C++ umožňuje generovat jak spravovaný tak nespravovaný kód). Všechny tyto jazyky pak sdílejí jednotnou sadu knihoven tříd a mohou být převedeny do již zmíněného Intermidiate Language. Termín spravovaný kód pak označuje skutečnost, že je tento kód prováděn *pod správou* virtuálního stroje (CLR).

A právě moduly pro MS Office je možno implementovat pomocí jazyků z .NET platformy. Nicméně zadání bakalářské práce požadovalo použití jazyků C++ či C#, ze kterých jsem nakonec pro implementaci zvolil druhý jmenovaný a to hned z několika důvodů:

- Obsahuje Garbage collector.
- Visual Studio ve spojení s C# umožňuje jednoduší správu Ressources.
- Větší část internetových zdrojů zabývajících se tvorbou MS Office modulů používá C#.
- Chci se v používání C# zlepšit.

Jako editor pro implementaci jsem chtěl původně použít Microsoft Visual Studio 2012, bohužel tady se ukázalo, že verze 2012 nepodporuje projektovou šablonu "Shared Add-in" (součást Extensibility), tedy šablonu pro tvorbu MS Office modulů kompatibilních pro několik MS Office aplikací současně. V mém případě tedy pro MS Word, Excel a PowerPoint. Naopak, verze 2012 obsahuje šablony jen pro jednotlivé aplikace, takže ve výsledků bych musel implementovat tři prakticky stejné knihovny. Pravděpodobně by bylo možné Visual Studiu 2012 tuto šablonu nějakým nestandardním postupem přidat, nicméně jsem měl k dispozici Visual Studio 2010, se kterým již mám zkušenosti a především, které "Shared Add-in" šablonu podporuje. Shared Add-in šablona je výhodná i z toho důvodu, že automaticky obsahuje šablonu pro vygenerování zaváděcích souborů typů exe a msi, z čehož právě druhý jmenovaný je vhodný pro registraci modulu a pro připojení knihoven, mezi které patří i knihovna SharpSvn (viz. následující kapitoly).

Pro implementaci modulů však samotné PIA nestačí, protože PIA nezahrnuje možnost, jak takový modul do MS Office aplikací připojit. Prvním a zároveň také nejefektivnějším řešením je použití Add-in Express[4], což je komerční placený nástroj zahrnující PIA, který přidává projektovou šablonu a

běhové knihovny do MS Visual Studia. Umožňuje tak tvorbu modulů pro MS Office kompatibilní skrze jejich různé verze (a pomocí kterého byl například implementován již zmíněný Google Cloud Connect). Také poskytuje celou řadu pomocných nástrojů jako je například editor pro tvorbu Ribbon panelů, které se jinak musí vytvářet ručně pomocí značkovacího jazyka XML. Jednou z výhod toho nástroje je i poměrně velké množství dostupných internetových materiálů, které však ve většině případů opomínají zmínit, že Add-in Express používají.

Kromě placeného Add-in Express, existují pouze dvě možnosti jak modul do MS Office aplikací připojit. První možností je použít Visual Studio Tools for Office (dále také VSTO), což je rozšiřující nadstavba pro PIA. Druhé řešení pak zahrnuje použití rozšiřujících rozhraní IDTExtensibility2 a IRibbonExtensibility.

2.1 Rozšíření pomocí VSTO

VSTO je sada nástrojů dostupná v podobě projektové šablony pro Visual Studio a runtime knihoven jež umožňují snadnější implementaci rozšiřujících modulů pro MS Office v programovacích jazycích z rodiny CLI z .NET platformy za pomoci právě PIA. V podstatě se tedy jedná o volitelné rozšíření PIA. Nicméně problém nastává při kompatibilitě jednotlivých verzí MS Office s verzemi VSTO a

Tabulka 1: kompatibilit VSTO napříč různými verzemi MS Office a .NET frameworku. Podrobnější informace jsou dostupné na [9].

Verze VSTO	Office 2003	Office 2007	Office 2010	.NET verze	Verze Visual Studia
2003	ano	-	-	1.1	2003 VSTO
2005	ano	-	-	2.0, 3.0, 3.5	2005
2005 SE	ano	ano	-	2.0, 3.0, 3.5	2005+
3.0	ano (2005 SE)	ano	-	3.5	2008+
4.0	-	ano	ano	3.5, 4.0	2010+

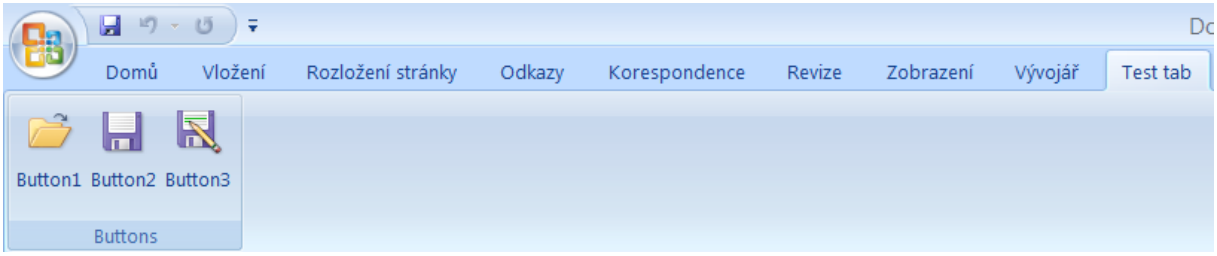
.NET frameforku. VSTO není totiž vázané na verzi .NET frameworku, ale na verzi Microsoft Visual Studia - což znázorňuje tabulka 1. To znamená, že pro MS Office modul určený pro verze 2003 a 2007 bych mohl použít Visual Studio 2008 případně 2005 a pro verze 2007 a 2010 pouze Visual Studio 2010. Ve výsledku bych tedy musel pravděpodobně implementovat nejméně dvě verze modulu abych pokryl zadanou podporu pro verze 2003, 2007 a 2010. S tím, že pokud bych chtěl implementovat podporu i pro verzi 2013 musel bych použít Visual Studio 2012, které nemusí podporovat MS Office 2007.

2.2 Rozšíření pomocí rozhraní IDTExtensibility2 a IRibbonExtensibility

Druhou možností je použít PIA pro manipulaci s MS Office objekty v kombinaci s IDTExtensibility2 rozhraním z jmenného prostoru Extensibility (definovaného v Msaddndr.dll - Microsoft Add-in Designer type library) jenž definuje následujících pět povinný metod pro definici obsluhy událostí:

- **OnConnection** je nejdůležitější a zároveň nejužitečnější metoda definující obsluhu události OnConnection, která umožňuje definovat požadované operace, jenž se provedou hned poté, co je modul úspěšně načten do některé z aplikací MS Office. Například tedy umožňuje načíst a zobrazit změnu UI pro MS Office 2003, kterou potřebuji.
- **OnDisconnection** a **OnBeginShutdown** jsou události převážně určeny k uvolnění potřebných zdrojů předtím než se modul či celá aplikace ukončí.
- **OnStartupComplete** obsluha je provedena poté co je vykonán OnConnection událost, tedy možnost jak provést operace po inicializaci modulu.
- **OnAddInUpdate** metoda definuje chování, které se vykoná, kdykoliv je připojen či odpojen nějaký rozšiřující modul. Pro potřeby modulu se jedná o nezajímavou událost.

A jelikož výsledný modul musí fungovat i na verzích MS Office 2007 a 2010, musíme vedle IDTExtensibility2 přidat ještě podporu pro modifikaci Ribbon nabídky, kterou samotné PIA v základní verzi nezahrnuje. Rozhraní IRibbonExtensibility z jmenného prostoru Microsoft.Office.Core pro tento účel definuje jedinou metodu GetCustomUI, jež vrací řetězec obsahující definici Ribbon nabídky v jazyce XML. Příklad načteného modulu v MS Office 2007 můžete vidět na obrázku 3, jehož definice uživatelského rozhraní je k dispozici na obrázku 4.



Obrázek 3: Příklad načteného modul pro MS Office 2007.

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="OnRibbonLoad">
  <ribbon>
    <tabs>
      <tab id="TestTab" label="Test tab">
        <group id="group" label="Buttons">
          <button id="bt_Open" label="Button1" size="large" onAction="ShowMessageBox" imageMso="FileOpen"/>
          <button id="bt_Save" label="Button2" size="large" onAction="ShowMessageBox" imageMso="FileSave"/>
          <button id="bt_SaveAs" label="Button3" size="large" onAction="ShowMessageBox" imageMso="FileSaveAs"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Obrázek 4: Ukázka definičního souboru pro záložku na obrázku 3.

3 Správa konfigurace software

Software configuration management (SCM) je proces zabývající se mapováním a kontrolou změn ve vývoji software. Díky tomu pak poskytuje informace jaké změny byly provedeny, kdo a kdy je provedl a proč je provedl. Tím to však nekončí. Hlavní silou SCM jsou nástroje umožňující tyto změny v případě potřeby vrátit nebo je naopak rozšířit napříč libovolným počtem uživatelů a to vše pokud možno bezpečně, efektivně a spolehlivě. Nicméně existuje celá řada různých SCM nástrojů, mezi kterými je nutno vybrat ten nejvhodnější pro účely MS Office modulu.

3.1 Git

Git je verzovací (version control) systém, původně implementován Linus Torvaldem jako alternativa k centralizovaným verzovacím systémům jako jsou CVS a SVN. Jeho hlavním a stěžejním rozdílem je, že je distribuovaný (peer-to-peer model), tedy každý uživatel má kopii repozitáře na svém vlastním počítači a nad touto kopií má plnou kontrolu, což umožňuje se zbavit celé problematiky commit oprávnění a komu je přidělit. Další výhodou lokálně přítomného repozitáře je možnost nahrávat změny i během nedostupnosti internetového připojení.

Mezi další výhody Gitu pak patří jeho výrazně vyšší rychlost, velice efektivní možnost větvení (branching), kdy například během implementace nějakého projektu je možné repozitář rozvětvit a sledovat několik různých vývojových cest. A nakonec ponechat jen tu, která přinesla neuspokojivější výsledky.

Git má samozřejmě několik dalších rozdílů, jako například rozdílný formát a struktura v jakém se soubory v repozitáři skladují či podpora datové jistoty, tedy podpora pro ověření, že soubory nejsou poškozeny.

3.2 WebDAV

WebDAV[6] (Web Distributed Authoring and Versioning) je rozšíření pro HTTP protokol, jenž umožňuje spolupráci mezi uživateli, správu a editování souborů. WebDAV přidává k metodám HTTP protokolu (GET, POST, atd.) metody jako například DELETE pro mazání souborů nebo LOCK a UNLOCK, které slouží k zamykání a odemykání souborů. Také nechybí metody pro přidávání dodatečných informací k souborům skrze metody PROPFIND a PROPPATCH.

Jednou z implementací WebDAV protokolu je například sabredav[7], jež je implementován čistě pomocí webového programovacího jazyka PHP5.

Nakonfigurovat WebDAV server není příliš složité a je poměrně jednoduché jej připojit jako síťovou jednotku podobně jako FTP. Problémem je však ta skutečnost, že se jedná v podstatě o holý protokol a implementace modulu, který by měl s takovým úložištěm komunikovat by z velké části zahrnoval pouze kontrolu a zotavení z chyb, jenž musí být precizní, aby takový modul byl dostatečně spolehlivý.

Také je tu fakt, že WebDAV sám o sobě nepodporuje verzování, třebaže podle názvu by měl. Skupina zabývající se jeho vývojem totiž zjistila, že vývoj verzování by byl příliš náročný a od tohoto záměru upustila. Jeho podpora byla přidána až pomocí rozšíření Delta-V (RFC 3253). Nicméně Sabredav toto rozšíření nepodporuje a ani jeho implementaci neplánuje[21].

3.3 Apache Subversion

Apache Subversion je centralizovaný verzovací systém založený na starším verzovacím systému CVS. Všichni uživatelé mají na svém počítači pracovní kopii centralizovaného repozitáře a nad tou mají plnou kontrolu. Změny souborů pak pomocí akce commit mohou nahrát do centrálního repozitáře nebo z něj naopak pomocí akce update změny mohou stáhnout na svůj počítač. Hrozí tady samozřejmě poměrně častý případ konfliktů, kdy dva či více uživatelů upraví jednu a tu samou verzi souboru a jeden ji pak nahraje do centrálního úložiště. Druhý uživatel však narazí, protože se snaží přepsat novější nebo stejnou verzi souboru. To se dá pak řešit buď operací merge nebo resolve, kdy se použije buď verze uživatele nebo ta již v repozitáři (tyto problémy jsou v Gitu z větší míry omezeny).

3.4 Srovnání SCM

Vlastnost	Git	Přínos pro modul	WebDAV	Přínos pro modul	Apache Subversion	Přínos pro modul
Rychlé operace a odezva	Ano	Ano	Ano	Ano	Ne	Ne
Distribuovaný systém	Ano	Ne	Ano / Ne	Ano	Ne	Ano
Větvení	Ano	-	Ne	-	Ano	-
Vlastnosti souborů	Ano	Ne	Ano	Ano	Ano	Ano
Implicitní podpora verzování	Ano	Ano	Ne	Ne	Ano	Ano
Práce offline	Ano	Ne	Ne	Ano	Ne	Ano
Jednoduché použití	Ne	-	Ne	-	Ano	Ano

Tabulka 2: Srovnání zvažovaných SCM nástrojů.

Tabulka 2 porovnává některé vlastnosti tří zmíněných SCM nástrojů, jež jsem zvažoval použít pro vývoj MS Office modulu. A zároveň hodnotí jestli daná vlastnost má přínos pro vývoj modulu.

Rychlost operací a odezvy není třeba příliš rozvádět - čím rychlejší, tím lepší. Distribuovaný systém je naopak pro účely modulu nevhodný, jelikož jedním z (v zadání nezmíněných) požadavků na modul, byla indikace, zda-li je soubor právě modifikován jiným uživatelem a to v případě distribuovaného systému není možné. Nemluvě o tom, že Git umožňuje práci offline. Ze stejného důvodu jsou záporně hodnoceny i jeho vlastnosti souboru a práce offline, čímž se Git či jakýkoliv jiný distribuovaný systém (Mercurial) prakticky vyřazuje.

Podpora větvení je důležitá především pro programátory, ale z pohledů dokumentů nepřináší až tak velkou výhodu, proto je hodnocení této položky neutrální. Zato implicitní podpora verzování je poměrně důležitá a z tohoto pohledu má WebDAV nevýhodu, jelikož Git a Svn jsou nástroje vytvořené speciálně pro tento účel. Tady stojí zmínit skutečnost, že Svn má poněkud čitelnější systém organizace verzí, tedy používá číslo revize souboru, které se při commitu nové verze inkrementuje. Git

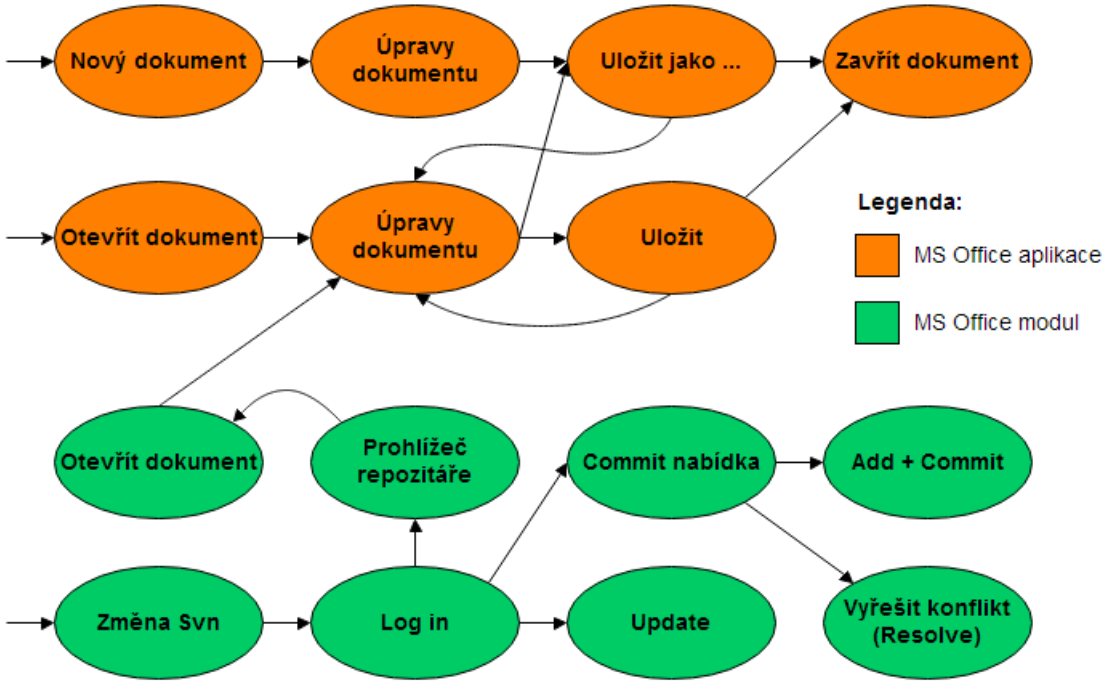
na místo toho používá hash, protože v distribuovaném systému nedává sekvenční označení příliš smysl. Uživatel tuto nepříjemnost pak muže řešit pomocí například takzvaných tagů a sám si poznačit změny.

Poslední hodnocená vlastnost je jednoduchost použití, kde vede Svn. Git má poměrně nepříznivou učební křivku a mnoho uživatelů z tohoto důvodu preferuje Svn. U WebDAV zaleží jak se tento nástroj využije, tedy je možné aby použití výsledného modulu bylo jednoduché nebo taky složité.

Srovnání pak ukazuje, že nejvíce kladů pro použití má Apache Subversion, který byl proto zvolen pro implementaci modulu.

4 Návrh řešení

4.1 Funkcionalita modulu



Obrázek 5: Vývojový cyklus použití modulu a tvorby dokumentů.

Funkcionalita modulu byla navržena tak, aby odrážela potřeby uživatele při tvorbě dokumentů - ty jsou v jejich základní posloupnosti znázorněny na obrázku 5. Nicméně návrh funkcionality modulu ještě přiblížím na reálné situaci použití modulu.

Poté co si uživatel nainstaluje modul, bude potřeba aby si zřídil Svn repozitář, přičemž výběr Svn serveru je plně v režii uživatele - pokud pracuje sám, tedy bude zároveň i správcem repozitáře. Následně pak vytvoří lokální kopii daného repozitáře na svém počítači.

Jakmile spustí jednu ze tří podporovaných MS Office aplikací, může pomocí rozhraní modulu, otevřít soubor, přičemž se zobrazí formulář, kde bude potřeba zadat přihlašovací údaje. Tyto údaje pak může uložit (údaje jsou ukládány v uživatelské složce) a příště pomocí ComboBoxu vybrat (nechybí ani možnost údaje smazat).

Po přihlášení budou ověřeny uživatelské údaje a zobrazen prohlížeč repozitáře, pomocí kterého uživatel otevře soubor, který potřebuje pozměnit nebo pouze prohlédnout. Nezapomene však zkontrolovat, zda-li je soubor používán (informace jež bude indikována v prohlížeči), aby se vyhnul případným konfliktům. Pokud by chtěl soubor pouze prohlížet, může tuto informaci ignorovat a soubor otevře.

Při otevření je soubor označen jako používaný, tak aby tuto informaci měli k dispozici ostatní uživatelé a po jeho zavření je příznak aktivity zrušen. Následně bude soubor uložen pomocí klasického uložení MS Office aplikace, načež ho může uživatel commitnout, pokud ho uložil do lokální kopie repozitáře.

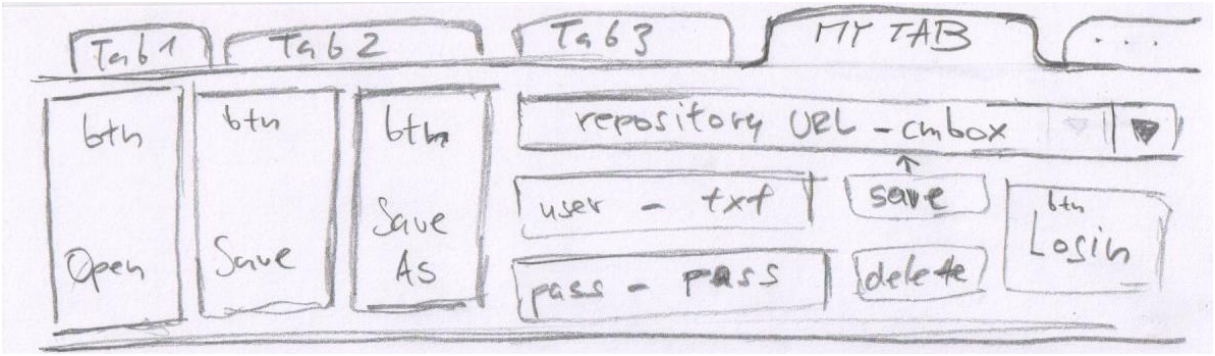
Při commitnutí se zobrazí jednoduchý seznam necommitnutých souborů (akce add a commit bude prováděna současně). Bude-li seznam obsahovat soubory v konfliktu, budou zvýrazněny a pomocí kontextového menu (stisk pravého tlačítka myši) bude konflikty možno vyřešit.

Modul si během práce bude uživatele samozřejmě pamatovat, aby se nemusel přihlašovat stále dokola. Proto v rozhraní modulu nechybí ani možnost změnit Svn za běhu aplikace (ta zastupuje i první volbu přihlašovacích údajů, která je povinná).

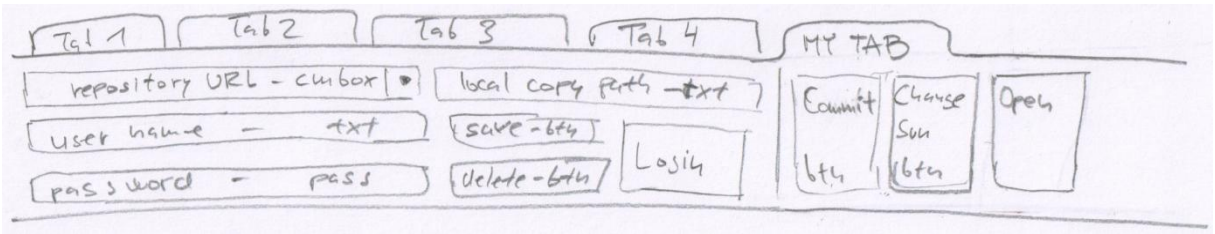
4.2 Uživatelské rozhraní

Návrh uživatelského rozhraní modulu je částečně spjat s MS Office. Verze 2007 a vyšší podporují Ribbon nabídku, která v prvotním návrhu rozhraní měla pokrýt část funkcionality modulu.

Jelikož modul pracuje s verzovacím systémem, jsou potřeba údaje jako je URL repozitáře, uživatelské jméno a heslo, pro které by byl normálně potřeba zvláštní formulář, jenž první návrhy vůbec neuvažovaly, jelikož tuto funkcionalitu přesunuly do Ribbon nabídky.



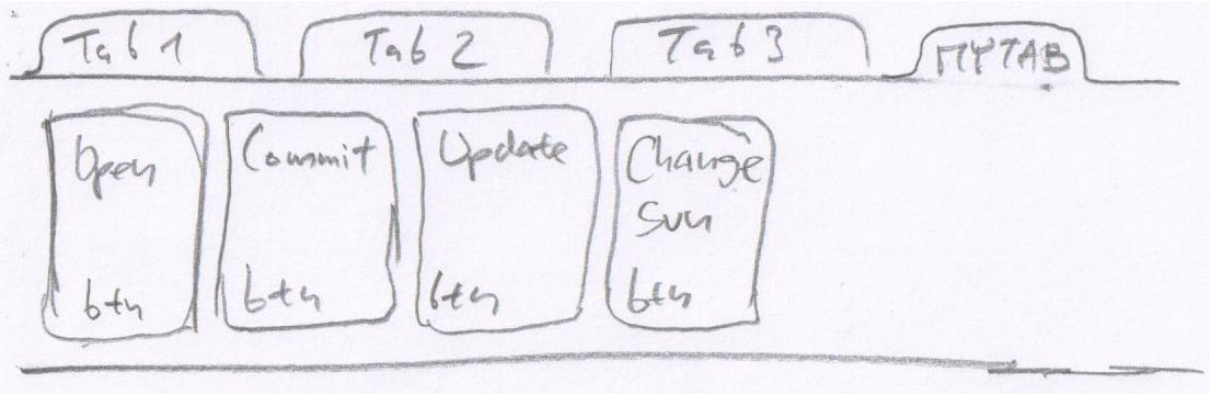
Obrázek 6: První návrh rozhraní pro MS Office 2007.



Obrázek 7: Druhý návrh rozhraní pro MS Office 2007.

Obrázky 6 a 7 znázorňují první a druhou verzi návrhu Ribbon nabídky pro MS Office 2007 (a vyšší). První návrh ještě uvažoval se situací, kdy by modul obsahoval vlastní rozhraní pro operace Open, Save a SaveAs (důvod proč tyto tři metody modul nakonec neobsahuje je vysvětlen v 5. kapitole, zabývající se implementací) a formulář pro vyplnění uživatelských údajů by byl zabudován přímo do Ribbon nabídky, což se neuskutečnilo, protože pro MS Office 2003 toto řešení není možné. Druhý

návrh pak reflektuje potřeby přidat do modulu potřebnou funkcionalitu, na kterou se v prvním návrhu zapomnělo.



Obrázek 8: Poslední návrh rozhraní pro MS Office 2007.

Poslední verze modulu na obrázku 8, reflektuje nutnost vytvořit formulář pro zadávání údajů uživatele a některé další faktory jež budou vysvětleny v kapitole 5. Implementace. Všechny prvky pro tento účel potřebné byly přesunuty do samostatného formuláře, což se později vyplatilo, jelikož PIA bez nadstavby VSTO disponuje jen velice omezenými možnostmi pro manipulaci s Ribbon nabídkou a zdaleka neposkytuje takovou kontrolu nad jejími prvky, jako klasické Windows formuláře či WPF (Windows Presentation Foundation).

Obrázek 9: Návrh uživatelského rozhraní dialogového okna prohlížeče repozitáře.

Další částí uživatelského rozhraní byl návrh prohlížeče repozitáře. Jelikož Visual Studio neobsahuje žádný formulářový prvek pro zobrazování souborové struktury, bylo nutné vytvořit dialogové okno, které bude sloužit k prohlížení obsahu repozitáře. Zároveň také bude umožňovat operace jako "Otevření souboru". Návrh okna je na obrázku 9.

5 Implementace

První část této kapitoly (podkapitoly 5.1 - 5.4) se zabývá implementací grafického rozhraní modulu. Podkapitola 5.5 rozebírá jak byl modul navázán na Svn repozitář a nakonec podkapitola 5.6 popisuje řešení některých zajímavý implementačních problémů.

5.1 Grafické rozhraní MS Office modulu

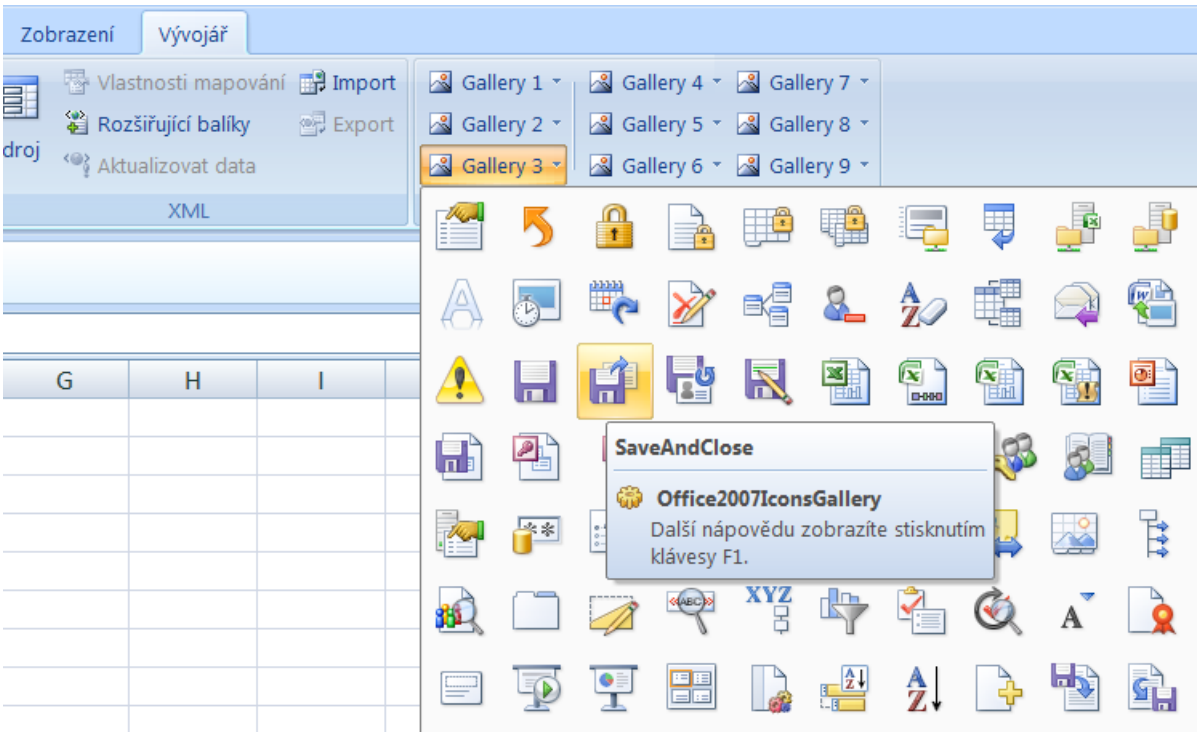
V MS Office 2007 lze přidat záložku a její Ribbon nabídku pomocí XML souboru, popisující jak taková nabídka má vypadat. Obsah tohoto souboru pak vrací funkce GetCustomUI z rozhraní IRibbonExtensibility, jenž definuje její povinnou implementaci pro třídy, které toto rozhraní dědí. Bohužel neexistuje žádná dokumentace jak takový soubor má vypadat. Jediné dostupné příklady mají nanejvýše dvě tlačítka, takže použití třeba ComboBoxu či jiných složitějších prvků není příliš jednoduché a ještě k tomu, kdykoliv je v XML sebemenší chyba, modul se prostě nenačte a MS Office ani neupozorní, že se měl nějaký modul načíst. Paralelní změna kódu a definice uživatelského rozhraní je proto více než nevhodná, jelikož pak při výskytu chyby není jasné, zda-li se vyskytla v kódu nebo uživatelském rozhraní.

Dalším již zmíněným faktem je, že MS Office verze 2003 nepodporují Ribbon nabídku a verze 2007 a vyšší mají kontrolu nad Ribbon menu poměrně omezenou, jelikož nepoužívám nadstavbu VSTO. Proto jsem se rozhodl, že samotného rozhraní MS Office (všech verzí) budu využívat co nejméně a s co nejjednoduššími prvky - především tlačítky - abych nemusel obstarávat zpětnou vazbu na uživatelské rozhraní MS Office modulu a většinu funkcionality obstarám pomocí klasických Windows formulářů, které poskytují daleko větší kontrolu nad obsahem.

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="OnRibbonLoad">
  <ribbon>
    <tabs>
      <tab id="CustomTab" label="Document Synchronization">
        <group id="gr_io" label="Input">
          <button id="bt_open" label="Open" size="large" onAction="Open_Click" imageMso="FileOpen"/>
        </group>
        <group id="gr_options" label="Svn Actions">
          <button id="bt_commit" label="Commit" size="large" onAction="Commit_Click" imageMso="ServerConnection"/>
          <button id="bt_update" label="Update" size="large" onAction="Update_Click" imageMso="RecurrenceEdit"/>
        </group>
        <group id="gr_other" label="Other">
          <button id="bt_changeSvn" label="Change Svn" size="large" onAction="ChangeSvn_Click"
                  imageMso="DatabasePermissionsMenu"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Obrázek 10: Příklad XML souboru definující vzhled záložky v Ribbon menu.

Posledním nepříjemným problémem bylo, že pro položku imageMso, definující ikonu která se na tlačítku zobrazí, neexistuje dokumentace. Respektive se z ní dá dozvědět pouze to, že tato položka definuje ikonu tlačítka, popřípadě jiného prvku. Ale galerie možných ikon a jejich jmen nebyla v dokumentaci k nalezení. Až po poměrně dlouhém hledání se mi podařilo sehnat rozšiřující modul[11] pro Microsoft Excel, který takovou galerii poskytuje - obrázek 11.



Obrázek 11: Ukázka rozšiřujícího modulu[11] pro MS Excel poskytujícího galerii ikon imageMso.

Stěžejní část modulu tvoří třída Connect dědící z tříd a rozhraní nutných pro komunikaci s MS Office aplikacemi. Pro tuto třídu je také při založení projektu vygenerován identifikační řetězec, takzvaný GuidAttribute, pomocí, kterého je registrován do operačního systému Windows:

```
[GuidAttribute("3A777E31-4EF7-4D21-9388-DD4E5B7C6D59"), ProgId("docsync.Connect")]
public class Connect :    Object,
                          Extensibility.IDTExtensibility2,
                          IRibbonExtensibility
```

Pro připojení modulu do MS Office jsem použil událost OnConnection, již zmíněného rozhraní IDTExtensibility2, kde jsem si do vlastnosti (proměnné) třídy Connect uložil instanci aktivní MS Office aplikace. Nutností bylo rozlišit běžící aplikaci a podle toho správně přetypovat objekt aplikace a pro verzi MS Office 2003 vytvořit jeho kompatibilní menu:

```
public void OnConnection(object application, Extensibility.ext_ConnectMode
connectMode, object addInInst, ref System.Array custom){

    if (application is MSWord.Application){
        wordApp = (MSWord.Application)application;
        officeType = OfficeType.OT_Word;
    }
    //podobně pro ostatní aplikace
    //...
    if (GetOfficeVersion() == "2003"){
        Office2003Menu(application);
    }
    //...
```

```
}
```

Jelikož metoda GetCustomUI z IRibbonExtensibility rozhraní se volá automaticky, musí i tato metoda obsahovat kontrolu, zda-li se jedná o MS Office verze 2007 - 2013, protože pokus o vytvoření Ribbon nabídky v MS Office 2003, způsobí chybu.

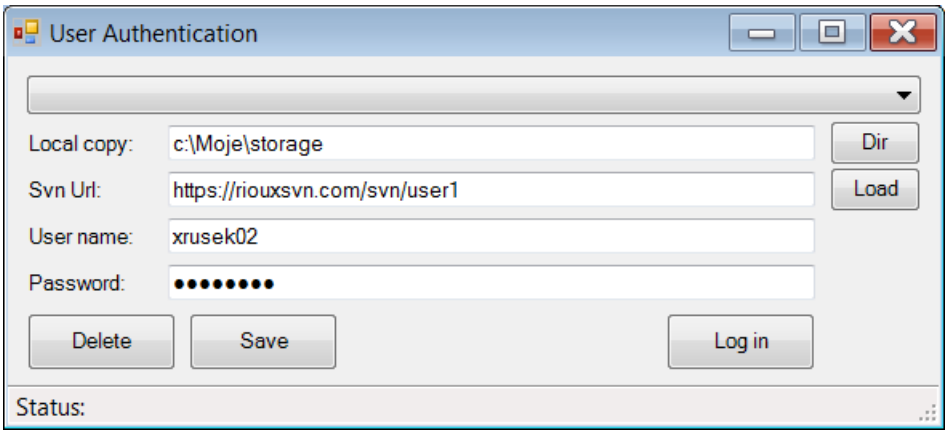
Další součástí třídy Connect jsou metody volané při události stisknutí tlačítka na Ribbon či klasické nabídce, dle verze Office aplikace. A jelikož tyto obsluhy událostí jsou definovány s různými počty parametrů, respektive pro Ribbon nabídku je to parametr jeden (typu object) a pro klasické menu jsou to parametry dva (typy object a ref bool), bylo by nutné vytvořit dvě sady prakticky duplicitních funkcí lišících se jenom parametry, nebo z jedné sady volat druhou. Ale protože je obecně vhodné se co nejvíce vyhnout duplicitním kódům - už jen z důvodu jejich možné změny - použil jsem vlastní delegát.

Delegát[22] je typ, jenž odkazuje na funkci (metodu) nejen pro programovací jazyk C#. Ve srovnání s ukazateli na funkce, jež jsou k dispozici například v jazycích C a C++ jsou však typově bezpečné. Jakmile se tedy delegátu přiřadí funkce, jenž má shodný návratový typ a typy parametrů, chová se tento delegát naprosto stejně jako ona funkce. Což umožňuje například předávat metody jako parametry, definovat callback metody (například porovnávání či řazení/seřazování), programově měnit volané metody či přiřadit mu anonymní funkci. Následuje příklad definice mého delegátu a jeho užití:

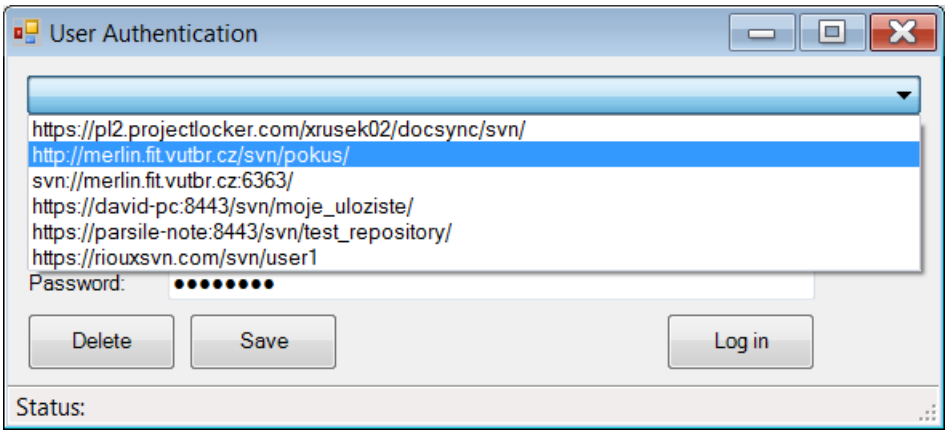
```
private delegate void MyClickCast(object control_, ref bool cancel_);

private void Office2003Menu(object application){
    //...
    MyClickCast myClickCast_ChangeSvn = delegate(object control_,ref bool cancel_)
    { ChangeSvn_Click(control_); };
    bt_svnChange = (CommandBarButton)menuBar.Controls.Add(
    MsoControlType.msoControlButton, missing, missing, missing, true);

    bt_svnChange.Caption = "Svn Change";
    bt_svnChange.Tag = "Svn Change";
    bt_svnChange.FaceId = 3198;
    bt_svnChange.Click += new  CommandBarButtonEvents_ClickEventHandler(
    myClickCast_ChangeSvn);
    //...
}
```

5.2 Formulář ověření uživatele

Všechny metody volané z některé nabídky obsahují kontrolu zda-li je uživatel přihlášen a pokud ne zobrazí se formulář pro zadání přihlašovacích údajů (obrázky 12 a 13).



Obrázek 12: Formulář ověření/přihlášení uživatele.



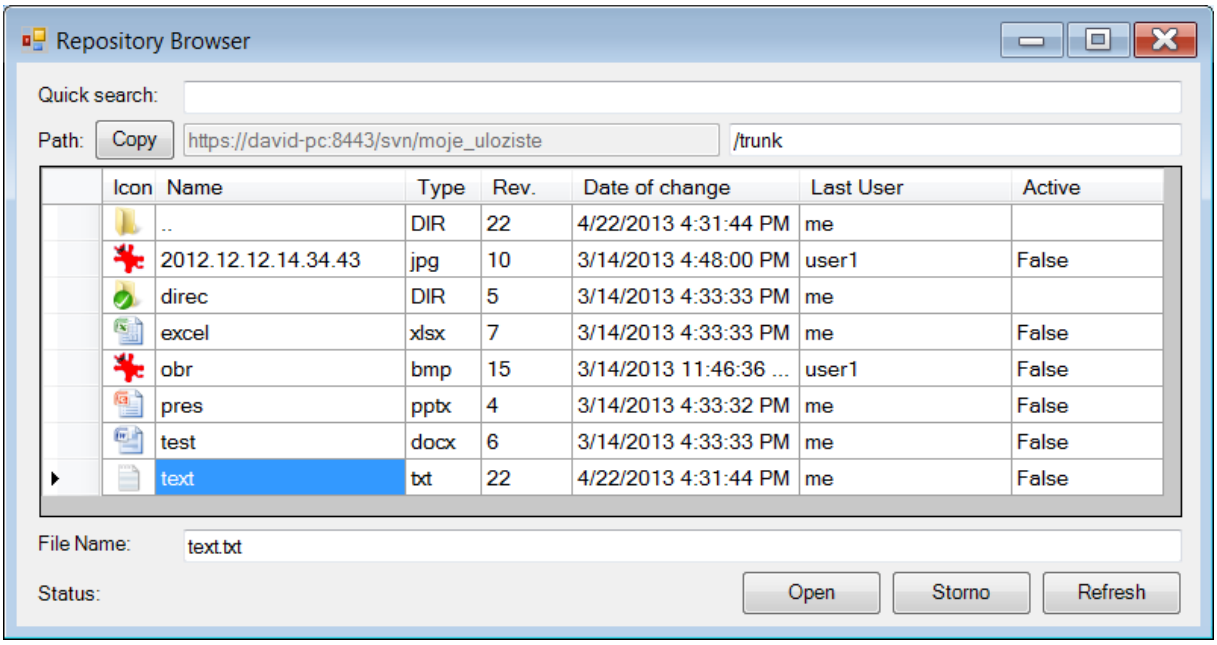
Obrázek 13: Formulář ověření/přihlášení uživatele se zobrazeným obsahem ComboBoxu.

Tlačítko Save a Delete na obrázcích 12 a 13 slouží k ukládání a případně mazání přihlašovacích údajů s automatickou aktualizací ComboBoxu zobrazeného na obrázku 13. Tlačítko Dir slouží k vybrání cesty k lokální kopii repozitáře a tlačítko Load načte URL repozitáře z lokální kopie. Aby Load funkce pracovala správně, je nutné aby lokální kopie repozitáře byla ve formátu Subversion 1.7.x. V předchozích verzích totiž URL v lokální kopii není uložena a v takovém případě se ve stavovém řádku objeví varování.

5.3 Formulář prohlížeče repozitáře

Přímo z menu rozšiřujícího modulu je k dispozici prohlížeč repozitáře (obrázek 14) a k jeho zobrazení musí být uživatel přihlášen.

K reprezentaci prohlížeče byly k dispozici dva podobné formulářové prvky, ListView a DataGridView. Druhý ze jmenovaných má lepší organizaci dat a proto byl použit pro zobrazení seznamu souborů a jejich vlastností. Zde je důležité si uvědomit, že prohlížeč zobrazuje aktuální stav dokumentů a souborů ve vzdáleném Svn úložišti a ne lokální stav. Jediná lokálně získávaná položka jsou ikony souborů a adresářů. Pro soubory, které nejsou přítomny v lokální kopii repozitáře, jsou ve zdrojích (ressources) uloženy defaultní ikony. Jedna pro soubor a druhá pro adresář. To znamená, že je možné otevřít soubor, který není na lokálním počítači uložen a v takovém případě se zobrazí dialogové okno s upozorněním a možností tento soubor z repozitáře stáhnout (update).

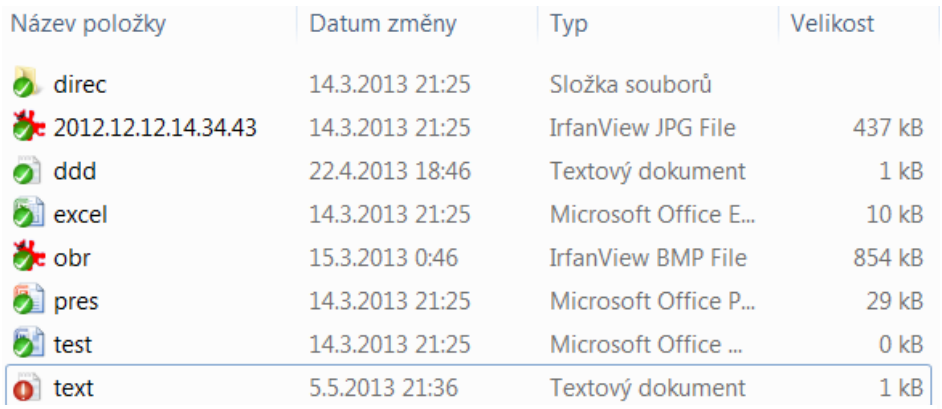


Obrázek 14: Formulář prohlížeče repozitáře.

Při zamítnutí možnosti aktualizovat lokální kopii repozitáře, se nic nestane a žádný dokument se neotevře. Naopak při souhlasu se aktualizuje pouze požadovaný soubor a následně se automaticky otevře.

V horní části prohlížeče je zobrazena URL právě zobrazené složky, která je pro názornost rozdělena na dvě části. Na neměnnou část URL a na dynamicky se měnící část současně s procházením repozitáře. Kvůli této úpravě je k dispozici i tlačítko Copy, které vloží do schránky (Clipboard) celou URL aktuálně prohlíženého adresáře. Výhodou tohoto přístupu je, že odstraňuje nutnost spojení dvou oddělených částí, což by pro uživatele bylo nepříjemné.

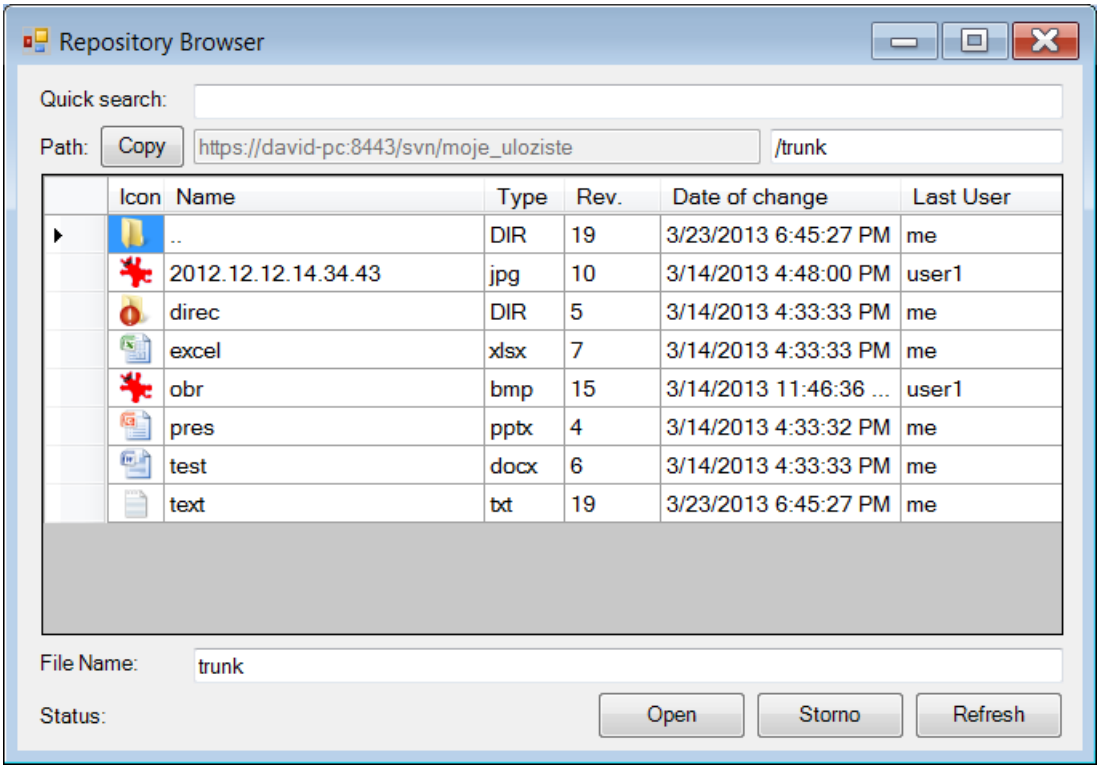
Z podobných důvodů jsem doplnil i Quick search funkcionalitu pro vyhledávání, jež při změně obsahu textového políčka zobrazí pouze soubory v daném adresáři, jejichž názvy obsahují daný řetězec. Poslední ovladatelným prvkem prohlížeče, je tlačítko Refresh, sloužící jako možnost aktualizace především v případě sdílení repozitáře s více uživateli.



Obrázek 15: Windows průzkumník se zobrazenými TortoiseSVN overlays.

Součástí implementace prohlížeče, byl i požadavek na zobrazování ikon zobrazených souborů tak, aby měl uživatel možnost ještě vizuální kontroly mimo pouze zkratky typu souboru. Dalším důvodem bylo, že například Subversion klient TortoiseSvn registruje v systému pro ikony souborů takzvaný overlay - transparentní překryv ikony informující o stavu složky či souboru. Na obrázku 15 je vidět obsah složky kde například soubor text není nahrán do repozitáře (není commitnut), tudíž jeho překryv je vykřičník na rudém pozadí.

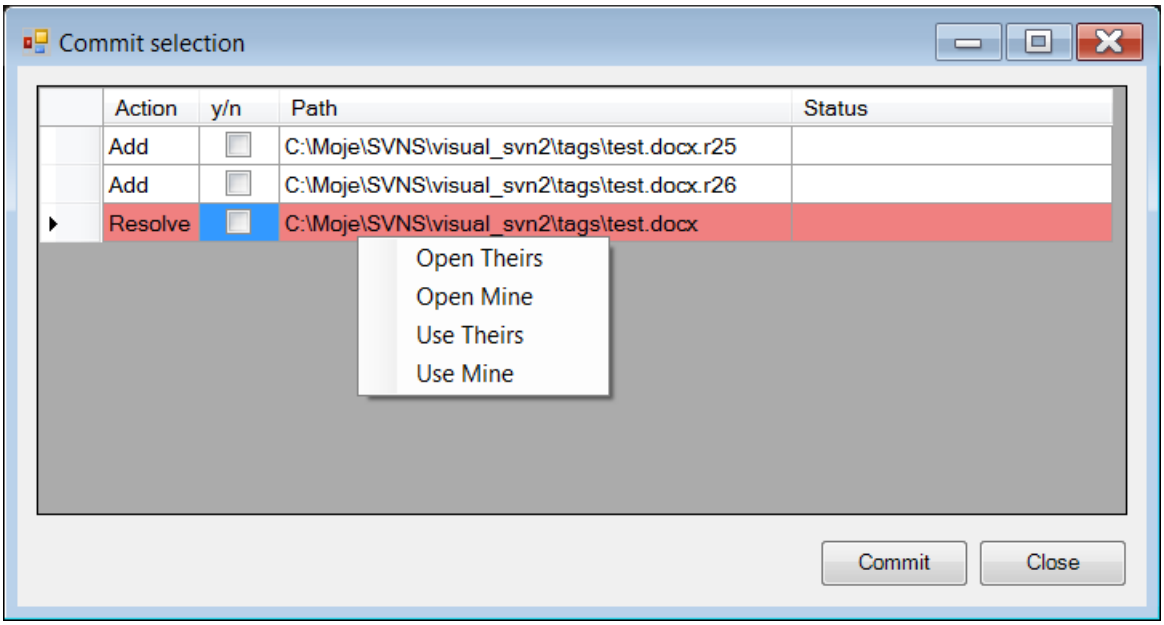
C#, respektive .NET obsahuje funkci System.Drawing.Icon.ExtractAssociatedIcon(string filePath), jenomže ta funguje jenom pro soubory a ne složky, které překryv mají samozřejmě také. Navíc v systému je většinou registrováno několik velikostí každé ikony a zmíněná funkce vrací pouze jednu velikost, která je pro potřeby prohlížeče příliš velká. Mnohem závažnější nedostatek je však skutečnost, že ikona získaná voláním této funkce požadovaný overlay neobsahuje.



Obrázek 16: Formulář prohlížeče repozitáře.

K řešením jsem použil funkci SHGetFileInfo, kterou bylo potřeba importovat z Windows API (shell32.dll). SHGetFileInfo umožňuje přesně specifikovat velikost ikony a potřebný overlay, podle potřeb prohlížeče repozitáře. Navíc funkce SHGetFileInfo nerozlišuje mezi soubory a adresáři. Ke správné implementaci mi pomohly zdroje [13] a [14].

5.4 Formulář pro commit souborů



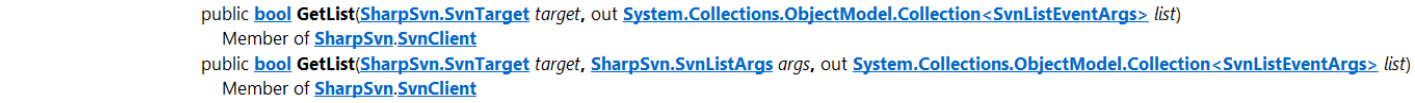
Obrázek 17: Formulář pro commit souborů.

Další formulář přístupný přímo z MS Office slouží pro commit souborů a stejně jako prohlížeč je tvořen DataGridView prvkem. Zobrazí všechny soubory jež jsou ve stavu Add (právě přidaný soubor do lokální kopie repozitáře Svn), Modified (soubor, který byl modifikován a který dosud nebyl nahrán do repozitáře) nebo Conflicted (například situace, kde soubor v repozitáři má vyšší číslo revize než uživatelem dosud modifikovaný soubor), pro které je možno buď provést akci commit (akce Add vždy následuje i Commit) nebo vyřešit jejich konflikt. Na obrázku 18 je vidět příklad takové situace.

5.5 Napojení modulu na SVN repozitář

K napojení modulu na Subversion repozitář jsem použil open source knihovnu SharpSvn[8], která poskytuje stejnou funkcionalitu jako klasický Svn klient pro příkazovou řádku. SharpSvn je organizována objektově. Tedy pro interakci s repozitářem se vytvoří instance třídy SharpSvn.SvnClient, která obsahuje metody pro akce jako je Commit, Update, Resolve, atd.

Pro získání dat prohlížeče repozitáře jsem použil například metodu SvnClient.GetList, jenž odpovídá funkcionalitě Svn klienta "svn list <args>". Tedy umožňuje získat obsah adresáře na zadané URL adrese. Bohužel jakákoliv dokumentace SharpSvn je ve stejně žalostném stavu jako dokumentace PIA, takže se dá opět maximálně tak zjistit, jakého typu jsou parametry funkce (příklad je k vidění na[12] - velice podobné tomu, co je vidět na obrázku 19).



Obrázek 18: Visual Studio Object Browser - dokumentace GetList metody.

Obě dvě verze metody se liší tím, že u druhé je možné upřesnit chování metody pomocí parametru args typu SharpSvn.SvnListArgs. Samozřejmě tu jsem jako první nepoužil a narazil na problém, že

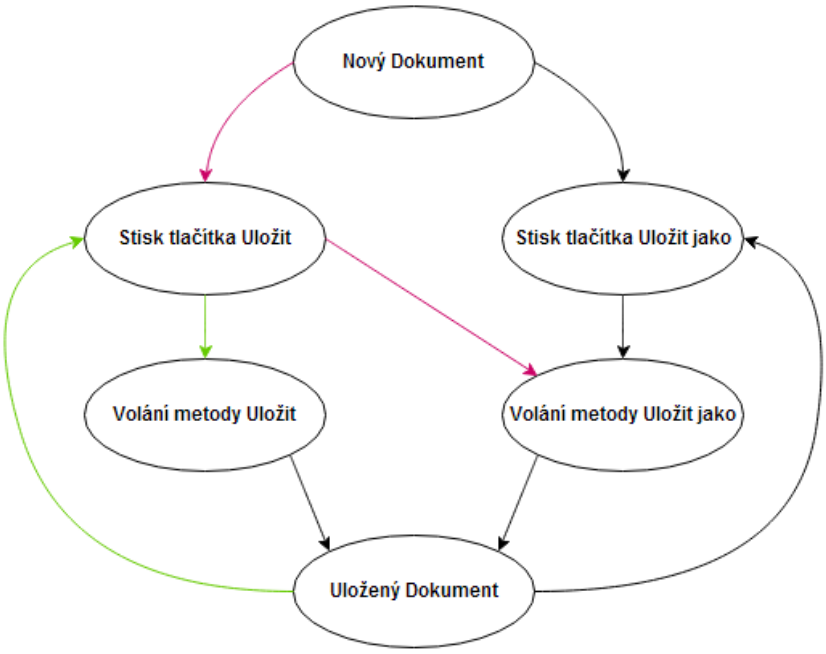
metoda vracela korektně pouze seznam názvů souborů, ale data jako revize souboru, poslední uživatel či časové razítko byly špatně nebo nebyly získány vůbec. To jsem pak řešil voláním dalších metod jako je GetStatus nebo GetInfo. A to samozřejmě vedlo k poměrně velkému nárůstu dotazů na Svn server a nepřijatelně dlouhou dobu odezvy i pro malé počty souborů ve složce. Proto jsem pak sám později zkusil parametr args definovat a kde se ukázalo, že pro informace které se vracely nekorektní, se prostřednictvím tohoto parametru nejprve muselo dát Svn serveru najevo, že je potřebuji.

Zpětně je logické, aby metoda potencionálně pracující s relativně velkým objemem dat měla možnost specifikovat jak náročná nakonec bude. Přesto to však nemění nic na faktu, že jsem nenašel ani jeden příklad použití druhé verze metody.

5.6 Implementační problémy

5.6.1 Implementace uložení souboru a následného commitu

V kapitole zabývající se grafickým návrhem modulu byla v prvotní verzi rozhraní plánována tlačítka "Uložit" respektive "Uložit jako". Měly to být prakticky identické funkce jako poskytuje MS Office ve svých aplikacích, jediným rozdílem by bylo, že by na ně bylo možné navázat akce jako je commit a automatizovat tak částečně práci s modulem. Nicméně vlastní implementaci této funkcionality zabránilo nelogické chování PIA.



Obrázek 19: Znázornění posloupnosti akcí při ukládání dosud neuloženého souboru.

Během události OnConnection z rozhraní IDTExtensibility2 se modul připojí k aktuálně aktivní instanci MS Office aplikace, která pak zahrnuje všechny aktivní dokumenty a okna - například MS Word - což je velice důležité. Skrze ni (konkrétně `Microsoft.Office.Interop.Word.Application` a obdobně pak existují třídy i pro MS Excel či MS PowerPoint) je možné získat instanci aktivního dokumentu, jenž obsahuje metody Save a SaveAs. Bohužel jsem při snaze vytvořit klasickou

posloupnost akcí při ukládání souboru (znázorněno na obrázku 17) narazil na naprosto nepochopitelnou implementaci metody SaveAs. Pokud se snažím uložit dosud neuložený dokument a zavolám metodu Save nebo SaveAs, obě povedou k vykonání metody SaveAs, což funguje správně. Respektive se vyvolá klasický MS Office dialog pro uložení dokumentu. Je důležité si uvědomit, že tento dialog není klasický ukládací dialog jako obsahuje .NET framework, ale značně upravený dialog, který kromě toho že umožňuje uložení přibližně do 20 různých formátů, také umožňuje přidat rozšiřující nastavení téměř ke každému z nich.

Když však zavolám metodu SaveAs na již uloženém dokumentu, žádný dialog se nevyvolá a je pouze na programátorovi se pokusit implementovat SaveAs dialog se stejnou funkcionalitou jako ten pro neuložený. Což se ukázalo natolik obtížné, že jsem od tohoto záměru upustil. Nejenom, že funkce SaveAs má přibližně patnáct parametrů a dohled jejich kombinaci tak, aby se SaveAs funkce chovala stejně jako klasická metoda se mi nepodařilo, ale především některé formáty jako docx a pdf, při uložení hlásily chybu, což je samozřejmě nepřijatelné.

Pokusil jsem se dokonce datové struktuře pro aktivní dokument vnutit stav, že dokument je neuložený, nastavením její položky Saved před uložením na false. Nicméně metody Save a SaveAs na tuto změnu vůbec nereagovaly, tudíž jsem se rozhodl metody Save a SaveAs nepoužívat a odstranit jejich tlačítka z rozhraní modulu. Uživatel tedy bude muset využít klasické ukládání prostřednictvím MS Office aplikací. Díky tomu bylo nutné rozšířit grafické rozhraní o možnost commitu souborů. Pro tento účel jsem implementoval nový formulář, kde jsem podobně jako v prohlížeči repozitáře použil DataGridView, pro zobrazení souborů nenahraných do repozitáře či souborů v konfliktu. Tady se však objevila jedna nepříjemná vlastnost metody SvnClient.Commit knihovny SharpSvn, která vyžaduje aby soubor, jež je nutno nahrát do repozitáře (commit), byl zavřený nebo maximálně otevřen pro čtení. Pokud tak není před akcí commit učiněno, aplikace na několik sekund zamrzne načež nastane výjimka.

Pro řešení existují tedy pouze dvě možnosti. Buď automaticky uzavřít všechny soubory jež jsou otevřeny, nahrát je do repozitáře a zase otevřít nebo před či po výjimce upozornit uživatele aby uzavřel všechny soubory.

První možnost by byla ideální, nicméně může velice jednoduše nastat, že budou otevřeny i soubory mimo MS Office (poznámkový blok, obrázky atp.), jelikož modul pro MS Office nezabraňuje nahrávat do repozitáře i jiné soubory, které mohou být editovaný v cizích aplikacích a nad kterými by modul neměl naprosto žádnou kontrolu. Proto jsem zvolil druhou možnost.

5.6.2 Implementace indikace zda-li je soubor používán

Jedním z úkolů implementace bylo také umožnit uživateli před otevřením souboru vizuální kontrolu, zda-li je tento soubor již otevřen. A jelikož jsem pro účely modulu zvolil Subversion jako verzovací systém, kde jsou konflikty poměrně častá záležitost, tím důležitějším se tento bod stal.

Vzhledem k tomu, že Svn tuto funkcionalitu nenabízí, navrhl jsem několik možností. První řešení byla možnost Subversion systému přidávat k jednotlivým souborům dodatečné informace, takzvané properties. Princip je jednoduchý. Při otevření dokumentů nastavit vlastnost IsOpened na true a při zavření ji nastavit na false, případně úplně zrušit. Bohužel se ukázalo, že Svn servery mají poměrně velkou odezvu, což způsobilo před otevřením dokumentu a po zavření dokumentu nepříjemné

zamrznutí aplikace na několik sekund. Tento problém bylo možné eliminovat například provedením operace v separátním vláknu a nezatěžovat tak vlákno starající se o běh aplikace. Avšak tento přístup má jiný zásadní nedostatek. Subversion totiž o vlastnosti (properties) uvažuje jako o pevné součásti souboru, což má za důsledek změnu čísla revize souboru při jakékoliv její změně. Takže při otevření a zavření souboru revize naroste o dvě, což by vyvolalo velké množství naprosto zbytečných aktualizací lokální i vzdálené kopie repozitáře Svn.

O něco lepší řešení bylo vytvořit v Svn repozitáři soubor, obsahující všechny jména otevřených souborů a jeho obsah pak získávat. Ale proces získání souboru z repozitáře a jeho nutná následná analýza by způsobila nutnost práce s dočasnými soubory. Nemluvě o již zmíněné dlouhé odezvě Svn a opět nadbytečným aktualizacím, které by toto řešení vyvolalo. Proto jsem toto řešení zamítnul.

Nakonec jsem zvolil řešení pomocí jednoduchého php scriptu hostovaného na mém školním účtu, který na přesně specifikovaný HTTP dotaz metodou POST vytvoří soubor, jehož název bude SHA256 hash url repozitáře (řetězec 64 znaků), za cílem zbavit se nepovolených znaků v url repozitáře a který bude obsahovat seznam jmen otevřených souborů i s cestou. A jelikož je HTTP dotaz mnohem rychlejší než Svn dotaz, uživatel nezaznamená prakticky žádné zpomalení MS Office aplikace.

K zasílání HTTP POST žádosti požívám HttpWebRequest třídu, která však vracela výjimku, kdykoliv by modul testován na cizím počítači. HttpWebRequest má totiž standardně nastaveno kontrolovat známé certifikáty. A jelikož hostuji php script na školním serveru (eva.fit.vutbr.cz), kde je certifikace při přístupu na webové stránky vyžadována, přestože nevlastní platný certifikát, HTTP požadavek byl pokaždé neplatný. Pro vývojové a testovací účely byla tedy kontrola certifikace zrušena.

Tento problém je možno vyřešit pomocí `ServicePointManager.CertificatePolicy`, jenž umožňuje nastavit chování (politiku) kontroly certifikátů, tedy ji i úplně vypnout. Nicméně je tato položka označena jako zastaralá a k tomuto účelu je nyní k dispozici `ServicePointManager.ServerCertificateValidationCallback`, což je delegát (.NET ukazatel na funkci), pomocí kterého uživatel může definovat svoji vlastní kontrolu. Pro její definici jsem použil lambda (anonymní) funkci, která jakékoliv kontrole certifikátu vnutí kladný výsledek.

```
ServicePointManager.ServerCertificateValidationCallback =
((sender, certificate, chain, sslPolicyErrors) => true);
```

6 Testování

Hlavní část vývoje rozšiřujícího modulu proběhla na operačním systému Microsoft Windows 7 za použití kancelářské programové sady Microsoft Office verze 2007, přesto však bylo nutné ověřit funkčnost modulu i na některých jejich dalších verzích. Přesné složení uvádí následující seznam:

- Microsoft Windows 7 Professional 64bit, Service Pack 1
  - Microsoft Office 2003 Professional
  - Microsoft Office 2007 Home and Student
  - Microsoft Office 2010
- Microsoft Windows XP Professional 32bit Service Pack 3
  - Microsoft Office 2007 Home and Student
- Microsoft Windows 8 Professional 64bit
  - Microsoft Office 2007 Home and Student

.NET verze	1.0	1.1	2.0	3.0	3.5	4.0	4.5
Windows 95	nelze	nelze	nelze	nelze	nelze	nelze	nelze
Windows NT	lze doinstalovat	lze doinstalovat (SP6a)	nelze	nelze	nelze	nelze	nelze
Windows 98, Windows 98 SE	lze doinstalovat	lze doinstalovat	lze doinstalovat	nelze	nelze	nelze	nelze
Windows Me	lze doinstalovat	lze doinstalovat	lze doinstalovat	nelze	nelze	nelze	nelze
Windows 2000	lze doinstalovat	lze doinstalovat	lze doinstalovat (SP3)	nelze	nelze	nelze	nelze
Windows XP	lze doinstalovat	lze doinstalovat	lze doinstalovat (SP2)	lze doinstalovat (SP2)	lze doinstalovat	lze doinstalovat (SP3)	nelze
Windows Server 2003	??	součást systému	lze doinstalovat	lze doinstalovat (SP1)	lze doinstalovat	lze doinstalovat (SP2)	nelze
Windows Vista	částečná kompatibilita	částečná kompatibilita	součást systému	součást systému	lze doinstalovat	lze doinstalovat (SP1)	lze doinstalovat (SP2)
Windows Server 2008	??	??	??	součást systému	lze doinstalovat	lze doinstalovat	lze doinstalovat (SP2)
Windows Server 2008 R2	??	??	??	??	součást systému	lze doinstalovat	lze doinstalovat (SP1)
Windows 7	částečná kompatibilita	částečná kompatibilita	součást systému	součást systému	součást systému	lze doinstalovat	lze doinstalovat (SP1)
Windows 8	??	??	lze doinstalovat	lze doinstalovat	lze doinstalovat	kompatibilita	součást systému

Obrázek 20: Tabulka rozšíření jednotlivých verzí .NET frameworku na různé verze operačního systému Windows[19].

Podmínkou pro správný chod navrženého modulu, respektive jeho samotného nainstalování, je mít na všech zmíněných operačních systémech nainstalován .NET framework verze 3.5, který je standardně obsažen pouze ve verzích Windows 7 a Windows Server 2008 R2. Přesné rozšíření verzí frameworku .NET a jeho kompatibilitu s různými verzemi operačního systému Windows, znázorňuje obrázek 20.

K testování také bylo zapotřebí přístupu k Subversion serveru. Mezi volně dostupné servery patří například:

- RiouxSVN - https://riouxsvn.com/
- ProjectLocker - http://projectlocker.com/

Bohužel jejich odezva byla velice pomalá a pro potřeby ladění nebyly vhodné. Proto jsem se rozhodnul použít lokálně běžící Visual SVN Server[20], jenž umožňuje velice jednoduchou správu SVN serveru, tvorbu repozitářů a jejich uživatelů s podporou komunikace přes HTTPS protokol.

Samotné testování pak sledovalo, jestli se v jednotlivých případech užití nevyskytne chyba.

7 Závěr

Rozšiřující modul pro Microsoft Office aplikace Word, Excel a PowerPoint byl úspěšně implementován za použití vysokoúrovňového jazyka C# z .NET platformy, verzovacího systému Apache Subversion, jehož funkcionalita byla s modulem propojena pomocí knihovny SharpSvn. Modul umožňuje sdílení, synchronizaci a management dokumentů vytvořených nejen pomocí MS Office aplikací. Ukázalo se však, že volba Subversion systému pro vývoj modulu nebyla nejlepší, přestože ze srovnání SCM nástrojů vyšla nejlépe. Jedním z důvodů byly properties jednotlivých souborů, které nakonec nebyly použitelné, protože způsobovaly změny revizí souborů. A kvůli čemu byl do vývoje modulu nakonec zapojen i webový programovací jazyk PHP5. Také nutnost pevného provázání centralizovaného repozitáře s lokální kopií, se ukázalo býti velice svazující, což dokazuje například nutnost ukládat soubory do lokální kopie repozitáře, aby mohly být commitnuty. Proto jeden směr budoucího vývoje modulu by mohla být náhrada repozitáře na bázi Subversion systému za úložiště v režii technologie (protokolu) WebDAV.

Zdroje

[1] Google Clound Connect [online]. [cit. 2013-2-15] Dostupné z URL: <https://tools.google.com/dlpage/cloudconnect>

[2] Google Inc.: Google Cloud Connect. In: Wikipedie: otevřená encyklopedie [online]. St. Petersburg (Florida): Wikimedia Foundation, aktualizováno 2013-4-2 [cit. 2013-4-2]. Dostupné z URL: <http://en.wikipedia.org/wiki/Google_Cloud_Connect>

[3] How to build an Office COM add-in by using Visual C# .NET. In: Microsoft Support [online]. Redmont, USA: aktualizováno 2007-5-11 [cit. 2013-5-6]. Dostupné z URL: <http://support.microsoft.com/default.aspx?scid=kb;en-us;302901>

[4] Add-in Express. c2004-2013 [cit. 2013-5-6]. Dostupné z URL: <http://www.add-in-express.com/downloads/adxvsto.php>

[5] Git [online].[cit. 2013-5-6]. Dostupné z URL: <http://git-scm.com/>

[6] WebDAV. In: Wikipedie: otevřená encyklopedie [online]. St. Petersburg (Florida): Wikimedia Foundation, aktualizováno 2013-4-22 [cit. 2013-5-6]. Dostupné z URL: <http://en.wikipedia.org/wiki/WebDAV>

[7] Sabredav project: In: Google Code [online]. California, USA: [cit. 2013-5-6]. Dostupné z URL: <https://code.google.com/p/sabredav/>

[8] Domovská stránka projektu SharpSvn knihovny. Dostupné z URL: <http://sharpsvn.open.collab.net/>

[9] Visual Studio Tools for Office. In: Wikipedie: otevřená encyklopedie [online]. St. Petersburg (Florida): Wikimedia Foundation, aktualizováno 2013-18-3 [cit. 2013-5-6]. Dostupné z URL: <http://en.wikipedia.org/wiki/Visual_Studio_Tools_for_Office>

[10] What is COM Add-in? In: MSDN: Microsoft Developer Network [online]. Redmont, USA: Dostupné z URL: [cit. 2013-5-6]. <http://msdn.microsoft.com/en-us/library/office/aa141383(v=office.10).aspx>

[11] Savraj Dhanjal: RibbonX Update for B2TR. In: Jensen Harris: An Office User Interface Blog [online]. Publikováno: 2006-9-15 [cit. 2013-5-6]. Dostupné z URL: <http://blogs.msdn.com/b/jensenh/archive/2006/09/15/755336.aspx>

[12] Dokumentace SharpSvn knihovny. Dostupné z URL: <http://docs.sharpsvn.net/current/html/M_SharpSvn_SvnClient_GetList.htm>

[13] How to use the SHGetFileInfo function to get the icons that are associated with files in Visual C# .NET. In: Microsoft Support [online]. Redmont, USA: aktualizováno 2006-9-22 [cit. 2013-5-6]. Dostupné z URL: <http://support.microsoft.com/kb/319350/en-us?fr=1>

[14] Need help with SHGetFileInfo API. In: MSDN: Microsoft Developer Network [online]. Redmont, USA: publikováno 2008-1-14 [cit. 2013-5-6]. Dostupné z URL: <http://social.msdn.microsoft.com/Forums/en-US/csharpgeneral/thread/6630dda6-6b6a-45bf-a852-b6a79cfe7f7>

[15] How to bypass server certificate error (underlying connection was closed). In: Share the Knowledge [online]. Publikováno 2012-2-15 [cit. 2013-5-6]. Dostupné z URL: <http://ashwini47-tts.blogspot.cz/2012/02/how-to-bypass-server-certificate-error.html>

[16] Managed Code. In: Wikipedie: otevřená encyklopedie [online]. St. Petersburg (Florida): Wikimedia Foundation, aktualizováno 2013-2-26 [cit. 2013-5-6]. Dostupné z URL: <http://en.wikipedia.org/wiki/Managed_code>

[17] Common Language Runtime. In: Wikipedie: otevřená encyklopedie [online]. St. Petersburg (Florida): Wikimedia Foundation, aktualizováno 2013-3-2 [cit. 2013-5-6]. Dostupné z URL: <http://en.wikipedia.org/wiki/Common_Language_Runtime>

[18] Registry Entries for Application-Level Add-Ins. In: MSDN: Microsoft Developer Network [online]. Redmont, USA: [cit. 2013-5-6]. Dostupné z URL: <http://msdn.microsoft.com/en-us/library/vstudio/bb386106.aspx#LoadBehavior>

[19] .NET. In: Wikipedie: otevřená encyklopedie [online]. St. Petersburg (Florida): Wikimedia Foundation, aktualizováno 2013-3-8 [cit. 2013-5-6]. Dostupné z URL: <http://cs.wikipedia.org/wiki/.NET>

[20] VisualSVN Server [online]. c2005-2013 [cit. 2013-5-6]. Dostupné z URL: <http://www.visualsvn.com/server/>

[21] Sabredav project: List of related RFCs and implementation status. In: Google Code [online]. California, USA: aktualizováno 2012-11-27 [cit. 2013-5-6]. Dostupné z URL: <https://code.google.com/p/sabredav/wiki/RFCs>

[22] Microsoft Corporation: Delagates (C# Programming Guide). In: MSDN: Microsoft Developer Network [online]. Redmont, USA: aktualizováno 2013 [cit. 2013-5-6]. Dostupné z URL: <http://msdn.microsoft.com/en-us/library/ms173171(v=vs.80).aspx>