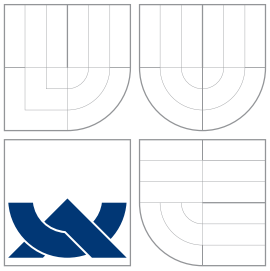
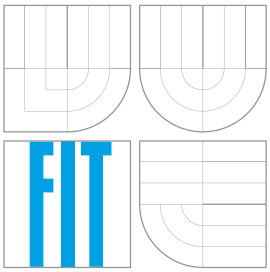


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ŘÍZENÍ TOKU DOKUMENTŮ

INFORMATION SYSTEM FOR DOCUMENT FLOW MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ BARBOŘÍK
AUTHOR

VEDOUCÍ PRÁCE — Mgr. MAREK RYCHLÝ
SUPERVISOR

BRNO 2009

Abstrakt

Tato diplomová práce řeší problematiku správy a životního cyklu dokumentu ve větší organizaci. Popisuje firemní požadavky na informační systém pro správu dokumentů. Srovnává různé nástroje pro popis dokumentu, analyzuje požadavky a nabízí návrh a implementaci vlastního řešení.

Klíčová slova

DMS, Správa dokumentů, Tok dokumentů, Životní cyklus, Historie, Revize, Informační systém

Abstract

This thesis solves the problem of a life cycle of a document and its management in a larger organization. It describes demands of companies on document management informating systems. It compares various tools for document desciption, analyzes the requirements and offers a project and implementation of the solution itself.

Keywords

DMS, Document management, Workflow, Life cycle, History, Revision, Information system

Citace

Jiří Barbořík: Informační systém pro řízení toku dokumentů, diplomová práce, Brno, FIT VUT v Brně, 2009

Informační systém pro řízení toku dokumentů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Mgr. Marka Rychlého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Barbořík
26. května 2009

Poděkování

Dovoluji si poděkovat panu Mgr. Marku Rychlému za odborné vedení diplomové práce. Dále bych chtěl poděkovat své rodině za podporu po celou dobu mého studia.

Obsah

1 Úvod **4**

2 Document management system **5**
2.1 Obecné pojednání o DMS 5
2.2 Základní funkcionalita 5
2.3 Oblasti využití 6
2.4 ISO 9000 7
2.4.1 Systém managementu kvality 7
2.4.2 Vedení a řízení dokumentace 7
2.5 Srovnání vybraných DMS 8
2.5.1 AIP Safe III 8
2.5.2 Captaris Alchemy DMS 8
2.5.3 Online Document Management 8
2.5.4 SpringCM 8
2.5.5 I.R.I.S. DocShare - Document Library Module 8
2.5.6 Hummingbird PC DOCS 9
2.5.7 FileNet Panagon 9
2.5.8 iXOS-ARCHIVE 9
2.5.9 IBM DB2 Content Manager 9
2.5.10 Documentum 6 10
2.5.11 Easy Enterprise 10
2.5.12 Livelink ECM - eDOCS DM 11
2.5.13 Filenet P8 Platform 11
2.5.14 Microsoft Office SharePoint Server 2007 11
2.5.15 IBM Lotus Domino 11

3 Základní terminologie workflow **12**
3.1 Workflow (Workflow) 12
3.1.1 Systém řízení workflow (Workflow management system) 12
3.2 Systém workflow a jeho výhody 12
3.2.1 Co by měl workflow systém poskytovat 13
3.2.2 Co lze od workflow systému očekávat 13
3.3 Typy workflow systémů 14
3.3.1 Produkční workflow 14
3.3.2 Administrativní workflow 15
3.3.3 Ad hoc workflow 15
3.3.4 Kolaborativní workflow 16
3.4 Terminologie workflow systémů 16

3.4.1 Podnikový proces (Business Process) . . . 16
3.4.2 Definice procesu (Process Definition) . . . 16
3.4.3 Činnost (Activity) . . . 16
3.4.4 Instance (Instance) . . . 17
3.5 Referenční model workflow . . . 17
3.5.1 Řídící služba workflow (Workflow Enactment Service) . . . 17
3.5.2 Výkonné jádro workflow (Workflow Engine) . . . 17
3.5.3 Workflow aplikace . . . 19

4 Informační systémy **20**
4.1 Vývoj informačních systémů . . . 20

5 Webové aplikace **22**
5.1 Vývoj a současný trend webových aplikací . . . 22
5.1.1 Ohlédnutí za desktopovými aplikacemi . . . 22
5.1.2 Přechod na webové aplikace . . . 22
5.1.3 Vývoj webových aplikací ASP.NET . . . 23
5.1.4 Vznik a historie HTML a CSS . . . 23

6 Specifikace a analýza požadavků **27**
6.1 Popis vlastního zadání . . . 27
6.2 Neformální specifikace . . . 27
6.2.1 Založení dokumentu . . . 27
6.2.2 Zabezpečení . . . 28
6.2.3 Nastavení práv přístupu . . . 28
6.2.4 Organizování dokumentů . . . 28
6.2.5 Verzování dokumentů . . . 28
6.2.6 Revize dokumentů . . . 29
6.2.7 Workflow . . . 29
6.2.8 Vázané dokumenty . . . 29

7 Návrh řešení **30**
7.1 Use Case diagram . . . 30
7.2 Slovní popis diagramů použití . . . 31
7.2.1 Aktéři - Role v systému . . . 31
7.3 ER diagram . . . 32
7.4 Workflow diagram . . . 33
7.5 Architektura . . . 34

8 Implementace **35**
8.1 Použité technologie . . . 35
8.1.1 Proč jsem si vybral ASP.NET . . . 35
8.1.2 Platforma ASP.NET . . . 35
8.1.3 Historie ASP.NET . . . 38
8.1.4 Microsoft Visual Studio . . . 40
8.1.5 Microsoft SQL Server 2008 . . . 40
8.1.6 Programovací jazyk C# . . . 42
8.2 Popis provedení některých významných vlastností aplikace . . . 42
8.2.1 Common Table Expression (CTE) . . . 42

8.2.2 LINQ 44
8.2.3 AJAX 48
8.3 Grafické uživatelské rozhraní 49
8.4 Popis jednotlivých sekcí aplikace 49
8.4.1 Sekce Administration 49
8.4.2 Sekce Tools 49
8.4.3 My desk 50
8.4.4 Categories 50
8.4.5 Documents 50
8.4.6 Detail dokumentu 50

9 Praktické použití aplikace **52**

10 Závěr **53**

Kapitola 1

Úvod

Vzhledem k masivnímu rozvoji informačních technologií, nárůstu množství dat, informací a dokumentů ve firemním sektoru, se do popředí zájmu stále více dostává problematika životního cyklu a správy dokumentů. Systémy pro správu dokumentů (DMS) nabízejí řešení, které znatelně zvýší efektivitu procesů a vytvoří prostředí pro bezpečné uložení, zpřístupnění a sdílení dokumentů, kde u mnoha firem právě zpracování dokumentů je jednou z hlavních částí administrativy firmy a patří mezi největší zdroje tzv. interních nákladů.

Důvodem pro nasazení DMS je, že firmy jsou nuceny stále rychleji uspokojovat potřeby zákazníků a větší množství informací, tak musí být zpracováno v mnohem kratším čase. Dalším důvodem pro nasazení DMS jsou také vzrůstající legislativní požadavky na uchovávání firemních dokumentů nebo zavádění politiky jakosti norem ISO.

Cílem této práce je rozebrat jednotlivé fáze životního cyklu dokumentu, porovnat jakými způsoby jsou většinou řešeny, analyzovat požadavky a navrhnout obecné řešení pro správu dokumentu a jeho oběhu ve větší organizaci.

Kapitola 2

Document management system

2.1 Obecné pojednání o DMS

Systém pro správu dokumentů (DMS) je možné definovat jako informační systém, který musí umět ukládat, uchovávat, vést historii, archivaci a sdílení obecného elektronického dokumentu. Záměrně je uveden termín obecný elektronický dokument, protože se může jednat nejen o nejčastěji užívané textové dokumenty, ale i o obrazové nebo zvukové záznamy, které vznikly již jako elektronické nebo byly do této podoby převedeny. V minulosti byly DMS provozovány na intranetových firemních sítích, kde bylo důvodem především zvýšení bezpečnosti a také internetové linky neměly takovou propustnost, což minimalizovalo efektivní práci s objemnějšími dokumenty. Dnes se častěji setkáváme s DMS jako internetovými aplikacemi, kde přístup do systému je přes tenkého klienta v podobě internetového prohlížeče. To s sebou přináší obrovskou výhodu, protože k dokumentu může přistupovat oprávněná osoba kdekoliv, kde je připojení k internetu.

S touto výhodou ovšem přichází i bezpečnostní úskalí, kde při návrhu a implementaci je třeba dbát na zvýšení zabezpečení přístupu do systému a na datový tok z něj. Proto je třeba systém vybavit vysokou úrovní zabezpečení a šifrovaným přenosem dat.

Práce s informacemi dnes tvoří jednu z nejdůležitějších částí činnosti firmy a tak není divu, že i document management nabývá na významu, a pokud je rozumně nasazen, může představovat znatelnou konkurenční výhodu.

2.2 Základní funkcionalita

Základním principem DMS je umožnit efektivně spravovat a sdílet jakékoliv dokumenty nebo informace. Toho je docíleno v první řadě implementací bezpečného centralizovaného úložiště. Nad tímto úložištěm běží aplikace DMS, která uživatelům nabízí širokou funkcionalitu pro zpracování dokumentů a zároveň tato aplikace řídí přístup k dokumentům podle nastaveného autorizačního konceptu, čímž je zamezeno zneužití informací při jejich sdílení. Mezi základní rysy DMS patří:

- organizace dokumentů do přehledné struktury
- automatická tvorba a řízení verzí a revizí dokumentů
- podpora práce více uživatelů s jedním dokumentem - funkce check in / check out
- efektivní vyhledávání dokumentů

- podpora vytváření standardizovaných dokumentů, přenos dat do dokumentu
- vytváření dynamických pohledů na dokumenty
- podpora elektronického schvalování a uvolňování dokumentů
- správa firemních šablon dokumentů
- evidence historie práce s dokumenty
- publikace dokumentů na intranet
- podpora převodu papírových dokumentů do elektronické podoby

2.3 Oblasti využití

Podle procesů a jejich vazeb na dokumenty můžeme oblasti využití DMS rozdělit na tři části[9]:

Dokumenty tvoří výstup určitého procesu – DMS nabízí podporu pro vznik a správu dokumentů během jeho celého životního cyklu. U těchto dokumentů se často aplikují funkcionality pro elektronické schválení dokumentu před jeho uvolněním. Do této části mohou spadat všechny dokumenty, které ve firmě vznikají - objednávka, žádanka, smlouvy, podklady pro poptávky/nabídky, směrnice, dopisy klientům, informace pro web, projektová dokumentace atd.

Dokumenty jsou vstupem, který procesy startuje - zde DMS primárně řeší evidenci dokumentu, jejíž součástí je obvykle i jeho naskenování a následné schválení, realizované pomocí elektronického oběhu dokumentů (workflow). Příkladem této skupiny mohou být například příchozí faktury, které jsou zaevidovány, prochází procesem schválení a poté jsou zaúčtovány. Obzvláště u těchto dokumentů je třeba zajistit bezpečnou archivaci a zamezení jejich změny, protože na jejich základě firma provádí své činnosti, které musí být zpětně prokazatelné. Kromě faktur sem můžeme zařadit veškeré příchozí dokumenty. Na archivaci všech příchozích dokumentů je například postaveno řešení spisové služby. V bankovním sektoru mohou být na tomto principu zpracovávány požadavky klientů na různé platební operace, v pojišťovnách podklady pro vypořádání škodní události apod.

Dokumenty podporují dané procesy - do třetí skupiny spadají například veškeré dokumenty, které zaměstnanci potřebují jako podklad pro plnění svých pracovních úkolů. Pro tyto dokumenty je důležité, aby byly snadno dostupné a vyhledatelné. V podstatě se jedná o bezpečný archiv dokumentů, který tvoří většina dokumentů z prvních dvou skupin, které již byly zpracovány a uzavřeny, nicméně musí být pracovníkům stále k dispozici. V rámci firmy jde většinou o aktuální interní předpisy, podepsané smlouvy, odeslané nabídky, schválené objednávky nebo faktury, ISO dokumentace a další.[4]

Výše uvedené rozdělení je však velmi zjednodušené. Obvykle se tyto oblasti vzájemně prolínají a jednotlivé dokumenty a procesy prochází napříč. DMS zároveň přináší řešení pro bezpečnou archivaci všech dokumentů, které jsou jeho součástí, což zajišťuje přístup k těmto informacím i za mnoho let. Systémy DMS jsou velmi flexibilní a jejich customizací[1] je možné je upravit a přizpůsobit konkrétním požadavkům zákazníka.

[1] Softwarové přizpůsobení aplikace podle požadavků zákazníka.

2.4 ISO 9000

Organizace ISO (International Orgranization for Standardization) byla založena v roce 1947 a v dnešní době organizaci tvoří síť národních institucí s centrem v Ženevě. V České republice je zástupcem pro ISO Český normalizační institut.

Hlavní činnost organizace je vývoj technických norem. Pro téma této práce je nejvhodnější uvést mezinárodně uznávané ISO 9000. Hlavním cílem těchto norem je poskytnout návod na vybudování systému řízení kvality. ISO 9000 není určeno pro specifický druh produktu, proto ho lze uplatnit ve všech oblastech výroby a služeb.

2.4.1 Systém managementu kvality

Systém řízení kvality je postaven na identifikaci procesů probíhajících ve firmě a zjištění jejich vzájemných vazeb. Jednotlivé procesy je třeba sledovat, analyzovat jejich efektivitu a využití zdrojů, odhalit nedostatky a provést kroky pro jejich zlepšení. Vysoká míra pozornosti je kladena na zákazníka, jehož potřeby a požadavky musí být v organizaci kvalitně a včas plněny.

Kromě řízení samotných procesů vyžaduje ISO 9000 také vedení firemní dokumentace a její řízení. Právě tomuto bodu se věnuje tato práce primárně, proto je zaměřeno především na normy týkající se struktury dokumentu.

2.4.2 Vedení a řízení dokumentace

Struktura dokumentu

Dokumentace je tvořena soustavou řízených dokumentů, které popisují všechny základní činnosti firmy a obsah řízeného dokumentu je závazný pro vymezenou skupinu zaměstnanců. Každý takový dokument má definovanou strukturu a formální úpravu.

Dokument je opatřen hlavičkou, kde jsou uvedeny základní vlastnosti dokumentu.

Povinné údaje v hlavičce:

- číslo a jméno dokumentu
- autor dokumentu a datum vytvoření
- schvalovatel a datum, kdy dokument vstupuje v účinnost
- seznam pracovních pozic, pro které je dokument určen
- typ dokumentu
- stručný popis obsahu dokumentu
- seznam klíčových slov

2.5 Srovnání vybraných DMS

V této kapitole bude stručně popsáno několik významnějších Document Management Systémů. Ve výčtu nebudou chybět enterprise řešení na špičkové úrovni, jako je například řešení od společnosti EMC. Popisuji také mnohem jednodušší produkty, o kterých je mnohdy jednoduší se dozvědět více informací než u některých robustních korporátních řešení.

2.5.1 AIP Safe III

Systém SAFE III využívá vícevrstvou architekturu, konkrétně controller-view-model. Realizuje všechny potřebné aktivity s digitálním dokumentem, jako zpracování do digitální podoby na vstupu a následné zařazení do příslušného pracovního toku. Umožňuje webový přístup k dokumentu a rychlé fulltextové vyhledávání prostřednictvím evidence metadat. Tento DMS je možné integrovat pomocí autorizačních mechanismů do standardních systému třetích stran.

2.5.2 Captaris Alchemy DMS

V případě Captaris Alchemy DMS jde o typickou architekturu klient-server. Základní kostru tvoří instalace standardního serveru, na který je možné instalovat dva druhy klientů. Prvním instalace nese název Index station a je to plnohodnotná instalace administrátorského klienta a vyhledávacího klienta (Search client), který slouží pouze jako přístup do databáze dokumentů. Klient i server jsou výrazně modifikovatelné dodatečnýma instalacemi rozšiřujících modelů.

2.5.3 Online Document Management

Systém od společnosti HyperOffice nabízí online prostředí pro komplexní správu dokumentů. Umožňuje sdílet dokumenty v rámci firmy i ve vztahu k veřejnosti. Podporuje workflow dokumentů ve firmě a samozřejmostí je uchovávání historie verzí. Velký důraz je kladen na zabezpečení. K systému je možné se přihlásit pouze pomocí hesla se 128-bitovým SSL šifrováním, přičemž není možný přístup přes firewall a VPN. Každému uživateli lze nastavit práva přístupu k dokumentům (prohlížení, editace, mazání nebo nedostupný). Systém prochází denním skenováním proti virům. Zajímavostí je fakt, že servery, na nichž jsou data uložena, jsou chráněny pomocí biometrických zámků.[5]

2.5.4 SpringCM

Klasické řešení pro správu dokumentů dostupné online. Umožňuje vkládání, uchovávání a zpřístupňování dokumentů dle definovaných přístupových práv a samozřejmě jejich organizaci do složek. I zde je kladen velký důraz na bezpečnost. Pro přístup a komunikaci se serverem se využívá 128-bitové SSL šifrování, pravidelně se provádí testování zabezpečení systému a stejně jako u předchozího řešení jsou servery s uloženými daty zabezpečeny v místnosti s biometrickými zámky. SpringCM podporuje workflow firmy a spolupráci na projektech. Z dalších funkcí nechybí fulltextové vyhledávání.[5]

2.5.5 I.R.I.S. DocShare - Document Library Module

I.R.I.S. DocShare patří mezi Content Management Systems (CMS). Jeho součástí je modul Document Library Module (DML), který je určen pro správu dokumentů ve firemním

prostředí. Mimo klasické funkce jako organizování dokumentů, navázání na workflow organizace, verzování, nastavení práv přístupu a fulltextové vyhledávání umožňuje převod všech dokumentů uložených na serveru do formátu PDF. Vzhledem k tomu, že je modul součástí komplexního CMS řešení, je zajištěno propojení s dalšími moduly, např. na modul eNews, který slouží k rozesílání novinek dle nastaveného emailového seznamu, s webovými stránkami a zejména s účetním modulem Invoice Management Module (všechny účetní doklady procházející tímto modulem se automaticky převedou do PDF a následně se ukládají do DML.[8]

2.5.6 Hummingbird PC DOCS

Systém PC DOCS americké společnosti Hummingbird, patří do kategorie pokročilých document management systémů. Přes svou lehce imagingovou orientaci (zviditelněnou zejména vlastním skenovacím subsystémem a prohlížečem více než 160 formátů) nejde o archivační systém, ale o skutečný management dokumentů.

Řešení PC DOCS je určen pro střední a větší organizace. Z funkčního hlediska lze vytknout absenci automatizace životního cyklu dokumentu a nedostatečnou hloubku řešení workflow. Aplikační server i klientské produkty vykazují spíše orientaci na prostředí Microsoft, a naopak nejsou explicitně podporovány některé unixové systémy.[8]

2.5.7 FileNet Panagon

Použití produktu FileNet Panagon 2000 je nejlepší v případě zpracování velkého množství dokumentů v kombinaci s elektronickým oběhem dokumentů. Ke zřejmým kladům patří velký záběr nabízených funkcí v podobě mnoha volitelných modulů resp. licencí, vlastní přímá podpora skenovacího imagingu a hlavně prohlížečů mnoha formátů.

Opět nejsou podporovány unixové systémy IBM AIX a Sun Solaris a databáze Sybase.

2.5.8 iXOS-ARCHIVE

Systém elektronické archivace společnosti iXOS, která je úzce spjata se systémem SAP. Portfolio produktů iXOS nabízí jednak iXOS-ARCHIVE for R/3 pro image enabling procesů v systému SAP a archivaci aplikačních dat SAP, dále iXOS-UniversalARCHIVE pro uživatele mimo systém SAP a konečně DocuLink pro přístup SAP uživatelů k dokumentům, které nejsou ze systému SAP. Pomocí integračních aplikací je pak umožněno ukládání MS Exchange objektů, MS Office a jiných desktop dokumentů či faxů.

Archivační orientace znamená samozřejmě vlastní přímou podporu imagingu a především skenovacího modulu a také přímé ukládání na velkokapacitní optická média. iXOS-Archive podporuje implementace v sítích LAN i WAN. Mezi podporovaná prostředí patří samozřejmě Windows a také některé unix platformy.[8]

2.5.9 IBM DB2 Content Manager

Základem je víceúrovňová distribuovaná architektura, která umožňuje snadnou škálovatelnost pro potřeby menších podniků až po koncerny velkých geografických rozloh. Podporuje práci s HTML a XML, multimediální dokumenty a elektronické dokumenty sady Office. Obsahuje nástroje na správu celého životního cyklu dokumentů, správu záznamů, správu elektronické pošty pro Lotus Notes a Exchange server.
Základní vlastnosti:

- Správa verzí - umožňuje transparentní evidenci jednotlivých dokumentů.
- ODMA podpora - umožňuje snadný vstup dokumentů z ODMA [2]
- Vícehodnotové atributy - umožňují, aby dokumenty mnohých typů jako: audio, video byly identifikovatelné více autory nebo skladateli.
- Index Class subsets - je možnost pro administrátora nastavit práva pro zobrazení citlivých informácí v dokumentech.
- LAN Cache - nabízí použít správce zdrojů jako LAN cache server, a tím urychlit přístup k těmto zdrojům. Uživatel si sám může určit, které dokumenty bude do této cache ukládat.

2.5.10 Documentum 6

Documentum je tradiční a úspěšný produkt společnosti EMC. Jde o vícevrstvou architekturu klient-server, která zvládá kromě klasické správy dokumentů, evidenci verzí, správu pracovních toků, přípravu auditů, zabezpečení dokumentů, zabezpečení dokumentů atd. Hlavní předností tohoto DMS je je celkový přístup formou SOA2 [3] a Documentum Composer. Je to platforma na základě Eclipse, která slouží k definování vlastních aplikací a k modifikacím a nastavením parametrů jednotlivých implementovaných funkcí.

2.5.11 Easy Enterprise

Produkt německé firmy Easy Software, který se pozvolna vyprofiloval z produktu Easy Archive. Opět se jedná o architekturu klient-server, tentokrát napsaný v jazyce Java. Podporuje Windows, Linux a všechny varianty platformy UNIX. Je navržený pro spravování velkého množství dat a vybavený jednoduchou administrativou.

Varianty klientů pro Easy Enterprise:

- **EASYiDOX** – Klient založený na nezávislé platformě, vybavený množstvím funkcí pro vytváření dokumentů.
- **EASYiDOX WEB** – Klient založený na platformě webového prohlížeče, vybavený lehkým vyhledáváním a vytvářením dokumentů. Dovoluje zobrazování MS Office dokumentů bez nutnosti instalace Microsoft Office.
- **EASYiDOX SMART CLIENT** – Klient integrovatelný jako rozšíření MS Office 2003. Přidává dodatečné funkce pro moduly Word, Excel a Access.

[2] Open Document Management API (ODMA) je aplikační programové rozhraní pro správu otevřených dokumentů.

[3] SOA je servisně orientovaná architektura chápána a přijímána jako další fáze budování podnikových informačních systémů. Informační systémy založené na SOA jsou sestaveny ze vzájemně provázaných procesů postavených na službách.

2.5.12 Livelink ECM - eDOCS DM

Produkt byl původně vyvíjený firmou Humingbird a nesl označení Hummingbird Enterprise - Data Management System. Jedná se o produkt určený firmám, které chtějí jednoduše a rychle spravovat, organizovat a sdílet dokumenty. Balík služeb eDOCS nabízí širokou paletu nástrojů pro správu dokumentů, možnost integrovat moduly pro práci v týmu, okamžité zasílání zpráv, správu záznamů, pracovních toků, publikačních analýz atd.

2.5.13 Filenet P8 Platform

Produkt od firmy IBA, která v divizi Filenet dlouhodobě připravuje software pro správu firemních dokumentů. Základ tohoto produktu tvoří následujících šest modulů:

- **Business Process Manager** – Slouží firemnímu managementu ke správě pracovních postupů v oblasti optimalizace produktivity nebo redukcím délky pracovních cyklů.
- **Content Manager** – Modul pro centralizovanou správu dokumentů včetně práv přístupu.
- **Email Manager** – Modul pro kompletní správu elektronické pošty, směrování prezentovaných dokumentů a sdílení dat.
- **Forms Manager** – Modul pro návrh, implementaci elektronických formulářů a následné zpracování vyplněných informací.
- **Image Manager** – Modul pro správu obrazových a vizuálních dokumentů.
- **Team Collaboration Manager** – Modul pro podporu práce v týmu a koordinaci pracovních procesů.

2.5.14 Microsoft Office SharePoint Server 2007

Jedná se o kvalitně škálovatelný portálový server nabízející jednoduchou správu, sdílené vyhledávání, výměnu a publikaci informací a dokumentů. SharePoint Server je v podstatě výkonné jádro, které je možné snadno rozšířit o vlastní implementace webových služeb nebo aplikacemi třetích stran.

2.5.15 IBM Lotus Domino

Lotus Domino od firmy IBM, který byl původně vyvíjený jako Lotus Notes Server, nabízí kline-server architekturu pro aplikace Lotus Notes. Od verze 7 může Domino server využívat databáze DB2, poskytuje centralizovanou správu dokumentů a záznamu a nabízí možnost přímé komunikace formou diskusních skupin.

Základní vlastnosti:

- Správa životního cyklu dokumentů od vytvoření po archivaci.
- Kooperace pracovních skupin a práce v týmu.
- Správa dokumentů balíku MS Office, Microsoft Outlook, Lotus Domino atd.
- Řeší bezpečný přístup k chráněným informacím.

Kapitola 3

Základní terminologie workflow

3.1 Workflow (Workflow)

Pojem workflow je používán v mnoha významech, od vlastního procesu až po počítačové systémy, které zajišťují jeho automatizaci. O sjednocení terminologie v této oblasti se snaží instituce Workflow Management Coalition, která vydala v roce 1996 terminologický slovník[1], kde je pojem workflow definován následovně:

Workflow je automatizace celého nebo části podnikového procesu, během kterého jsou dokumenty, informace nebo úkoly předávány od jednoho účastníka procesu ke druhému podle sady procedurálních pravidel. Přičemž proces je definován jako množina jedné nebo více propojených činností společně přispívajících k dosažení podnikového cíle.[2]

3.1.1 Systém řízení workflow (Workflow management system)

Řízení workflow zajišťuje Workflow management system, který definuje, vytváří a řídí průběh procesu je schopen komunikovat s účastníky workflow a v případě potřeby spustit další aplikace.[1]

Systém řízení workflow zajišťuje procedurální automatizaci podnikového procesu řízením posloupnosti pracovních činností a vyvoláním odpovídajících lidských nebo technických zdrojů. Poskytuje administrativní a monitorovací funkce, jako je například zrušení procesu, změna účastníka procesu, kontrola stavu procesu apod. WfMC vytvořila referenční model popisující jeho strukturu a rozhraní.

Workflow systémy obvykle pokrývají nejenom fázi realizační, ale i fázi přípravnou, kde dochází k nastavení všech procesů. Obvykle se nastavují parametry návazností procesu, schvalování, autorizace atd. Poslední fází je i sledování a vyhodnocování, kde je možné monitorovat a vyhodnocovat reálný průběh procesu.

3.2 Systém workflow a jeho výhody

Komplikované soustavy operací nebo počítačových transakcí, které je nutno organizovat do příslušných sekvencí a koordinovat jejich provádění nejen v závislosti na potřebném

pořadí, ale i docílených výsledcích či selhání jednotlivých kroků, představují výzvu k implementaci podpůrného počítačového systému. Počítačové modely podnikových procesů specifikují všechny potřebné parametry pro provedení těchto procesů. Tyto parametry zahrnují ustanovení pořadí a podmínek, za nichž mohou být jednotlivé kroky procesů vykonány.

Instrukce podniku vytváří kombinace všech jeho procesů. Obvykle však tato infrastruktura není kompletně dokumentována, protože její velkou část představují postupy, které jsou zkonstruovány a udržovány pouze v hlavách zaměstnanců nebo jsou roztroušeny po směrnicích či jsou respektovány v rámci neformálních pravidel.

Jako další aspekt stojí za zmínku rychlost provádění procesů. Moderní řídící metody mají vždy v centru pozornosti rychlost jednotlivých podnikových cyklů. Jde-li o rozhodující procesy, bude jejich rychlejší a efektivnější provedení příznivě ovlivňovat rozvoj podnikání a boj s konkurencí.

3.2.1 Co by měl workflow systém poskytovat

grafický návrh workflow – tím je míněno grafické vytvoření map workflow procesů, které definují tok činností a úkolů vykonávané od startu do cíle

role – schopnost přiřadit jednotlivým činnostem role nebo pracovní funkce, aby definice workflow nemusela být měněna vždy se změnou pracovníka

pravidla – schopnost vložit do definice workflow logiku procesu bez potřeby programování

řešení výjimek – možnost řešit výjimečné situace (dlouhodobá nepřítomnost zodpovědného pracovníka apod.)

monitoring – monitorovat jednotlivé výskyty procesů; ideálním je řešení, kdy je tato funkce přístupná všem účastníkům průběhu procesu administrátorovi workflow

měřitelnost – schopnost generovat statistické zprávy, které jsou podkladem pro zjištění časového průběhu procesu a jeho nákladů

simulace – možnost testovat workflow procesy na jednom počítači před jeho spuštěním v síti

aktivita – workflow musí uživatele informovat o nových úkolech, upozorňovat je na termíny úkolů a případně přesměrovat úkoly na jiné uživatele

připojování dokumentů – dokumenty jsou klíčovou součástí řady podnikových procesů, a proto musí poskytovat efektivní prostředky pro jejich integraci do workflow

3.2.2 Co lze od workflow systému očekávat

Workflow systém přispívá ke:

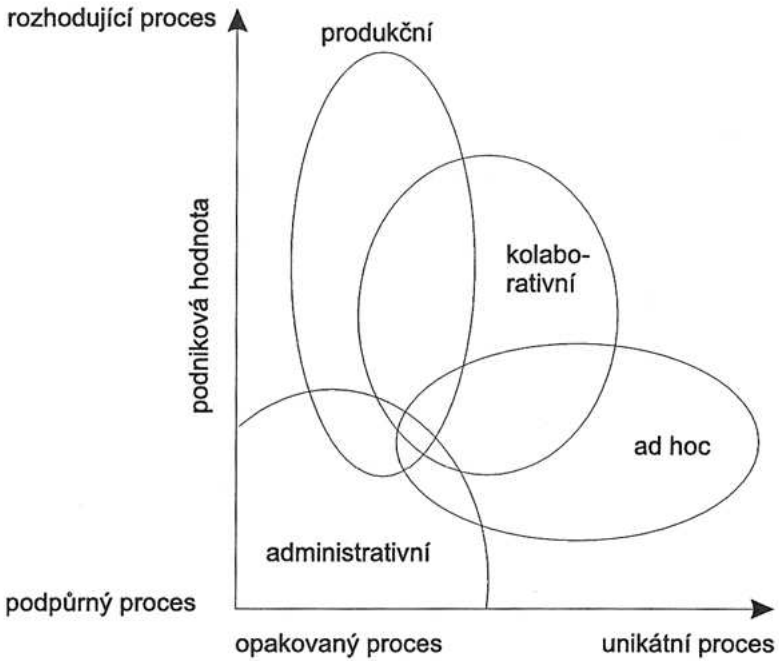
- změně podnikových procesů, zlepšuje organizaci a kvalitu práce,
- zavedení standardních postupů zvyšuje efektivitu práce a snižuje provozní náklady,
- pracovní postupy jsou uchovány v systému, ne v hlavách odcházejících pracovníků,
- vyřizování případů se značně urychluje,

- v každém okamžiku je zjistitelný stav konkrétního případu,
- veškeré verze změn v kolujících dokumentech jsou autorizovány,
- průběh každého případu je zachycen v historii, kterou nelze dodatečně měnit,
- je podporováno řízení kvality.

3.3 Typy workflow systémů

Podle charakteru procesů jsou rozlišovány čtyři typy workflow systémů[2]:

- produkční,
- administrativní,
- kolaborativní,
- ad hoc.



Obrázek 3.1: Typy workflow systémů podle charakteru procesů

3.3.1 Produkční workflow

Produkční workflow podporuje hlavní podnikové procesy. Ty procesy, které vytvářejí přidanou hodnotu k finálnímu produktu, a na kterých závisí spokojenost zákazníka (např. vyřízení žádosti o poskytnutí půjčky, likvidace pojistné události apod.). Procesy jsou dobře strukturovatelné, výskyt případů procesů je častý, práce s aktivitami procesu zabírá uživateli

většinu pracovní doby. Důležitá je integrace s dalšími firemními aplikacemi, čím je kratší doba mezi jednotlivými kroky procesu, tím je systém efektivnější, pružnost změn definice procesu není důležitá, protože změny procesu nejsou denní záležitostí, souvisí většinou s rozsáhlejšími změnami v celé organizaci.

Produkční workflow má většinou tyto vlastnosti:

- pružnost změn definice procesu není důležitá, protože jejich výskyt není vysoký,
- změna definice procesu není záležitostí koncových uživatelů, ale specialistů,
- čím kratší je doba mezi jednotlivými kroky procesu, tím je systém produktivnější.

3.3.2 Administrativní workflow

Administrativní Workflow je určeno k vyřizování běžné každodenní agendy (např. vystavení objednávky, sledování výdajů, vyřízení reklamace, povolení pracovní cesty apod.). Tyto procesy jsou dobře strukturovatelné, často se opakují, bývají jednoduché - s malým počtem alternativních možností, obvykle jsou vázány na standardizované formuláře a dokumenty. Dobré řešení musí respektovat, že téměř každý v organizaci je jejich potencionálním účastníkem, proto je důležitá dostupnost systému pro každého a že účastnící administrativního workflow jsou příležitostní, workflow není jejich hlavní pracovní náplní.

Administrativní workflow má většinou tyto vlastnosti:

- téměř každý v organizaci je jeho potencionálním účastníkem, proto je velice důležitá dostupnost systém,
- účastníci administrativního workflow jsou pouze příležitostní, workflow není jejich hlavní náplní,
- administrativní workflow podléhá občasným změnám,
- administrativní workflow jednotlivých organizací se velice liší.

3.3.3 Ad hoc workflow

Ad hoc workflow je založeno na náhodnosti vzniku workflow procesu. Procesy jsou většinou jedinečné, je možné je definovat až v okamžiku jejich vzniku (např. odpověď na dotaz zákazníka, vypracování výroční zprávy). Vyžadují od uživatelů vysokou míru samostatnosti. Důležitá je zde široká přístupnost Workflow produktu a snadná definice workflow procesu.

Ad hoc workflow má většinou tyto vlastnosti:

- široká přístupnost workflow produktu,
- snadná definice workflow procesu.

3.3.4 Kolaborativní workflow

Kolaborativní workflow je zaměřeno na podporu skupinové spolupráce. Typická je existence dokumentu, jehož prostřednictvím si účastníci vyměňují své poznatky, a který se stane výsledkem jejich společné práce (např. zpracování kupní smlouvy, tvorba propagačního materiálu, návrh nové služby apod.). Očekáván je dokument, na kterém spolupracuje několik uživatelů, a který prochází několika schvalovacími cykly. U těchto workflow je klíčový proces a dokument, spolupracují s ním většinou tvůrčí pracovníci, musí tedy umožňovat kreativitu pracovníků, musí být pružné, neboť tvůrčí pracovníci často využívají nepředddefinované cesty.

Kolaborativní workflow má většinou tyto vlastnosti:

- účastníci mohou nebo mají pracovat společně,
- procesy jsou méně rigidní, charakteristická je dynamická změna definice procesu,
- musí být pružné, protože tvůrčí pracovníci často využívají nepředddefinované cesty.

3.4 Terminologie workflow systémů

3.4.1 Podnikový proces (Business Process)

V souvislosti s workflow je v [1] podnikový proces definován jako množina jedné nebo více propojených činností přispívající k dosažení podnikového cíle, obvykle ve vazbě na organizační strukturu, která definuje funkční role a vztahy.

3.4.2 Definice procesu (Process Definition)

Reprezentace podnikového procesu ve formě, která podporuje automatickou manipulaci jako je modelování nebo zpracování systémem řízení workflow. Definice procesu se skládá ze sítě činností a vztahů mezi nimi.

Definice procesu musí obsahovat přiřazení rolí, stanovení pravidel pro přechod na další činnost. Při přiřazování rolí se definice procesu odkazuje na organizační model obsahující organizační strukturu a role uvnitř organizace.

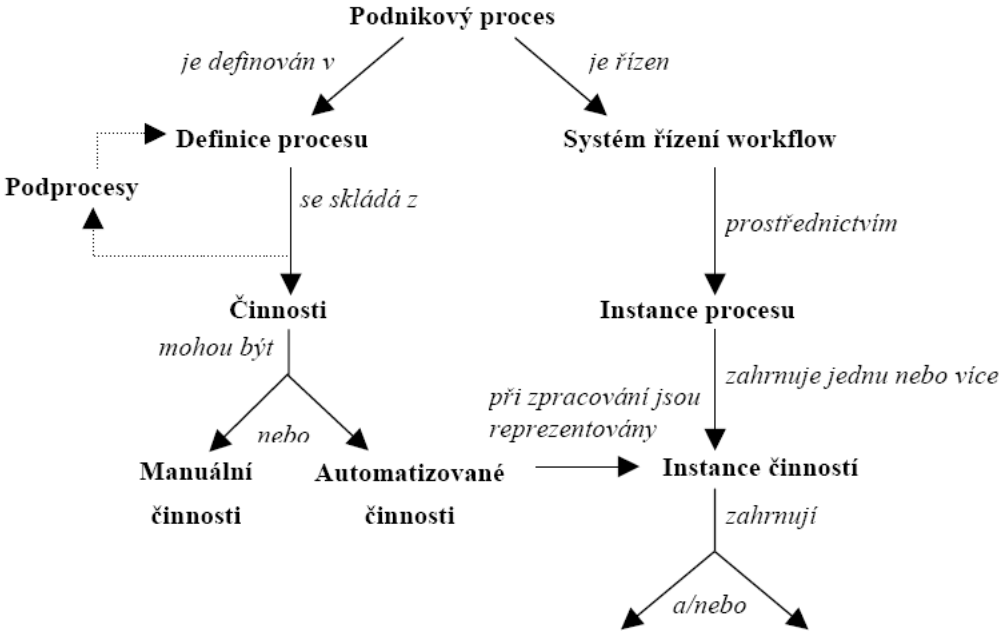
3.4.3 Činnost (Activity)

Činnost je popis části práce, která tvoří jeden logický krok procesu. Činnost může být manuální, která není podporována počítačovou automatizací, nebo workflow (automatizovanou) činností. Workflow činnost vyžaduje podporu zpracování procesu lidskými nebo strojovými zdroji.

Definice procesu obvykle obsahuje mnoho činností, které jsou logicky propojeny tak, jak se podílejí na celkové realizaci podnikového procesu. Činnost je typicky nejmenší jednotkou práce, která má časový rámec.

3.4.4 Instance (Instance)

Reprezentuje jedno konkrétní zpracování procesu nebo činnosti procesu včetně použitých dat. Každá instance představuje samostatně vykonávané vlákno procesu nebo činnosti, které může být řízeno nezávisle a má svůj interní stav a zvenčí viditelnou identitu, která může být použita k manipulaci, například k získání auditačních údajů dané instance.



Obrázek 3.2: Vztah mezi základními pojmy

3.5 Referenční model workflow

Referenční model workflow, který vznikl z implementačního modelu workflow systému, vytvořila organizace WfMC. Referenční model představuje základní architekturu workflow systémů.

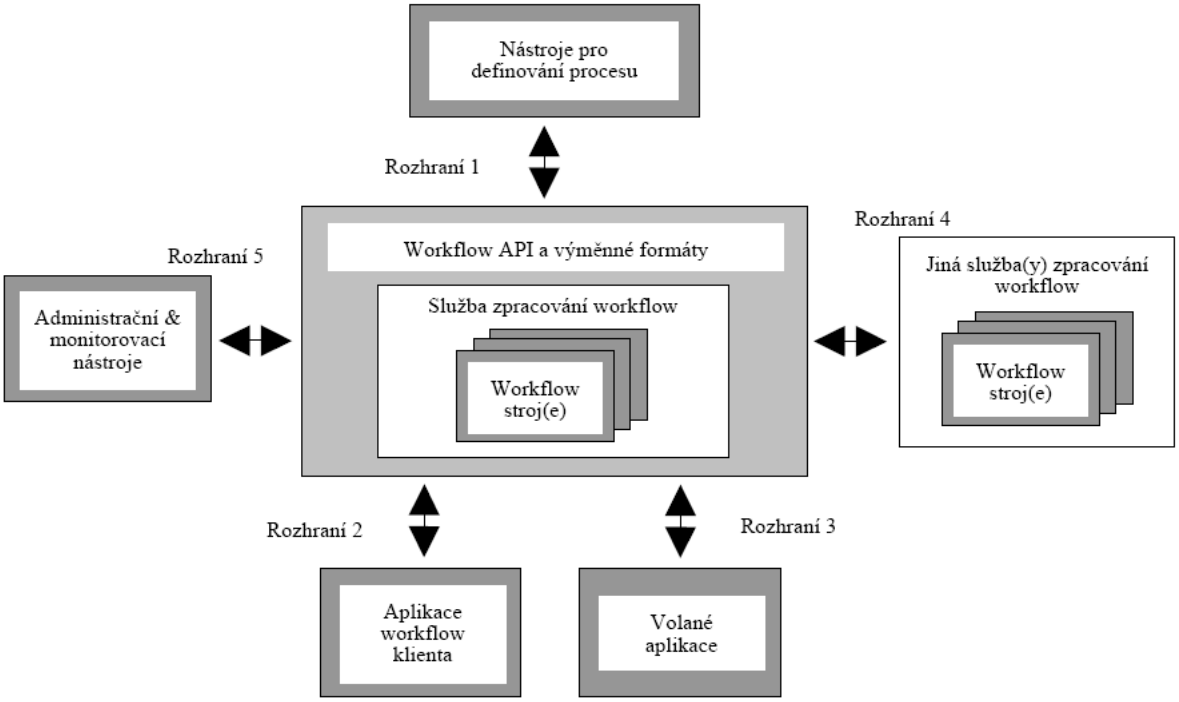
3.5.1 Řídící služba workflow (Workflow Enactment Service)

V [1] je definovaná jako software , který prostřednictvím výkonného jádra workflow vytváří, spouští a řídí výskyty procesů. Ostatní aplikace mohou komunikovat s touto službou prostřednictvím aplikačního programového rozhraní workflow (WAPI, Workflow APIs and Interchance Formats, Workflow API a výměnné formáty)

3.5.2 Výkonné jádro workflow (Workflow Engine)

Výkonné jádro workflow je software, který zajišťuje zpracování výskytů procesů.[1]

Dokáže interpretovat definici procesu, řídí zpracování výskytů procesů včetně jejich spuštění, zastavení, přerušení, obnovení, atd. Naviguje mezi činnostmi, které vyžadují sekvenční či paralelní zpracování, hlídá lhůty apod. Zajišťuje kontrolní a řídící funkce. Většinou nezahrnuje funkce na zpracování fronty úkolů.



Obrázek 3.3: Referenční model workflow

Rozhraní 1 je určeno k propojení nástrojů pro modelování a definici procesů s workflow produkty.

Rozhraní 2 je rozhraním mezi řídící službou workflow a klientskou wokflow aplikací.

Rozhraní 3 je rozhraní mezi workflow a externí aplikací.

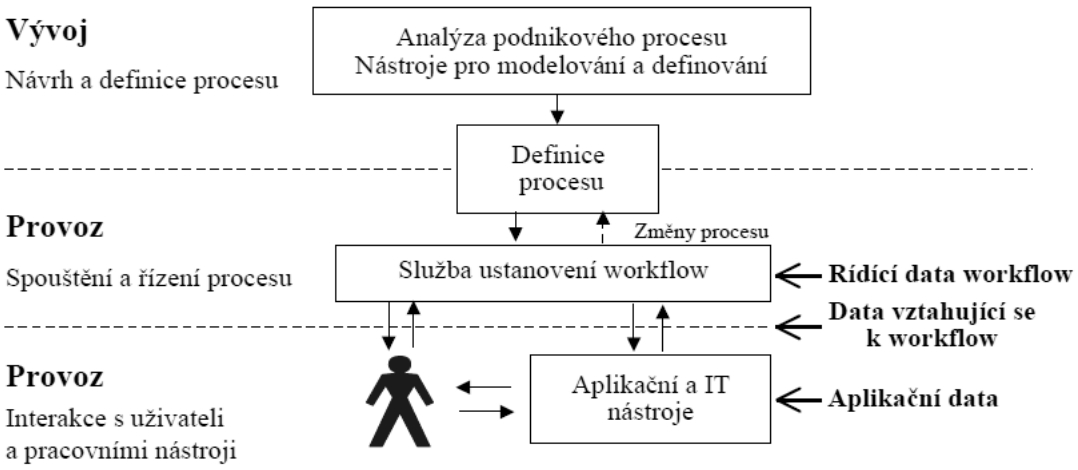
Rozhraní 4 je rozhraní mezi jednotlivými řídícími službami workflow mezi nimiž má být zajištěna interoperabilita.

Rozhraní 5 je rozhraní řídící služby workflow s nástrojem pro správu a monitorováním.

3.5.3 Workflow aplikace

Za workflow aplikaci je možné považovat SW program, který spolupracuje se službou ustanovení workflow a ošetřuje konkrétní části zpracovávání instancí workflow. Referenční model workflow obecně rozeznává dva typy workflow aplikací:

- Klientské aplikace, které vyžadují technické prostředky a služby workflow stroje.
- Volané aplikace, které podporují zpracování určitých úloh nebo pracovních položek a jsou spouštěny systémem řízení workflow.



Obrázek 3.4: Typy dat v řízení systémech workflow

Kapitola 4

Informační systémy

Mezi základní cíle každé organizace patří rozvojová, reprodukční a finační funkce. Úspěchem plnění těchto základních funkcí je ve velké míře podmíněné kvalitou používaného informačního systému. Informační systém organizace přestavuje soubor činností, které zabezpečují sběr, přenos, uchovávání, zpracovávání, distribuci a prezentaci informací. Tyto informace slouží především pro potřeby organizace činností a rozhodování pro management organizace.

4.1 Vývoj informačních systémů

K získání informačního systému pro své specifické potřeby má klient v zásadě dvě možnosti. První možnost je návrh a implementace informačního systému úplně od začátku. Takový systém má řadu výhod v podobě přesně nastavených a implementovaných procesů dle zákazníkových potřeb, předem zvolení hardwarové platformy pro provoz, vlastní firemní design atd. Na druhou stranu je tento vývoj tzv. na zelené louce velmi nákladný a časově náročný.

Většina procesů v různých organizacích se často opakuje, proto se nabízí řešení, kde je možné využít již navržený a implementovaný IS. Takový IS zpravidla tvoří základní jádro s běžnými firemními procesy jakou jsou: správa uživatelů, bezpečnostní politika, správa zdrojů atd. Speciální procesy konkrétní organizace jsou doprogramovány v podobě přídavných modulů.

Změny na již navrženém a implementovaném SW, v podobě úprav stávajících modulů nebo přidáváním vlastních modolů, se nazývají customizace.

S návrhem IS úzce souvisí obor zvaný Softwarové inženýrství. Tento obor se zabývá především systematickým přístupem k analýze návrhu, zaváděním a údržbou nasazeného softwaru.

Proces návrhu systému:

1. Definice parametrů systému - jasná definice toho co očekáváme od IS.
2. Definice pracovních procesů - je třeba definovat jaké úkony bude uživatel v systému provádět a co od něj bude očekávat.
3. Seskupení myšlenkového datového modelu - je sestavení myšlenkového datového modelu a definice datových struktur pro uložení dat.

4. Příprava databázového schématu - převod datového modelu do konkrétních fyzických pojmů.

5. Návrh uživatelského rozhraní - jedná se o nástavbu celého systému. Z hlediska klienta je to však prioritní věc, která ho zajímá.

První čtyři body spadají do analýzy systému a hrají tedy velmi důležitou roli v návrhu celého informačního systému. Pokud v této části uděláme závažnější chybu, může to mít katastrofální důsledky na celý systém. Realizace datového modelu se provádí nejčastěji pomocí ER diagramů, diagramů případu použití nebo univerzálního jazyka UML.

Dalším důležitým aspektem je výběr vhodného programovacího jazyka a databázového systému. Je důležité zvážit, který systém je pro vývoj systému nejvýhodnější. Podstatnou roli hraje rychlost zpracování v daném vývojovém prostředí a samozřejmě také cena produktu.

Kapitola 5

Webové aplikace

5.1 Vývoj a současný trend webových aplikací

5.1.1 Ohlédnutí za desktopovými aplikacemi

Vývoj desktopových aplikací historicky předcházel před webovými aplikacemi, proto pro vývoj desktopových aplikací bylo a stále je mnoho moderních vývojářských prostředí, které ve velké míře usnadňují a rapidně zrychlují vývoj aplikace. Příkladem je Delphi nebo Visual Basis, kde programátor navrhuje aplikaci z navazujících formulářů. Každý takový formulář je složen z ovládacích prvků, jako jsou tlačítka, rozbalovací nabídky, textová pole, kalendáře atd.

Vývojové prostředí nabízí celou paletu nejčastěji užívaných prvků, které přehledně rozděluje do kategorií a nabízí tak programátorovi snadné vkládání těchto prvků do vlastní aplikace. Většina těchto předpřipravených ovládacích prvků je možné graficky i funkčně přizpůsobit.

Každý ovládací prvek je po softwarové stránce objektem. Jeho vzhled a chování je určeno vlastnostmi jakou jsou výška, šířka, zobrazený text, pozadí atd. Na základě uživatelských akcí vyvolávají tyto prvky události, kde vývoj uživatelského rozhraní většinou spočívá ve vytváření obslužného mechanismu těchto událostí.

Jednoduchost a rychlost vývoje desktopových aplikací vedla k jejímu velkému nárůstu na popularitě. Také z pohledu koncových uživatelů jsou tyto aplikací velmi příjemné. Nabízí velký komfort v ovládání a plně využívají dostupné hardwarové vybavení a periferie.

Vedle výše popsaných kladů desktopových aplikací, se vyskytuje minimálně jeden nezanedbatelný nedostatek. Jedná se o správu takových aplikací, která spočívá v instalacích a aktualizacích na cílových stanicích. Tato správa se tak stává velmi náročnou, jak po stránce časové, tak po stránce využití lidských zdrojů. Ve finále jsou tyto zdroje problému vždy spojeny s vysokými finančními náklady.

5.1.2 Přechod na webové aplikace

Právě problematická správa desktopových aplikací vedla v posledních letech k velkému rozmachu aplikací klient-server, které jsou nejrozšířenější v podobě webových aplikací. Výhodou webových aplikací je to, že na straně klienta potřebují pouze libovolný HTML prohlížeč, který je součástí většiny operačních systémů. Tento přístup s sebou nese také negativa. Webové aplikace mají své limity v uživatelském rozhraní a ve využívání lokálního počítače a jeho periferií.

Tento stav se zlepšuje každým rokem, především s příchodem nových technologii jakou jsou Adobe Flash, Microsoft Silverlight, Windows Presentation Foundation (WPF) atd. Samozřejmě nemohu opomenout jazyk JavaScript a s ním spjatou technologii AJAX, která nám umožňuje vytvářet stále více komfortnější webové aplikace podobné těm desktopovým. V aplikacích využívající ve velké míře jazyk JavaScript, který běží na straně klienta, může nastat problém s nízkým výkonem klientského počítače.

Ještě větším problémem při vývoji webových aplikací je neefektivita a cenová nákladnost jejich vývoje. V prohlížečích jsou jednotlivé stránky zobrazovány, na základě HTML, v textovém souboru, kde pomocí jazyka HTML je možné zobrazit pouze jednoduché vizuální prvky. Pokud programátor chce použít složitější prvky, jako jsou rozbalovací stromy, menu nebo kalendáře, musí mít pokročilé znalostí jazyků CSS a JavaScript. Webové aplikace začínají svojí funkcionalitou a designem dohánět aplikace desktopové, ale s tím také rostou hardwarové nároky, které jsou často vyšší než u dříve zmíněných desktopových aplikací.

Vlastní vývoj webových aplikací v tradičních skriptovacích technologií jako je jazyk PHP spočívá v kombinování HTML značek s kódem skriptovacího jazyka. Výsledkem takové aplikace je velmi špatně čitelný zdrojový kód, který se obtížně udržuje a je téměř nemožné ho sdílet mezi více projekty. Především chybějící objektový model je důvodem, proč nevzniklo mnoho kvalitních vývojových prostředí jako u desktovpových aplikací.

5.1.3 Vývoj webových aplikací ASP.NET

Platforma ASP.NET s sebou přináší osvědčené principy desktopového vývoje do prostředí vývoje webového. Podobně jako při vývoji desktopových aplikací je možné využit širokou paletu ovládací prvků. Vzhled je možné opět přizpůsobit pomocí vlastností ovládacího prvku a stejné také zůstává vyvolávaní události dle akcí, které vykonává uživatel.

Pokud by byla nabídka předpřipravených prvků pro programátora nedostatečná, může si nové prvky a komponent obstarat či vytvořit. Existuje mnoho serverů, kde je možné komponenty zakoupit nebo volně stáhnout.

Stejně jako u webových aplikací vytvořené pomocí jiných jazyků je výsledně generován čistý HTML kód v kombinaci s JavaScriptem. Proto na klienta není třeba instalovat žádné podpůrné technologie a moduly. Webové aplikace napsané v ASP.NET tak fungují libovolném internetovém prohlížeči s podporou HTML a JavaScriptu.

5.1.4 Vznik a historie HTML a CSS

Jelikož nedílnou součástí této diplomové práce je návrh a implementace webové aplikace, je dobré zmínit značkovací jazyk HTML, který je základním kamenem všech webových aplikací.

HTML

Jazyk HTML je zkratka z anglického HyperText Markup Language, značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu.

Historie HTML

Vývoj jazyka HTML začal už ve 40. letech 20. století, kdy Vannevar Bush představil hypertextový systém Memex. V dalších letech ho následovali Douglas Engelbart (1963), Ted Nelson, věnující celý svůj život globálnímu hypertextovému systému - Xanadě a Bill Atkinson známý populárním HyperCardem (1985), který byl později překonán systémem NoteCard (1987).

První neoficiální verzi HTML vyvinuli Tim Berners-Lee a Robert Caillau v roce 1989. Oba pracovali v CERNu[1]. Tuto neoficiální verzi spojili s jednoduchým protokolem HTTP (HyperText Transfer Protocol). Prvním prohlížečem byl Mosaic vyrobený v NCSA [2]. Ten umožnil další rozvoj webu a s ním i HTML. Před první oficiální verzí HTML přišlo na svět ještě několik dalších verzí, první v roce 1992, další HTML+ v roce 1994 a poslední HTML 3.0.

Oficiální HTML jazyk vznikl v první verzi v roce 1996. Nejdříve to bylo HTML 2.0 a poté HTML 3.2, které se velice hojně používalo ještě před nedávnem. Předposlední verzi schválilo W3C (World Wide Web Consortium) 18.12.1997 a označuje se jako HTML 4.0. Jelikož i tato verze obsahovala mnoho nedostatků, tak 24.12.1999 W3C uvolnilo opravnou verzi HTML 4.01. Ta se stala zatím poslední verzí HTML a s ní pravděpodobně skončil i vývoj jazyka s názvem HTML. Vývoj HTML již skončil, což ale neznamená, že by neměl své následníky. Své místo si našly jazyky postavené na HTML. Dnes již do světa webu pronikl nový jazyk XHTML, který je velice podobný HTML, a další, složitější - XML (XML Extensive Markup Language). XHTML je v podstatě takovým přechodem mezi velice jednoduchým a hodně volným HTML a složitějším, striktním XML.

Verze jazyka HTML

HTML 2.0

Zachycuje stav jazyka v polovině roku 1994. Standard vydala komunita IETF[3]. Je to první verze, která odpovídá syntaxi SGML (Standard Generalized Markup Language). Přidává k původní specifikaci interaktivní formuláře.

HTML 3.2

Byla vydána v květnu 1996. Připravovaná verze 3.0 byla zastaralá dříve, než byl její návrh dokončen, protože nově vydané verze prohlížečů již byly s vývojem dál. Standard už vydalo konsorcium W3C, stejně jako následující verze. Přidává k jazyku tabulky, zarovnávání textu a stylové elementy pro ovlivňování vzhledu.

HTML 4.0

Byla vydána v prosinci 1997. Do specifikace jazyka přidala nové prvky pro tvorbu tabulek, formulářů a nově byly definovány rámy. Tato verze se snažila dosáhnout původního účelu - prvky by měly vyznačovat význam (sémantiku) jednotlivých částí dokumentu, vzhled má být ovlivňován připojovanými styly. Některé prezentační elementy byly zavrženy.

[1] CERN z franc. Conseil Européen pour la recherche nucléaire - výzkumné centrum fyziky se sídlem v Ženevě

[2] NCSA National Centre for Supercomputer Applications

[3] IETF Internet Engineering Task Force

HTML 4.01

Tato verze opravuje některé chyby verze předchozí a přidává některé nové značky. Je to poslední verze HTML, které se již dále nevyvíjí a je nahrazena novějším XHTML, jehož základem je právě tato poslední verze HTML spolu s jazykem XML.

XHTML

Jazyk XHTML je nástupcem HTML založený na XML. Akceptuje většinu značek a atributů jazyka HTML, ale dodržuje striktní syntaxi.

HTML 1.0 Strict

Čistě strukturální značkování, které neobsahuje žádné značky spojené s formátováním vzhledu.

XHTML 1.0 Transitional

Povoluje atributy pro formátování textů a odkazů v elementu body a některé další atributy.

XHTML 1.0 Frameset

Používá se při využití rámců pro rozdělení okna prohlížeče na dvě nebo více částí.

XHTML 1.1

Poslední verzí je XHTML 1.1, která akceptuje XHTML 1.0 Strict a ještě více se přibližuje jazyku XML.

CSS

Jazyk CSS česky nazývaný jako "kaskádové styly" vznikl za účelem umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Výhodou CSS oproti starému formátování v HTML je, že kód a obsah webu je uložen v souboru .html a veškerý design a formátování se načítá z jednoho souboru .css, který je většinou společný pro celý web.

CSS1

Tato verze byla dokončena již v roce 1996, ovšem podpory ze strany prohlížečů se dočkala až kolem roku 1999. První verze CSS nebyla příliš rozsáhlá, proto umožňovala pouze úpravu písma a barev. Stále chybělo dnes hojně využívané pozicování elementů, které se poprvé objevilo ve verzi CSS-P. Ovšem tato verze nenašla větší uplatnění, protože od prohlížečů se jí nedostalo jednotné interpretace.

CSS2

Tato verze byla dokončena kolem roku 2000 a přináší s sebou, kromě nových vlastností pro formátování písma, především podporu pozicování. Ale i tato verze se nedočkala úplné podpory od většiny standardních webových prohlížečů. Asi nejlépe je na tom prohlížeč Mozilla, který se snaží dodržovat všechny standardy W3C.

Kapitola 6

Specifikace a analýza požadavků

6.1 Popis vlastního zadání

Zadáním této práce je navrhnout a implementovat webový informační systém pro řízení toku dokumentů. Proto bylo nutné se seznámit s problematikou správy a životního cyklu dokumentu ve větší organizaci. Zároveň bylo nutné prohloubit teoretické znalosti o workflow systémech a seznámit se typickými business procesy v organizaci.

6.2 Neformální specifikace

Navržená aplikace se bude soustředit především na výrobu nového elektronického dokumentu a podrobně bude řešit jednotlivé kroky životního cyklu od založení, výrobu až po exspiraci dokumentu.

Aplikaci DMS je vhodné rozdělit do několika modulů. V první řadě je třeba zajistit autorizovaný přístup do systému, proto nemůže chybět modul pro administraci uživatelů IS. V tomto modulu bude mít administrátor možnost nastavit všechna možná oprávnění pro jednotlivé role systému.

Dalším modulem bude modul dokumentů, který bude kompletně spravovat všechny dokumenty. Tento modul bude jistě nejrozsáhlejší a implementačně nejnáročnější.

Pro správu revizí, workflow a přístupovým práv k dokumentům bude sloužit modul nástrojů Tools, který analyzuji především z pohledu bezpečnostní politiky jako také více náročný.

Jako poslední modul bude tzv. modul monitorovací, který bude mít funkci především kontrolní a statickou. Tento modul bude využíván především manažery a vedoucími pracovníky firmy.

6.2.1 Založení dokumentu

Při založení dokumentu jsou uložena následující metadata:

- název
- popis
- klíčová slova
- kategorie dokumentu

- autor/kdo dokument založil
- vedoucí dokumentu
- přístupová práva k dokumentu
- časové razítko založení
- workflow
- termín revize

V následujícím kroku má autor možnost dokument rozdělit na jednotlivé části nebo použít již připravenou šablonu. Realizaci jednotlivých částí dokumentu je možné přiřadit konkrétnímu uživateli. Dále je možné nastavit workflow pro schvalování částí dokumentů a napojení na další procesy výroby dokumentu.

6.2.2 Zabezpečení

Jelikož se bude jednat o webovou aplikaci, bude na zabezpečení systému kladen zvýšený důraz. Aplikace nepochybně bude typu klient-server, kde by měla komunikace probíhat přes šifrovaný přenos SSL. Autentizace uživatele bude realizována pomocí uživatelského jména a hesla. Práva k užívání systému budou korespondovat s rolí, do které bude uživatel přiřazen.

6.2.3 Nastavení práv přístupu

Tato funkce je závislá na složitosti systému a potřebách organizace. V naší aplikaci bude možné přidělovat práva k dokumentu a jeho částem (pokud je takto rozdělen).

6.2.4 Organizování dokumentů

Organizování dokumentů bude prováděno pomocí záložek/kategorií do kterých budou dokumenty řazeny. DMS bude obsahovat také rozšířené vyhledávání, kde bude moct uživatel vyhledávat fulltextově v metadatech dokumentu, ale i dle typu dokumentu, data vložení, verzí dokumentu atd.

6.2.5 Verzování dokumentů

V systému bude možné ukládat všechny změny dokumentu. Každá změna bude mít vlastní verzi, která se bude skládat ze 3 číslic.

Popis čísel verzí:

1. **první číslo** – největší priorita, která se mění při největším zásahu do dokumentu, při této změně se další 2 čísla vynulují,
2. **druhé číslo** – střední priorita, která se zvýší při větší změně obsahu, např. při dodáni nové kapitoly atd. Při této změně se vynuluje poslední číslo verze,
3. **třetí číslo** – nejmenší priorita, která se zvýší jen při drobných úpravách.

6.2.6 Revize dokumentů

Ke každému dokumentu bude možné nastavit jeho životnost. To znamená, že dokumentu bude přidělena nějaká doba, vyjádřena v kalendářních dnech, po kterou je dokument platný. Po vypršení této doby je dokument třeba většinou aktualizovat nebo skartovat. Jako příklad využití revize je kontrola dokumentů, které se odkazují na platné zákony ČR, které se po dobu konání našeho parlamentu mohou měnit.

6.2.7 Workflow

Workflow bude v aplikaci zastoupeno především ve formě schvalování dokumentů, případně jeho částí je-li dokument rozdělen. Dále bude možné přesně definovat, kdo může dokument upravovat, kdo má dokument schválit a také procesní návaznosti při výrobě dokumentu. Na všech úrovních workflow bude možné přidávat komentáře, které budou sloužit k lepší informovanosti a orientaci.

6.2.8 Vázané dokumenty

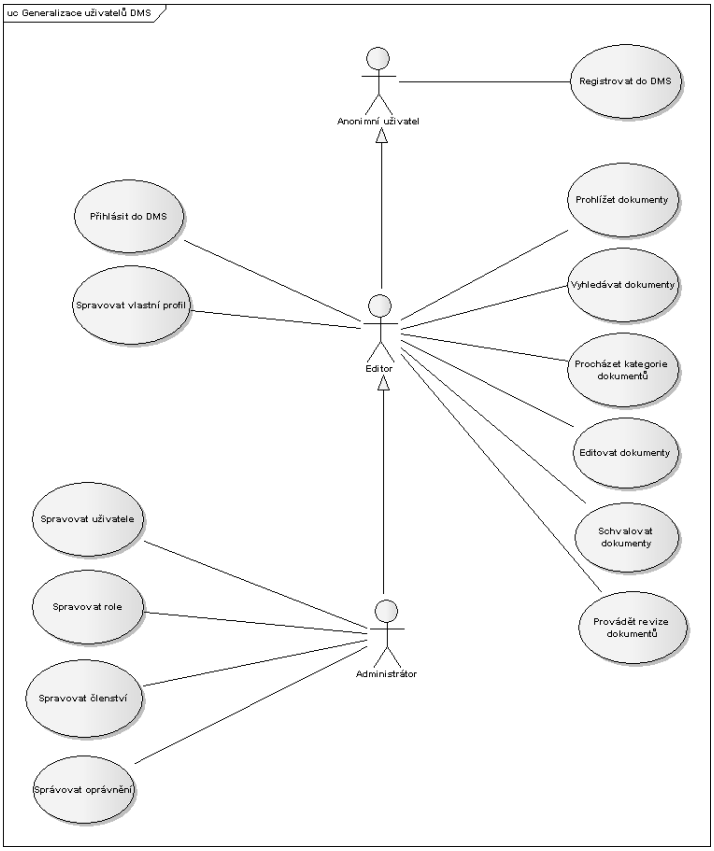
Ke každému dokumentu bude možné nastavit jeho životnost. To znamená, že dokumentu bude přidělena nějaká doba, vyjádřena v kalendářních dnech, po kterou je dokument platný. Po vypršení této doby je dokument třeba většinou aktualizovat nebo skartovat. Jako příklad využití revize je kontrola dokumentů, které se odkazují na platné zákony ČR, které se po dobu konání našeho parlamentu mohou měnit.

Kapitola 7

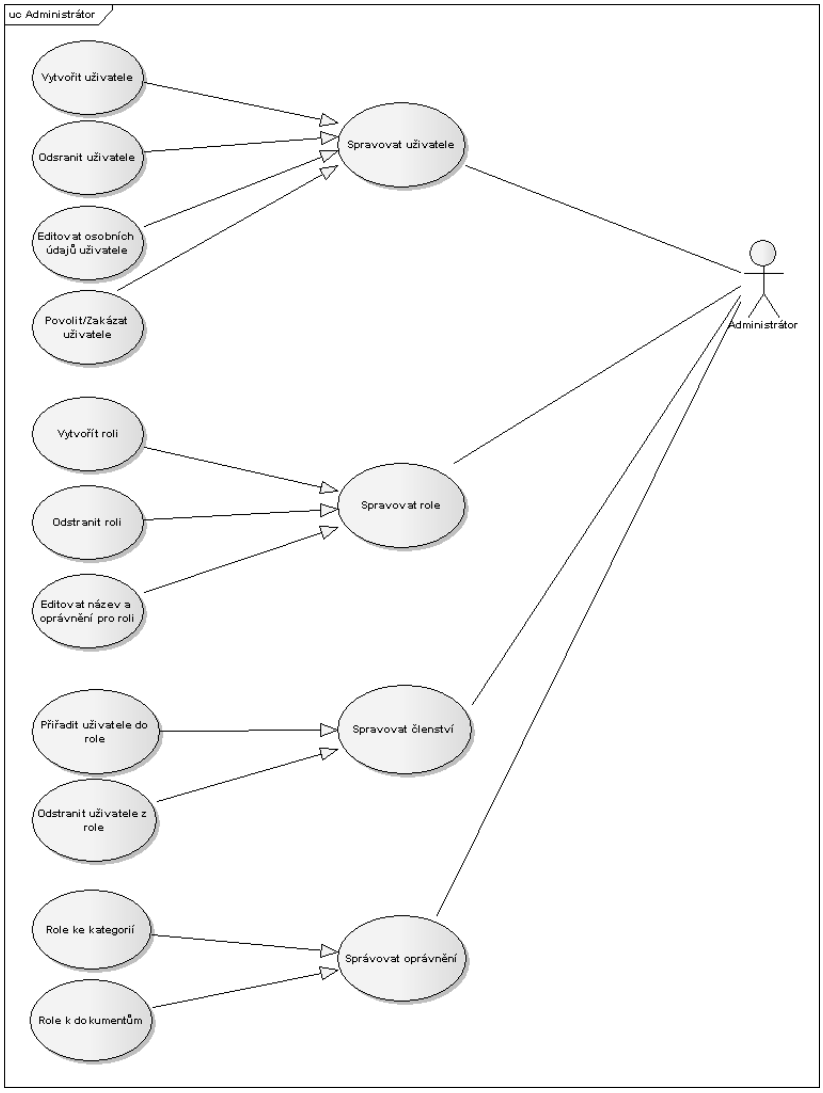
Návrh řešení

7.1 Use Case diagram

Diagram případů použití se využívá především při komunikaci se zákazníkem, následně je možné diagram využít při podrobnějším návrhu aplikace. Důležitou součástí informačního systému je bezpečnost. Důraz se klade na politiku rolí, kde daná role může v systému vykonávat pouze činnosti, které této roli umožňují předem přidělená práva.



Obrázek 7.1: Use Case diagram - Generalizace uživatelů DMS



Obrázek 7.2: Use Case diagram - administrátor

7.2 Slovní popis diagramů použití

7.2.1 Aktéři - Role v systému

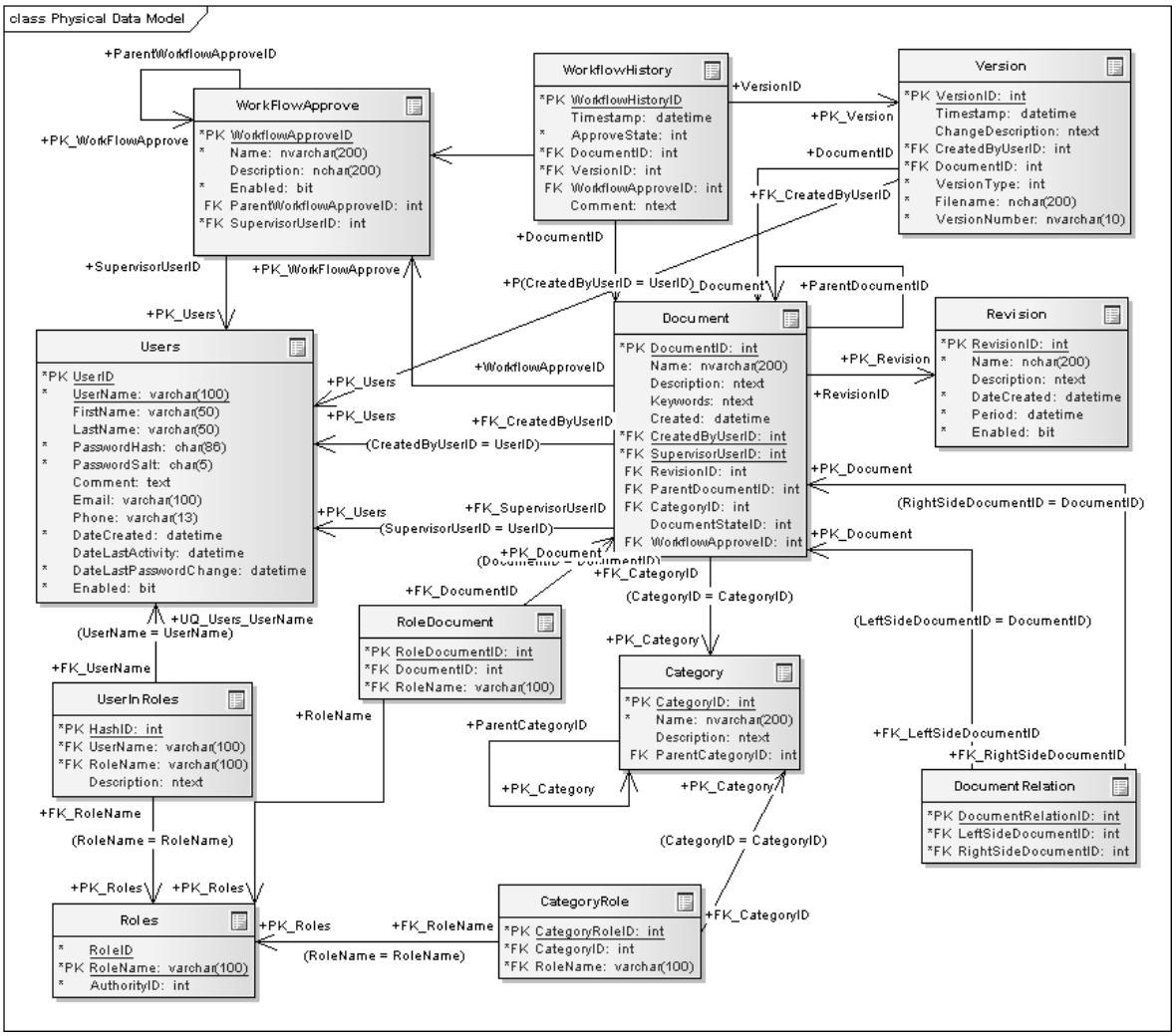
Anonymní uživatel – je uživatel, který může provést pouze registraci do systému.

Editor – uživatelé v roli editor mají přístup do správy dokumentů a pomocných nástrojů. Také mohou modifikovat své osobní údaje a přístupové heslo.

Administrátor – uživatelé v roli administrátor dědí všechny funkce od editora a přidává k nim administraci uživatelů, rolí a přístupových práv k dokumentům.

7.3 ER diagram

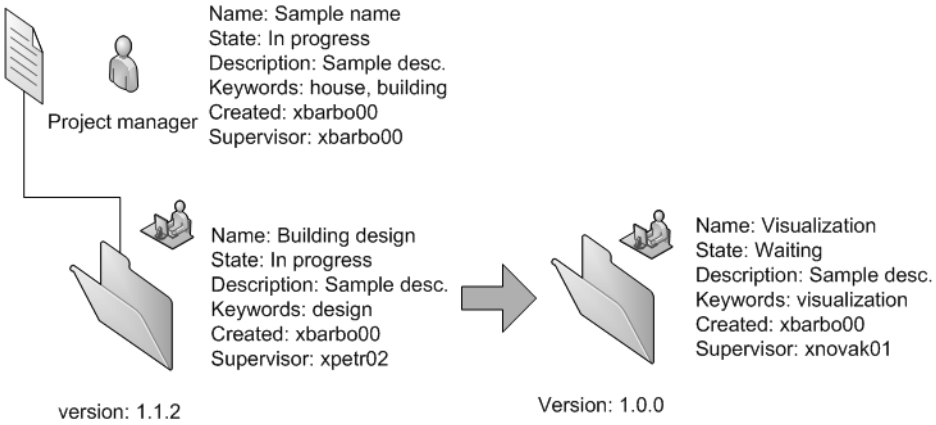
Entitně relační diagram nabízí model, který se opírá o možnosti relační databáze. Předpokladem pro nasazení IS bylo použití databáze splňující SQL standard. Databáze byla navržena tak, aby IS umožňoval provádět změny, které mohou přijít až za nějaké časové období a byl rozšiřitelný pro další funkce. Zohledněno bylo také kritérium pro jednoduchost a přehlednost. Cílem tohoto návrhu bylo především dosažení delší životnosti aplikace.



Obrázek 7.3: Entitně relační diagram

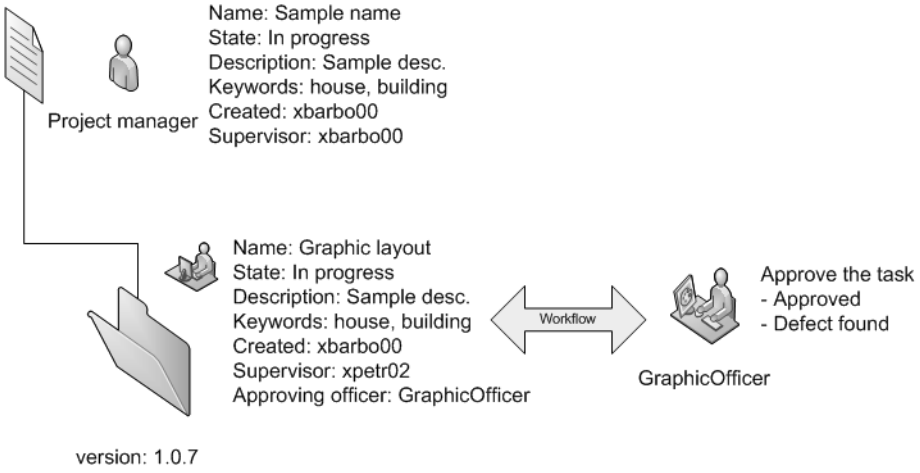
7.4 Workflow diagram

Workflow diagram nabízí zobrazení pracovního procesu/toku mezi jednotlivými aktéry procesu. V následujícím diagramu je znázorněn proces zpětné kontroly, kde nadřazený pracovník v podobě uživatele GraphicOffice kontroluje kvalitu od podřízených.



Obrázek 7.4: Workflow: závislost na cizích zdrojích při výkonu vlastního procesu

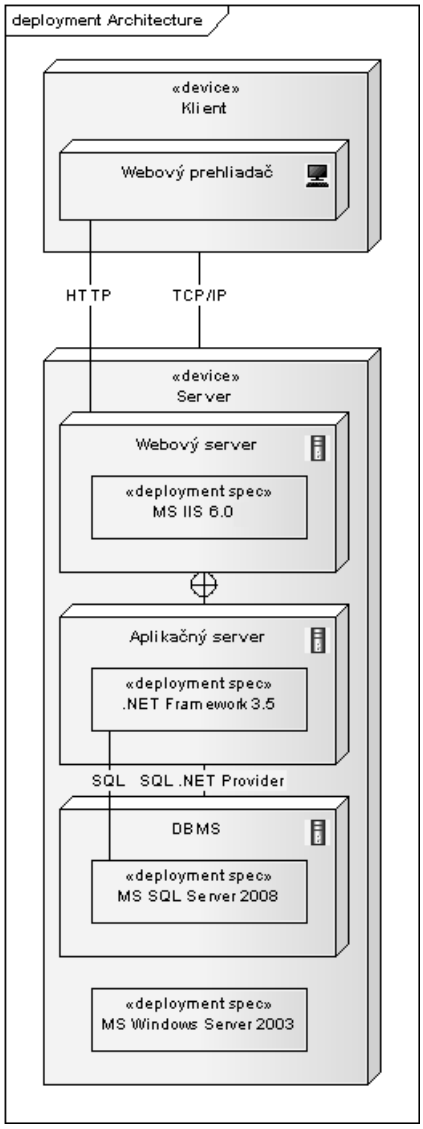
Druhý diagram znázorňuje workflow proces závislého procesu. Aktér procesu Visualization musí vyčkat se svým výkonem až na dokončení procesu Building design.



Obrázek 7.5: Workflow: kontrola kvality dokumentu

7.5 Architektura

Bylo navrženo řešení na bázi ASP.NET. Jedná se o webovou aplikaci běžící na webovém serveru Microsoft IIS a používající Microsoft SQL Server jako DBMS. Následující diagram zobrazuje typický scénář webových aplikací založených na Microsoft technologiích.



Obrázek 7.6: Architektura klient-server

Kapitola 8

Implementace

8.1 Použité technologie

8.1.1 Proč jsem si vybral ASP.NET

ASP.NET jsem si vybral především z důvodu mnoha výhod, které tato platforma pro vývoj webových aplikací nabízí. Výběr byl samozřejmě ovlivněn jistým návykem na vývojové prostředí Visual Studio, obslužný SW pro správu databází a nakonec i operační systém pro nasazení webové aplikace.

ASP.NET sice nepřináší revoluci ve vývoji webových aplikací, protože tradiční pojmy pro vytváření webových aplikací stále platí, ale v pozadí ASP.NET pracuje odlišně než skriptovací technologie, mezi které patří například velmi populární PHP, na kterém jsem jako vývojář webových aplikací začínal.

ASP.NET se v porovnání s dřívějšími platformami webového vývoje odlišuje především těchto bodech:

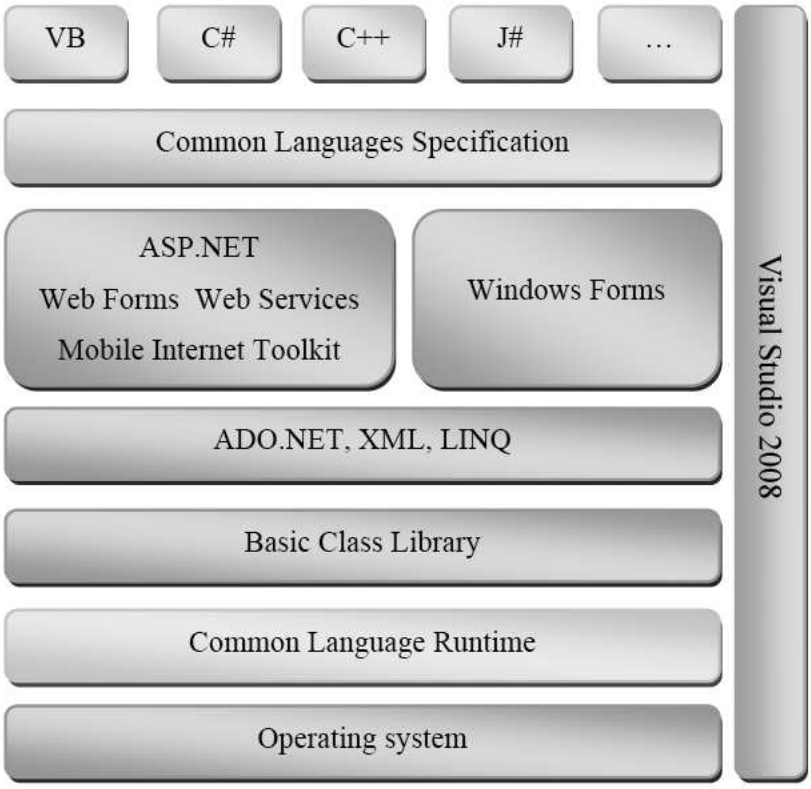
- ASP.NET nabízí úplný, objektově orientovaný programovací model, který obsahuje architekturu řízenou událostmi, založenou na ovládacích prvcích, což podporuje zapouzdření kódu a jeho opětovné využívání
- ASP.NET dává možnost psát kód v kterémkoliv z programovacích jazyků .NET
- V ASP.NET je vše podřízeno vysokému výkonu. Stránky ASP.NET a komponenty se kompilují na požádání, a tudíž se neinterpretují pokaždé, když se používají. ASP.NET také obsahuje vyladěný model pro přístup k datům a flexibilní ukládání dat do cache pro ještě větší zvyšování výkonu.

8.1.2 Platforma ASP.NET

Mluvíme-li o platformě ASP.NET, máme většinou na mysli její infrastrukturu. Infrastruktura se skládá ze čtyř systémů .NET Framework, Microsoft Visual Studio .NET, .NET Enterprise Servers a Microsoft Windows .NET. Infrastruktura platformy zahrnuje všechny technologie, které tvoří prostředí pro vytváření a spouštění aplikací. Část platformy ve které je možno tyto aplikace vytvářet se nazývá .NET Framework.

Systém .NET Framework se skládá ze dvou částí, běhového systému CLR (Common Language Runtime) a knihoven tříd známých pod zkratkou BCL (Base class library). Knihovna je přehledně členěná do mnoha jmenných prostorů a tříd a mají k ní přístup všechny

programovací jazyky. Běhový systém je možné demonstrovat jako virtuální stroj, ve kterém pracují aplikační funkce platformy .NET a je jádrem celé infrastruktury. V tomto běhovém prostředí mohou být bez problémů spouštěny aplikace psané v různých programovacích jazycích. Jde o princip, který je někdy nazýván spolupráce mezi jazyky (cross-language interoperability). Všechny jazyky, které chceme použít pro vývoj aplikací běžících na CLR, musí mít kompilátor dodržující specifikaci CLS (Common Language Specification). Jde o jakýsi souhrn pravidel, které musí překladač jazyka dodržovat, aby byl kompatibilní s platformou .NET. Mezi jazyky dodržující CLS patří například C#, C++, VB .NET nebo J#. [6]



Obrázek 8.1: Diagram .NET Frameworku 3.5

ASP.NET se neinterpretuje, ale kompiluje

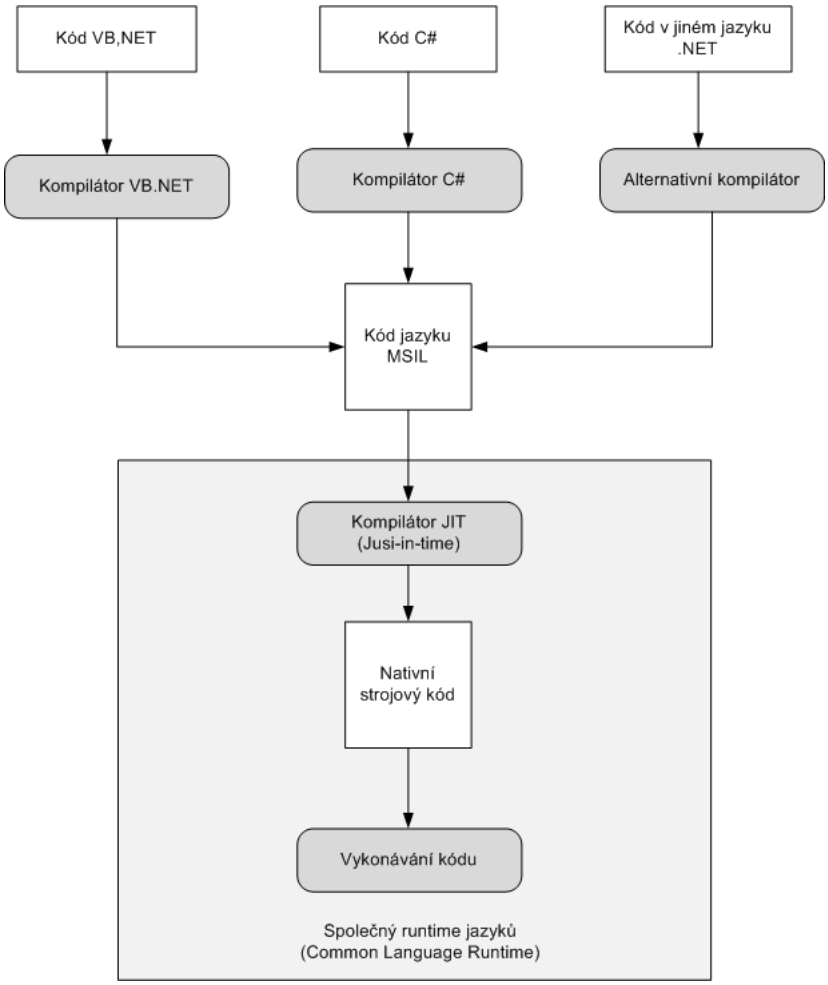
Aplikace ASP.NET se vždy kompilují a procházejí dvěma kompilačními etapami. V první etapě se kód C# zkompiluje do přechodného jazyka, který se nazývá Microsoft Intermediate Language (MSIL). Tento první krok je příčinou toho, že v .NET je možné používat různé programovací jazyky. Všechny jazyky .NET se zkompilují do virtuálně identického kódu MSIL.

První kompilační krok může nastat automaticky, když se stránka poprvé požaduje, nebo se může vykonat předem pomocí předběžné kompilace (Precompiling).

Druhá úroveň kompilace nastává těsně předtím, než se stránka skutečně vykoná. V tomto okamžiku se kód MSIL zkompiluje do nativního kódu. Tato kompilace se nazývá just-in-time (JIT) a probíhá stejně pro všechny aplikace .NET.

Kompilace .NET je rozdělena do dvou kroků proto, aby bylo možno vývojářům nabídnout co největší pohodlí a co nejlepší přenositelnost. Předtím, než může kompilátor vytvořit nízkoúrovňový strojový kód, potřebuje znát typ operačního systému a hardwarovou platformu, kde bude aplikace běžet (například 32bitový nebo 64bitový operační systém Windows). Jakmile jsou obě kompilační etapy dokončeny, je možné vytvořit zkompilovanou assembly s kódem .NET, kterou je možné distribuovat na více než jednu platformu.

Kompilace JIT by samozřejmě nebyla tak užitečná, kdyby se musela provádět pokaždé, když nějaký uživatel požádá o zobrazení nějaké webové stránky. Aplikace APS.NET se nemusejí kompilovat při každém požadavku na webovou stránku, protože kód MSIL se vytvoří pouze jednou. Generuje znovu jen tehdy, pokud dojde k modifikaci zdroje.[6]



Obrázek 8.2: Kompilace webové stránky ASP.NET

ASP.NET běží uvnitř společného runtime jazyků

Jedním z nejdůležitějších aspektů ASP.NET je to, že běží uvnitř runtimového prostředí CLR. Na .NET Framework se odkazuje jako na řízený (managed) kód. Tato architektura nabízí hned několik výhod:

Automatická správa paměti CLR alokuje pro objekt prostor na řízeném heapu (man-

aged heap) vždy, když aplikace vytvoří na něj instanci. Paměť se nemusí uvolňovat manuálně. Jakmile se reference na objekt dostane mimo obor nebo aplikace skončí, garbage collection automaticky vyžaduje navrácení nepoužívané paměti.

Strukturované zpracování chyb Jazyky .NET nabízí strukturované zpracování výjimek, které umožňuje uspořádat kód stručně a výstižně. Je možné vytvářet oddělené bloky pro zpracování různých druhů chyb.

Multithreading CLR poskytuje kolekci vláken (pool of threads), který mohou využívat různé třídy. Je možné například volat metody, číst soubory, nebo komunikovat s webovými službami asynchronně bez nutnosti explicitního vytváření nového vlákna.

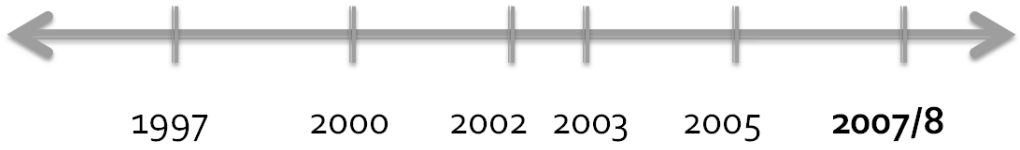
ASP.NET podporuje různá zařízení a různé prohlížeče

Jeden z významných problémů pro webové vývojáře je navrhnout a implementovat webovou aplikaci tak, aby se ve všech podporovaných prohlížečích chovala stejně. Tento problém je nejčastěji spojován s interpretací jazyka JavaScript, kde jsou nutné časté optimalizace ušité na míru pro různé prohlížeče. Někdy je třeba řešit i designovou stránku webu optimalizací CSS stylů. Tento problém je, dle mého názoru, často zveličován programátory, kteří rozumí kaskádovým stylům spíše okrajově.

ASP.NET tento problém řeší velmi inteligentně. Každý ovládací prvek ze široké palety webových ovládacích prvků realizuje svůj HMTL kód adaptivně, což znamená, že bere v úvahu schopnosti jednotlivého webového klienta. To ocení především programátor, protože pro podporu různých prohlížečů nemusí psát, někdy i značně rozsáhlé optimalizace.

8.1.3 Historie ASP.NET

ASP.NET se začal rodit v roce 1997 v rukou Scotta Guithie, který během vánočních svátků naprogramoval prototyp zvaný XPS. O tři roky později vyšla první betaverze a zrodila marketingová značka ASP.NET. V roce 2002 vychází první oficiální verze ASP.NET 1.0, následně v periodách dvou až tří let vychází nové verze. V této kapitole budou podrobněji rozebrány klíčové verze 2.0 a 3.5.[6]



Obrázek 8.3: Důležité milníky ve vývoje ASP.NET

1997 – Během vánočních svátku Scott Guithrie naprogramoval pomocí Javy prototyp pod názvem XPS.

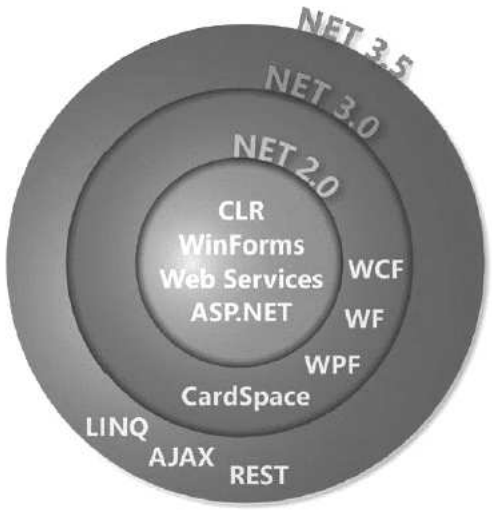
2000 – Vychází první betaverze, XPS bylo přejmenováno na ASP+ a po vytvoření marketingové značky .NET na ASP.NET.

2002 – Vychází verze ASP.NET 1.0, která přináší především objektově orientované programování (OOP) a model Web Forms.

2003 – Vychází verze ASP.NET 1.1, která spíše dolaďuje předchozí verzi. Nově vychází vývojové prostředí Visual Studio 2003.

2005 – Vychází verze ASP.NET 2.0, která přináší významné změny v .NET Frameworku. Navržené CLR 2.0 tvoří silný základ i v současné nejnovější verzi. Jako novinky pro vývoj samotný je možné uvést: Automatická kompilace, Master Pages, WebParts, podpora XHTML, Themes, Skins a navigace.

2007/8 – Vychází verze ASP.NET 3.5, která přináší C# 3.0, VB 9.0 a Visual Studio 2008. Dalšími novinkami jsou větší podpora technologie AJAX a integrace s IIS 7.0.



Obrázek 8.4: Diagram postupné evoluce ASP.NET

ASP.NET 2.0

Tato verze přinesla mnoho nového, jak již výše zmíněných změn v .NET Frameworku v podobě kvalitně navrženého CLR 2.0, tak i v novinkách pro snadnější vývoj aplikací. Nejdůležitější funkcionalita ASP.NET 2.0 je stručně popsána v následujících bodech.

Lepší vybavenost ovládacích prvků – V ASP.NET 2.0 bylo zavedeno více jak 40 nových ovládacích prvků, počínaje dlouho očekávanými základními věcmi, jako je TreeView s funkcemi pro sbalování a rozbalování, až po prvek Menu poháněný JavaScriptem.

Vzory stránek (Master pages) – Vzory stránek jsou opětovně využitelné šablony stránky. Pomocí vzorů je možné například zajistit, aby každá stránka měla stejné záhlaví, zápatí a navigační ovládací prvky.

Motivy – Motivy umožňují nadefinovat standardizovanou sadu charakteristik vzhledu webových ovládacích prvků. Jakmile jsou jednou nadefinované formátovací předvolby, je

možné je následně použít pro celý web a docílit jednotného vzhledu všech webových stránek.

Bezpečnost – Do ASP.NET 2.0 byla přidáno sousta funkcí vztahujících se k bezpečnosti. Jako příklad je možné uvést automatickou podporu sledování přihlašovacích dokladů uživatele (Credentials), autorozici založenou na rolích, a předem zabudované ovládací prvky určené pro správu bezpečnosti.

Ovládací prvky pro zdroje dat – Model ovládacích prvků pro zdroje dat umožňuje deklarativně určit, jak bude stránka komunikovat s datovým zdrojem, což znamená, že odpadá povinnost ručně psát ekvivalentní kód pro přístup k datům.

Webové části (WebParts) – Jednou z běžných webových aplikací je portál, který zobrazuje na jediné webové stránce různé informace pomocí oddělených panelů. WebParty poskytují předem zabudovaný kompletní portálový pracovní rámec.

ASP.NET 3.5

Možná by se někdo mohl divit, proč po verzi 2.0 nenásleduje verze 3.0. .NET Framework 3.0 byl skutečně vydán, ale tohle označení použil Microsoft pouze pro uvedení několika nových technologií. Například je možné uvést WPF (Windows Presentation Foundation), což je velmi zajímavá technologie uživatelského rozhraní přinášející klientům pestřejší zážitky. Další je možné uvést WCF (Windows Communication Foundation), což je technologie pro budování služeb orientované na zprávy a WF (Windows Workflow Foundation), které umožňuje modelovat složité obchodní procesy.

.NET Framework 3.0 však neobsahoval ani novou verzi CLR a hlavně ani novou verzi ASP.NET, proto po verzi ASP.NET 2.0 uvádím až verzi ASP.NET 3.5, kde nové funkce byly soustředěny především do oblastí AJAX a LINQ, které budou podrobněji rozebrány v následujících kapitolách.

8.1.4 Microsoft Visual Studio

Většina rozsáhlejších webů na platformě ASP.NET se vyvíjejí pomocí Visual Studia. Je to profesionální vývojářský nástroj, který podporuje soustavu různých designérských nástrojů. Mezi nejoblíbenější nástroje určitě patří IntelliSense. Tato technologie zachytává při psaní kódu chyby a nabízí různá doporučení pro opravu. Visual Studio také podporuje robustní model kódu v pozadí (code-behind), kde se kód .NET odděluje od webových značek. Jako poslední výhodu je možné zmínit také to, že Visual Studio disponuje vlastním webovým serverem, který programátor ocení především při častém testování.[6]

8.1.5 Microsoft SQL Server 2008

MS SQL Server je databázový stroj, umožňující ostatním programům ukládat, zpracovávat a vyhledávat větší objemy dat. Systém řízení báze dat MS SQL Server je natolik robustní a spolehlivý, že tvoří databázovou vrstvu aplikacím mnoha velkých podnicích a korporací po celém světě. Celý systém obsahuje mnoho užitečných aplikací s grafickým uživatelským rozhraním, které velmi efektivně dokážou ulehčit a zpříjemnit práci.

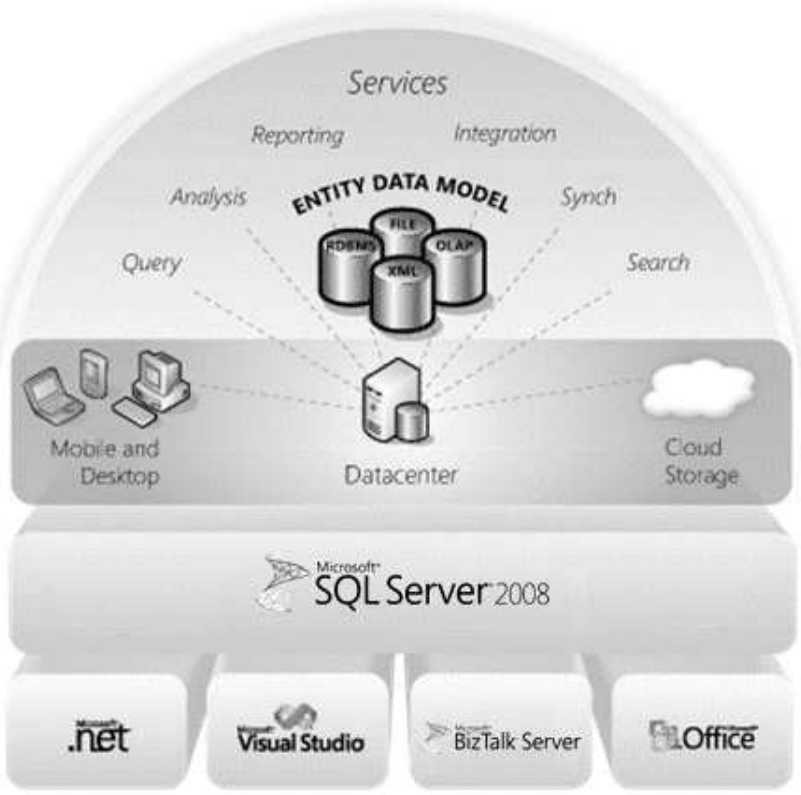
Pro tuto práci byl použit SQL server s označením 2008, který přináší od předchozí verze s označením 2005 zejména tyto novinky:

Nové datové typy – V MS SQL 2008 přibylo několik nových datových typů. Konečně je možné uložit samostatně DATE a TIME. K ukládání data a času jsou k dispozici ještě datové typy DATETIME2 (širší rozsah hodnot) a DATETIMEOFFSET (údaj s časovou zónou), které jsou pouze přesnější, než samotný DATETIME. Mezi další nové datové typy se řadí datový typ pro ukládání geometrických dat a pro ukládání geografických dat, kde především ukládání geografických dat čekám v následujících letech nárůstu na popularitě. Další novinkou je, že datový typ File Stream umožňuje uložit nestrukturovaná data přímo ze souborového systému. Tohle rozšíření dovoluje ukládat velké objemy binárních dat, na které se lze dotazovat a modifikovat jazykem SQL.

Upravené UTDs – Uživatelem definované datové typy již nejsou velikostně omezeny. Dříve byly omezeny na 8kb na stránku.

Integrované fulltextové vyhledávání – Fulltextové vyhledávání je plně integrováno do SQL Serveru, což přináší zejména zvýšení výkonu.

Ostatní změny – SQL Server 2005 SSIS engine nemohl paralelně zpracovávat úlohy na více než dvou procesorech. Verze 2008 obsahuje podporu multiprocesorového zpracování, což lze zohlednit již při návrhu SSIS balíčku. Řadu nových a vylepšených funkcí přinesla integrace Reporting Services s MS Office 2007, která například umožňuje efektivnější správu a analýzu reportů.



Obrázek 8.5: Modelové schéma SQL Server 2008

8.1.6 Programovací jazyk C#

Jak již bylo dříve uvedeno, programovou logiku v .NET platformě je možné psát v několika dostupných programovacích jazycích. Já jsem si vybral jazyk C#, a nebál bych se říct, že je to nejčastější volba většiny programátorů na platformě .NET.

Tento objektově orientovaný programovací jazyk byl vyvinut z historicky starších jazyků C/C++ a Java. Zavedením tohoto jazyka byly eliminovány některé potencionálně nebezpečné rysy jazyka C/C++, kterými jsou ukazatele či nepřímá adresace proměnných.

Jazyk C# je úzce spjat s platformou .NET, což mu umožňuje využít její bohatou knihovnu funkcí. .NET Common Language Runtime je prostředí založené na komponentách, proto se v C# programují všechny objekty jako komponenty. K těmto komponentám lze přiřazovat atributy, které mohou předávat informace ostatním součástem systému.

Další výhodou, která vyplývá z provázanosti s platformou .NET, je snadná správa paměti. Automatické přidělování paměti ulehčuje mnohdy komplikovanou práci programátora při alokaci a uvolňování paměti. Jazyk je typově bezpečný a značně tak zamezuje práci s neinicializovanými proměnnými a používání nebezpečného přetypování.[6]

8.2 Popis provedení některých významných vlastností aplikace

8.2.1 Common Table Expression (CTE)

Novinkou v SQL 2005 jsou Common Table Expressions, zkráceně CTE. Common Table Expression je výraz, který vrací dočasnou množinu výsledků zevnitř příkazu. Množina výsledků je podobná hybridu mezi Derived Table a deklarovanou Temporary Table. CTE obsahuje prvky podobné oběma zmíněným. Nejčastějším použitím CTE je vytváření tabulek za běhu uvnitř zanořeného příkazu SELECT a provádění rekurzivních dotazů. CTE mohou být použity jak pro příkaz SELECT, tak pro DML příkazy. Přestože jsou CTE v SQL Serveru novinkou, jsou již součástí ANSI SQL 99 a SQL3. Co je však nejdůležitější, CTE jsou mocným nástrojem, pomocí kterého je možné provádět rekurzivní a zanořené dotazy v syntaxi. CTE jsou z hlediska kódování a kontroly kódu ve většině případů snazší než jiné použitelné metody.

CTE v praxi - Rekurzivní dotazy

Jednou z nejsilnějších vlastností CTE je jejich schopnost odkazovat na sebe samotné pomocí jména, což umožňuje vytváření rekurzivních dotazů. Rekurzivní dotaz je takový dotaz, který volá sebe sama. Níže naleznete příklad rekurzivního dotazu. Účelem příkazu je zobrazení všech hierarchicky nadřazených dokumentů (ParentDocumentID). Tento příkaz je využíván při mazání částí nebo celého dokumentu. Aby nebyla porušena referenční integrita, je třeba mazat ve stromové terminologii, od listů postupně ke kořenu větve nebo celého stromu.

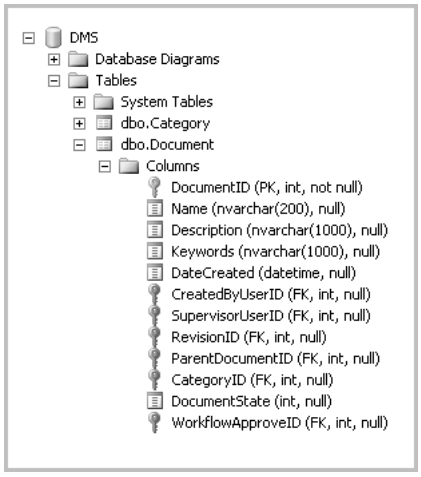
```
ALTER PROCEDURE [dbo].[DocumentTree]
  -- Specifies where to begin mining data
  @ParentDocumentID int = NULL
AS
BEGIN
  -- CTE (Common Table Expression)
  -- Creates a temporary result refferencing itself from inside
  WITH DocumentTree
  AS
  (
    -- Selects first record of desired data from Document table
    -- (Uses parameter declared earlier)
    SELECT  T1.DocumentID, T1.ParentDocumentID
    FROM  Document T1
    WHERE T1.DocumentID = @ParentDocumentID

    -- Unions this record with recursively retrieved data (other rows)
    UNION ALL

      -- Joins every other yielded record with temporary result (DocumentTree) on PrentDocumentID
      -- By default SQL Server 2008 performes 100 recursive calls, that gives us 101 nested records
    SELECT  T1.DocumentID, T1.ParentDocumentID
    FROM  Document T1 INNER JOIN DocumentTree T2
    ON    T2.DocumentID = T1.ParentDocumentID
  )

  -- Select all columns from Document table for records belonging to the tree
  SELECT  *
  FROM  Document
  WHERE DocumentID IN
  (
    SELECT DocumentID
    FROM DocumentTree
  )
  ORDER BY ParentDocumentID DESC
END
```

Obrázek 8.6: Rekurzivní dotaz pomocí CTE



Obrázek 8.7: Tabulka Document s referencí na sebe sama (sloupec ParentDocumentID)

8.2.2 LINQ

Jedním z obvyklých problémů ve vývojovém prostředí je přístup k různým datovým zdrojům, proto se hledal způsob, jak zefektivnit a sjednotit jejich zpracování.

V aplikacích postavených nad platformou .NET bylo možné pracovat z relačními či hierarchickými daty několika způsoby. V případě dat, která jsou uložena v relační databázi je možné přistupovat skrze rozhraní ADO.NET a to jak v připojeném tak odpojeném režimu. Pokud jsou data reprezentována hierarchicky v populárním formátu XML, lze využívat typů z assembly System.XML a pracovat s těmito daty buď pomocí DOM, nebo je zpracovávat sekvenčně. Nyní ovšem přišla technologie, která umožňuje zcela nový způsob práce s daty na platformě .NET.

LINQ (Language Integrated Query) je sada rozšíření pro jazyky C# a Visual Basic, která umožňuje psát kód manipulující s daty v paměti podobně, jako dotazovací jazyky nad nějakou databází. Další velmi příjemnou novinkou, která plyne z integrace dotazování přímo do .NET programovacího jazyku je odhalení chyb již v době kompilace, takže dotazy jsou kontrolovány hned a drtivá většina chyb se neprojeví až za běhu aplikace.

Pomocí tohoto nového přístupu je možné pracovat takřka s jakýmikoliv daty, protože architektura technologie LINQ je navržena tak, že je možné tvořit její implementace pro jednotlivé datové zdroje. Je to podobné jako v ADO.NET, kde je možné implementací rozhraní, vytvořit .NET provider pro specifický typ databáze, avšak v LINQ tato rozšiřitelnost nezůstává jenom u relačních databází a je mnohem abstraktnější. Této volnosti je dosaženo hlavně díky novince v .NET 3.5, kterou jsou Expression Trees, které umožňují pracovat s kódem jako z daty a tím pádem může být LINQ dotaz konkrétním providerem přeložen pro adekvátní datový zdroj a je jedno jestli se jedná o data relační, hierarchická či nějaká jiná. Tento, dle mého názoru skvělý koncept, má mimo jiné za následek, že díky LINQ se sjednocuje způsob práce s daty různých druhů.[7]

.NET framework 3.5 obsahuje několik implementací LINQ, které jsou uvedeny následujících bodech:

- LINQ to Objects – Implementace LINQ pro standardní kolekce nacházející se v paměti.
- LINQ to SQL – Implementace LINQ pro Microsoft SQL Server 2000 a vyšší.
- LINQ to XML – Implementace LINQ pro práci s XML daty.
- LINQ to DataSet – Implementace LINQ pro práci s ADO.NET datasety .

LINQ to Objects a LINQ to SQL jsou nejvýraznější a nejužívanější implementace LINQ. Také já jsem po nich sáhl při implementaci DMS, proto tyto dvě technologie budou níže podrobněji popsány.

LINQ to Objects

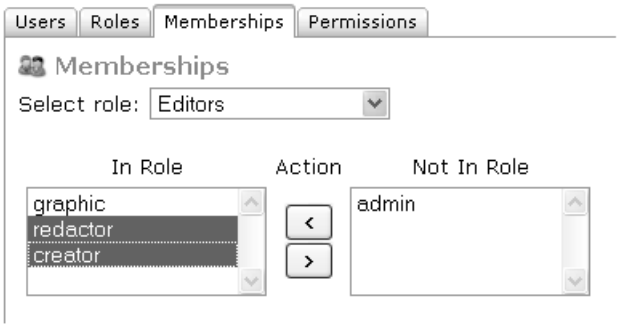
LINQ to Objects je nejjednodušší použitelnou implementací LINQ. Tato implementace je určena, jak bylo zmíněno výše, pro použití se standardními kolekcemi, které se nacházejí pouze a jenom v paměti. Takže je možné velmi jednoduše provádět dotazy nad kolekcemi stylem podobným jazyku SQL, bez nutnosti výroby vlastních algoritmů pro vyhledávání.

Kolekcí, která může být použita v LINQ, je kterákoli kolekce, která implementuje generické rozhraní IEnumerable<T>.

V nové verzi jazyka C# bylo rozhraní IEnumerable<T> rozšířeno o množinu nových metod, která slouží k provádění dotazů nad daty. Tato množina metod je označována také jako standard query operators a obsahuje metody jako například Where pro definici omezení prvků ve výsledkové sadě, metodu Select pro implementaci projekce, či také OrderBy metodu pro seřazení výsledku a samozřejmě mnoho dalších. LINQ extension metody pro typ IEnumerable<T> jsou implementovány ve třídě Enumerable nacházející se uvnitř jmenného prostoru System.Linq.

LINQ to Objects v praxi

Zde nabízím ukázku využití LINQ to Objects ve vlastní implementaci DMS. Jedná se o kód ze souboru administration.aspx.cs, který přidává uživatele do jednotlivých rolí pomocí dvou listboxů a přesouvacích tlačítek.



Obrázek 8.8: Membership dialog pro řazení uživatelů do rolí

V listboxech je možné vybrat více uživatelů najednou. Pomocí LINQ to Objects je možné jednoduchým dotazem získat pole označených uživatelů.

```
IEnumerable<ListItem> items = usersNotInRole.Items.Cast<ListItem>().Where(item => item.Selected);
string[] userNames = items.Select(item => item.Value).ToArray();
SiteHelper.RoleProvider.AddUsersToRoles(userNames, new[] { drpRoles.SelectedValue });
```

Běžným způsobem bychom museli procházet celé pole v listboxu a každého uživatele zkontrolovat, zda je označen nebo ne.

```
foreach(ListItem item in usersNotInRole.Items)
{
...kontrola uživatele na příznak označení a následné zařazení/nezařazení do pole
}
```

LINQ to SQL

LINQ to SQL je aplikační programové rozhraní (API) pro práci s SQL databázemi, které řeší komunikaci mezi relačními databázemi. Při psaní aplikací modelujeme části reálného světa reprezentované třídami. Potřebujeme zajistit perzistenci těchto objektů, aby při restartu aplikace byly zachovány všechny její objekty. Většina databázových systémů je relační, což znamená, že ukládáme záznamy tabulek, nikoliv objektů jak bychom potřebovali.

Další problém může nastat v tom, že databázový server používá jiné datové typy, než programovací jazyk, ve kterém je aplikace napsána. Programátor tak musí hlídat datové

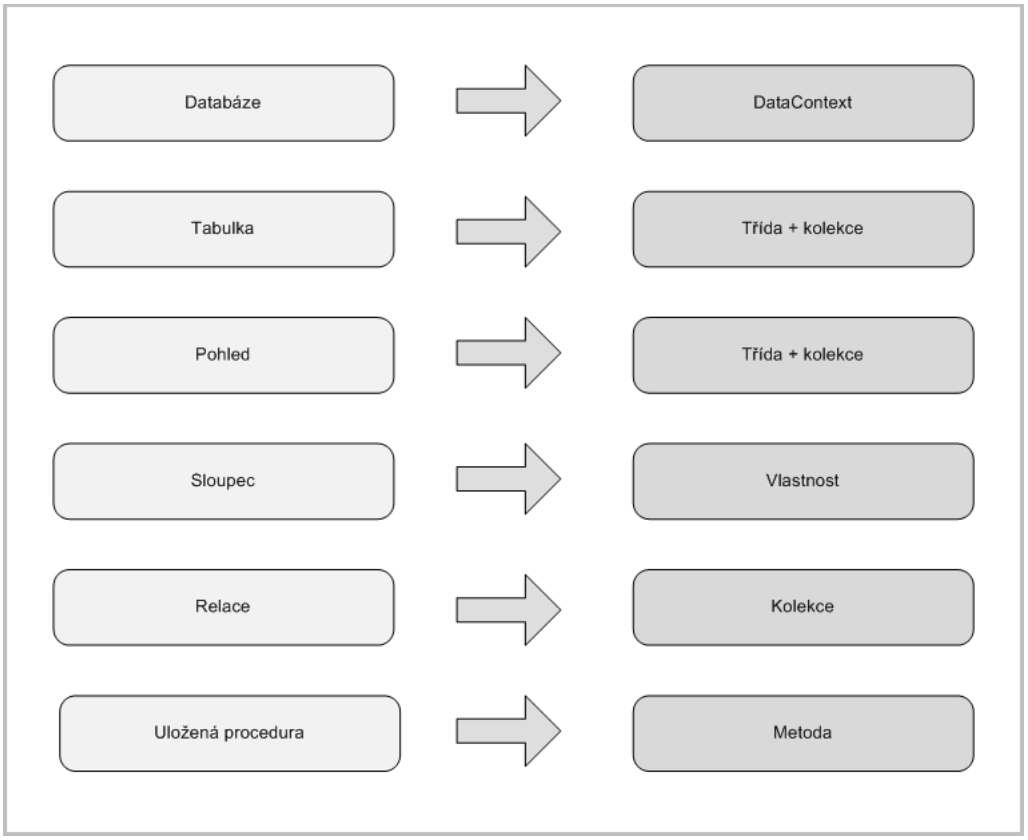
typy, které používá databáze a které může použít v programovacím jazyce. Při nesprávně zvoleném typu může nastat záměna hodnoty, nebo v horším případě pád aplikace.

Jako problém může být považováno, že LINQ to SQL je možné používat jen databází SQL Server 2000 a novější. Pro práci s jinými databázemi lze použít pouze SQL to Dataset. [6]

Mapování a překlad LINQ to SQL

Při použití LINQ to SQL se typicky používá třída odvozená od třídy DataContext. Jméno třídy je obvykle totožné se jménem databáze. Odvozená třída od DataContext obsahuje veřejné proměnné typu Table<T> pro každou databázovou tabulku, kde T je typ entitní třídy, která je instancí pro každý získaný záznam z příslušné databázové tabulky.

LINQ to SQL používá entitní třídy, kde je obvykle každá entitní třída vázána na jednoduchou databázovou tabulku a vlastnosti entitní třídy jsou vázány ke sloupcům této tabulky. Uložené procedury a uživatelsky definované funkce jsou v LINQ to SQL reprezentovány jako metody.



Obrázek 8.9: Diagram mapování LINQ to SQL

LINQ to SQL dotazy vrací sekvence typu IQueryable<T> , které nejsou kompilované do mezikódu, jako obyčejné LINQ dotazy. Místo toho jsou převedeny do výrazových stromů, které jim umožní vyhodnocení přeložení do příslušných optimálních SQL dotazů. Na rozdíl od LINQ dotazů, které jsou vykonány v lokální paměti, LINQ to SQL jsou přeloženy na SQL volání a vykonány v databázi.

Obecná forma jednoduchého LINQ dotazu pomocí query keywords v C# 3.0:

```
from [typ] proměnná in datový_zdroj
[where] podmínka_restrikce
[orderby] klíč_řazení [descending]
select výraz_projekce
```

Klíčové slovo	**Metoda na IEnumerable<T>**	**Popis**
from	žádná	První klíčové slovo v dotazu. Slouží ke specifikaci datového zdroje, nad kterým je prováděn dotaz.
where	Where	Slouží k definici podmínky restrikce pro výsledek. Pokud podmínka vrátí true, prvek je zahrnut do výsledku.
select	Select	Používá se k implementaci projekce, kde jsou pouze některé datové složky objektu použity ve výsledku. Uvádí se vždy na konci dotazu.
group	GroupBy	Slouží k seskupování prvků ve výsledku podle určitého klíče. K použití toho slova se váže použití nového klíčového slova by. Může jím být zakončen dotaz.
into	žádná	Používá se v kombinaci se slovem group, join nebo select k uložení jeho výsledku a další možné práci s tímto výsledkem.
orderby	OrderBy, OrderByDescending	Slouží k řazení prvků ve výsledku podle definovaných kritérií. Pro sestupné řazení je možné použít s novým klíčovým slovem descending.
join	Join, GroupJoin	Používá se k propojení prvků z různých datových zdrojů na základě definované podmínky ekvivalence. V kombinaci s tímto slovem se používají nová klíčová slova equals a on.
let	žádná	Slouží k definici lokální proměnné v rámci dotazu, do které může být přiřazena jak sekvence elementů, tak jednoduchá hodnota.

Tabulka 8.1: Seznam klíčových slov pro LINQ[7]

LINQ to SQL v praxi

Opět nabízím ukázky implementace z DMS, která demonstruje operace INSERT, UPDATE a DELETE nad databází.

INSERT

```
//vytvořím nový objekt
        Category newCategory = new Category
              {
                  Name = txtCategoryName.Text,
                  Description = txtCategoryDescription.Text,
                  ParentCategoryID = selectedCategory.CategoryID
              };
//oznámím kontextu, že až se bude jednou ukládat, nemá zapomenout na náš nový objekt
        DataProvider.DMSContext.Categories.InsertOnSubmit(newCategory);

//ukládání provádím ihned, protože nikde nedochází ke změnám velkých dat,
//navíc data budou stále aktuální a nemůže dojít ke kolizím
        DataProvider.DMSContext.SubmitChanges();
```

UPDATE

```
//vyberu lambda výrazem konkrétní vybranou kategorii
//lambda výrazy je možné používat místo klíčových slov např. Where atd.
 Category newCategory =  DataProvider.DMSContext.Categories.SingleOrDefault(
            DataProvider.DMSContext.Categories.Single(
                theItem => theItem.CategoryID == selectedCategory.CategoryID));
// změním nějakou property a dám opět uložit změny
newCategory.Name = "Changed category name";
DataProvider.DMSContext.SubmitChanges();
```

DELETE

```
//vyberu lambda výrazem jeden záznam, který chci smazat dle jeho ID)
 DataProvider.DMSContext.Categories.DeleteOnSubmit(
            DataProvider.DMSContext.Categories.Single(
                theItem => theItem.CategoryID == selectedCategory.CategoryID));
 DataProvider.DMSContext.SubmitChanges();
```

8.2.3 AJAX

AJAX (Asynchronous JavaScript and XML) je programová technika aplikovaná u klienta, která umožňuje, aby stránka zavolala server a aktualizovala svůj obsah, aniž by spustila kompletní odesílání zpět na server. Vlastní implementované DMS využívá jeden společný UpdatePanel, který odchytává všechny PostBacky prohlížeče a pomocí XMLHttpRequest probíhá asynchronní výměna dat se serverem. AJAX nepředstavuje konkrétní novou technologii, ale označuje nasazení několika technologií za určitým cílem.

Použití technologie Ajax je možné dobře demonstrovat na komponentě pro vyhledávání dokumentů, kde při každém stisknutí klávesy se vyvolá PostBack pomocí skrytého tlačítka. Tento způsob řešení, je běžný v ASP.NET aplikacích, protože PostBack musí vždy vyvolat nějaký ovládací prvek.

```
// Register onkeyup event (initiates postback to search button)
txtSearch.Attributes.Add("onkeyup", Page.ClientScript.GetPostBackEventReference(btnSearch, null));
```

8.3 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní bylo navrženo podle posledních standardů a trendů. V horní části aplikace je standardní hlavička s logem, která je navíc opatřena informačním boxem s daty o přihlášeném uživateli. Navigace mezi sekcemi je vytvořena pomocí záložek, se kterými poprvé přišel Microsoft a do současné doby je hojně používá nejen ve svých operačních systémech.

Názvy sekcí a většina tlačítek byla opatřena ikonkami za účelem lepší orientace v aplikaci. Jako zajímavý grafický prvek byla použita komponenta GoogleCharts, která nám na základě statistických číselných údajů generuje koláčové grafy.

8.4 Popis jednotlivých sekcí aplikace

V této kapitole bude stručně popsán význam a funkcionalita jednotlivých sekcí a podsekcí celé aplikace.

8.4.1 Sekce Administration

Tato sekce je přístupná pouze uživatelům v roli Global Administrator. Důvodem je především to, že v této sekci se spravují kompletní bezpečnostní práva aplikace. Proto by nebylo žádoucí, aby si například oprávnění k dokumentům mohl každý uživatel měnit sám.

Hlavní sekce Administration obsahuje tyto záložky:

Users – obsahuje seznam všech uživatelů IS. Administrátor má právo přidávat nové uživatele, modifikovat nebo odstranit existující.

Roles – obsahuje seznam všech rolí IS. Administrátor má právo přidávat nové role, modifikovat nebo odstranit existující.

Memberships – je záložka, kde je možné přiřadit uživatele do jednotlivých rolí. Administrátor ma možnot vybrat roli a následně do ní přiřadit nebo z ní vyřadit vybrané uživatele.

Permissions – je záložka, kde je možné nastavit rolím práva pro modifikace kategorií a dokumentů. Administrátor má možnost vybrat roli a následně ji navolit, do kterých kategorií nebo dokumentů má přístup.

8.4.2 Sekce Tools

Sekce Tools slouží rolím Editors a Global Administrators pro správu podpůrných funkcí pro tvorbu dokumentů.

Hlavní sekce Tools obsahuje tyto záložky:

Workflows – obsahuje seznam všech workflow procesů, které je možné na dokumenty uplatnit. Uživatel má právo nová workflow vytvářet, modifikovat nebo odstraňovat již existující. Krom těchto klasických funkcí má možnost také nastavovat pořadí jednotlivých stepů.

Revisions – obsahuje seznam všech revizí, které je možné na dokumenty uplatnit. Kromě standardních dat, jako je název a popis revize, udává uživatel číselný údaj, který se rovná počtu dnů životnosti dané revize.

Sectors – obsahuje všechny sektory, do kterých je možné zařadit všechny uživatele IS. Slouží pouze jako doplňková funkce pro organizaci uživatelů.

Watchdog – v překladu hlídací pes, kontroluje všechny dokumenty, na které je uplatněná revize. Watchdog varuje již 30 dnů před vypršením životnosti dokumentu.

Statistics – zobrazuje statistické údaje k dokumentům IS.

8.4.3 My desk

Sekce obsahuje všechny důležité informace, které se vážou pouze k přihlášenému uživateli. Hlavní sekce My desk obsahuje tyto záložky:

My details – zobrazuje data o přihlášeném uživateli, které si každý může libovolně modifikovat.

Chnage password – je záložka pro změnu hesla do IS.

Waiting for my approval – zobrazuje všechny dokumenty, kde se očekává schválení.

My documents – zobrazuje všechny dokumenty, které přihlášený uživatel v IS založil.

Supervised documents – zobrazuje všechny dokumenty, za které jsem zodpovědný po stránce obsahové.

8.4.4 Categories

Sekce zobrazující stromové úložiště dokumentů, kde kategorie je možné zanořovat nebo přesouvat do jiných větví. Do každé kategorie je možné vložit libovolný počet dokumentů. Hlavní sekce Categories obsahuje tyto záložky:

Info – zobrazuje základní metadata vybrané kategorie.

List documents – zobrazuje seznam všech dokumentů, které patří do této vybrané kategorie.

8.4.5 Documents

Tato sekce zobrazuje všechny dokumenty IS. Uživatel má možnost vyhledávat dokumenty na základě názvu, popisu a klíčových slov dokumentu.

8.4.6 Detail dokumentu

Detail dokumentu, který se otevírá pomocí komponenty GrayBox, obsahuje stromovou strukturu dokumentu a také několik záložek pro správu vybraného dokumentu.

General – zobrazuje metadata a tři selectory pro nastavení supervizora, revize a stavu vybraného dokumentu.

Version history – zobrazuje seznam verzí dokumentu. Po kliknutí na verzi je možné danou verzi otevřít nebo uložit na lokální disk.

Related documents – zobrazuje všechny poddokumenty kořenového dokumentu pod který vybraný dokument náleží. Pomocí checkboxu má uživatel možnost zvolit, na kterých dokumentech je vybraný závislý. To znamená, že pro vykonání svého procesu potřebuje mít uzavřené tyto závislé dokumenty.

Workflow – uživatel má možnost k dokumentu vybrat předem vytvořené workflow. V této záložce následně probíhá zasílání výzev pro kontrolu dokumentu.

Upload new version – je záložka pro ukládání nových verzí dokumentu. Uživatel vybere soubor ze svého lokálního disku vybere, o jakou změnu se jedná a odešle soubor s metadaty na server.

Kapitola 9

Praktické použití aplikace

Vlastní aplikace pro řízení a správu dokumentů byla navržena a implementována tak, aby byla příjemným a usnadňujícím nástrojem při každodenní práci s elektronickými materiály. Rozsah funkcí současného řešení je velmi vhodné pro menší a střední firmy, které disponují 50 až 150 zaměstnanci. Z vlastní praxe mám ověřeno, že nasazení informačních systému do reálného provozu obnáší často odhalení řady skrytých chyb a nedostatků, které je nutné během nasazení a zahřívacího provozu odhalit a vyřešit.

Současný stav aplikace se ani zdaleka neblíží korporátním řešením, které nabízí společnosti jako EMC nebo IBM. Vstoupit na trh, kde kralují tito velikáni, je pro malou a střední firmu téměř nemožné. Důvodem je především to, že za největšími společnostmi stoji dobrá a dlouhá historie, finanční stabilita, nabyté zkušenosti a hlavně obrovské zázemí lidských zdrojů pro vývoj a prodej.

Na druhou stranu se odvažuji prohlásit, že tato aplikace po několika vylepšeních, by si mohla najít místo na trhu v sektoru malých a středně velkých společností.

Kapitola 10

Závěr

Cílem této diplomové práce bylo seznámení se s potřebami pro snadnou správu dokumentů, kterými by měly aplikace s označením DMS disponovat. Pro objasnění těchto funkcí bylo nutné nastudovat základní principy workflow a životního cyklu ve větší organizaci.

Jedním z bodů této diplomové práce bylo také popsat stávající systémy, které se dobře, v oblasti správy dokumentů, uplatnily. Tento bod byl proveden především získáváním informací přímo z webů výrobce, protože dema nebo trial verze softwaru této kategorie nejsou běžně dostupné. Dalším krokem byl sběr informací o životním cyklu dokumentu, kde se jednalo především o ujasnění funkcionalit pro návrh systému tohoto zaměření.

Po získání a analýze požadavků bylo navrženo vlastní řešení, které se domnívám, že plně vyhovělo zadání a dalo základ pro implementaci úspěšné aplikace.

Poslední fází bylo implementovat webovou aplikace dle vytvořeného návrhu, kde za použití dobře zvolených a osvědčených technologií, bylo docíleno vytvoření dobře ovladatelné a funkčně nabyté aplikace.

Literatura

[1] Terminology and Glossary, Workflow Management Coalition, 1996

[2] Carda A., Kunstová R.: Workflow, Řízení firemních procesů, Grada Publishing 2001, ISBN 80-247-0200-2

[3] Albrecht D., Workflow - základní terminologie, 1999, Dokument firmy Sefira spol s r.o,

[4] Krupička K., Proč nasadit systém pro správu dokumentů, 2003, Dokument dostupný na: http://www.systemonline.cz/clanky/proc-nasadit-system-pro-spravu-dokumentu-a-obsahu.htm

[5] Krčál M., Document management systems. Inflow: information journal [online], 2008, Dokument dostupný na: http://www.inflow.cz/document-management-systems, ISSN 1802-9736

Matthew Macdonald

[6] MacDonald M., Szpuszta R.: ASP.NET 3.5 a C# 2008 tvorba dynamických stránek profesionálně, Zoner Press 2008, ISBN 978-80-7413-008-3

[7] Petr Puš, Úvod do LINQ, Vyvojar.cz: Server se zaměřením na Windows programátory [online], 2008, Článek dostupný na: http://www.vyvojar.cz/Articles/563-uvod-do-linq.aspx

[8] Přehled systémů pro elektronické zpracování dokumentů na českém trhu, SystemOnLine: zpravodajský portál časopisu IT Systems [online], 2003, Článek dostupný na: http://www.systemonline.cz/clanky/prehled-systemu-pro-elektronicke-zpracovani-dokumentu-na-ceskem-trhu.htm

[9] Flessig S., Dokumenty ve firmě - jak na ně?, Databázový svět: informační portál ze světa databázových technologií [online], 2006, Článek dostupný na: http://www.dbsvet.cz/rservice.php?akce=tisk&cisloclanku=2006071101