Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Sestavování dokumentů z fragmentů textu

Plzeň, 2013 Bc. Jan Rabušic

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2013

Bc. Jan Rabušic

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Martinovi Bíklovi za čas věnovaný konzultacím a cenné rady, bez kterých by tato práce nemohla vzniknout. Také bych chtěl poděkovat všem blízkým, kteří se mnou měli během mé tvorby nesmírnou trpělivost.

Abstrakt

Cílem práce je nalézt formát dokumentů, který je vhodný pro automatizované vytváření dokumentů s možností doplňování zadaných informací, opatřování elektronickým podpisem a časovým razítkem a je vhodný pro užití v podnikovém prostředí. Tato práce zahrnuje výběr formátu, analýzu a implementaci řešení automatizovaného vytváření a opatřování elektronickým podpisem a časovým razítkem. Integrace řešení do systému Microsft SharePoint 2010 demonstruje využití zvoleného řešení. Práce hodnotí právní platnost a význam elektronických podpisů a časových razítek.

Abstract

The aim of the work is to find a document format that is suitable for automatic document composition with filling the entered information, obtaining electronic signature and time stamp and is suitable for use in a corporate environment. The work includes analysis and implementation of solution for automatic creation and obtaining electronic the signature and the time stamp. Integration of the solution into the Microsoft SharePoint 2010 demonstrates the use of the solution. Thesis evaluates the juristic validity and importance of electronic signatures and time stamps.

Obsah

1 Úvod

1.1 Pozadí problému

Téma této diplomové práce bylo zadáno společností CCA Group a.s. se sídlem Karlovo náměstí 17, 120 00 Praha 2, dále jen CCA Group.

1.1.1 Správa dokumentů

V poslední době roste nebývalou měrou množství elektronických dokumentů. Důvodem je stále větší objem elektronické komunikace a postupný trend opouštění papírových dokumentů. Podle Cite World (www.citeworld.com) bylo v roce 2012 odesláno 89 miliard firemních e-mailů denně. Odhaduje se, že do roku 2016 vzroste počet těchto e-mailů na 143,8 miliard denně.[1] Očekává se tedy téměř dvojnásobný objem komunikace prostřednictvím e-mailu. Detailnější průzkum společnosti Radicati nabízí vývoj průměrného počtu obdržených e-mailů průměrného firemního uživatele za den. V roce 2011 to bylo 72 e-mailů za den. Pokud nebudeme počítat nevyžádanou poštu, pak to bylo 58 e-mailů za den. Očekává se, že v roce 2016 to bude 71 e-mailů denně bez nevyžádané pošty.[2] Kromě objemu elektronické komunikace roste počet digitalizovaných dokumentů i v jiných oblastech. Zavedením legislativy k elektronickému podpisu v České republice (Zákon č. 227/2000 Sb.) a podporou elektronizace státní správy (e-Government) se firmám otevírá další oblast, ze které lze očekávat nárůst objemu elektronických dokumentů.

„Správa dokumentů se zaměřuje na ukládání a uspořádání dokumentů s cílem podporovat aktuálně rozpracovanou práci“[3]. To zahrnuje především vytváření a sdílení obsahu. Účel je zřejmý – když organizace nevyužívá žádný systém správy dokumentů, obsah se zpravidla vytváří a ukládá neuspořádaně a decentralizovaně. To vede k časté redundanci informací, neschopnosti efektivně vyhledávat ve firemních dokumentech, spolupracovat na vytváření obsahu a jeho hromadnému užívání.[3]

Tato problematika je aktuálním tématem i v České republice. Potvrzuje to i IT internetový portál SystemOnLine.cz, který uvádí, že nezbytnost nasazení DMS a zjevný zájem hlavně větších firem dokazuje interní průzkum společ-

nosti Infinity z ledna roku 2012, který uskutečnila ve 130 středně velkých a velkých českých firmách. Podle tohoto průzkumu až 44% firem hodnotí problematiku správy dokumentů jako klíčovou.[4]

1.1.2 Důvod vzniku práce

Záměr společnosti CCA Group nabízet systémy pro správu dokumentů vyžaduje průzkum současného stavu dokumentových formátů, možnosti jejich využití a míry jejich podpory. Z tohoto důvodu vznikla práce, která má za úkol prozkoumat tento stav, vybrat dokumentový formát vhodný k podporování a vyzkoušet práci s ním na základních operacích (skládání dokumentů, jejich podepisování) v desktopové aplikaci a případně její integraci do již existujícího DMS systému (SharePoint). Zároveň vyžaduje průzkum z hlediska platnosti elektronických podpisů.

1.2 Přehled práce

V teoretické části se budu věnovat srovnání jednotlivých textových dokumentových formátů, teorii elektronických podpisů a časových razítek včetně kontextu české legislativy a představení vývoje modulů pro systém správy dokumentů Microsoft SharePoint. V praktické části provedu analýzu požadavků na dokumentový formát. Na základě srovnání dokumentových formátů provedeného v teoretické části vyberu, který dokumentový formát podporovat. Navrhnu a naimplementuji aplikaci pro skládání dokumentů, jejich plnění a podepisování tak, aby byla s co nejmenším množstvím úprav použitelná jako serverové řešení, respektive integrovatelná do systému Microsoft SharePoint. Následně ji do tohoto systému ve formě monolitického řešení zakomponuji. Řešení otestuji a ohodnotím z hlediska výkonu. Na závěr zhodnotím způsob řešení elektronických podpisů ve formátu Office Open XML.

2 Formáty složených textových dokumentů

Jedním z úkolů je určit formát dokumentů, který je vhodné používat v oblasti podnikových informačních systémů. Pro tento účel představím vybrané dokumenty se zaměřením na tři hlavní kritéria.

2.1 Kritéria formátu složeného dokumentu

Z reálných požadavků zákazníků a standardů kladených na správu dokumentů lze odvodit několik hlavních kritérií, které musí složené formáty dokumentů splnit.

2.1.1 Dostupnost

Primárním požadavkem na formát dokumentu je, aby byl široce dostupný veřejnosti, tedy aby se nejednalo o proprietární formát, jehož specifikace není známá, a tak aby zákazníka pro práci s ním nic nevázalo k používání konkrétní aplikace. To platí bez výjimky pro firemní dokumenty. Ačkoliv by bylo možné zavést proprietární dokumentový formát uvnitř podniku, musíme počítat s tím, že dokumenty nejsou vázány pouze na podnikové prostředí, ale jejich prostřednictvím probíhá interakce s okolím (jiné organizace, státní správa a další). Zde bychom naráželi na problémy s akceptací takových dokumentů.

2.1.2 Automatické vytváření dokumentu

Dalším požadavkem je, aby existovala nekomerční knihovna, která by značně usnadnila práci s tímto formátem. Přinášela by možnost skládat dokument z menších definovaných celků – šablon, které by umožňovala následně plnit textovými fragmenty.

2.1.3 Archivace

Posledním hlavním požadavkem je, aby struktura dokumentu podporovala, nebo alespoň umožňovala připojování elektronických podpisů a časových razítek, důležitých pro archivaci.

2.2 Zkoumané modely

Již podle kritéria 2.1.1, které klade důraz na otevřenost formátu, mohu mezi porovnávané formáty dokumentů zahrnout následující:

- Portable Document Format (PDF)
- Office Open XML (OOXML)
- Open Document Format for Office Aplications (ODF)
- eXtensible Stylesheet Language - Formatting Objects (XSL-FO)
- Rich Text Format (RTF)

Tyto formáty splňují kritérium 2.1.1, to však neznamená, že jsou to jediní zástupci, kteří tuto podmínku splňují. Byly vybrány také proto, že jsou poměrně rozšířené, nebo mají alespoň specifické vlastnosti, které by mohly mít pro odvětví správy dokumentů význam. V dalších podkapitolách, kde budou podrobně rozebrány, neopomenu zhodnotit, do jaké míry vyhovují bodům 2.1.2 a 2.1.3.

Následující informace o formátech dokumentů vycházejí z poznatků zjištěných v práci na téma „Modely dokumentů a metody jejich skládání“, kterou jsem vypracoval jako součást Oborového projektu. Zde se však na rozdíl od výše uvedeného projektu více zaměřuji na dostupnost knihoven pro platformu .NET, která byla uvedena jako požadavek společnosti CCA Group, a podle tohoto kritéria zohledňuji závěrečné hodnocení.

2.3 Portable Document Format

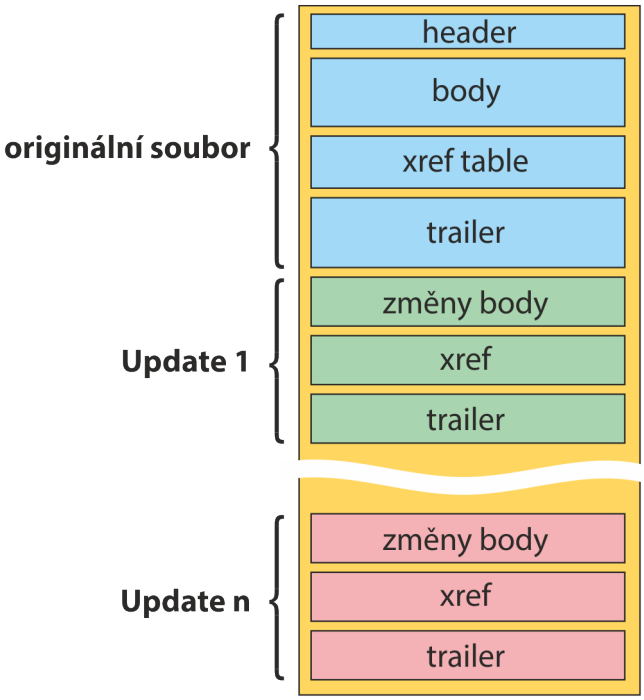
Portable Document Format (dále jen PDF) je souborový formát vyvinutý původně jako proprietární formát společnosti Adobe Systems. Přestože byla v roce 1993 uveřejněna volně šiřitelná specifikace formátu PDF, zůstal formát pod kontrolou společnosti Adobe až do 1. června 2008, kdy byl publikován jako otevřený standard ISO 32000-1.[5] Jedná se o formát, který slouží k různým účelům podle toho, jaký standard splňuje:

- **PDF/X - PDF for Exchange** – grafické technologie – předtisková výměna digitálních dat,
- **PDF/A - PDF for Archive** – správa dokumentů – formát elektronického dokumentu pro dlouhodobou úschovu,
- **PDF/E - PDF for Engineering** – správa dokumentů – formát pro technickou dokumentaci,
- **PDF/VT - PDF for exchange of variable data and transactional (VT) printing** – grafické technologie – podpora variable data printing,
- **PDF/UA - PDF for Universal Access** – správa dokumentů – usnadnění navigace v dokumentu (pro lidi s postižením).[6]

2.3.1 Struktura PDF

PDF je strukturovaný formát, jehož kód je rozdělen implicitně do následujících komponent: *header*, *body*, *xref table*, *trailer* (obrázek 2.1).

Část header obsahuje pouze jednu řádku kódu, která označuje verzi PDF. Část body obsahuje veškeré informace o objektech (písma, texty, formulářová pole, obrázky a další). Xref table obsahuje informace o tom, kolik objektů dokument obsahuje, kde začínají a jejich délku v bytech. Trailer obsahuje ukazatele na položky v xref table a na klíčové objekty. Pokud chceme PDF měnit, máme dvě možnosti, jak následně dokument uložit. Volbou **uložit jako** (*Save As*) uložíme pouze změněné PDF. Volbou **uložit** (*Save*) dojde k uložení původního PDF, ke kterému jsou připojeny update části: body changes, tref a trailer nového dokumentu. Pokud chceme PDF dokument podepsat, musíme jej ukládat volbou uložit, jinak bychom změnili původní PDF



Obrázek 2.1: Struktura PDF souboru po mnohonásobném ukládání (zdroj: autor).

a podpis by nebyl validní.[7]

PDF využívá pro generování layoutu a grafiky podmnožinu popisného jazyka *PostScript*. Ten navíc rozšiřuje o dříve zmíněné strukturování, systém vkládání a přenosu písma a další.

2.3.2 Vytváření PDF

Pro práci s PDF existuje celá řada knihoven. Jednou z nich je například *iText*, který má placenou i AGPL variantu a existuje i ve verzi pro C# .NET. Umožňuje automaticky vytvářet PDF jak po chunk blocích (po kusech) a odstavcích, tak přímo po bytech. Dále umožňuje automaticky plnit interaktivní formuláře a podepisovat, šifrovat, popisovat, spojovat a rozdělovat dokumenty formátu PDF.

2.3.3 Shrnutí PDF

Na jedné straně je velkou výhodou PDF jeho rozšířenost a standardizace, která se navíc stále vyvíjí. Na druhé straně je pro „obyčejného uživatele“ nevýhodou, že již vytvořený dokument může měnit pouze s omezeními (přidání popisku, podepsání atd.) nebo za použití pokročilých nástrojů (zpravidla se jedná o placené nástroje).

2.4 Office Open XML

Formát *Office Open XML* (dále jen OOXML) byl vyvinut společností Microsoft a využívá kombinace ukládání dat ve formátu XML, které jsou strukturovány v ZIP archivu. První standard byl vydán organizací Ecma (ECMA-376) a o něco později ISO a IEC (ISO/IEC 29500). Od vydání Microsoft Office 2007 se stal OOXML defaultním formátem Microsoft Office. Díky otevřené specifikaci lze nalézt mnoho programů napříč operačními systémy, které s OOXML umí pracovat. Formát OOXML byl navržen tak, aby podporoval následující oblasti:

- Jednoduchá integrace business informací s dokumenty – podporuje rychlou tvorbu dokumentů za použití datových zdrojů.
- Otevřenost, bezplatnost – formát je založen na technologiích XML a ZIP, které jsou všeobecně dostupné, specifikace je též bezplatná a otevřená.
- Interoperabilita – díky tomu, že je jádro dokumentu ve formátu XML, lze formát OOXML snadno použít pro výměnu dat mezi informačními systémy a aplikacemi Microsoft Office.
- Robustnost – formát OOXML je navržen tak, aby byl robustnější než klasické binární formáty (poškození podsouboru neomezí čitelnost ostatních podsouborů atd.).
- Efektivita – ZIP formát umožňuje komprimovat data (až o 75% menší soubory).
- Bezpečnost – otevřenost formátu umožňuje snadno identifikovat a odstranit osobní či jiné citlivé údaje. Další vylepšení v oblasti bezpečnosti

přinášejí OOXML dokumenty tím, že defaultně nevykonávají vložený kód, tedy není možné touto cestou podstrčit škodlivý kód k vykonání.

- Zpětná kompatibilita – předešlé .doc, .xls a .ppt formáty jsou plně kompatibilní s jejich OOXML nástupci.[8]

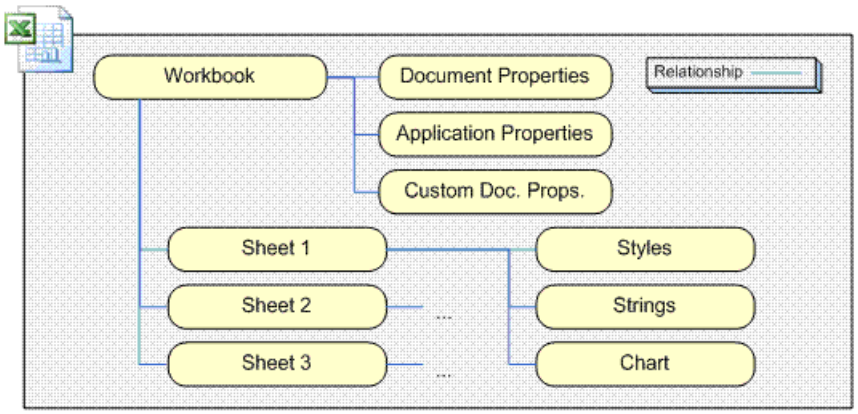
2.4.1 Struktura OOXML

Jak bylo řečeno výše, OOXML kombinuje technologie ZIP a XML. Jedná se tedy o zip archiv převážně XML souborů, které obsahují informace o formátování, stylech, kapitoly, textový obsah, metadata a další. Jednotlivé části dokumentu lze samostatně měnit, nebo nahrazovat. Existuje zde systém odkazů, které ukazují na jednotlivé části dokumentu. Z toho je patrné, že struktura adresáře zde není pevně stanovená a záleží jen na odkazech. XML části dokumentu mohou být různých typů, některé jsou společné pro všechny aplikace, které s OOXML pracují, další jsou pak podřízeny přímo konkrétním typům aplikací (například *list* je implicitně určen aplikaci Excel, nebo *slide master* aplikaci PowerPoint). Další objekty, například obrázky, jsou ukládány přímo v jejich nativním formátu.[8]

Důležité součásti:

- **Složka _rels** obsahuje kořenový soubor .rels.
- **Soubor .rels** obsahuje popis částí, které se váží k (virtuální) startovní části dokumentu, pomocí vztahů. Vztahy (Relationships) obsahují informaci o typu části, na kterou odkazují, dále obsahují unikátní identifikátor a umístění části. Příklad vztahů naleznete na obrázku 2.2.
- **Soubor [Content_Types].xml** obsahuje definice použitých typů obsahu.

Odkazované části mohou být buď jednoduché, nebo opět složené pomocí odkazování v _rels. V každém rels je definována hlavní část dokumentu (u PowerPoint dokumentu je to presentation, u Excel dokumentu workbook a u Word dokumentu je to document). Podle typu hlavní části dokumentu se odvíjejí ostatní vztahy. Vztahy (relationships) nehrají roli pouze v rels částech, ale používají se jako způsob propojení v celém OOXML. Příklad implicitního a explicitního odkazování ve formátu OOXML naleznete na obrázcích 2.3 a 2.4.



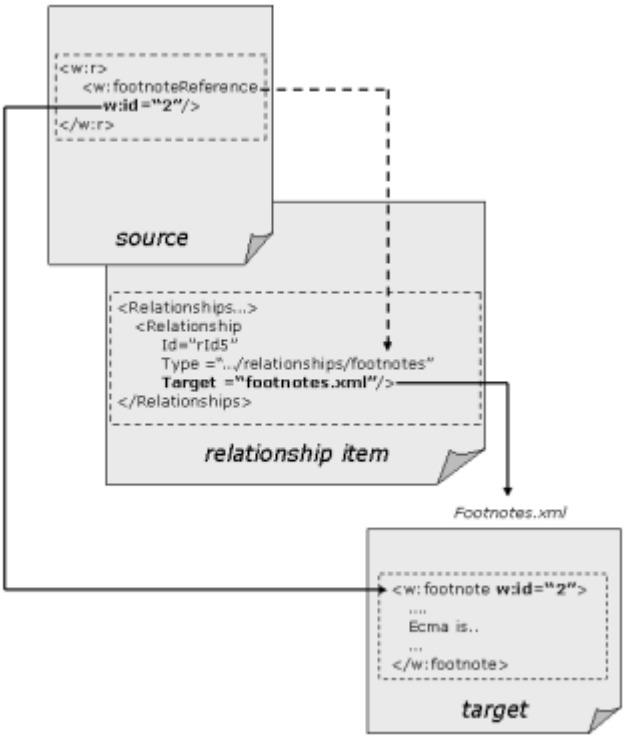
Obrázek 2.2: Příklad vysokoúrovňových vztahů v sešitu aplikace Excel (zdroj: msdn.microsoft.com).

Užitečnou vlastností formátu OOXML je také to, že nabízí vztahování nejen na interní, ale i na externí zdroje. Jak už bylo zmíněno výše, formát OOXML defaultně nepodporuje makra. Je možné uložit dokument s podporou maker, v tom případě ale uživatel rozezná takový typ dokumentu (je uložen s koncovkou „m“ místo „x“). Pokud bude dokument s podporou maker otevírán jako dokument bez podpory (změna koncovky), aplikace vrátí chybu. [8]

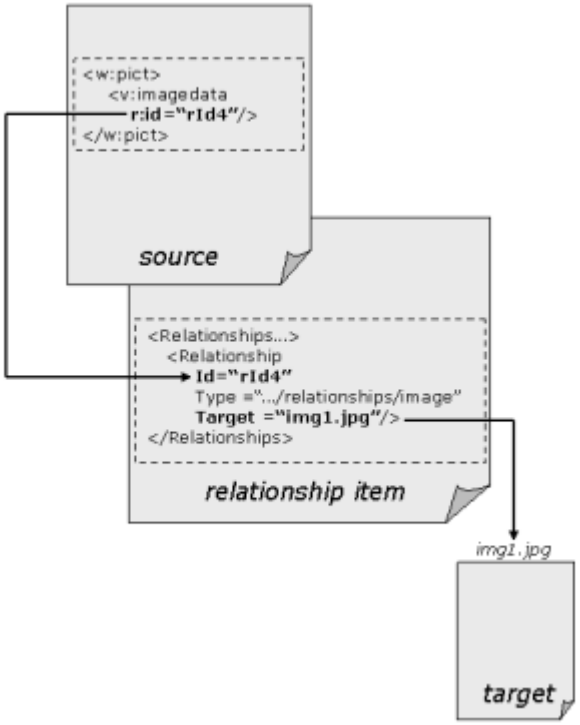
2.4.2 Tvorba OOXML

Pro práci s OOXML teoreticky stačí pouze prostředek pro manipulaci se ZIP archivy. Přesto existuje řada nástrojů pro práci s OOXML vydaných přímo společností Microsoft. Nejvýznamnějšími nástroji jsou patrně *Office Open XML SDK* (nyní již ve verzi 2.0) a *PowerTools for Open XML*. Tyto nástroje umožňují vytváření dokumentů, změnu jejich obsahu a mnohé další. Pokud se zaměřím na vytváření dokumentů spojením již existujících, jsou známy minimálně dvě různé možnosti, jak takový dokument vytvořit:

- **altChunk** je speciální vysokoúrovňový přístup skládání Word OOXML dokumentu pomocí značkování, který umožňuje do těla dokumentu vkládat jiný OOXML dokument, nebo html stránku. Použité dokumenty jsou přibaleny do balíku cílového dokumentu a v těle jsou na ně vytvořeny odkazy.



Obrázek 2.3: Implicitní odkazování v OOXML (zdroj: msdn.microsoft.com).



Obrázek 2.4: Explicitní odkazování v OOXML (zdroj: msdn.microsoft.com).

- **DocumentBuilder** je třída z PowerTools for Open XML určená k vytváření dokumentů. Cílový dokument je přímo vytvořen ze zdrojových a nejedná se o pouhé odkazování.

Rozdíl mezi těmito přístupy je zřejmý – dokument pomocí altChunk bude snadné a rychlé vytvořit, ale vzroste zátěž stroje při otevírání dokumentu. Naproti tomu se bude nový dokument pomocí DocumentBuilder vytvářet déle, avšak jeho otevírání bude méně náročné.

Podepisování OOXML dokumentů je možné realizovat mnoha způsoby. Nejsnazší možností je vkládání interních podpisů ve formátu *XMLDsig*, stačí k tomu knihovny *System.IO.Packaging* (.NET Framework 3.0 a vyšší) a *System.Security.Cryptography* (.NET Framework 1.1 a vyšší). V aplikacích Office od verze 2010 existuje možnost podepisování OOXML ve formátu *XAdES* (viz kapitola 3.6).

2.4.3 Shrnutí OOXML

Formát OOXML je mnohem mladší a také méně rozšířený než formát PDF. Pravděpodobně z těchto důvodů není zvykem využívat dokumenty ve formátu OOXML k archivaci. Díky podpoře ze strany tvůrců balíku Office vidím snahu o zavedení uznávaných standardů podpisových formátů, což by mohlo formátu OOXML z pohledu archivace v budoucnu prospět. V současnosti lze tento nedostatek řešit konverzí OOXML dokumentů do formátu PDF (k tomu může posloužit například knihovna DocX, určená pro platformu .NET). Nevýhodou OOXML je, že mimo platformu .NET má méně kvalitní podporu. K formátu OOXML existuje otevřený standard a mnoho aplikací tak s tímto formátem umí zacházet. Dále je pro uživatele příjemné, že si může snadno vytvořit šablony pro tvorbu dokumentů v aplikaci typu Office. Nakonec nemohu nezmínit, že se jedná o formát, který podporuje ukládání podpisů v různých formátech.

2.5 OpenDocument

OpenDocument (OD) je v plném znění znám jako *Open Document Format for Office Applications* (ODF). Byl původně vyvinut společnosti SunMicrosystems jako formát *OpenOffice.org XML*. Poté byla jeho specifikace vydána or-

ganizací OASIS, navíc byl při vydání verze 1.1 publikován také jako ISO/IEC standard (ISO/IEC 26300:2006/Amd 1:2012).

2.5.1 Struktura ODF

ODF podporuje dvě odlišné varianty reprezentace:

- **Reprezentace pomocí jediného XML souboru.** Známá také jako ploché XML (*Flat XML*), nebo nekomprimované XML (*Uncompressed XML*). Tato reprezentace není široce užívaná a nemá v technické specifikaci standardizované přípony. Běžně se ale používají přípony .xml, .fodt, .fods aj.
- **Jako kolekce poddokumentů v standardním ZIP archivu s pevně danou strukturou.** Každý z poddokumentů má vlastní kořenový dokument. Používá standardizované přípony .odt, .ods, .odp a .odg.

Stejně jako OOXML, ODF podporuje šablony (templates). Jedná se soubory, které obsahují informace o formátování (včetně stylů) bez samotného obsahu.

ODT podporuje ukládání metadat. Pro tento účel má vlastní sadu metadat, kterou lze uživatelsky rozšířit.

Objekty, které může OD obsahovat, jsou dvojího druhu:

- **Objekty, které jsou v OD zastoupeny** (vzorce, grafy, tabulky, textové dokumenty, kresby, prezentace)
- **Objekty, které nemají XML reprezentaci a mají pouze binární reprezentaci.** Jedná se například o OLE objekty, bitmapy atd.

ODF umožňuje referencování uvnitř balíku pomocí takzvaných *IRIs* (*Internationalized Resource Identifiers*) lokátorů. Neumožňuje tedy na rozdíl od OOXML referencovat externí zdroje.[9]

Uncompressed varianta obsahuje standardně následující části:

- XML soubory:

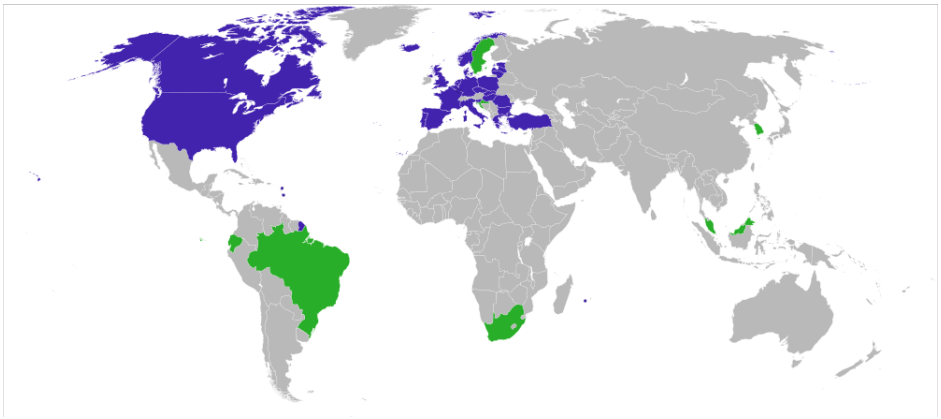
  - content.xml – nese nebinární obsah dokumentu
  - styles.xml – obsahuje informace o stylech
  - meta.xml – obsahuje metadata
  - settings.xml – obsahuje data o nastaveních, která se netýkají obsahu ani rozložení (například zoom, pozice kurzoru atd.)
- Ostatní soubory:
  - mimetype – jednořádkový soubor s informací o mimetypu dokument
- Složky:
  - META-INF – obashuje manifest.xml s informacemi o souborech obsažených v OD balíku (seznam všech obsažených souborů, jejich media typ a pokud jsou šifrované, tak i informace potřebné k dešifrování)
  - Thumbnails – thumbnail.png – mělo by se jednat o náhled na první stranu dokumentu o rozměrech 128x128 pixelů[9]

2.5.2 Tvorba ODF

Pro práci s ODF existuje řada knihoven, ale pouze několik je jich určeno pro platformu .NET. Nejznámější z nekomerčních knihovnen pro .NET je *AODL*. Její kvalita je však nízká, podporuje ODF pouze do verze 3.2 a nyní se zdá být její vývoj neaktivní. Pro platformu .NET tedy nezbývá jiná možnost než použít knihovny pro Javu (*ODFDOM*, *Simple ODF API for Java*, nebo *JOpenDocument*) a Java VM pro .NET, nebo zaplatit zhruba 7500 Kč za komerční knihovnu *ODF.NET* od společnosti Independent. Nekomerční nástroje pro Javu umožňují veškerou základní práci s dokumentem a na této úrovni jsou postačující. Pokud bych chtěl spojovat již existující dokumenty, musel bych si tuto funkčnost sám doplnit. Vysokoúrovňové skládání, které by samo zajišťovalo shodu stylů, buď nefunguje zcela dobře, nebo jej neumí vůbec. Knihovna ODF.NET tuto funkčnost poskytuje.

2.5.3 Shrnutí ODF

ODF a OOXML jsou v mnohých směrech podobné formáty. Používají stejné technologie k ukládání dat. ODF umožňuje interní připojování elektronických



Obrázek 2.5: Státy, které si ODF osvojily (zeleně), a členové NATO (modře), kterým byl uložen jako standard povinně (zdroj: wikipedia.org).

podpisů obdobně jako formát OOXML. ODF ale, na rozdíl od OOXML, nepodporuje například referencování externích zdrojů a také má výrazně horší podporu pro platformu .NET. Důvodem, proč je tento formát zajímavý, je fakt, že jej dosud jako národní standard osvojila řada zemí (například Brazílie, Chorvatsko, Japonsko a další) a také byl zaveden povinně jako standard členům NATO (obrázek 2.5)[10]

2.6 XSL-FO

Extensible Stylesheet Language - Formatting Objects (XSL-FO) je značkovací jazyk, který lze chápat jako speciální případ dokumentového formátu. Zjednodušeně se jedná o spojení XML dokumentu s XSL transformací, která určuje výsledný vzhled dokumentu. XSL-FO spadá do technologií W3C. V roce 2001 byla vydána specifikace 1.0 a v roce 2006 vydala organizace W3C specifikaci verze 1.1. XSL-FO využívá výhradně prezentačního a nikoliv sémantického značení (na rozdíl od HTML).

Jeho hlavní myšlenkou je, aby byl používán jako formát pro transformace. Dokument by se tedy netvořil přímo jako XSL-FO, ale byl by popsán v nějakém jiném XML jazyce (XHTML, DocBook, ale i jakýkoliv jiný XML jazyk). Dále by se takový dokument převedl pomocí XSLT transformace na XML-FO formát. XSL-FO dokument pak zpracovává takzvaný FO procesor, který z dokumentu může vygenerovat jiný typ dokumentu (zpravidla PDF

či PostScript, může ale například i RTF), nebo jej může přímo zobrazit na obrazovku.[11]

2.6.1 Struktura XSL-FO

XSL-FO má v zásadě velmi jednoduchou strukturu. Jedná se o XML dokument, který nemusí nutně vyhovovat konkrétnímu DTD schématu. Stačí, když bude vyhovovat XSL-FO specifikaci. XSL-FO dokument obsahuje dvě povinné části. První obsahuje seznam pojmenovaných rozložení stránky. Druhá část obsahuje seznam dat se značkováním, který odkazuje na rozložení stránek, a tak určuje, jak obsahová data vyplní jednotlivé stránky.

První část definuje veškeré nastavení stránek. Mezi ty patří hlavně velikost stránek, odsazení textu a například i definování sekvencí pro sudé a liché stránky.

Druhá část je rozdělena na toky, které vždy patří určité definici stránky. Ty pak dále obsahují seznam bloků, kde každý blok obsahuje seznam textových dat, řádkových značkovacích prvků anebo kombinaci obojího. Bloky a řádkové prvky fungují podobně jako v CSS (tedy i většina formátovacích prvků se dědí na vnořené FO).[12]

2.6.2 Tvorba XSL-FO

Tvorbu XSL-FO lze rozdělit do několika fází:

1. **Tvorba XML dokumentu s obsahem.** V této fázi lze použít celou řadu přístupů. Je možné vygenerovat si vlastní XML, které bude mít námi určenou strukturu. Také je možné využít některý formát, který je vyjádřen pomocí XML (například *XHTML*, *DocBook* a další) - mnoho nástrojů bude podporovat export do takových formátů.
2. **XSLT transformace na XSL-FO.** K tomuto kroku je zapotřebí *XSLT* transformace a *XSLT procesor*. Na internetu je k nalezení celá řada šablon XSLT transformací pro převod XML na XSL-FO (zvláště pak pro typizované formáty typu DocBook). Mezi kompilační nástroje patří například *Saxon*, *xsltproc* nebo *MSXML*.

3. **Konverze do jiného formátu / zobrazení na monitor.** Pokud je k dispozici XSL-FO soubor, je potřebné mít ještě *XSL-FO procesor*, aby bylo možné si dokument prohlédnout nebo převést na jiný formát. Zde se nabízí kvalitní open-source nástroj *FOP* (od společnosti Apache) s podporou českého jazyka, který je určen pro Javu a existuje k němu port pro .NET. Jmenuje se *nfop* a jedná se o port v jazyce J#. Alternativou je projekt *FO.NET*. Tato knihovna nabízí kvalitní tvorbu PDF, včetně komprese. Knihovna FOP však podporuje více výstupních typů souborů (například PostScript nebo PCL) a umožňuje zobrazit zformátované XSL-FO na obrazovku.

2.6.3 Shrnutí XSL-FO

XSL-FO zdaleka není ideálním formátem. Neexistuje k němu dosud žádný nástroj, který by poskytl uživatelsky příjemné prostředí pro formátování (žádný WYSIWYG editor), tvorba složitějšího vzhledu je náročná nebo nejsou některé vlastnosti vůbec podporovány. Navíc si běžný uživatel tento dokument ani nezobrazí ve zformátovaném stavu na monitor. Proč jsem jej tedy mezi zkoumané formáty zahrnul? Důvodem je, že tímto směrem XSL-FO nemíří. Jeho cílem je stát se prostředkem pro dosažení cílového formátu (většinou PDF). Typickým příkladem užití bude generování PDF z dat obsažených v objektu uloženém v databázi – chceme do PDF vygenerovat seznam zaměstnanců, fakturu, smlouvu – tedy cokoliv, co má jasně danou, málo proměnlivou strukturu a nelze očekávat časté změny vzhledu dokumentu. Lze zajisté oponovat, že takových konvertorů existuje celá řada nad různými formáty (XHTML i DocBook mají také své konvertory do PDF), XSL-FO se ale neomezuje na jeden formát XML dokumentu, ale umožňuje generování nad jakýmkoli XML. XSL-FO také podporuje vkládání libovolných XML dat pomocí tagu foreign-object (nejoblíbenějšími objekty jsou SVG křivky).

2.7 Rich Text Format

Rich Text Format (dále jen RTF) je platformě nezávislý, proprietární dokumentový formát vyvinutý společností Microsoft. Jeho specifikace byla uveřejněna již v roce 1987 a jedná se tedy o nejstarší dokumentový formát z těch, které zde porovnávám. V průběhu let byly vydávány novější verze specifikace (převážně spolu s vydáním nové verze MS Office) a v roce 2008 byla vydána

dosud poslední verze 1.9.1. Po vydání MS Office 2010 vydal Microsoft stanovisko, že nebude specifikaci RTF dále rozvíjet, a tak některé vlastnosti MS Word 2010 a novějších nebudou s formátem RTF kompatibilní.[13]

2.7.1 Struktura RTF

RTF je ukládán jako jeden soubor obsahující prostý text. Je konstruován tak, že je i v neinterpretovaném stavu čitelný člověkem (*Human readability*). Pokud se jedná o RTF ukládané aplikací MS Word, je čitelnost ztížena, protože je přidáváno značné množství řídících znaků. Ve formátovacím vyznačování se autoři (Richard Brodie, Charles Simonyi a David Luebbert) nechali inspirovat značením formátu *TeX*.

Řídící znaky:

- **Složené závorky {}** definují skupinu. Skupiny se mohou libovolně zanořovat.
- **Zpětné lomítko** \ začíná řídící kód RTF. Každý RTF dokument by měl začínat řídícím kódem \rtf (například formátování tučného textu je řízeno sekvencí \b atd.).

RTF dokument umožňuje použít pouze 7bitové ASCII znaky. Pokud mají být ve výsledném textu jiné znaky, budou v RTF souboru zastoupeny pomocí escape sekvence. Znaky z kódování Windows budou vkládány ve tvaru \'ff (kde ff představuje dvě hexadecimální číslice). Znaky z kódování Unicode budou ve tvaru \u0000 (kde 0000 představuje znaménkové 16bitové decimální číslo).

Vložené objekty:

- **OLE objekty** – pokud interpret formátu OLE objektu nerozumí, může ho v zobrazení nahradit bitmapou, nebo jej nezobrazit vůbec.
- **Obrázky** – je umožněno vkládat v několika vybraných formátech (JPEG, PNG). Pokud program, který RTF vytváří, má vložit obrázek jiného formátu, zpravidla si program obrázek nejprve zkonvertuje do podporovaného formátu, nebo jej nevloží vůbec. Některé programy zapisují obrázky ve dvou různých formátech zároveň, kvůli lepší kompatibilitě s produkty MS Windows. Jeden bude ve formátu JPEG, nebo PNG a

druhý ve formátu Windows Meta File (WMF). Ukládání obrázku ve dvou formátech pochopitelně značně zvětšuje velikost výsledného RTF dokumentu. atd.).

- **Písma** – mohou být vložena a zároveň jsou podporovány generické názvy rodiny písma pro nahrazení chybějících písem. Ani jedna z těchto vlastností však není interprety široce podporována.
- **Tvary** – RTF podporuje základní kreslící objekty (obdélníky, čáry, šipky, mnohoúhelníky, elipsy a další). Ani RTF tvary nejsou interprety příliš podporovány.

RTF nepodporuje spouštění maker, což zvyšuje bezpečnost formátu vůči šíření virů. Samotná přípona ale nic neznamená, takže je možné, aby šiřitel viru uložil dokument obsahující škodlivý kód ve formátu DOC a příponu souboru následně změnil na .rtf. V aplikaci MS Word je tento dokument otevřen jako jakýkoliv jiný dokument formátu DOC (tedy včetně provedení maker), aniž by byl uživatel upozorněn, že se nejedná o RTF. Nejedná se tedy o zcela bezchybný ochranný prvek. Pokud by byla přípona zmíněného dokumentu změněna z DOC na DOCX, je otevírání souboru v aplikaci MS Word přerušeno chybou.[14]

2.7.2 Tvorba RTF

Pro práci s formátem RTF existují dva druhy knihoven:

- pro generování RTF (*jRTF*, *RTF Document Constructor Library*)
- pro manipulaci s RTF (*NRTFTree*, *Gios WORD .NET Library*)

Právě druhá kategorie je zajímavější, protože je využívána nejen pro tvorbu nového RTF, ale i pro skládání z již existujících dokumentů. Takovou funkcionalitu sice neposkytují přímo, dá se ale poměrně snadno pomocí jejich prostředků naimplementovat. Větší problém je ohledně souladu se specifikací formátu. Knihovny buď odpovídají pouze některé z historických verzí RTF, nebo nepodporují žádnou konkrétní verzi.

2.7.3 Shrnutí RTF

Specifikace RTF je zveřejněna poměrně dlouho dobu. Zpočátku ale veřejně dostupná nebyla, a proto měli vývojáři společnosti Microsoft oproti ostatním vývojářům výhodu. To vedlo k aktuálnímu stavu, ve kterém s tímto formátem (alespoň v nějaké verzi) umí pracovat většina textových procesorů, málokterý interpret však podporuje všechny vlastnosti RTF dokumentu a výsledky interpretace se liší. Další nevýhodou je, že k formátu RTF neexistují knihovny, které by umožňovaly vysokoúrovňovou manipulaci, zároveň by plně podporovaly nejnovější specifikaci a byly by aktivně vyvíjené. Poslední zjištěnou nevýhodou je, že plochá struktura formátu RTF neumožňuje interně připojovat elektronický podpis ani časová razítka. Jakýkoliv podpis tedy musí být připojován externě. Spolu se skutečností, že Microsoft nebude formát RTF dále rozvíjet, jsou to důvody, proč se podporou RTF příliš nezabývat.

V kapitole 5.1 jsem provedl analýzu výše uvedených informací, zhodnocení přínosu jednotlivých formátů a jejich vzájemné porovnání.

3 Elektronické podpisy a časová razítka

3.1 Základní pojmy

Podle zákona č. 227/2000 Sb. se **elektronickým podpisem** rozumí *„údaje v elektronické podobě, které jsou připojené k datové zprávě nebo jsou s ní logicky spojené, a které slouží jako metoda k jednoznačnému ověření identity podepsané osoby ve vztahu k datové zprávě“*.[15] Pokud bych měl tuto definici vysvětlit, pak se elektronickým podpisem míní obecný identifikátor připojený k datové zprávě, který umožňuje jednoznačnou identifikaci podepsané osoby. V takovém smyslu se nejedná o elektronický podpis, jak je běžně chápán, a neposkytuje žádnou významnou informaci. Zákon č. 227/2000 Sb. však zavádí další pojmy: **zaručený elektronický podpis a uznávaný elektronický podpis**.

Zaručený elektronický podpis je takový *„elektronický podpis, který splňuje následující požadavky*

1) je jednoznačně spojen s podepisující osobou,

2) umožňuje identifikaci podepisující osoby ve vztahu k datové zprávě,

3) byl vytvořen a připojen k datové zprávě pomocí prostředků, které podepisující osoba může udržet pod svou výhradní kontrolou,

4) je k datové zprávě, ke které se vztahuje, připojen takovým způsobem, že je možno zjistit jakoukoliv následnou změnu dat.“[15]

Podle tohoto znění je zaručený elektronický podpis tím, čím je běžně chápán elektronický podpis. Proto pokud bude v následujícím textu uveden elektronický podpis a zároveň nebude zdůrazněno, že se jedná o elektronický podpis „bez přívlastku“, bude myšlen jako zaručený elektronický podpis.

Uznávaný elektronický podpis je podle téhož zákona zaručený elektronický podpis, který je navíc založen na kvalifikovaném certifikátu, vydaném buď akreditovaným poskytovatelem certifikačních služeb (jeho definice, práva a

povinnosti jsou uvedeny v témž zákoně), nebo jeho zahraniční obdobou.

Kvalifikované časové razítko je „datová zpráva, kterou vydal kvalifikovaný poskytovatel certifikačních služeb a která důvěryhodným způsobem spojuje data v elektronické podobě s časovým okamžikem, a zaručuje, že uvedená data v elektronické podobě existovala před daným časovým okamžikem“[15].

3.2 Princip elektronického podpisu

V definici zaručeného elektronického podpisu je důležitá fráze „ je k datové zprávě, ke které se vztahuje, připojen takovým způsobem, že je možno zjistit jakoukoliv následnou změnu dat“. Jedná se o zcela zásadní fakt, který udává smysl elektronického podpisu. Jakmile kdokoli po podepsání (prakticky) jakkoliv změní dokument, je to zjistitelné.

Aby byla zajištěna tato vlastnost, využívá se k podpisu asymetrické šifrování. Používá se dvojice klíčů (soukromý, veřejný). Soukromý klíč je určen pouze pro uživatele, který k dokumentu připojuje svůj podpis. Veřejný klíč je pak přidán přímo do podpisu a může ho použít kdokoliv, aby zjistil, zda nebyla porušena integrita dokumentu.

3.2.1 Proces podepisování

Proces podepisování funguje následovně. Hashovací funkcí (například sha-1, sha-2) je spočten hash dokumentu, ten je následně pomocí soukromého klíče zašifrován a spolu s veřejným klíčem a případnými dalšími informacemi (certifikát, systémový čas podepsání) připojen jako elektronický podpis k dokumentu na určené místo (mluvíme pak o interním podpisu), nebo je připojen jako samostatný soubor (externí podpis).

3.2.2 Proces ověřování

Uživatel, který si chce ověřit platnost dokumentu, spočte stejnou hashovací funkcí hash dokumentu (pokud byl podpis připojen jako interní, pak samozřejmě bez něj) a dešifruje pomocí veřejného klíče přiložený hash. Pokud jsou

tyto dva hashe totožné, znamená to, že nebyl dokument změněn (autentičnost). Z principu je možné, aby byl dokument pozměněn a uživateli, který podpis ověřuje, byl poskytnut falešný veřejný klíč.

3.3 Proces ověřování

K zabránění podvržení veřejného klíče slouží certifikační autority, které certifikáty vydávají. Certifikační autority potvrzují, že veřejný klíč opravdu patří osobě, která je v certifikátu uvedena. Pokud uživatel důvěřuje certifikační autoritě, může důvěřovat i vydanému certifikátu. Certifikační autoritu může provozovat každý sám a pak je problém v určení, komu může věřit. V praxi lze jako certifikační autority uznávat například banky.

3.4 Platnost elektronického podpisu

K zabránění podvržení veřejného klíče slouží certifikační autority, které certifikáty vydávají. Certifikační autority potvrzují, že veřejný klíč opravdu patří osobě, která je v certifikátu uvedena. Pokud uživatel důvěřuje certifikační autoritě, může důvěřovat i vydanému certifikátu. Certifikační autoritu může provozovat každý sám a pak je problém v určení, komu může věřit. V praxi lze jako certifikační autority uznávat například banky.

3.4.1 Nezpochybnitelnost

Pokud však uživatel komunikuje se státními institucemi, je zákonem dáno, že jako nezpochybnitelný je brán pouze uznávaný elektronický podpis. Ten je založen na kvalifikovaném certifikátu, tj. certifikátu vydaném kvalifikovanou certifikační autoritou. V České republice byly v roce 2012 pouze tři kvalifikované (a zároveň akreditované) certifikační autority – PostSignum, I.CA a Eldentity. Další možností je použít uznávané zahraniční autority. Kvalifikovaný certifikát je vydán pouze na základě ověření identity žádající osoby, a to pouze na omezenou dobu (podpisový certifikát na dobu 1 roku). Certifikační autorita ručí za časově omezenou platnost vydaného certifikátu, pokud během této doby nedojde k revokaci certifikátu (zneplatnění na žádost majitele).

3.4.2 Okamžik posuzování platnosti podpisu

Jak bylo řečeno, certifikát platí pouze po určitou dobu. Nejjednodušší možností je zkoumat platnost certifikátu v okamžiku, kdy podpis ověřujeme. Nejedná se však o efektivní využití elektronického podpisu. Pokud uživatel podepsal dokument certifikátem, který byl vydán 1. 1. 2012 a k podpisu byl využit 1. 12. 2012 (v té době stále platný), pak při posuzování platnosti 2. 1. 2013 by bylo zjištěno, že certifikát, na jehož základě byl podpis vystaven, není platný, a proto nemůže být podpis vyhodnocen jako platný. Mnohem logičtější je tedy posuzovat platnost certifikátu vůči okamžiku, kdy byl podpis vytvořen. Problémem je takový okamžik určit. Elektronický podpis v sobě sice zpravidla nese informaci o čase vystavení, ta se však vytváří podle systémového času, a může tak být podvržena. Z tohoto důvodu se zavádí nezpochybnitelný mechanismus – takzvané časové razítko.

3.4.3 Časové razítko a kvalifikované časové razítko

Časové razítko je značka, podobně jako elektronický podpis, kterou si však uživatel nevystavuje sám, ale je mu vystavena časovou autoritou. Postup vytváření časového razítka připomíná do jisté míry podepisování dokumentu elektronickým podpisem. Uživatel, který má v úmyslu dokument k danému okamžiku opatřit časovým razítkem, nejprve vypočte hash dokumentu. Pokud má časové razítko dokazovat existenci elektronického podpisu v tomto čase, je potřeba zahrnout i elektronický podpis do výpočtu hashe. Tento hash uživatel odešle poskytovateli certifikačních služeb. Ten k hashi připojí přesný časový údaj a celou tuto informaci zašifruje vlastním soukromým klíčem. Zároveň poskytne veřejný klíč pro ověření platnosti časového razítka a výsledek pošle uživateli, který může následně časové razítko připojit k dokumentu. Proces ověřování pak probíhá analogicky jako u elektronického podpisu.

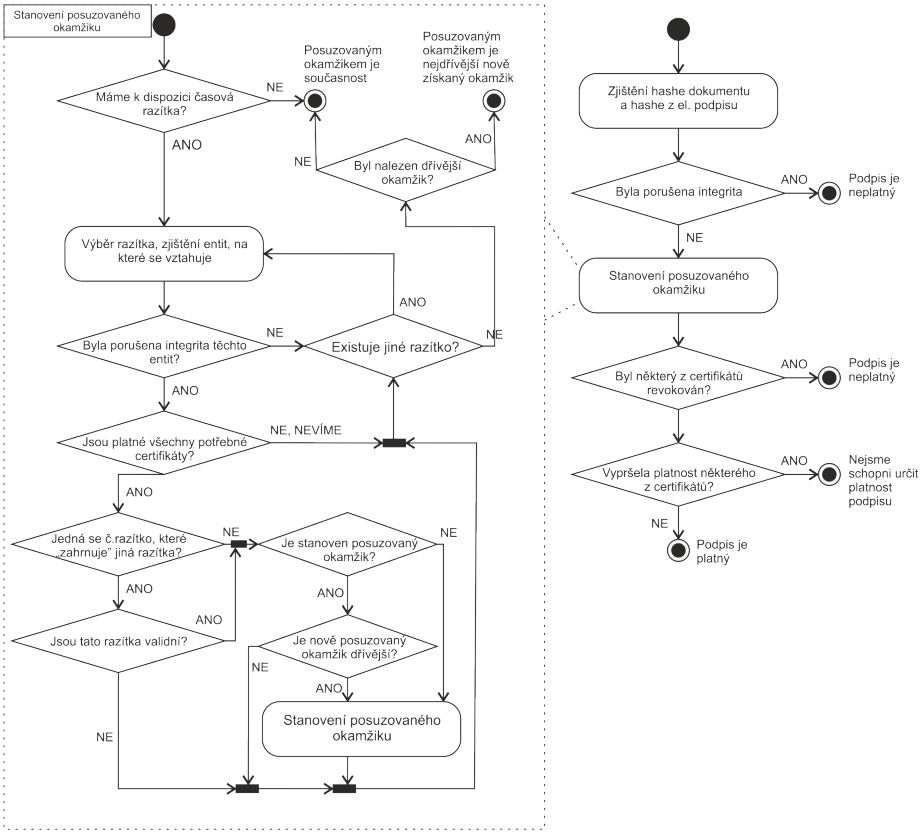
Kvalifikované časové razítko vydávají pouze kvalifikovaní poskytovatelé certifikačních služeb, vydávající kvalifikovaná časová razítka (v roce 2012 to byly v ČR pouze společnosti PostSignum, I.CA a Eldentity). Česká legislativa nepoužívá pojem časové razítko, ale rovnou zavádí pojem kvalifikované časové razítko.

Pomocí časových razítek lze získat ověřený časový údaj, ke kterému lze o platnosti podpisu uvažovat. Pokud se vrátíme k příkladu uvedenému v kapitole 3.3.2 a dokument z 1.12.2013 by uživatel nejen podepsal, ale i opatřil časo-

vým razítkem, byl by dokument 2.1.2013 ověřen jako platný. Pro certifikát časového razítka však stejně jako pro certifikát elektronického podpisu existuje doba platnosti (na rozdíl od elektronického podpisu je to delší časový interval, běžně od 3 do 6 let). Proto i zde musí uživatel počítat s tím, že platnost certifikátu vyprší. To je, hlavně u dokumentů určených k archivaci, nežádoucí. Aby uživatel umožnil ověřit podpis jako platný po neomezeně dlouhou dobu, musí ještě před vypršením certifikátu autority časového razítka opatřit dokument dalším časovým razítkem. U dokumentů určených k archivaci to představuje značnou režii spojenou s kontrolou toho, zda se neblíží konec platnosti certifikátu.

3.4.4 Revokace certifikátu a její vyhodnocení při ověřování podpisu

V kapitole 3.3.1 bylo zmíněno, že si uživatel může nechat certifikát elektronického podpisu na vlastní žádost revokovat. Prvořadým důvodem pro revokaci bude odcizení certifikátu, respektive jeho soukromého klíče. Konkrétní proces revokace zákon neupravuje a je na každé certifikační autoritě, jak jej zavede. Obecně však bude dostupný mechanismus, pomocí kterého se držitel autentizuje a zažádá autoritu o revokaci. Ta má ze zákona 24 hodin na to, aby revokaci certifikátu zveřejnila. Během vyhodnocování platnosti elektronického podpisu musí být bráno v úvahu, zda byl certifikát revokován. Zjištění této informace může být provedeno v reálném čase, nebo opožděně. Zjišťování platnosti certifikátu v reálném čase bývá nejčastěji realizováno pomocí protokolu OSCP (Online Certificate Protocol). Protokol OSCP umožňuje vznést dotaz a obdržet odpověď serveru certifikační autority v reálném čase. Jedná se o způsob, který promítá revokace v nejkratším možném čase, který však není konkrétně specifikován. Z českých certifikačních autorit jej poskytuje od roku 2011 I.CA a nově od prosince 2012 i PostSignum. Důvodem, proč tuto službu české certifikační autority neposkytují vůbec, nebo až v poslední době, je, že není vyžadována zákonem. Zákon vyžaduje poskytování opožděné informace pomocí CRL (Certificate Revocation List) – seznam revokovaných certifikátů. Ani zde nejsou služby poskytující CRL jednotné. Jedním druhem CRL jsou kumulativní seznamy, které obsahující informace o všech revokovaných certifikátech dané certifikační autority (po dobu alespoň 10 let, po kterou má autorita ze zákona povinnost udržovat informace). Takové CRL poskytuje v ČR autorita EIdentity. Druhým druhem jsou intervalové seznamy, které obsahují souhrn revokací za určitý časový interval. Zákonem jsou přípustné, ale nikde není určena délka takového intervalu. V ČR tímto



Obrázek 3.1: Vývojový diagram ověření platnosti podpisu (zdroj: autor).

způsobem poskytují CRL I.CA a PostSignum. Délka takového intervalu u PostSignum je zhruba 5 měsíců (ačkoliv to není explicitně řečeno). V praxi je délka platnosti ceritfikátu elektronického podpisu jeden rok, uživatel tedy musí informaci zjišťovat z několika CRL.

3.5 Postup ověřování podpisu

Z výše uvedených požadavků lze sestavit proces, jakým musí být podpis ověřen (schéma na obrázku 3.1).

3.6 Elektronický podpis v praxi

3.6.1 Odpovědnost za škodu

Jedním z důvodů, proč se elektronickými podpisy zabývat, je osvěta uživatelů, kteří mnohdy neznají jejich principy a vystavují se tak riziku, že dojde ke škodě, která na nich může být vymáhána. Zejména se jedná o povinnost držitele soukromého klíče, aby se o klíč staral tak, aby nemohlo dojít k jeho zneužití a pokud by k něčemu takovému došlo, povinnost využít revokaci certifikátu. Tato povinnost je uvedena v §5 zákona č. 227/2000 Sb.

„Podepisující osoba je povinna

a) zacházet s prostředky, jakož i s daty pro vytváření zaručeného elektronického podpisu s náležitou péčí tak, aby nemohlo dojít k jejich neoprávněnému použití,

b) uvědomit neprodleně poskytovatele certifikačních služeb, který vydal kvalifikovaný certifikát, o tom, že hrozí nebezpečí zneužití jejích dat pro vytváření zaručeného elektronického podpisu.“[15]

Z této povinnosti vyplývá i odpovědnost za škodu:

„Za škodu způsobenou porušením povinností podle odstavce 1 odpovídá podepisující osoba podle zvláštních právních předpisů. Odpovědnosti se však zprostí, pokud prokáže, že ten, komu vznikla škoda, neprovedl veškeré úkony potřebné k tomu, aby si ověřil, že zaručený elektronický podpis je platný a jeho kvalifikovaný certifikát nebyl zneplatněn.“[15]

Druhá část výše uvedeného odstavce je taktéž důležitá pro běžného uživatele. Pokud mu vznikne škoda způsobená tím, že spoléhal na podpis vzniklý zneužitím podpisového certifikátu a zároveň neprovedl všechny kroky pro ověření platnosti podpisu, pak mu nárok na kompenzaci škody zaniká.

3.6.2 Současný stav podpory elektronických podpisů v aplikacích

Pokud si uvědomíme, jaké povinnosti máme, je dobré si zjistit, v jakém stavu je vůči legislativě podpora elektronických podpisů.

V knize Báječný svět elektronického podpisu RNDr. Ing. Jiří Peterka poukazuje na to, že Adobe Reader nemůžeme považovat za bezpečný nástroj pro vyhodnocování elektronického podpisu. Jeho chování musíme nejprve nastavit, aby fungovalo v souladu s českou legislativou. Například určení „posuzovaného okamžiku" platnosti certifikátu podpisu se řídí nastavením. Toto nastavení může nabývat jedné ze tří hodnot – platný čas (aktuální systémový čas), bezpečný čas (pokud je připojeno časové razítko, zjistí se časový údaj z něj, jinak se vezme aktuální systémový čas) a třetí možnost - čas, kdy byl podpis vytvořen (bez ohledu na to, zda pochází z časového razítka, nebo ze systémových hodin). Zarážející je fakt, že implicitní nastavení aplikace Adobe Reader je možnost třetí, tedy čas, kdy byl podpis vytvořen, bez ohledu na původ. To je pochopitelně špatně, neboť může jít o podvržený čas (viz kapitola 3.3.2), uživatel může být klamán, že se jedná o správný podpis a jelikož neprovedl řádnou kontrolu elektronického podpisu v souladu s legislativou, může přijít o nárok na náhradu škody. Správnou volbou tedy má být „bezpečný čas". Další nebezpečnou funkcionalitou může být zobrazování identity z popisku podpisu a nikoliv z certifikátu, může tedy dojít k mylné identifikaci (ačkoliv v detailech podpisu je již uvedeno správné jméno osoby).[16]

Po prostudování dalších případů aplikací v téže knize RNDr. Ing. Jiřího Peterky lze konstatovat, že stav aplikací není ideální a téměř všude nalezneme alespoň zavádějící informace, které by si uživatel bez znalostí digitálních podpisů mohl špatně vyložit.

3.7 Formáty pokročilých el. podpisů

3.7.1 Přehled

V této práci budu často zmiňovat formáty pokročilých elektronických podpisů (*Advanced Electronic Signatures*). Z hlediska procesu podepisování dokumentů elektronickým podpisem a z hlediska následného ověřování elektro-

nického podpisu nepřináší formáty pokročilých elektronických podpisů nic nového. Jedná se o formáty, které jsou svojí strukturou uzpůsobeny k tomu, aby veškeré informace, které jsou k ověření elektronického podpisu potřebné, byly uchovány v jedné struktuře. Kromě samotného elektronického podpisu se jedná o certifikát, na jehož základě byl podpis vystaven, veškeré jemu nadřazené certifikáty, ke všem uvedeným certifikátům příslušné CRL seznamy a časová razítka, která udrží časovou kontinuitu od doby podepsání po dobu ověření. Dohromady tyto informace naplňují koncept dlouhodobého ověření – LTV (*Long Term Validation*). Evropský institut pro telekomunikační standardy (ETSI) vytvořil sadu norem, které se vztahují k elektronickým podpisům a časovým razítkům, mezi nimi i normy, které popisují tři standardy pokročilých elektronických podpisů:

- **PAdES – PDF Advanced Electronic Signatures** (určené pro elektronický podpis dokumentů ve formátu PDF).
- **XAdES – XML Advanced Electronic Signatures** (určené pro elektronický podpis ve formátu XML, který umožňuje podepsání části XML dokumentu nebo externího obsahu bez ohledu na to, v jakém formátu tento obsah je).
- **CAdES – CMS Advanced Electronic Signatures** (určené pro elektronický podpis ve formátu CMS, který umožňuje podepisování zpráv včetně jejich veškerého obsahu).

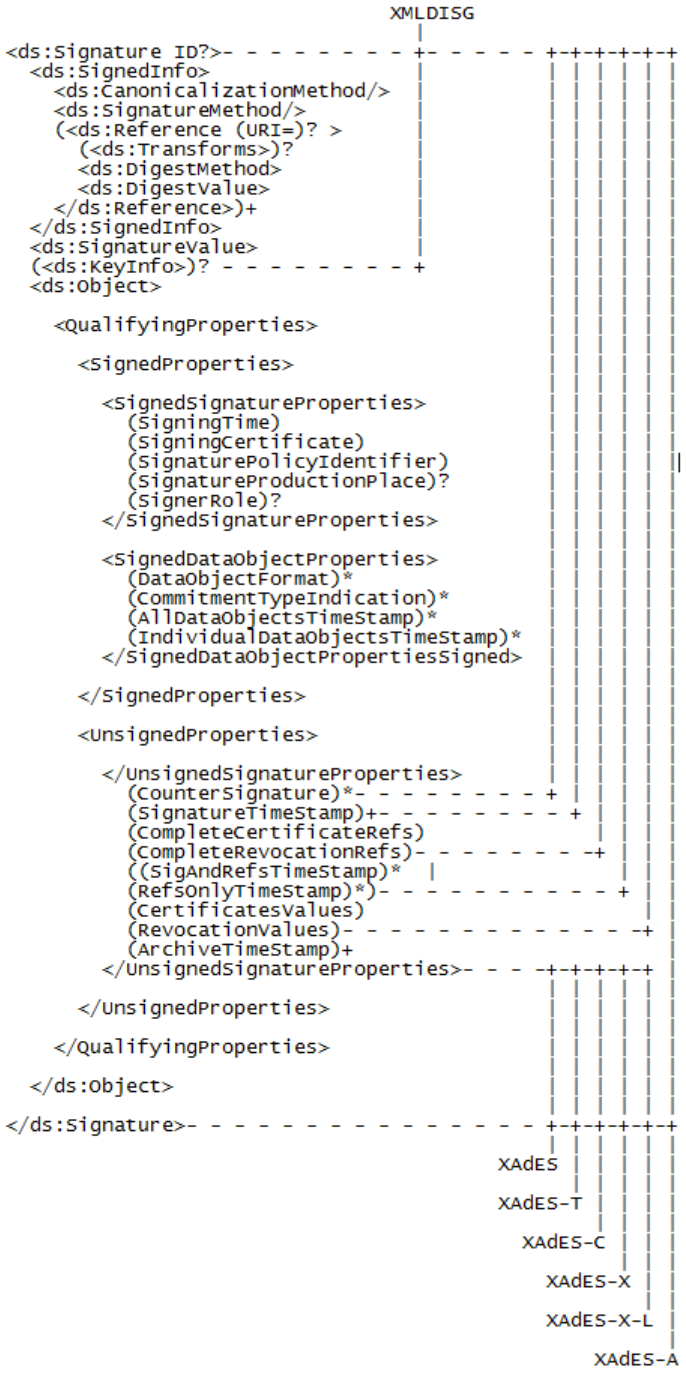
Evropská komise svým výnosem 2011/130/EU tyto formáty (v únoru 2011) označila jako referenční a uložila členským státům, aby je přijaly do své legislativy. Orgány státní správy členských států EU mají povinnost přijmout elektronický dokument, pokud je řádně podepsán elektronickým podpisem v jednom z výše uvedených formátů. Výnos připouští i použití jiného formátu elektronického podpisu, pak ale musí být splněna řada dalších podmínek. Česká legislativa tyto formáty přijala ve své novele zákona o archivnictví a spisové službě (499/2004 Sb.) a zákona o elektronickém podpisu (227/2000 Sb.).

3.7.2 XAdES

V této práci se budu podrobněji zabývat formátem XAdES. XAdES formát vychází z formátu XMLDSig a je rozlišován v několika úrovních podle toho, jaké informace obsahuje:

- **XAdES-BES (Basic Electronic Signature)** – je základní rozšíření podpisu (XMLDSig), které přidává vlastnosti podpisu, jež jsou určeny k podepsání, a vlastnosti, jež k podepsání určeny nejsou.
- **XAdES-EPES (Explicit Policy based Electronic Signature)** – je dalším základním rozšířením, které je alternativou k XAdES-BES. Přináší to samé, co XAdES-BES, a navíc obsahuje element, v němž jsou vyjádřena pravidla podpisu (Signature Policy).
- **XAdES-T (XML Advanced Electronic Signature with Time)** – je rozšíření, které vychází z formátu XAdES-BES, nebo XAdES-EPES, a které přidává časové razítko podpisu.
- **XAdES-C (Electronic signature with complete validation data references)** – je rozšíření formátu XAdES-T o kompletní soubor odkazů na všechny certifikáty, které jsou potřebné k validaci podpisu, a na jejich CRL seznamy.
- **XAdES-X (Extended signatures with time forms)** – je rozšíření formátu XAdES-C, ke kterému je navíc přidáváno časové razítko, u kterého rozlišujeme dva různé druhy: RefsOnlyTimeStamp zahrnuje pouze odkazy na certifikáty a revokační seznamy. SigAndRefsTimeStamp zahrnuje navíc i hodnotu podpisu.
- **XAdES-X-L (Extended long electronic signatures with time)** – je rozšíření formátu XAdES-X, které navíc přidává i hodnoty certifikátů a revokačních seznamů.
- **XAdES-A (Archival electronic signatures)** – je rozšířením formátu XAdES-A a zároveň nejkompletnější možná úroveň tohoto formátu. Umožňuje navíc vkládat archivační časová razítka a splňuje tak koncept LTV.

Struktura formátu XAdES podle úrovní znázorňuje schéma na obrázku 3.2.



Obrázek 3.2: Úrovně formátu XAdES (zdroj: W3C.org).

4 Systém správy dokumentů Microsoft SharePoint

Mezi nejrozšířenější *systémy správy dokumentů* (*Document Management System - DMS*)patří Microsoft *SharePoint*. V této práci se budu zabývat systémem Microsoft SharePoint 2010. SharePoint 2010 je vydáván ve třech různých edicích – *Foundation*, *Standard* a *Enterprise*, které se od sebe liší svými funkcemi a cenou licencí. Pokud budou patřit některé informace pouze ke konkrétní edici, bude to explicitně uvedeno.

4.1 Přehled služby

Microsoft rozděluje služby platformy SharePoint do šesti oddílů:

- **Sites** – tento oddíl poskytuje organizacím strukturu, na které mohou postavit své interní weby.
- **Communities** – představuje nástroje pro interakci s jinými uživateli (poskytuje vlastní, zjednodušenou Wiki a podobně).
- **Content** – z pohledu dokumentů se jedná o nejdůležitější oddíl SharePointu. Umožňuje škálovat dokumenty a položky do typů, které lze definovat na základě metadat. Jednotlivým typům obsahu lze nastavovat předpisy ve formě pracovních postupů (workflow), podle kterých je pak s dokumenty zacházeno.
- **Search** – je modulem pro vyhledávání. Zde se liší pokročilost vyhledávání podle zvolené edice. Například edice Foundation poskytuje klasické vyhledávání. Edice Standard nabízí navíc integraci s vyhledáváním v operačním systému Windows 7, fonetické vyhledávání, návrhy dotazů a nastavování rozsahu. Edice Enterprise přidává využívání metadat, kontextové vyhledávání na základě uživatelského profilu a celkově rozšiřitelnou platformu vyhledávání (ve skutečnosti poskytují edice Standard a Enterprise mnohem více pokročilých funkcí).
- **Insights** – tento oddíl poskytuje řadu služeb pro analýzu dat, Business Intelligence, Excel Services (tvorba interaktivních Excel listů a jejich

možná integrace do jiných aplikací pomocí REST API) a další. Tento oddíl je dostupný pouze v edici Enterprise.

- **Composites** – je oddíl pro vytváření vlastních řešení. Jednoduše jsou to nástroje pro přizpůsobení SharePointu – upgrade ze SharePoint 2007, SharePoint designer, klientské i serverové API, dotazování pomocí jazyka LINQ, podpora technologie Silverlight a v edici Enterprise i integrace se službami InfoPath (tvorba formulářů).[17]

Služba SharePoint vyžaduje ke svému běhu Windows Server 2008. Je hostována službami IIS (Internet Information Services for Windows Server). Kromě výše uvedených edic je nabízen v rámci programu Office 365 i SharePoint Online. Jedná se o službu SharePoint, která je hostována v Microsoft Data Center.

4.2 Architektura

Pro pochopení jednotlivých pojmů, se kterými budu zacházet, je nutné znát architekturu SharePoint 2010.

Služba SharePoint se skládá z jednoho, nebo více SharePoint serverů (lze je specializovat na poskytování konkrétní SharePoint služby). Množina všech SharePoint serverů, které jsou navzájem propojeny, je takzvaná Sharepoint *farma* (*Farm*). Každý ze serverů farmy může hostovat jednu, nebo více *webových aplikací* (*Web Application*). Každá webová aplikace může hostovat jednu, nebo více *kolekcí stránek* (*Sites Collection*), ty obsahují jednu nebo více stránek (*Site*) a každá stránka může obsahovat seznamy (*Lists*). Speciálním případem seznamu mohou být knihovny různých typů (dokumentů, multimedií, vlastních typů).

Kompozice služeb SharePointu na úrovni farmy je dostupná všem webovým aplikacím, bez ohledu na to, na kterém serveru webová aplikace běží.

4.3 Word Automation Services

Z hlediska práce s dokumenty nás budou na službě SharePoint zajímat hlavně *Word Automation Services*. Jedná se o služby, které jsou spuštěny na Share-

Point Serveru 2010, o služby, které jsou navrženy pro doplnění Open XML SDK na straně SharePoint Serveru. Word Automation Services jsou určeny výhradně pro volání ze strany serveru, nemůžeme je tedy využívat v aplikaci, která je spuštěna na stanici klienta SharePoint služeb. Tyto služby jsou dostupné pouze ve verzích Standard a Enterprise.

5 Analýza

5.1 Formáty složených dokumentů

Na základě informací, které jsem o jednotlivých dokumentových formátech uvedl v kapitole 2, a na základě kritérií, které jsem si stanovil v kapitolách 2.1.1, 2.1.2 a 2.1.3, mohu mezi sebou dokumentové formáty porovnat a doporučit některé k podpoře v DMS systému.

5.1.1 PDF

Uživatelská manipulace	Není určeno pro běžnou uživatelskou manipulaci, je umožněna zpravidla placenými jednoúčelovými programy.
Automatické skládání dokumentů	Existují kvalitní knihovny pro skládání dokumentů, je možné skládat PDF dokumenty pomocí XSL-FO.
Archivace (el. podpisy a časová razítka)	Podpora standardů pro podepisování a archivaci, vnitřní struktura uzpůsobená ke vkládání podpisů a časových razítek.
Další aspekty	Jedná se o archivační a grafický standard.

Tabulka 5.1: Shrnutí vlastností formátu PDF.

5.1.2 OOXML

Uživatelská manipulace	Podporována celou řadou (i bezplatných) aplikací. Možnost tvorby šablony.
Automatické skládání dokumentů	Kvalitní knihovny i pro vysokoúrovňovou manipulaci s obsahy dokumentů.
Archivace (el. podpisy a časová razítka)	Vnitřní struktura uzpůsobená ke vkládání podpisů a časových razítek.
Další aspekty	Silně podporováno na platformě .NET, standardní formát pro SharePoint.

Tabulka 5.2: Shrnutí vlastností formátu OOXML.

5.1.3 ODF

Uživatelská manipulace	Podporována celou řadou (i bezplatných) aplikací. Možnost tvorby šablon.
Automatické skládání dokumentů	Kvalitní knihovny jsou. Vysokoúrovňová manipulace s obsahem spíše není.
Archivace (el. podpisy a časová razítka)	Vnitřní struktura uzpůsobená ke vkládání podpisů a časových razítek.
Další aspekty	Přijat jako národní standard v některých zemích.

Tabulka 5.3: Shrnutí vlastností formátu ODF.

5.1.4 XSL-FO

Uživatelská manipulace	Není možná.
Automatické skládání dokumentů	Náročná příprava dokumentu (transformace). Následné plnění snadné.
Archivace (el. podpisy a časová razítka)	Dokument není určen k archivaci. Lze jej podepsat připojením externího podpisu, ale po XSL transformaci nebude cílový dokument podepsán.
Další aspekty	Výborná automatická tvorba PDF dokumentů (pokud nepotřebujeme často měnit vzhled).

Tabulka 5.4: Shrnutí vlastností formátu XSL-FO.

5.1.5 RTF

Uživatelská manipulace	Úroveň uživatelské manipulace se značně liší - záleží na zvoleném programu. Obecně lze říci, že uživatelská manipulace je podporována.
Automatické skládání dokumentů	Absence spolehlivých knihoven, které by odpovídaly konkrétní specifikaci. Pouze nízkoúrovňová manipulace obsahu.
Archivace (el. podpisy a časová razítka)	Struktura dokumentu není přizpůsobena pro vkládání podpisů a razítek. Je možné vytvořit externí podpis (respektive razítko), pokud máte vhodný program.
Další aspekty	Silně rozšířen. Mnoho verzí specifikace, programy většinou zvládnou všechny verze alespoň částečně zpracovat. Výsledky se liší.

Tabulka 5.5: Shrnutí vlastností formátu RTF.

5.1.6 Porovnání

Tyto souhrnné aspekty jsem objektivně obodoval v jednotlivých kritériích v rozmezí 0 - 5 bodů (5 - nejlepší) podle míry naplnění těchto oblastí (tabulka 5.6). Všem kritériím jsem přiřadil stejnou váhu.

	Formát dokumentu				
Oblast zkoumání	PDF	OOXML	ODF	XSL-FO	RTF
Uživatelská manipulace	1	5	5	0	3
Automatické skládání dok.	4	5	3	3	1
Archivace	5	3	3	0	1
Další aspekty	3	3	2	5	2
Součet	13	16	13	8	7

Tabulka 5.6: Ohodnocení a porovnání vlastností zkoumaných formátů.

5.1.7 Výběr

Z výsledků porovnání mohu konstatovat, že mezi nejlepšími formáty se umístilo OOXML, PDF a ODF. Mým doporučením je používat formát PDF k archivaci dokumentů, neboť je k tomu ideálně přizpůsoben. Pro ostatní účely doporučuji použít formát OOXML, neboť je silně podporován na platformě .NET, je standardním formátem systému MS SharePoint, hodí se pro uživatelskou editaci a v neposlední řadě je vhodný pro automatizované skládání. Formát ODF hodnotím také jako použitelný, avšak jedná se o formát s podobnými vlastnostmi jako OOXML. Na rozdíl od OOXML nemá takovou podporu. O jeho zavedení bych uvažoval v zemích, které jej přijaly jako svůj

standard. Formát XSL-FO pochopitelně v mnohém ztrácí, dá se však využít jako doplňkový formát pro tvorbu PDF. Formát RTF dopadl nejhůře a k užívání jej nedoporučuji.

Na základě tohoto závěru upravila společnost CCA Group své požadavky a zadala implementaci formátu OOXML, konkrétně formátu dokumentů DOCX.

5.2 Kritéria aplikace

5.2.1 Platforma .NET

Společnost CCA Group je partnerem společnosti Microsoft a nabízí řešení systému správy dokumentů od této firmy (Microsoft SharePoint) a také vyvíjí systémy na bázi ASP.NET. Zároveň nemá pro tuto platformu vlastní řešení, které by umožňovalo skládání dokumentů a připojování elektronických podpisů. Mimo to lze předpokládat, že je na pracovních stanicích uvnitř podniků používán operační systém Windows (často ve verzi 7 nebo vyšší) a mnohdy i kancelářský balík Office, který by bylo možné touto aplikací obohatit. Tyto důvody vyústily v požadavek společnosti CCA Group, aby byla aplikace napsána pro platformu .NET.

5.2.2 Nezávislost datových zdrojů

Aplikace musí být navržena tak, aby umožňovala využití různých datových zdrojů. Nelze se spolehnout, že bude možné číst a zapisovat soubory na pevný disk stanice (respektive serveru). Je velmi pravděpodobné, že systém bude fungovat tak, že dokumenty budou ukládány a načítány z databáze a aplikace nebude mít práva přistupovat k pevnému disku stanice.

5.2.3 Modularita jednotlivých částí aplikace

Aplikace se bude skládat z oddělených logických celků (skládání dokumentů, plnění dokumentů, připojování podpisů). Požadavkem je, aby byly tyto části navrženy tak, aby byly snadno vyměnitelné.

5.2.4 Integrace aplikace do systému správy dokumentů

Společnost CCA Group chce nad rámec standardního zadání analyzovat stav možnosti skládání dokumentů a připojování elektronických dokumentů v systému Microsoft SharePoint a případně mnou navrženou aplikaci integrovat do tohoto systému.

6 Základní aplikace – MergeSignModul

6.1 Architektura

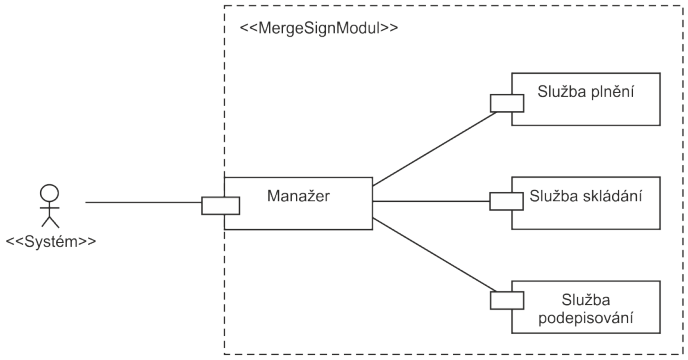
Na základě výše uvedených požadavků a našich dřívějších zkušeností jsem pro tvorbu aplikace vybral jazyk C#. Aplikace bude mít formu dynamické knihovny tříd.

6.1.1 Kontext aplikace

Je naznačen v diagramu na obrázku 6.1. Uživatel „Systém“ představuje aplikaci, která využívá funkcionalitu knihovny. Může se jednat například o webovou, nebo desktopovou aplikaci.

6.1.2 Struktura modulu

Aplikace poskytuje tři základní funkčnosti, které jsou po ní požadovány:



Obrázek 6.1: Kontextový diagram aplikace (zdroj: autor).

- Spojování dokumentu typu OOXML z dílčích dokumentů.
- Plnění námi předdefinovaných OOXML šablon vlastními daty.
- Podepisování OOXML dokumentů elektronickým podpisem a připojování elektronických časových razítek.

Defaultním jmenným prostorem aplikace je MergeSignModul. V tomto jmenném prostoru je implementován manažer modulu. Další jmenné prostory odpovídají výše uvedené funkcionalitě a každý z nich představuje samostatně fungující celek:

- **MergeSignModul.Merge** (spojování dokumentů)
- **MergeSignModul.Fill** (plnění šablon daty)
- **MergeSignModul.Sign** (podepisování elektronickým podpisem a připojování časových razítek)

Pro co nejvyšší modularitu jsem ke každé funkcionalitě navrhl rozhraní, které třídy s odpovídající funkcionalitou implementují. Zároveň jsem modul navrhl tak, aby odpovídal návrhovému vzoru *dependency injection*. Na nejvyšší úrovni aplikace je definován manažer. Ten vytváří instance tříd, které implementují daná rozhraní, a zastává tak funkci třídy, která bývá v dependency injection označována *assembler*[18]. Informaci o tom, jakou implementaci má zvolit, získává manažer z konfiguračního kontejneru config.xml. Tímto způsobem se odstiňuje volba konkrétních implementací od programového kódu a centralizuje se na jednom místě.

6.1.3 Předávání dat a odstínění datových zdrojů

V kapitole 5.2.2 jsem zmínil, že modul musí být nezávislý na datových zdrojích. Zároveň je vhodné, aby jednotlivé části modulu fungovaly nezávisle na sobě, aby bylo například možné dokument pouze podepsat. Z těchto důvodů se dokumenty předávají a zpracovávají ve formě obecných datových proudů (zděděných od abstraktní třídy `System.IO.Stream`).

6.1.4 Přehled tříd

Důležité třídy jsou vyznačeny v diagramu na obrázku 6.1.4. Podrobnější popis důležitých tříd naleznete v kapitole 6.2.

6.2 Implementace

V této kapitole budou popsány hlavní třídy aplikace.

6.2.1 Třída MFSManager

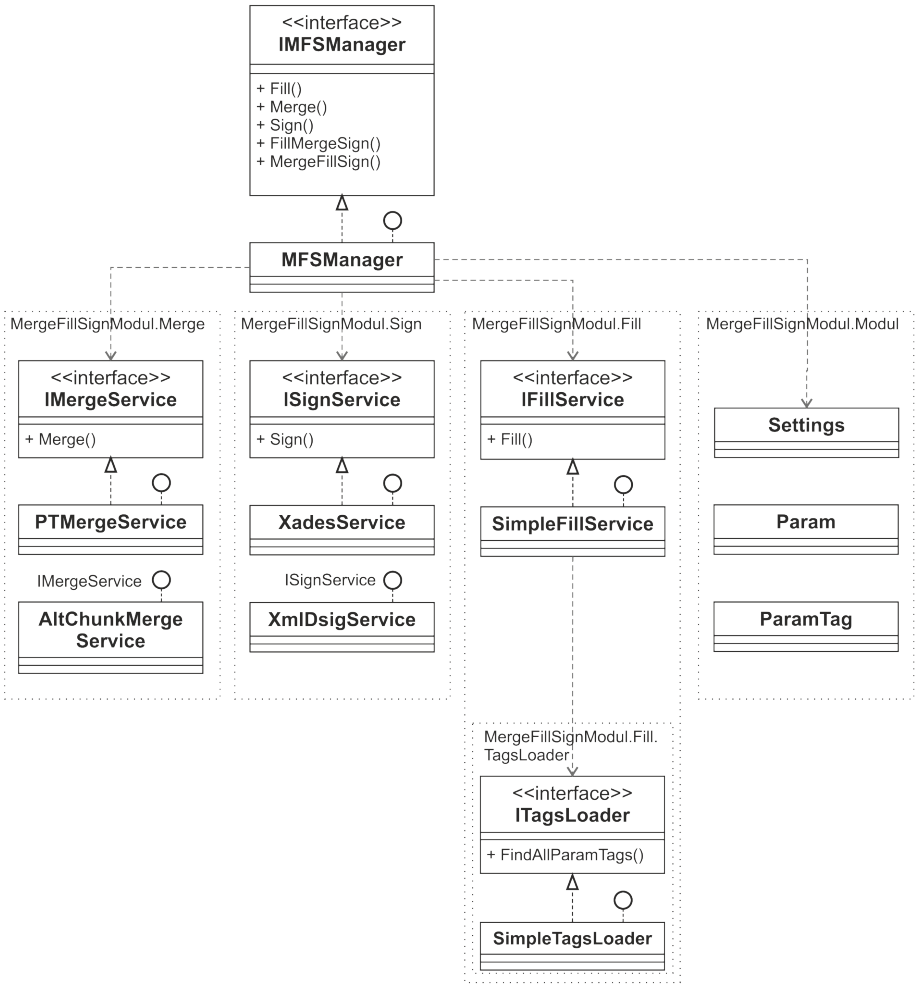
`MFSManager` je třída, která implementuje rozhraní `IMFSManager` a slouží jako vstupní bod modulu. Konstruktory této třídy zajišťují načtení konfigurace modulu a vytvoření instancí ostatních potřebných tříd na základě této konfigurace. Příklad vytváření instancí podle nastavení:

```
this.mergeService = (IMergeService)System.Reflection.Assembly
      .GetExecutingAssembly().CreateInstance(
      "MergeSignModul.Merge." +(String)settings["MergeService"]);
```

Třída dále obsahuje veřejné metody pro volání skládání dokumentu `Merge()`, plnění dokumentu `Fill()` a podepisování dokumentu `Sign()`. Obsahuje také metody pro vyvolání všech úkonů `MergeFillSign()` a `FillMergeSign()`.

6.2.2 Třída AltChunkMergeService

`AltChunkMergeService` je třída, která implementuje rozhraní `IMergeService`. Tato třída poskytuje funkčnost spojování dokumentů pomocí odkazování na existující celky. To je realizováno pomocí *AltChunk* elementů a obsahu ve formátu *AlternativeFormatImportPart*, které poskytuje *OpenXML SDK*. AltChunk je element Wordprocessing dokumentu, který slouží jako kotva pro umístění importovaného obsahu do hlavní části dokumentu. Umožňuje vložení externího obsahu nebo obsahu, který je přímo součástí OOXML balíku. V obou případech je tento obsah importován jako AlternativeFormatImportPart. Tento kontejner umožňuje import dat v různých formátech –



Obrázek 6.2: UML diagram tříd se znázorněním jmenných prostorů (zdroj: autor).

Text (.txt), Extensible HyperText Markup Language File (.xhtml), HyperText Markup Language File (.htm), Rich Text Foramt (.rtf), Wordprocessing (.docx) a dalších. Právě vkládání celého DOCX dokumentu využívá tato metoda. Ve výsledku `AltChunkMergeService` vrací dokument, jehož součástí jsou nezměněné dílčí dokumenty, a dokument, který definuje jejich umístění. Zajímavou vlastností je, že pokud dokument s importovaným obsahem v aplikaci Microsoft Word znovu uložíme, jsou importované části sloučeny do hlavního dokumentu.

6.2.3 Třída PTMergeService

`PTMergeService` je další třída, která implementuje rozhraní `IMergeService` a umožňuje spojovat více DOCX dokumentů v jeden. Představuje alternativu k použití `AltChunkMergeService` a na rozdíl od této třídy nevyužívá import celků, ale vytváří zcela nový dokument pomocí `DocumentBuilder` třídy, kterou poskytuje knihovna *OpenXmlPowerTools*. Třída `DocumentBuilder` se sama stará o správné sloučení stylů jednotlivých dokumentů a umožňuje tak čistě vysokoúrovňový přístup. `AltChunkMergeService` a `PTMergeService` využívají zcela odlišné přístupy a tomu odpovídají i specifické vlastnosti ohledně výkonnosti a kvality výsledků. Tyto vlastnosti budou později v kapitole 9 rozebrány a podloženy testováním.

6.2.4 Třída SimpleFillService

`SimpleFillService` je třída, která implementuje rozhraní `IFillService` a umožňuje plnit šablony dokumentu textem. Možností, jak naplnit dokument textem, je celá řada. *WordprocessingML* například umožňuje definovat vlastní metadata a zároveň tato metadata vkládat do těla dokumentu. Další možností je nadefinovat si vlastní značku jako posloupnost znaků a tuto značku nahrazovat konkrétním textem. Vyhýbám se tak používání metadat v těle dokumentu a uživatel, který bude chtít tvořit šablonu, nebude potřebovat pokročilé znalosti ovládání aplikace Microsoft Word. `SimpleFillService` používá metodu `FindTags()` z rozhraní `ITagsLoader`, která v šabloně nalezne značky pro umístění vlastního textu. Metoda `Fill()` třídy `SimpleFillService` v dokumentu nahradí text značky za uživatelský text.

6.2.5 Třída SimpleTagsLoader

`SimpleTagsLoader` je třída, která implementuje rozhraní `ITagsLoader` a vyhledává v textu značky ve tvaru `<<jmeno_parametru>>`.

Mnou navržený mechanismus umožňuje snadnou výměnu značek pro umístění textu. Stačí vytvořit jinou implementaci rozhraní `ITagsLoader` a třída `SimpleFillService` bude fungovat bez jakékoliv změny.

6.2.6 Třída XmlDsigService

Třída `XmlDsigService` implementuje rozhraní `ISignService` a umožňuje vkládat interní podpisy ve formátu XMLDSig do WordprocessingML dokumentů pomocí standardních knihoven platformy .NET (*System.Security* a *System.IO.Packaging*).

6.2.7 Třída XadesService

Další třídou, která implementuje rozhraní `ISignService`, je třída `XadesService`. Na rozdíl od `XmlDsigService` neposkytuje pouze základní podpis typu XMLDSig, ale jeho rozšířenou verzi – XAdES. Jak jsem uvedl v kapitole 3.6.2, podpis formátu XAdES je rozlišován v několika úrovních podle informací, které jsou v něm obsaženy. `XadesService` poskytuje úrovně T (Timestamp), C (Complete), X (eXtended), X-L (eXtended Long term). Neposkytuje nevyšší úroveň A (Archival) a nejnižší úroveň BES, nebo také EPES (která se v podstatě neliší od základního formátu XMLDSig). Třída tedy řeší nejen samotné podepisování, ale i připojení časového razítka, přidání odkazů na veškeré použité certifikáty v certifikačním řetězci, který byl k podpisu použit, připojení časového razítka po přidání těchto informací a nakonec i přidání samotných certifikátů certifikačního řetězce a hodnoty revokačních listů. `XadesService` využívá prostředků knihovny *Microsoft.Xades*, která byla vytvořena jako open source projekt *Signature avancée (XAdES) pour Microsoft .NET Framework v3.5* francouzského týmu vývojářů společnosti Microsoft a která byla publikována pod svobodnou CeCILL-B licencí. Samotná knihovna *Microsoft.Xades* není zcela bezproblémová, přesto představuje oproti ostatním knihovnám tu nejlepší možnou volbu. Další možností byla knihovna *XadesNet*. Její autoři uvádějí, že ve finální verzi bude poskytovat podpisy

formátu XMLDsig, XAdES BES, T a C. Avšak již delší dobu se knihovna nachází ve stavu „Beta“, ve kterém poskytuje pouze Formát XMLDsig a XAdES BES. Neřeší tak vůbec získávání časových razítek ani další pokročilé části XAdES podpisu. Získávání časových razítek naopak poskytuje knihovna *BouncyCastle*, která však neposkytuje žádnou podporu XAdES formátu, ani formátu XMLDSig. Nejlépe tak dopadla knihovna *Microsoft.Xades*, i přes skutečnost, že bylo nutné ji rozšířit o podepisování externích souborů ve formátu datových proudů, o možnost získávat CRL přes HTTP protokol a opravit chybu, která v případě vkládání více certifikátů certifikačního řetězce vypočítávala jejich hash pouze z prvního z nich. Podpisy vytvářené třídou `XadesService` je možné v případě potřeby dále rozšířit o archivační časová razítka (úroveň A).

6.2.8 Logování modulu

Jelikož je modul určen pro různé typy aplikací, včetně webových, je vhodné, aby bylo chování modulu logováno. Pro tyto účely jsem v základní implementaci zvolil nástroj NLog. Tato knihovna umožňuje snadné nastavení ve vlastním konfiguračním souboru, nebo ve web.xml. Podporuje logování do souborů i do databáze a použití je snadné. Stačí získat instanci loggeru:

```
static Logger logger = LogManager.GetCurrentClassLogger();
```

a pak už pouze zapisovat logovací zprávy:

```
logger.Warn("Spojování voláno bez vstupních datastreamů.");
```

6.2.9 Konfigurace modulu

V kapitole 6.1 jsem uvedl, že konfigurace modulu je uchovávána v souboru config.xml. K získání informací z tohoto souboru slouží třída `Model.Settings`, která k datům v config.xml přistupuje jako k *data setu*. Tento přístup umožňuje v případě potřeby snadnou změnu úložiště ze souboru na databázi.

7 Pomocné aplikace

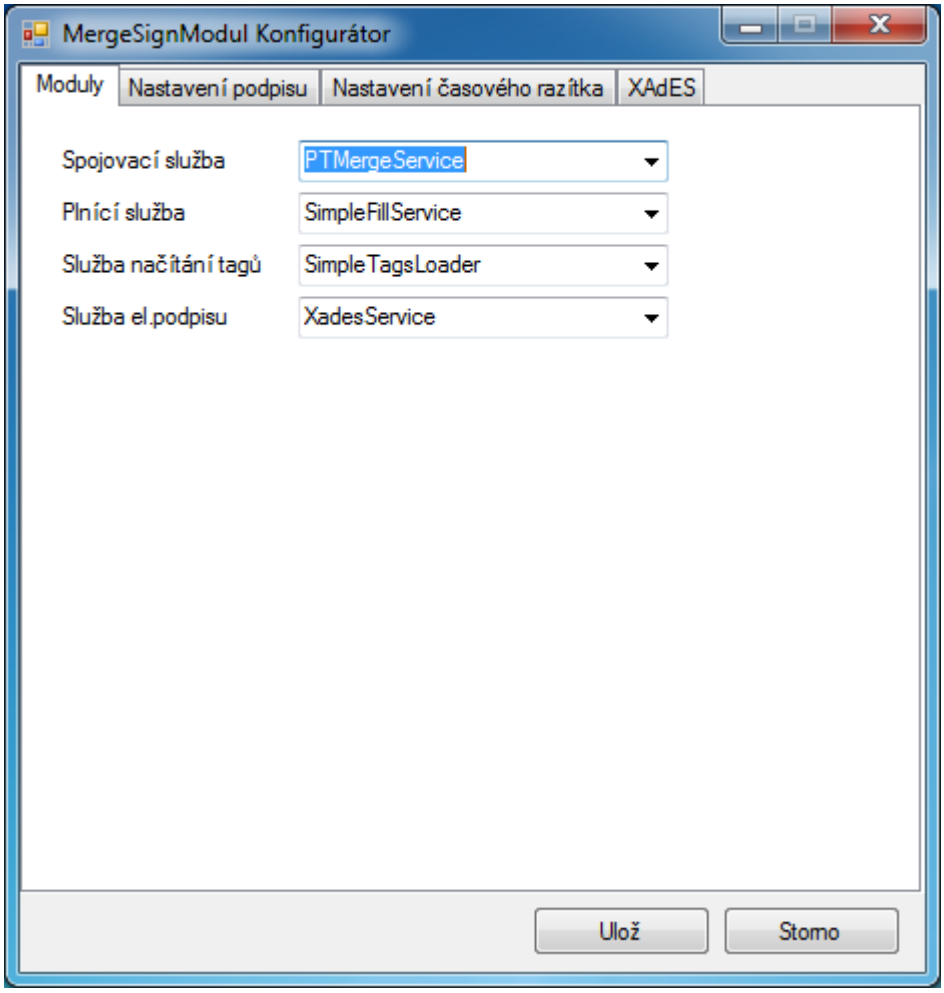
Pro jednodušší nastavování modulu, ukázku jeho činnosti a otestování výsledků vznikaly menší jednoúčelové aplikace. V této kapitole je ve stručnosti představím.

7.1 MergeSignModulConfig

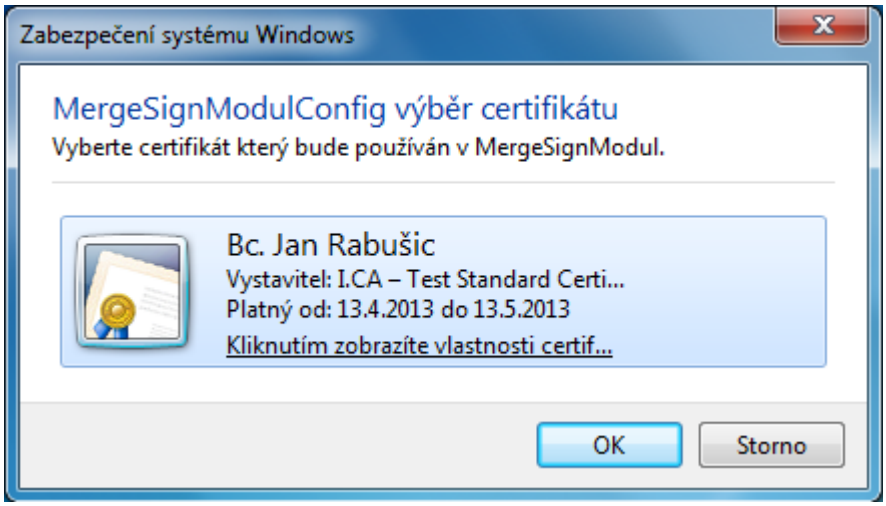
Jak bylo uvedeno v kapitole 6.1, v konstruktoru třídy `MFSManager` dochází k načítání konfiguračního souboru config.xml. Aby byl modul snáze nastavitelný, vytvořil jsem *MergeSignModulConfig*. Jedná se o jednoduchou *Windows Forms* aplikaci, kterou uživatel modulu umístí do složky, ve které se bude nacházet config.xml. Při jejím spuštění si načte aktuální verzi konfiguračního souboru, pokud je dostupná. Pokud není dostupná, je vytvořena zcela nová konfigurace. V podstatě je to velmi jednoduchý nástroj a zajímavější funkcionalitu představuje pouze tlačítko výběru certifikátu pro elektronický podpis. Zmíněné tlačítko otevírá úložiště certifikátů uživatele s dostupnými platnými certifikáty a uživatel si tak může vybrat konkrétní certifikát bez znalosti, jak se do tohoto úložiště dostat za pomoci systémových prostředků. Ukázku aplikace naleznete na obrázku 7.1 a příklad výběru certifikátu na obrázku 7.2.

7.2 MergeSignApp

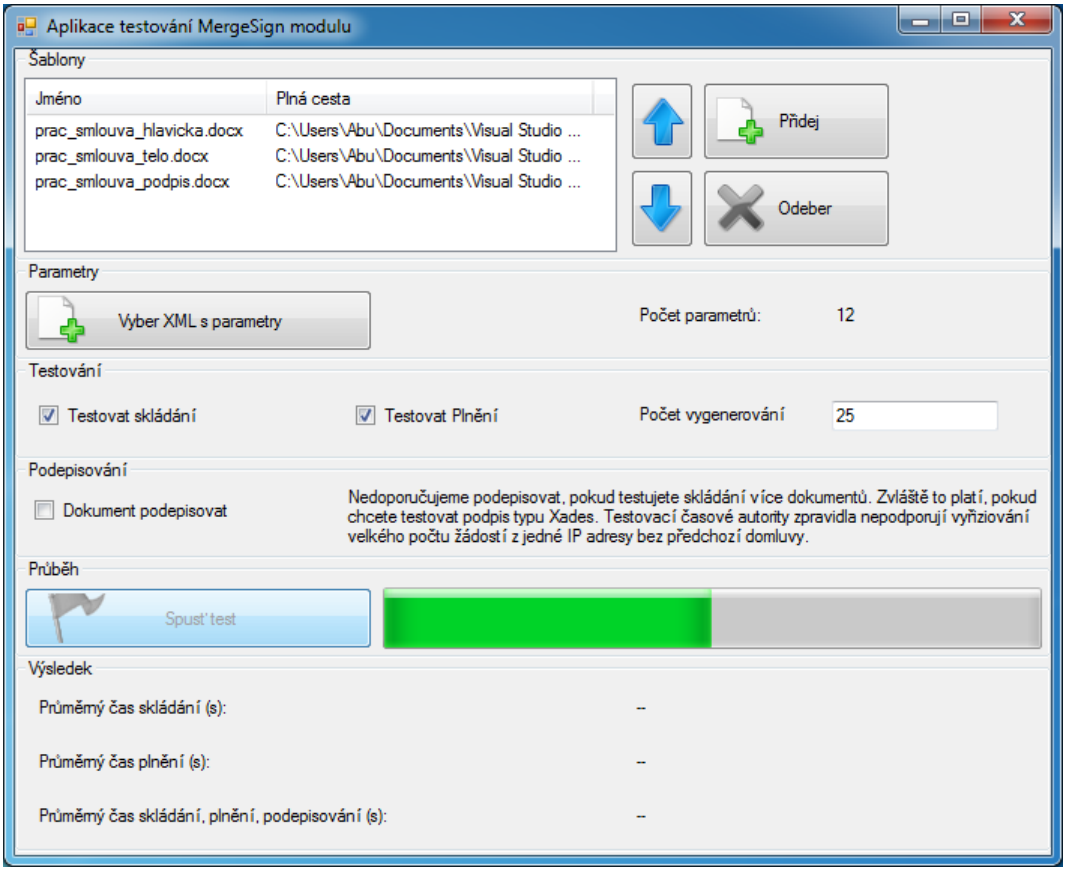
MergeSignApp je jednoduchá *Windows Forms* aplikace, která demonstruje použití modulu MergeSignModul a zároveň umožňuje testovat dobu potřebnou pro spojování a plnění dokumentu (obrázek 7.3). Aplikace poskytuje uživatelské rozhraní pro volbu šablon a možnost načtení parametrů ve formátu XML. XSD schéma a ukázku XML s parametry naleznete v příloze A. Dále si uživatel zvolí počet opakování, kolikrát se má dokument vytvořit. Tato funkcionalita ukazuje znovupoužitelnost šablon a zároveň umožňuje získat přesnější výsledky měření doby skládání a plnění, pokud zvolíme, aby se tento proces několikrát opakoval.



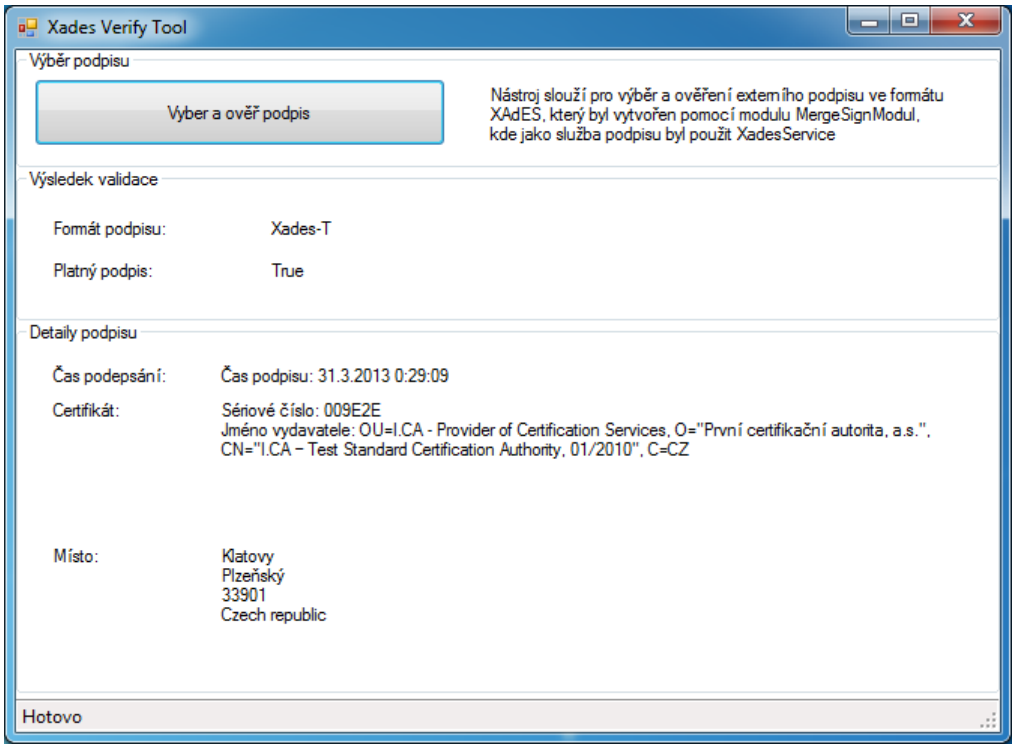
Obrázek 7.1: Ukázka konfigurační aplikace (zdroj: autor).



Obrázek 7.2: Ukázka výběru certifikátu (zdroj: autor).



Obrázek 7.3: Běh ukázkové aplikace (zdroj: autor).



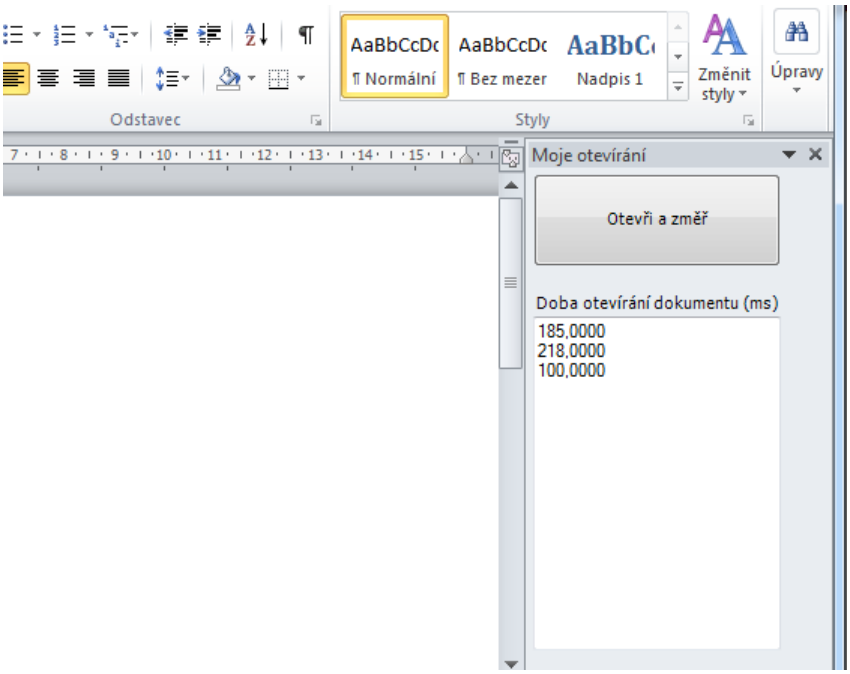
Obrázek 7.4: Ukázka ověřeného podpisu (zdroj: autor).

7.3 XadesVerifyTool

Další z řady podpůrných aplikací je jednoduchá *Windows Forms* aplikace, která je určena k ověřování podpisů ve formátu XAdES. Využívá prostředků, které poskytuje knihovna *Microsoft.Xades* k určení validity podpisu a navíc rozpoznává úroveň formátu XAdES, které podpis dosahuje (obrázek 7.4).

7.4 TestTimeToOpenDocument

Poslední z podpůrných utilit je *VSTO (Visual Studio Tools for Office) Word 2010 doplněk*, který rozšiřuje aplikaci Microsoft Word 2010 o možnost načítání dokumentu s měřením doby otevírání dokumentu (obrázek 7.5).



Obrázek 7.5: Doplněk do aplikace Microsoft Word 2010, který umožňuje měřit čas otevírání dokumentu (zdroj: autor).

8 Integrace modulu do systému Microsoft SharePoint 2010

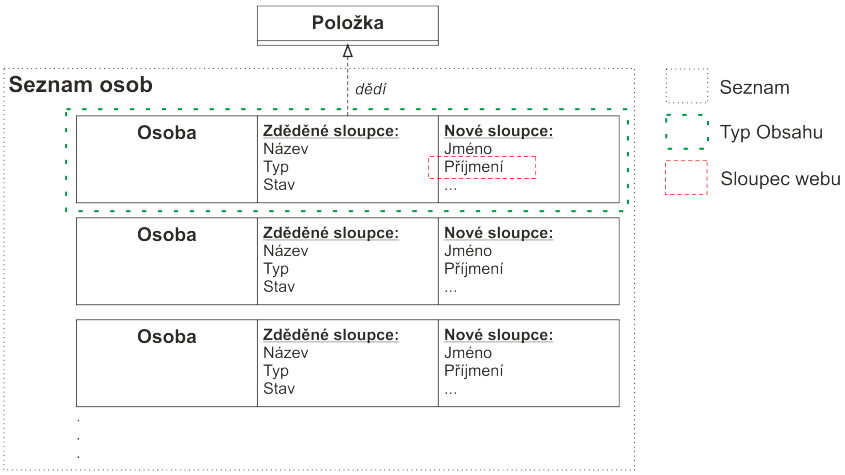
Úkolem v zadání diplomové práce od společnosti CCA Group bylo vytvořit modul, který by umožnil automaticky skládat dokumenty, plnit je textem a připojovat k nim elektronické podpisy a časová razítka. Společnost CCA Group na základě výsledků z kapitoly 6 vznesla návrh, zda by bylo možné diplomovou práci rozšířit o praktickou integraci vzniklého modulu do systému Microsoft SharePoint 2010 Standard. Vzhledem k tomu, že jsem implementaci v kapitolách 6 a 7 zvládl s časovým předstihem a že integrace modulu do systému SharePoint bude znamenat velký přínos pro společnost CCA Group, rozhodl jsem se tento návrh přijmout a vytvořit praktickou ukázku použití modulu v tomto systému, doplněk **SPMergeSign**.

8.1 Návrh integrace

Integrace modulu neznamená pouze uvést tento modul do fungujícího stavu, ale také navrhnout přístup k jeho využívání.

Systém SharePoint 2010 poskytuje možnost definovat vlastní *seznamy* (*lists*), *typy obsahu webu* (*content types*) a *sloupce webu* (*site columns*). Sloupce webu umožňují definovat datový typ na úrovni stránek. Typy obsahu webu jsou typy, které jsou odděděné od jiných typů obsahu webu (těch, které jsou implicitně definované v systému SharePoint, nebo těch, které jsme si sami vytvořili) a které je možné rozšířit o další vlastnosti přidáním sloupců webu (ukázka vztahů na obrázku 8.1).

Seznam má definovaný typ obsahu, který může obsahovat, a také šablonu dokumentu, kterou může tímto typem obsahu plnit. Sloupce webu, které tento typ obsahuje, se stanou metadaty dokumentu, a pokud byly v dokumentu umístěny značky pro umisťování metadat, jsou vytvářeny automatizované dokumenty, jak bylo popsáno v kapitole 6.3.4. Z tohoto hlediska má systém SharePoint automatické plnění dokumentů vyřešené. Na první pohled je však patrné, že se tento způsob silně váže na typ obsahu webu a na konkrétní seznam, se kterým je spojena šablona dokumentu.



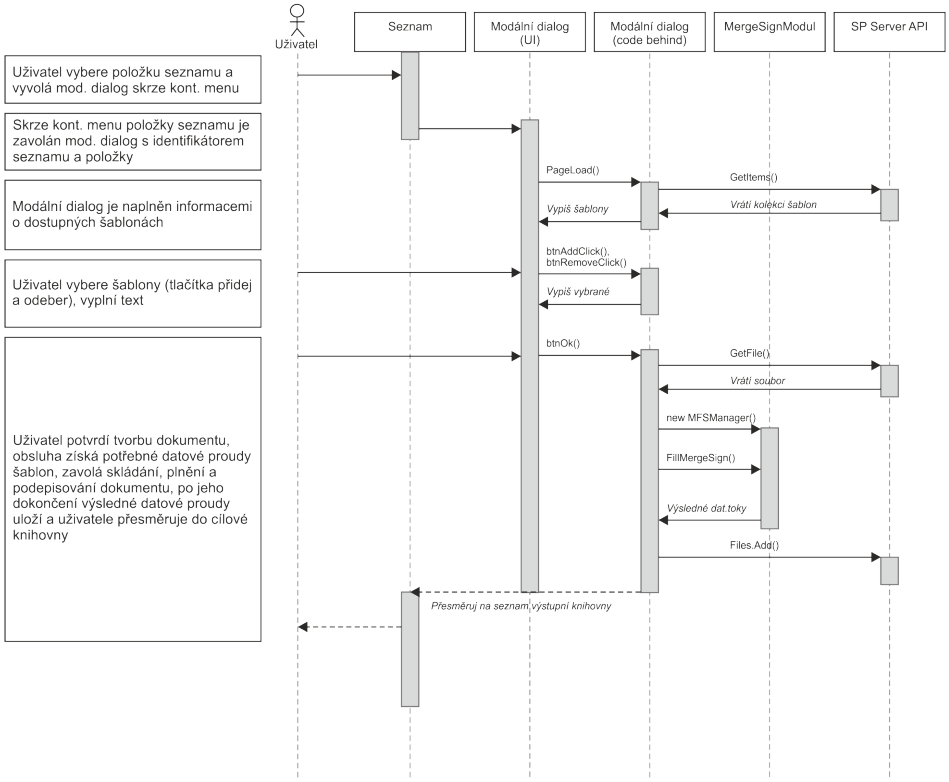
Obrázek 8.1: Příklad seznamů, typů obsahu a sloupců webu (zdroj: autor).

Mým přínosem je doplnit funkčnost, která bude poskytována nad obecnými datovými typy a výběr šablony nebude omezován.

Za tímto účelem rozšířím kontextové menu položky seznamu (v terminologii systému SharePoint je nazýváno *EditControlBlock*) o položku vytváření dokumentu. Tato položka bude vyvolávat modální dialogové okno, jehož obsahem bude nová aplikační stránka (ASPX), která bude zobrazovat dostupné šablony ve složce, která byla určena konfigurací modulu. Bude umožňovat jejich výběr, nastavení názvu výstupního dokumentu a po potvrzení získá datové proudy zvolených šablon, použije modul MergeSignModul, následně uloží výsledný datový proud (respektive výsledné datové proudy, pokud jsme vytvářeli dokument s externím elektronickým podpisem) do zvolené knihovny systému SharePoint a nakonec uživatele přesměruje na tuto knihovnu. Proces vytváření dokumentu pomocí navrhovaného rozšíření je popsán sekvenčním diagramem na obrázku 8.2.

8.2 Použití SharePoint 2010 SDK

Pro rozšíření funkcionality platformy SharePoint slouží *SharePoint 2010 SDK* a vývojové prostředí *Microsoft Visual Studio 2010*. Visual Studio obsahuje



Obrázek 8.2: Sekvenční diagram popisující proces tvorby dokumentu (zdroj: autor).

sadu šablon projektů, které jsou určeny platformě SharePoint a zaměřují se na rozšíření specifických částí této platformy (vizuální webové části, sekvenční workflow, definice seznamu, typ obsahu a další). Mnou vytvářená integrace modulu bude využívat různé části, a proto jsem zvolil obecný SharePoint projekt. Vývojář pro platformu SharePoint je při vytváření projektu vyzván, aby rozhodl, zda se bude jednat o *sandbox* řešení (řešení na omezeném „pískovišti“), nebo *farm* řešení (řešení, které ovlivňuje chod celé SharePoint farmy). Výhoda sandbox řešení je, že běží v omezeném prostoru, který je systémem SharePoint plně kontrolován a v případě, že by spotřebovával příliš mnoho zdrojů, jej SharePoint může dočasně ukončit. Zároveň se jedná o bezpečné řešení, které nemá přístup k pevným diskům serveru a mnohá další omezení. Tyto výhody sandbox řešení jsou zároveň i jeho nevýhody. Bohužel není například možné v sandbox řešení využívat assembly, které nejsou poskytovány systémem SharePoint. Z tohoto důvodu musel být projekt implementován jako farmové řešení.

8.3 Implementace UI a kontroloru

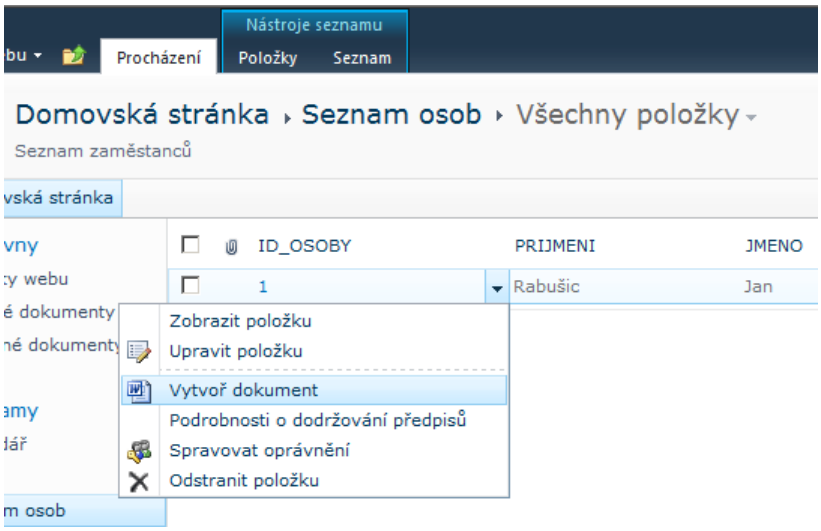
V kapitole 8.1 jsem uvedl, že k vytváření dokumentu budu přistupovat skrze novou položku v kontextovém menu položky seznamu a samotné ovládání bude realizováno skrze modální dialogové okno.

8.3.1 Položka kontextového menu

Je vytvářena jako *SharePoint 2010 Empty Element*, který navážeme na obecný typ obsahu webu, a kterému vytvoříme akci na kliknutí. XML kód pro tento element bude následující:

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- MergeSign polozka v~kontextovem menu -->
  <CustomAction Id="SPMergeSign.ContextMenuMergeSignItem.Click"
                RegistrationType="ContentType"
                RegistrationId="0x0100"
                Location="EditControlBlock"
                ImageUrl="/_layouts/IMAGES/DOC16.GIF"
                Sequence="600"
                Title="Vytvoř dokument"
                Description="Složení dokumentu ze zvolených dat." >
```

```
    <!-- Vyvolani akce (modalni dialog z˜pripravene aspx stranky) -->
    <UrlAction Url="javascript:
                OpenPopUpPageWithTitle('/_layouts/SPMergeSign/Select' +
                    Template.aspx?List={ListId}&amp;ID={ItemId}',
                    RefreshOnDialogClose,300,400,'Výběr šablony');"/>
  </CustomAction>
</Elements>
```



Obrázek 8.3: Položka kontextového menu (zdroj: autor).

8.3.2 Modální dialogové okno

Vyvolání tohoto okna je ukázáno v programovém kódu v kapitole 8.3.1. Tělo okna je běžná stránka aplikace (ASPX) a k ní patřičný *code-behind*. Pro co nejjednodušší integraci modulu do systému SharePoint nebylo prováděno vrstvení aplikace, a tak byl kód kontroloru jednoduše vložen do code-behind stránky. Jedná se hlavně o komunikaci se *SharePoint Server-Side API* (získání názvů šablon, získání a ukládání datových proudů) a volání modulu MergeSignModul. Dále jsou uvedeny příklady volání SharePoint Server-Side API:

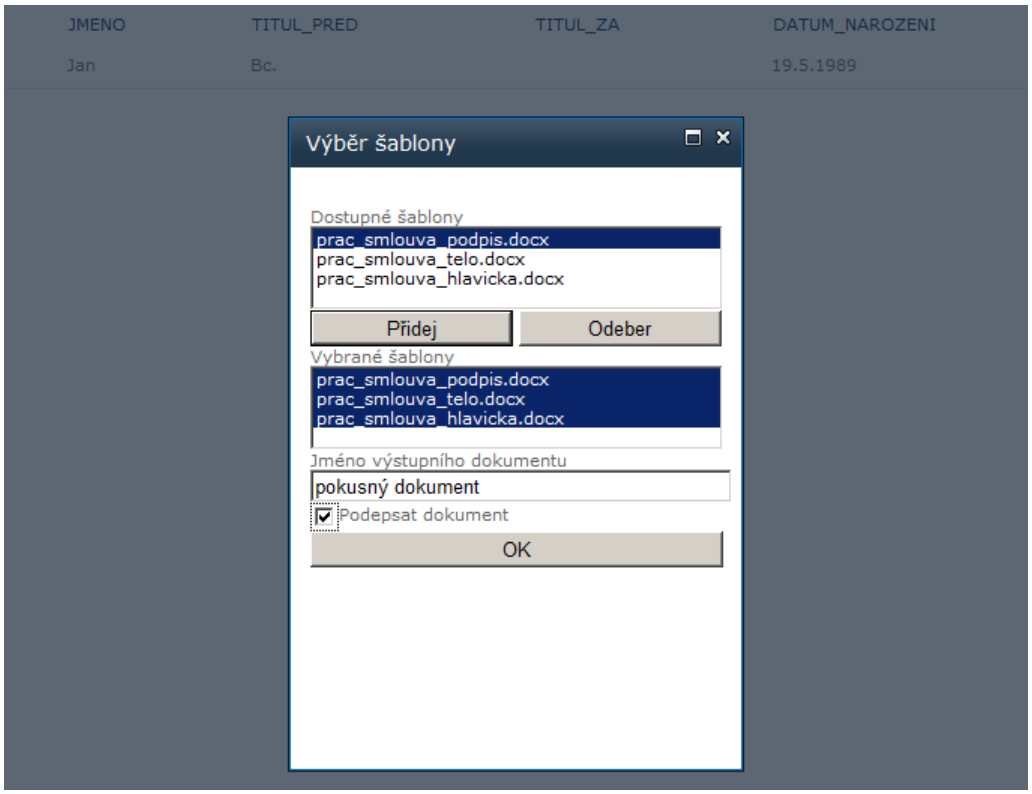
Příklad získání názvů a adres šablon:

```
SPFolder templatesFolder = SPContext.Current.Web.Lists[ioListName]
.RootFolder.SubFolders[infolderName];
```

```
foreach (SPFile file in templatesFolder.Files)
{
       ListBoxTemplAvailable.Items.Add(new ListItem(file.Name, file.Url));
}
```

Příklad uložení souboru do specifického adresáře:

```
SPFolder outputFolder = SPContext.Current.Web.Lists[ioListName]
                      .RootFolder.SubFolders[outfolderName];
outputFolder.Files.Add(outputName, outStream, true);
```



Obrázek 8.4: Ukázka modálního dialogu (zdroj: autor).

8.4 Změny v modulu oproti základní implementaci

Přestože jsem projekt integrace vytvořil jako farmové řešení, přinesl systém SharePoint některá omezení a doporučení, kvůli kterým bylo nutné provést několik zásahů do modulu.

8.4.1 Logování modulu

V základní implementaci modulu byl pro logování zvolen nástroj NLog. S jeho pomocí je stále možné i v aplikace běžící na platformě SharePoint logovat, avšak doporučené logování je za pomoci standardních logovacích prostředků systému SharePoint. Pro logování je pak použit následující kód:

```
SPDiagnosticsService.Local.WriteTrace(802,
          new SPDiagnosticsCategory("SPMergeSign",
          TraceSeverity.Monitorable, EventSeverity.None),
          TraceSeverity.Monitorable,
          "Spojování voláno bez vstupních datastreamů.", null);
```

8.4.2 Konfigurace modulu

Použité farmové řešení umožňuje číst soubory z disku, bylo by tedy možné číst konfigurační soubor config.xml. Zároveň SharePoint server využívá MS SQL databázi. Bylo by možné použít načítání a ukládání konfigurace i z databáze. SharePoint platforma má však pro ukládání nastavení vlastní standardní přístup – takzvaný *Property Bag*. Jedná se o standardizovaný způsob ukládání nastavení, který využívá databázi SharePoint serveru a zároveň umožňuje divergovat nastavení podle kontextu stránek. Farmové řešení tak může být aplikace, která běží na více kolekcích stránek, která má zároveň pro jednotlivé kolekce stránek individuální nastavení. Příklad použití Property Bag:

```
SPSite site = SPContext.Current.Site;
SPWeb sitePropWeb = site.RootWeb;
SPPropertyBag properties = sitePropWeb.Properties;
if (properties.ContainsKey(valueName))
{
            property = (String)properties[valueName];
}
```

8.4.3 IfillService

Ukázalo se, že v prostředí systému SharePoint nefunguje metoda `SearchAndReplace()` třídy `TextReplacer` z knihovny *OpenXmlPowerTools* pro OpenXml soubor zpracovávaný jako datový proud. Bylo tak nutné nahradit implementaci rozhraní `IfillService` třídu `SimpleFillService` za jinou třídu, která by nevyužívala této metody. Pro potřebu doplňku SPMergeService jsem vytvořil novou třídu, která implementuje rozhraní `IfillService` – `SPEWFillService`. Ta třídu `TextReplacer` nahrazuje třídou `SearchAndReplacer`, jejímž autorem je Eric White.[19]

8.4.4 Úložiště digitálních certifikátů

Aplikace, která běží pod kontextem SharePoint serveru, nemá právo exportovat primární klíče digitálních certifikátů z úložiště stávajícího uživatele. Nemůže tedy tyto certifikáty použít k tvorbě elektronického podpisu. Proto ve třídách `XmlDSigService` a `XadesService` bylo nutné změnit používané úložiště za úložiště stroje.

8.5 Výsledek integrace

Integrace proběhla tak, jak bylo v kapitole 8.1 navrženo. Uživatel si vybere položku seznamu, kterou chce použít k naplnění dat, zvolí „Vytvořit dokument“, vybere jednu nebo více šablon a zvolí, zda výsledný dokument také podepsat. Výsledek je pak uložen do předem určené složky. Problém nastává během ukládání dokumentu do systému SharePoint. Ten jej patrně při ukládání pozmění a tím zneplatní podpisy, které jsou k tomuto dokumentu vytvořeny. Problém zcela jistě není v implementaci modulu, ale je způsoben systémem SharePoint. V případě, že výsledný soubor neuložíme do systému SharePoint, ale uložíme jej přímo na souborový systém, je vždy podpis validní.

Mým návrhem je provést výzkum na zjištění, k jakým změnám v dokumentu při ukládání dochází a zda není možné toto chování systému explicitně zakázat, či obejít.

9 Porovnání různých metod skládání dokumentů

V kapitole 6.3 bylo zmíněno, že jsem implementoval dvě různé třídy, které jsou určeny pro skládání DOCX dokumentů z již existujících dokumentů tohoto typu. Každá z těchto tříd využívá zcela odlišné přístupy a bylo naznačeno, že každá bude mít specifické vlastnosti.

Pro připomenutí, třída `AltChunkMergeService` využívá možnosti importovat na určité místo v hlavní části dokumentu (AltChunk kotva) jiný dokument jako AlternativeFormatImportPart. Tuto možnost poskytuje knihovna OpenXML SDK. Na první pohled je zřejmé, že tento přístup bude umožňovat rychlé skládání dokumentů, neboť se nejedná o nic jiného než přidání referencí v určitém místě dokumentu. Otázkou je, zda bude mít toto řešení dopad na dobu otevírání dokumentu a na velikost výsledného dokumentu. Další nevýhodou řešení pomocí importu je, že importovaný obsah nelze prohledávat a nahrazovat pomocí třídy SimpleFillService. Musíme tedy nejprve šablony naplnit a teprve poté je spojit. Tato nevýhoda může být zásadní, pokud potřebujeme do vytvořeného dokumentu doplnit data až později. V tom případě není možné třídu AltChunkMergeService použít.

Druhá třída `PTMergeService` používá třídu DocumentBuilder z knihovny OpenXmlPowerTools. Zde je možné třídě DocumentBuilder předat jednotlivé dokumenty a třída sama je dokáže spojit i vyřešit případné kolize (stylů atd.). Lze předpokládat, že toto řešení bude pro vytváření dokumentu pomalejší než první uvedené. Na druhou stranu očekávám, že bude mít pozitivní dopad na dobu otevírání dokumentu a na jeho výslednou velikost. Na závěr je dobré dodat, že je možné při použití třídy `PTMergeService` dokument nejprve spojit a pak teprve pomocí třídy `SimpleFillService` naplnit.

Testování probíhalo na testovací sestavě s operačním systémem Windows 7 Professional 64-bit, procesorem Intel Core i5-2410M (frekvence 2,3 GHz – 2,6 GHz, L1 Cache 128 KB, L2 Cache 512 KB, L3 Cache 3072 KB), velikost paměti RAM 4096 MB.

9.1 Porovnání doby běhu modulu a výsledné velikosti dokumentu

Porovnávání doby skládání dokumentu probíhalo za pomoci aplikace **MergeSignApp** (kapitola 7.2). V každém zkoumaném případě jsem dokument vygeneroval stokrát. Výsledný čas je průměr naměřených časů. V tabulce 9.1 naleznete výsledky měření. Zkratka FM u názvu třídy znamená, že nejprve došlo k plnění šablon a poté k jejich skládání. Zkratka MF znamená, že nejprve byly šablony spojeny a pak plněny.

Z výsledků vyplývá několik faktů:

- Třída `AltChunkMergeService` je při skládání větších dokumentů až několikanásobně rychlejší než třída `PTMergeService` (na námi zvolených datech byla i více než 5krát rychlejší).
- Třída `PTMergeService` generuje až několikanásobně menší soubory (záleží pochopitelně na velikosti jednotlivých souborů) než třída `AltChunkService` (v extrémních případech našeho měření byl výsledkem i 15krát menší výsledný soubor).
- Třída `AltChunkMergeService` generuje výsledný soubor o velikosti, která zhruba odpovídá velikosti všech spojovaných dokumentů dohromady.
- Použití opačného pořadí (nejdříve spojovat a poté plnit) vede k mnohonásobně horším výsledkům doby plnění než u plnění a následného spojování. Z tohoto zjištění a z výsledků doby plnění největších souborů vyplývá, že plnění dokumentu daty pomocí třídy `SimpleFillService` roste nelineárně s množstvím textu dokumentu.
- Doby spojování pomocí obou tříd jsou vůči dobám plnění zanedbatelné.

Výsledky měření byly pro názornost zaneseny do grafů, které naleznete v příloze B.

		Velikosti (kB)		Doby běhu (ms)		
Použitá třída	n	Vstupní	Výstupní	Skládání	Plnění	Celková
ACMS (FM)	3	37	37	9,4	70,1	79,5
PTMS (FM)	3	37	16	25,5	63,1	87,5
PTMS (MF)	3	37	16	28,5	93,5	122,0
ACMS (FM)	12	147	149	16,7	244,5	261,2
PTMS (FM)	12	147	21	68,2	250,2	317,3
PTMS (MF)	12	147	21	76,9	802,9	879,8
ACMS (FM)	24	294	298	27,1	471,3	498,4
PTMS (FM)	24	294	28	125,0	519,1	644,1
PTMS (MF)	24	294	28	141,8	2999,0	3140,8
ACMS (FM)	36	441	448	38,2	694,1	732,4
PTMS (FM)	36	441	35	192,2	729,1	921,4
PTMS (MF)	36	441	35	209,7	6255,6	6465,3
ACMS (FM)	48	589	597	45,8	943,2	991,0
PTMS (FM)	48	589	41	244,6	953,8	1197,8
PTMS (MF)	48	589	41	283,1	11224,5	11507,7

ACMS (FM) – AltChunkMergeService (Fill -> Merge)
PTMS (FM) – PTMergeService (Fill -> Merge)
PTMS (MF) – PTMergeService (Merge -> Fill)
n – Počet vstupních šablon

Tabulka 9.1: Doba běhu skládání dokumentu a velikost výsledného souboru.

Výsledky předchozího testu otevřely otázku, jak roste doba plnění dokumentu v závislosti na délce jednotlivých souborů. Připravil jsem tedy test, ve kterém plním jeden dokument, jehož obsah má 1000 znaků a 25 nahrazovaných položek. Tento dokument jsem postupně zvětšoval kopírováním původního obsahu na 5000 znaků a 125 položek, 10000 znaků a tak dále. V testu jsem porovnával čas plnění jednoho spojeného souboru s časem plnění a následného složení odpovídajícího počtu původních souborů o rozsahu 1000 znaků.

Z tabulky 9.2 je vidět, že pokud plníme elementární soubory a ty teprve skládáme v jeden, roste doba skládání v závislosti na počtu znaků lineárně. Doba plnění jediného souboru v závislosti na počtu znaků roste zhruba kvadraticky (viz graf č. 3, příloha B).

Počet znaků	Počet nahrazení	t_1 (ms)	t_2 (ms)
1 000	25	152,3	160,4
5 000	125	2136,0	847,3
10 000	250	7528,5	1575,2
15 000	375	16852,0	2339,4
20 000	500	30977,2	3030,1
25 000	625	48097,6	3960,2

Tabulka 9.2: Doba běhu plnění dokumentu v závislosti na velikosti šablon (t_1 - doba plnění jediného vstupního souboru, t_2 - doba plnění elementárních souborů a jejich složení).

9.2 Porovnání doby otevírání dokumentu v aplikaci Office Word 2010

Porovnání doby otevírání dokumentu probíhalo pomocí doplňku **TestTimeToOpenDocument**, jenž byl popsán v kapitole 7.4. Testovány byly dokumenty vytvořené během prvního měření kapitoly 9.1.

Výsledky (v tabulce 9.3) potvrdily předpoklad, že doba otevírání dokumentů vytvořených třídou `AltChunkMergeService` je delší než doba otevírání dokumentů vytvořených třídou `PTMergeService`.

n	t_{ACMS}(ms)	t_{PTMS} (ms)
3	103,5	74,4
12	230,4	87,9
24	422,2	103,2
36	601,8	115,1
48	785,9	124,9

Tabulka 9.3: Doba běhu otevírání dokumentu podle počtu souborů, které byly použity k sestavení testovacího dokumentu (t_{ACMS} - doba otevírání dokumentu vytvořeného třídou AltChunkMergeService,t_{PTMS} - Doba otevírání dokumentu vytvořeného třídou PTMergeService, n - počet vstupních šablon, ze kterých byl dokument vytvořen).

9.3 Závěry porovnání

Pokud shrnu výše uvedené výsledky, mohu konstatovat, že doba skládání dokumentu i doba otevírání výsledného dokumentu je v obou případech lineárně závislá na počtu zdrojových dokumentů. V porovnání s dobou plnění šablon jsou zanedbatelné a pro výběr používané třídy nehraje zcela zásadní roli. Důležitější roli hraje fakt, že po použití třídy `AltChunkMergeService` pro složení dokumentu nelze do výsledného dokumentu vkládat data pomocí třídy `SimpleFillService` a že velikost výsledného dokumentu bývá mnohonásobně větší než při složení dokumentu třídou `PTMergeService`. Z těchto důvodů je praktičtější používat třídu `PTMergeService` vždy, když nemá systém problém s výkonem.

10 Zhodnocení přínosu implementovaných elektronických podpisů a časových razítek

Zhodnocení výsledků implementace elektronických podpisů a časových razítek není směrodatné provádět na základě měření doby běhu ani velikosti výsledného razítka. Mnohem důležitější je posuzovat je z hlediska ověřitelnosti platnosti, z hlediska dostupnosti nástrojů pro ověření a z hlediska jedinečnosti.

10.1 XmlDsigService

V kapitole 6.3.6 jsem popsal, že tato třída poskytuje interní podpis DOCX dokumentů ve formátu XMLDSig pomocí standardních knihoven platformy .NET. XMLDSig je jednoduchý formát elektronického podpisu, který neumožňuje vkládání časových razítek, digitálních certifikátů ani revokačních listů. Jeho využití je v praxi možné, ale kvůli komplikovanému ověření platnosti není příliš vhodné. Uživatel, který dokument ověřuje, musí posuzovat platnost certifikátu digitálního podpisu vůči okamžiku podepisování, nebo mít k dispozici samostatně uložené časové razítko. Zároveň bude muset zjišťovat, zda nedošlo k revokaci certifikátu z jiných zdrojů než z obsahu elektronického podpisu, což znemožňuje ověření platnosti s delším časovým odstupem, pokud by už tyto informace certifikační autorita neposkytovala a uživatel by si nevedl jejich záznamy.

Důvodem, proč se tímto typem podpisu vůbec zabývat je to, že je možné jej poskytnout standardními prostředky. Zároveň existuje nástroj, který jej dokáže ověřit - Microsoft Word. Ověření podpisu v aplikaci Microsoft Word se však liší podle použité verze této aplikace. Microsoft Word 2007 kontroluje pouze integritu dokumentu a neřeší důvěryhodnost certifikátu, pomocí kterého byl podpis vystaven. Microsoft Word 2010 kontroluje i důvěryhodnost certifikátu, podle této aplikace je ale podpis poskytnutý třídou XmlDsigService vyhodnocován jako „částečný“. Přínos této třídy je proto celkem malý – z hlediska funkcionality nepřináší nic nového (tato implementace je už známá), z hlediska ověřování jej nástroje částečně podporují a k archivaci

dokumentu se rozhodně nehodí.

10.2 XadesService

Třída `XadesService` poskytuje zcela odlišný přístup než třída `XmlDsigService`. Je určena k tvorbě externě ukládaného elektronického podpisu ve formátu XAdES v úrovních (T, C, X, X-L). Umožňuje tak vkládání časových razítek, odkazy na digitální certifikáty celého certifikačního řetězce a na revokační listy, dokonce i hodnoty těchto certifikátů a revokačních listů. Tímto způsobem řeší problémy s ověřením podpisu, které vznikají při použití jednoduchého formátu XMLDSig. Na rozdíl od formátu XMLDSig ověřování podpisu typu XAdES podporuje aplikace Microsoft Word až od verze 2010 a to pouze interně připojované podpisy tohoto typu.

Mojí původní myšlenkou bylo implementovat podpis ve formátu XAdES také jako interní podpis, aby bylo možné jej ověřit v aplikaci Word. Tato myšlenka byla implementována, ale aplikace Word podpis vyhodnocovala jako nevalidní (neshodoval se hash dokumentu), z tohoto výsledku však nebylo dohledatelné, ve které části podpisu se vyskytoval problém.

Porovnáváním s podpisem, který vytváří aplikace Word, jsem zjistil, že jednotlivé části dokumentu (document parts) měly shodnou hodnotu hash v obou podpisech a že zde problém patrně nebyl. Další součástí interního podpisu jsou odkazy na části dokumentu a část XAdES podpisu samotného. Tyto části, které jsou ve formátu XML, nejsou podepisovány jako celek, ale jsou nad nimi prováděny transformace pro výběr podepisovaných elementů (Relationship Transform) a transformace pro standardizaci formátu XML (C14N Transform). Ačkoliv jsem tyto transformace aplikoval tak, jak je popsáno v ECMA specifikaci formátu OpenXML, výsledný hash se vždy lišil oproti hash vypočtenému v podpisu aplikace Word. Bohužel nebylo možné porovnat výsledná XML, ze kterých byl vypočten hash, protože se toto XML po aplikovaných transformacích nikde neukládá.

Dalším místem, ve kterém se podpis třídy `XadesService` neshodoval s podpisem vytvářeným v aplikaci Word, bylo pořadí jednotlivých referencí na jednotlivé podepisované části. Podle ECMA specifikace (Open Packaging Conventions) by měly být reference v podpisu řazeny v abecedním pořadí, v podpisu vytvořeném aplikací Word jsou však ukládány neřazené (nebo ře-

zené podle jiného klíče). Otázkou je, zda jiné pořadí referencí uvnitř podpisu může mít vliv na ověřování podpisu. Po vznesení dotazu, jakým postupem jsou aplikovány transformace na části rels v aplikaci Word, na vývojářském fóru MSDN zaměřeném na formát OpenXML jsem byl Microsoft vývojáři odkázán na obecnou literaturu o OpenXML a zároveň mi bylo sděleno, že implementace formátu XAdES je řešením, ke kterému Microsoft neposkytuje podporu.

Při zkoumání interního podpisu typu XAdES, který vytváří aplikace Word, jsem narazil na značné nedostatky této implementace. Mezi částmi dokumentu, které jsou do podpisu zahrnuty, chybí například metadata. Absence těchto dat uvnitř podpisu se mi jeví jako zcela zásadní problém. V podepsaném dokumentu může například kdokoliv změnit autora a podpis dokumentu bude stále validní.

Kvůli všem výše uvedeným důvodům jsem se rozhodl pro implementaci externě ukládaného XAdES podpisu. Ten má stejnou strukturu jako XAdES interní, je ukládán do samostatného souboru a v referencích obsahuje URI podepisovaného dokumentu. Nevýhodou tohoto formátu je pouze slabá podpora v oblasti nástrojů na ověření platnosti. Z tohoto důvodu také vznikla aplikace na ověření platnosti podpisu - XadesVerifyTool (viz kapitola 7.3). Lze očekávat, že externí XAdES podpisy budou stále populárnější. Důvodů je několik:

- XAdES podpis umožňuje připojovat všechny potřebné náležitosti pro automatizované ověřování podpisu, bez nutnosti získávat jakékoliv externí informace.
- Externě ukládaný XAdES podpis umožňuje podpis souborů v jakýchkoliv formátech a podepisování několika souborů najednou.
- Zákon o elektronickém podpisu a zákon o spisové službě a archivnictví zahrnují formát XAdES mezi formáty podpisu, které musí být akceptovány orgány státní správy.

V budoucnu doporučuji rozšířit řešení o úroveň archivační (XAdES-A), která počítá s přidáváním archivních razítek. Podpis typu XAdES-A lze vytvořit z již existujícího formátu XAdES-X-L. Celkový přínos třídy `XadesService` shledávám v tom, že implementace takového řešení nebyla dosud nikde publikována. Řešení totiž využívá knihovnu Microsoft.Xades, která je v tuto chvíli

patrně nejkvalitnější open source knihovna pro práci s XAdES podpisy, kterou zároveň rozšiřuji o možnost tvorby podpisu pro soubory zpracovávané jako datové proudy a o možnost získávat revokační listy pomocí HTTP protokolu.

11 Závěr

Prvním úkolem bylo prozkoumat dokumentové formáty, které jsou vhodné pro automatické skládání dokumentů na platformě .NET, které jsou zároveň vhodné pro další uživatelskou manipulaci a které umožňují připojování elektronických podpisů a časových razítek. Tento úkol jsem naplnil. Mezi zkoumané formáty jsme určil těchto pět zástupců – PDF, OOXML (respektive DOCX), ODF (resp. ODT), XSL-FO a RTF. Po jejich prozkoumání a následné analýze se mezi nejlépe hodnocené formáty podle kritérií uvedených v kapitole 2.1 dostaly formáty OOXML a PDF. Ve speciálních případech bych doporučil také ODF a XSL-FO. Po konzultaci se zadavatelem byl pro implementaci určen formát OOXML.

Dále byly představeny principy elektronických podpisů a časových razítek, včetně jejich právní platnosti. Následně byla naznačena aktuální úroveň podpory elektronického podpisu v běžně používaných aplikacích. Na závěr byly uvedeny pokročilé dokumentové formáty jako vhodná úložiště pro ukládání kompletní informace o elektronickém podpisu.

Následným úkolem bylo vytvořit řešení automatického skládání a plnění dokumentu a připojování elektronických podpisů a časových razítek. Pro splnění tohoto úkolu jsem nejprve vytvořil řešení ve smyslu modulu, který poskytuje zmíněnou funkcionalitu, je snadno rozšiřitelný a konfigurovatelný. Skládání dokumentu z již existujících dokumentů jsem implementoval dvěma různými způsoby, které jsou zcela odlišné a každý z nich má specifické vlastnosti, které byly potvrzeny testováním v kapitole 9.1. Plnění dokumentu textem bylo implementováno pouze jedním způsobem. Následně jsem testoval, jaký je rozdíl doby běhu plnění, pokud plním dílčí soubory a ty poté skládám, nebo pokud nejprve vytvořím jeden složený soubor a na závěr jej naplním textem. Podepisování dokumentu elektronickým podpisem bylo implementováno dvěma různými způsoby. První způsob poskytuje základní tvorbu elektronického podpisu ve formátu XMLDSig, jak je používáno v aplikaci Microsoft Word 2007. Druhá implementace umožňuje vytvářet externě ukládaný podpis v pokročilém formátu XAdES (T, C, X a X-L). Nakonec byla hodnocena použitelnost obou řešení v praxi a další možná rozšíření.

Nad rámec zadání diplomové práce jsem navrhl a naimplementoval integraci modulu do systému pro správu dokumentů SharePoint 2010 (ve verzi Standard). Předvedl jsem tak použití modulu jako součást webové aplikace.

Literatura

[1] *The incredible persistence of email*[online]. Cite World [cit. 2.5.2013] Dostupné z: http://www.citeworld.com/business/21082/email-persists-social-collaboration

[2] *Email Statistics Report*[online]. Radicati Group [cit. 2.5.2013] Dostupné z: http://www.radicati.com/wp/wp-content/uploads/2012/04/Email-Statistics-Report-2012-2016-Executive-Summary.pdf

[3] *Úvod do správy dokumentů*[online]. Microsoft [cit. 2.5.2013] Dostupné z: http://office.microsoft.com/cs-cz/sharepoint-server-help/uvod-do-spravy-dokumentu-HA010241399.aspx?CTT=1

[4] *Dobře zvládnutá implementace DMS usnadňuje práci a šetří čas*[online]. SystemOnLine.cz [cit. 2.5.2013] Dostupné z: http://www.systemonline.cz/sprava-dokumentu/dobre-zvladnuta-implementace-dms-setri-cas.htm

[5] ORION, Egan, *PDF 1.7 is approved as ISO 32000*, 2007 [online]. The Inquirer [cit. 2.5.2013] Dostupné z: http://www.theinquirer.net/inquirer/news/1030411/pdf-approved-iso-32000

[6] *AIIM to Facilitate ISO Standards Process for Leading Electronic Document Format*, 2007 [online]. Adobe Systems [cit. 2.5.2013] Dostupné z: http://www.adobe.com/aboutadobe/pressroom/pressreleases /200701/012907OpenPDFAIIM.html

[7] *Adobe PDF 101 — Quick overview of PDF file format*[online]. Adobe Systems [cit. 2.5.2013] Dostupné z: http://partners.adobe.com/public/developer/tips /topic_tip31.html

[8] *Introducing the Office (2007) Open XML File Formats*, 2006 [online]. MSDN [cit. 2.5.2013]
Dostupné z: http://msdn.microsoft.com/en-us/library/office/aa338205(v=office.12).aspx

[9] *Open Document Format for Office Applications (OpenDocument) v1.0*, 2005 [online] OASIS Standard [cit. 2.5.2013]
Dostupné z: https://www.oasis-open.org/committees/download.php /12572/OpenDocument-v1.0-os.pdf

[10] *NNEC Core Enterprise Services.*[online] NATO Interoperability Standards & Profiles [cit. 2.5.2013]
Dostupné z: http://nhqc3s.nato.int/architecture/_docs/NISPv2/ volume2/ch03s04.html

[11] HOLMAN, Ken G., *What Is XSL-FO*, 2002 [online] O'Reilly's XML.com [cit. 2.5.2013]
Dostupné z: http://www.xml.com/pub/a/2002/03/20/xsl-fo.html

[12] *Extensible Stylesheet Language (XSL) Version 1.1*, 2006 [online] W3C Recommendation [cit. 2.5.2013]
Dostupné z: http://www.w3.org/TR/xsl11/

[13] *Changes in Word 2010 (for ITPros)*, 2010 [online] Microsoft TechNet [cit. 2.5.2013]
Dostupné z: http://technet.microsoft.com/en-us/library/cc179199.aspx#BKMK_Changed

[14] *Rich Text Format (RTF) Specification, version 1.6*, 1999 [online] MSDN [cit. 2.5.2013]
Dostupné z: http://msdn.microsoft.com/en-us/library/aa140277(v=office.10).aspx

[15] *Zákon č. 227/2000 Sb., o elektronickém podpisu*, 2012 [online] Ministerstvo vnitra ČR [cit. 2.5.2013]
Dostupné z: http://www.mvcr.cz/soubor/zakon-c-227-2000-sb-o-elektronickem-podpisu.aspx

[16] Peterka Jiří, RNDr. Ing, *Báječný svět elektronického podpisu*, 2011, CZ.NIC, z.s.p.o.,1. vydání.
ISBN: 978-80-904248-3-8

[17] *Informace o produktu (Microsoft SharePoint 2010* [online] Microsoft [cit. 2.5.2013]

Dostupné z: http://sharepoint.microsoft.com/cs-cz/product/capabilities/Pages/default.aspx

[18] FOWLER, Martin, *Inversion of Control Containers and the Dependency Injection pattern*, 2004 [online] [cit. 2.5.2013]
Dostupné z: http://martinfowler.com/articles/injection.html

[19] WHITE, Eric, *Search and Replace Text in an Open XML WordprocessingML Document*, 2011 [online] OpenXMLDeveloper.org [cit. 2.5.2013]
Dostupné z: http://openxmldeveloper.org/blog/b/openxmldeveloper/archive/2011/05/12/148357.aspx

A Příloha - Uživatelská dokumentace

A.1 MergeSignApp

Je aplikace, která slouží k testování funkcionality knihovny MergeSignModul. Tato aplikace je spustitelná z **bin\MergeSignApp\MergeSignApp.exe**. Před spuštěním doporučuji provést konfiguraci modulu pomocí aplikace **MergeSignModulConfig.exe** (z téhož adresáře). Nejdůležitějším nastavením je výběr certifikátu (na záložce **Nastavení podpisu**), který bude používán k tvorbě elektronického podpisu. Aplikace je otestována v prostředí Windows 7 Professional (64-bit) a předpokládá nainstalovaný .NET Frawork 4.0 nebo vyšší.
Samotné použití programu je intuitivní. Program umožňuje vybrat a seřadit šablony vytvářeného souboru. Minimální množství je jedna šablona, maximální počet není nijak omezený. Dále je možné zadat parametry ve formátu XML. Jeho XSD naleznete v příloze B.

A.2 XadesVerifyTool

Je aplikace, která slouží k testování validity podpisu. Lze jsi spustit z **bin\XadesVerifyTool\XadesVerifyTool.exe**. Po spuštění zvolte Vyber a ověř podpis a následně vyberte XML soubor, ve kterém je uložen externí podpis (podepisovaný dokument musí být dostupný v téže složce). Pokud validace neproběhla úspěšně, budete informováni dialogovým oknem s textem výjimky. Aplikace je otestována v prostředí Windows 7 Professional (64-bit).

A.3 TestTimeToOpenDocument

Je doplněk aplikace Microsoft Word 2010, který slouží k testování doby otevírání dokumentu v aplikaci Microsoft Word 2010. Spust'te instalaci z **bin\TestTimeToOpenDocument\setup.exe** a potvrd'te **Install**. Doplněk se automaticky spouští s aplikací Word. *Poznámka: Pokud byla aplikace Word otevřena nad novým dokumentem, je nutné před použitím tlačítka* ***Otevři a změř*** *provést nejprve jakoukoliv změnu dokumentu (jinak by se*

původní aplikace zavřela a výsledky měření by se nezobrazily). Pro odinstalování doplňku jděte na **Start** -> **Ovládací panely** -> **Progamy a funkce**. Vyberte **TestTimeToOpenDocument** a zvolte **Odinstalovat**. Aplikace je otestována v prostředí Windows 7 Professional (64-bit).

A.4 SPMergeSign

Stručný postup nasazení řešení je popsán pro operační systém Windows Server 2008 R2 Standard. Řešení zahrnuje také zkrácený popis instalace SharePoint Serveru. Instalaci je možné provést i na operační systém Windows 7, či Windows Vista, avšak nejedná se o standardní instalaci a je nutné provést řadu dalších kroků. Část z nich je popsána na webu MSDN (http://msdn.microsoft.com/en-us/library/ee554869(office.14).aspx). Budu předpokládat, že je na cílovém stroji nově nainstalovaný **Windows Server R2 2008 Standard (English)**. Návod popisuje instalaci **SharePoint Server 2010 SP1 64bit (Czech)**, která je pro výukové účely dostupná z programu MSDNAA.

A.4.1 Instalace SharePoint serveru

1. Nainstalujte **WCF Hotfix**. Pro Windows Server 2008 R2 je ke stažení zde: http://go.microsoft.com/fwlink/?LinkID=166231 (může se stát, že tato oprava je na vašem serveru již nainstalována, v tom případě bude instalátor ukončen).

2. Nainstalujte **ADO.NET Data Services Update for .NET Framework 3.5 SP1**. Pro Windows Server 2008 R2 je ke stažení zde: http://go.microsoft.com/fwlink/?LinkId=163524 (může se stát, že tato oprava je na vašem serveru již nainstalována, v tom případě bude instalátor ukončen).

3. Před samotnou instalací SharePoint Server 2010 nainstalujte předpokládané součásti systému. Ty je možné nainstalovat pomocí **PrerequisiteInstaller.exe**, který je dostupný na instalačním médiu SharePoint Server 2010. Tento instalátor je možné použít pouze pro operační systémy Windows Server 2010 SP2 a Windows Server 2010 R2. Na systémech Windows 7 a Vista je nutné tyto předpokládané součásti nainstalovat jiným způsobem.

4. Nyní můžete spustit instalaci SharePoint Server 2010. Na instalačním médiu spusťte **Setup.exe**.

5. Přijměte licenční podmínky a zvolte typ instalace **Samostatně**.

6. Dokončete instalaci a **povolte spuštění konfigurační utility**.

A.4.2 Nasazení řešení

1. Otevře se uvítací obrazovka nové SharePoint instance, která vybízí k výběru šablony webu. Přejděte odkazem do **Galerie řešení**, otevřete záložku **Řešení** a zvolte položku **Nahrát řešení**. Vyberte přiložené řešení **bin\SharePoint\sablona_sp_dp.wsp** a potvrďte nahrání. Ujistěte se, že je řešení aktivováno.

2. Přejděte zpět na domovskou stránku, která vám v tuto chvíli nabízí výběr šablony webu. Přejděte na záložku **Vlastní**, vyberte šablonu **sablona_sp_dp** a potvrďte volbu. Pokud bude nastavení šablony úspěšné, budete přesměrováni na **Domácí stránku** přednastaveného webu.

3. Nyní je potřeba nasadit řešení SPMergeSign. Zároveň nasadíte řešení, které umožňuje měnit nastavení hodnot v Property Bag. Obě řešení jsou „farmového typu“, proto je nutné je do farmy nahrát pomocí SharePoint 2010 Management Shell.

4. Spusťte **Start** -> **All programs** -> **Microsoft SharePoint 2010 Products** -> **Prostředí SharePoint 2010 Management Shell**.

5. Zadejte příkaz **Add-SPSolution –LiteralPath <cesta k wsp balíku PropertyBagSettings2010.wsp>** (balík je umístěn na DVD v adresáři
bin\SharePoint).

6. Zadejte příkaz **Add-SPSolution –LiteralPath <cesta k wsp balíku SPMergeSign.wsp>** (balík je umístěn na DVD v adresář **bin\SharePoint**).

7. Nyní můžete obě řešení spustit na SharePoint farmě. To lze provést z administrátorského rozhraní aplikace, které spustíte: **Start** -> **All programs** -> **Microsoft SharePoint 2010 Products** -> **Centrální správa služby SharePoint 2010**.

8. Přejděte na **Systémové nastavení** -> **Spravovat řešení farmy**. Zde klikněte na **propertybagsettings2010.wsp** a zvolte **Nasadit řešení**. Vyberte nasadit **nyní** a potvrďte **OK**.

9. Opakujte předchozí krok pro **spmergesign.wsp**.

10. Pokud proběhlo nasazení úspěšně, uvidíte stav nasazených řešení **Globálně nasazeno**.

11. Přejděte na domovskou stránku ShrarePoint site (defaultně je použit shodný název stroje, jako u centrálního nastavení, port 80). Zde zvolte v levé horní části okna **Akce webu** -> **Nastavení webu**. Vyberte **Spravovat funkce webu** (pod kategorií **Akce webu**)

12. Aktivujte funkci **SPMergeSign Feature1**.

A.4.3 Ovládání SPMergeSign

1. Nyní můžete vytvářet dokumenty z existujících seznamů. Příkladem je **Seznam zaměstnanců**. Vyvolejte **kontextové menu nad položkou** a zvolte **Vytvoř dokument**.

2. Při prvním vytváření dokumentu budete vyzváni k zadání knihovny, která bude sloužit pro vstupy a výstupy při vytváření dokumentů (defaultně **Sdílené dokumenty**), složku **knihovny**, ve které budou uloženy šablony (defaultně **Šablony**) a službu, kam se budou ukládat vytvořené soubory (defaultně **Výstupy**).

3. Nyní vyberte šablony, které chcete plnit a spojovat a potvrďte. Pokud chcete dokument zároveň podepsat, musíte si nastavit certifikát, který chcete k podpisu použít).

A.4.4 Nastavení podpisu SPMergeSign

1. Spusťte **Start** -> **MMC.exe**. Zvolte **File** -> **Add/Remove Snap-In**. Přidejte položku **Certificates**. Vyberte možnost **Computer account** a následně zvolte **Local Computer**. Potvrďte **Finish** a následně **OK**.

2. V levém sloupci přejděte na **Certificates** -> **Personal** -> **Certificates**. V prostřední části okna se vám zobrazí již nainstalované certifikáty.

Defaultně by měl být alespoň jeden certifikát serveru nainstalovaný. Pokud ne, budete si jej muset nejprve vytvořit (nebo získat) a poté naimportovat.

3. Dvojklikem na certifikát otevřete detaily. V záložce Details zkopírujte položku **serial number**.

4. Přejděte na SharePoint stránkách do nastavení. **Akce webu** -> **Nastavení webu** -> **Property Bag Settings**. Upravte položku **spmergesign_signaturecertificate** – vložte tam zkopírované sériové číslo certifikátu (bez mezer).

5. Nyní můžete testovat skládání včetně připojování elektronického podpisu. Další nastavení můžete měnit taktéž v Property Bag (názvy všech položek nastavení SPMergeSign začínají **spmergesign_**).

B Příloha - XSD

Pro možnost vytváření vlastních XML souborů, jejichž příklady jsou k práci přiloženy, uvádím jejich XSD schémata. Tato schémata naleznete na přiloženém médiu ve složce **doc\XSD**.

B.1 Config.xsd

Toto XSD definuje strukturu XML konfiguračního souboru knihovny MergeSignModul.

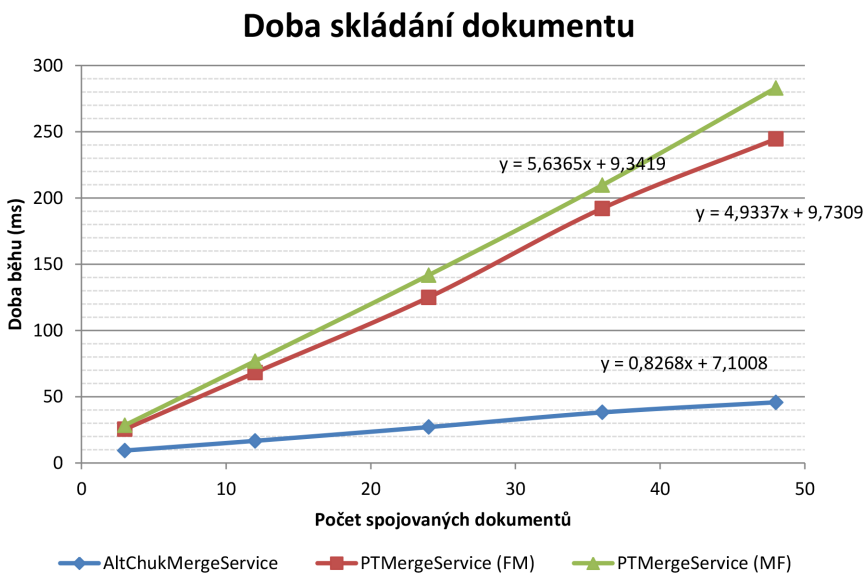
```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="Config" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
 <xs:element name="Config" msdata:IsDataSet="true"
    msdata:UseCurrentLocale="true">
  <xs:complexType>
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Settings">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SignService" type="xs:string" minOccurs="1" />
       <xs:element name="LoadTagService" type="xs:string" minOccurs="1" />
       <xs:element name="FillService" type="xs:string" minOccurs="1" />
       <xs:element name="MergeService" type="xs:string" minOccurs="1" />
       <xs:element name="SignatureCertificate" type="xs:string" minOccurs="0" />
       <xs:element name="TimeStampAthorityUrl" type="xs:string" minOccurs="0" />
       <xs:element name="Country" type="xs:string" minOccurs="0" />
       <xs:element name="PostalCode" type="xs:string" minOccurs="0" />
       <xs:element name="Province" type="xs:string" minOccurs="0" />
       <xs:element name="City" type="xs:string" minOccurs="0" />
       <xs:element name="CrlUri" type="xs:string" minOccurs="0" />
       <xs:element name="Role" type="xs:string" minOccurs="0" />
       <xs:element name="TimeStampX" type="xs:string" minOccurs="0" />
       <xs:element name="XadesLevel" type="xs:string" minOccurs="0" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:choice>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

B.2 Params.xsd

Toto XSD definuje strukturu XML souboru, který nese informace o parametrech plnění pro aplikaci MergeSignApp.

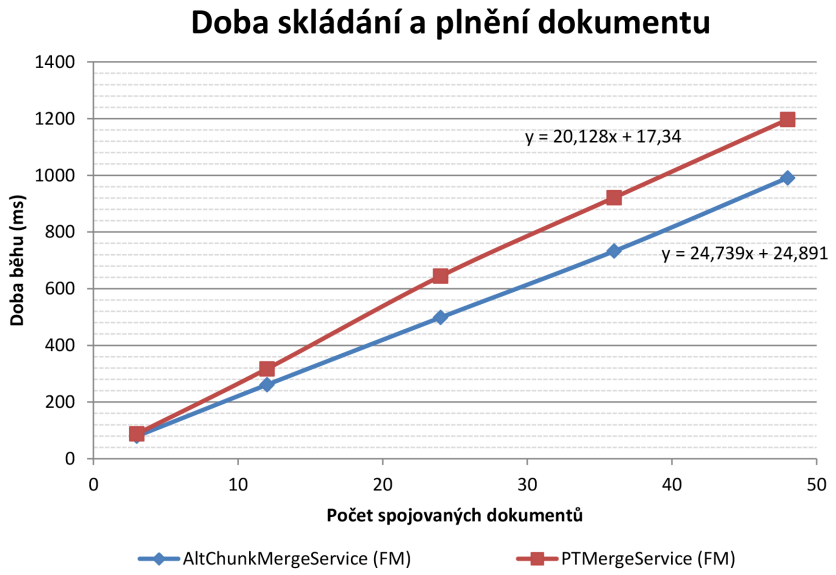
```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="Params" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
 <xs:element name="Params" msdata:IsDataSet="true"
    msdata:UseCurrentLocale="true">
  <xs:complexType>
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Param">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Name" type="xs:string" minOccurs="1" />
       <xs:element name="Value" type="xs:string" minOccurs="1" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:choice>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

C Příloha - Výsledky testů v grafech

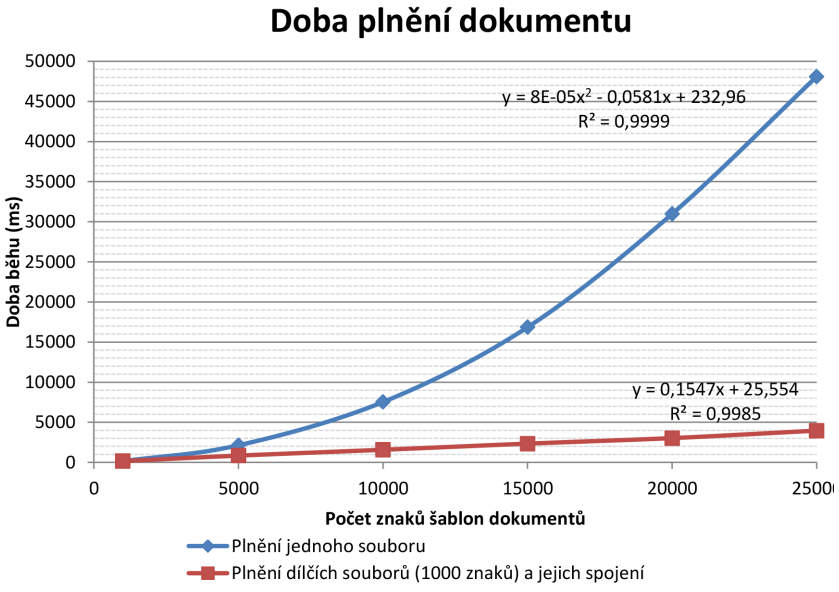


Obrázek C.1: Graf času skládání dokumentu v závislosti na počtu spojovaných dokumentů (zdroj: autor).

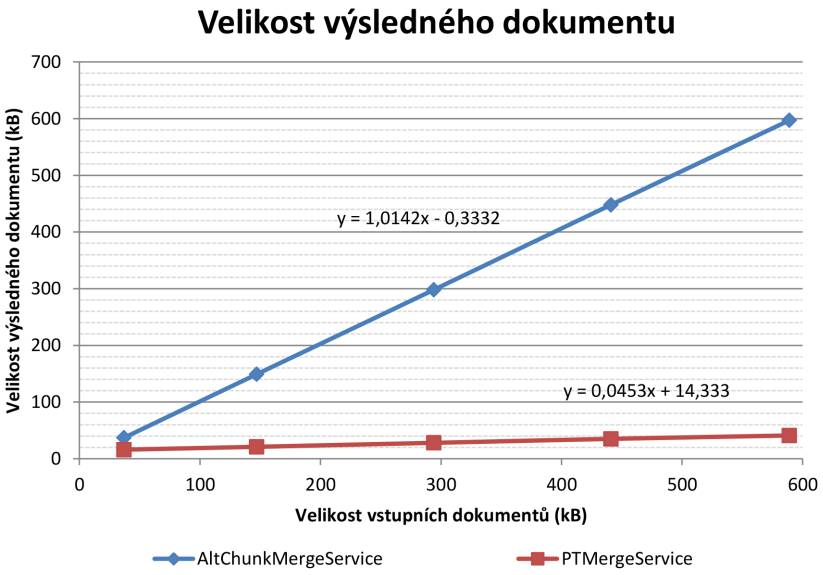
Z prvního grafu (Obrázek C.1) je patrné, že doba skládání dokumentu pomocí třídy DocumentBuilder (PTMergeService) je mnohonásobně pomalejší, než při použití altChunk kotev a importu obsahu (AltChunkMergeService). Pokud také budeme dokument plnit daty, jsou tyto rozdíly zanedbatelné (graf na obrázku C.2). Doba plnění v závislosti na velikosti šablony představuje hrozbu, kterou lze obejít plněním dílčích souborů a jejich následným spojováním (graf na obrázku C.3). Grafy na obrázcích C.4 a C.5 demonstrují výhodu použití třídy DocumentBuilder ve velikosti výsledného souboru a doby jeho otevírání aplikací Microsoft Word 2010.



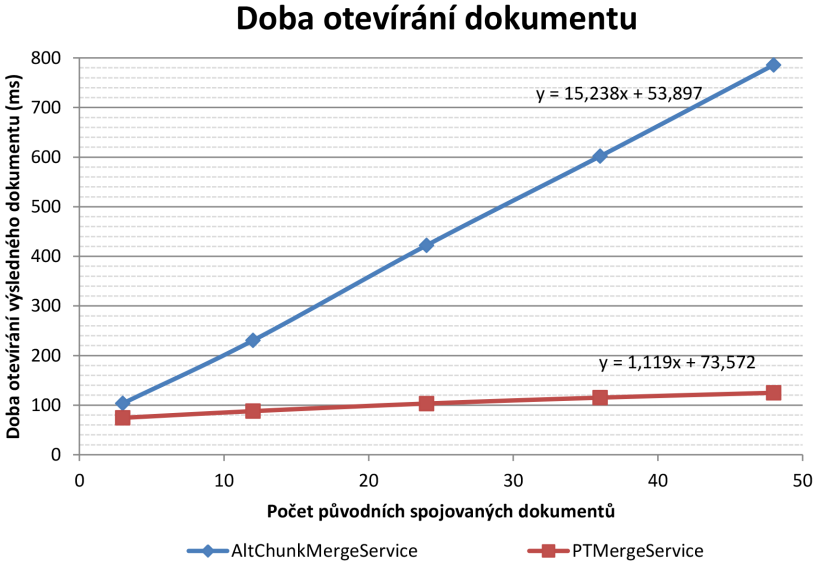
Obrázek C.2: Graf času skládání a plnění dokumentu v závislosti na počtu spojovaných dokumentů (zdroj: autor).



Obrázek C.3: Graf času plnění dokumentu v závislosti na počtu znaků šablony (zdroj: autor).



Obrázek C.4: Graf velikosti výsledného dokumentu v závislosti na velikosti vstupních dokumentů (zdroj: autor).



Obrázek C.5: Graf doby otevírání dokumentu v závislosti na počtu spojovaných dokumentů (zdroj: autor).