VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

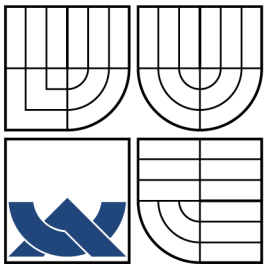
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉM PRO PODPORU PRÁCE PŘEKLADATELŮ PROGRAMOVÁNÍ V MS WINDOWS - VISUAL STUDIO NEBO DELPHI

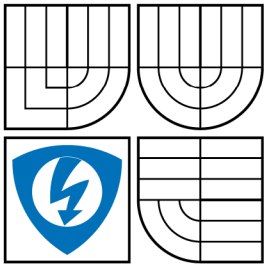
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE LUKÁŠ KOZEMPEL
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉM PRO PODPORU PRÁCE PŘEKLADATELŮ PROGRAMOVÁNÍ V MS WINDOWS - VISUAL STUDIO NEBO DELPHI

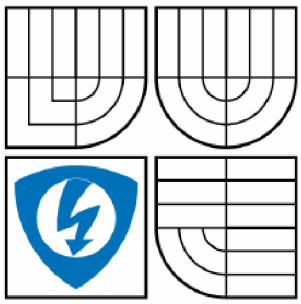
SYSTEM FOR SUPPORT FOR WORK OF PROGRAMMING TRANSLATORS IN MS WINDOWS - VISUAL STUDIO OR DELPHI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ KOZEMPEL
AUTHOR

VEDOUCÍ PRÁCE — ING. LUBOMÍR CVRK, PH.D.
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Kozempel Lukáš	***ID:***	74840
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Systém pro podporu práce překladatelů programování v MS Windows - Visual Studio nebo Delphi

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte program, který bude podporovat práci překladatele, který překládá texty v programu MS Word. Program musí umět otevřít dokument ve Wordu prostřednictvím OLE rozhraní. V otevřeném dokumentu nechť je zvýrazněn (např. podtržen) aktuálně překládaný řádek. Musí být možné posunout toto zvýraznění o řádek níže či výše za pomocí klávesové zkratky, která musí fungovat i když je aktivní jiná aplikace. Podpořte uživatele i rychlým posunem o stránku níže či výše, včetně přesunu zvýrazněného řádku do oblasti viditelné na monitoru. Program nechť umožňuje uložit různá nastavení (vazbu mezi otevřeným dokumentem a zvýrazněným řádkem) pro libovolné dokumenty. Formátem dat pro ukládání nechť je XML.

DOPORUČENÁ LITERATURA:

[1] Miller, T., Powell, D. Mistrovství v DELPHI 3, Computer Press, 1998.
[2] Microsoft Development Network, MSDN, msdn.microsoft.com
[3] Sharp, J., Jagger, M. Visual C#.NET, iDnes Knihy, 2005

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Lubomír Cvrk, Ph.D.

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA

POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Lukáš Kozempel
Bytem: Lhotka 186, 56002, Česká Třebová
Narozen/a (datum a místo): 5.1.1986, Svitavy

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Systém pro podporu práce překladatelů programování v MS Windows - Visual Studio nebo Delphi
Vedoucí/školitel VŠKP: Ing. Lubomír Cvrk, Ph.D.
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

Tématem bakalářské práce je praktická realizace systému, který ovládá program MS Word a v jeho otevřeném dokumentu zajišťuje podtržení řádku, jenž překladatel právě překládá. V první části jsou rozebrány technologie použité pro vytvoření tohoto systému. Ve druhé části je pak popsána činnost a funkce vytvořeného programu.

KLÍČOVÁ SLOVA

podtržení řádku, OLE, COM, DOM, API SAX, XML, zprávy systému, háky, MS Word

ABSTRACT

Subject of my bachelor's thesis is practical realization of the system, which controls program MS Word. It ensures underline that line in document, which is currently translating by translator. In the first part technologies used for create this system are analyzed. In the second part I have described function of created program.

KEYWORDS

underline of line, OLE, COM, DOM, API SAX, XML, system messages, hooks, MS Word

KOZEMPEL L. *Podpora práce překladatelů programování v MS Windows - Visual Studio nebo Delphi*. Místo: VUT v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. Počet stran 41 s., 2 s. příloh. Vedoucí práce byl Ing. Lubomír Cvrk Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Systém pro podporu práce překladatelů programování v MS Windows - Visual Studio nebo Delphi“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tématem mé bakalářské práce je praktická realizace systému pro podporu práce překladatelů. Tento systém zajišťuje podtrhávání jednotlivých řádků v překládaném originálu, aby se překladatel v překladu neztratil a mohl rychle nalézt, který řádek zrovna překládá.

Systém obsahuje uživatelské rozhraní, ve kterém je možno zvolit originální dokument a dokument, do kterého se bude zapisovat překlad (oba ve formátu .doc). Dokumenty se otevírají pomocí technologie OLE (Object linking and embedding) a OLE Automation. Tato technologie dovoluje systému využívat služeb aplikace MS Word a programově ji tak ovládat. Pomocí čtyř tlačítek je možné provádět skok na následující nebo předchozí řádek nebo stránku (provede se podtržení příslušného řádku). To je možné také pomocí klávesových zkratek. Zkratky jsou realizovány pomocí háků (hooks). Háky slouží pro zachytávání a přeposílání zpráv systému Windows. Můj program potom testuje zachycené kódy stisknutých kláves a pokud jsou stisknuty správné kombinace klávesových zkratek, provede příslušné akce. Pomocí háků také program sleduje, které okno Wordu je aktivní a zda uživatel některé okno nezavřel. Informace o podtrženém řádku, poloze a velikosti oken se ukládají do dokumentu XML. Překladatel tak může při příštím spuštění pokračovat od místa, kde naposledy skončil. Pro práci s XML jsou využity technologie API SAX a model DOM (Document object model). Pro vytvoření programu jsem zvolil program Borland Delphi 7, protože s ním mám již nějaké zkušenosti.

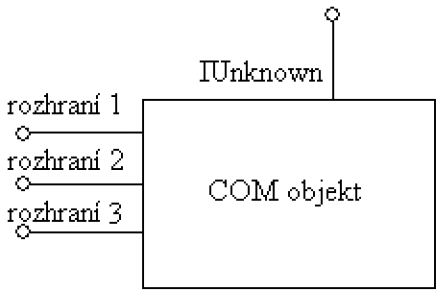
V následující kapitole jsem rozebral technologie potřebné pro realizaci programu: OLE, COM, Háky a práci s XML. Po ní následuje kapitola, která obsahuje řešení programu. V příloze je UML diagram tříd vytvořeného programu a obsah přiloženého CD-ROM.

1 TEORETICKÝ ROZBOR TECHNOLOGIÍ

1.1 Component Object Model (COM)

1.1.1 Úvod

Technologie OLE umožňuje ostatním aplikacím přistupovat k objektům, manipulovat s nimi (nastavováním vlastností a voláním metod OLE objektu) a sdílet objekty s jinými aplikacemi. Pro nízkoúrovňový přístup k objektům využívá model COM (Component object model), který umožňuje komunikaci mezi objekty přes standardní binární rozhraní. Každý objekt COM obsahuje jedno nebo i několik rozhraní (obr. 1.1), které umožňují ostatním aplikacím a komponentám využívat jeho metody. Rozhraním se rozumí paměťová struktura, ve které je pole ukazatelů na metody uvnitř komponenty. Tato paměťová struktura se nazývá VTBL (Virtual Function Table). Když chce klient zavolat některou metodu rozhraní, musí požadovanou metodu zavolat přes ukazatele ve VTBL. Ukazatel předá volání metodě uvnitř komponenty a ta provede požadovanou operaci. Každý COM objekt musí obsahovat rozhraní IUnknown. Toto rozhraní řídí všechna ostatní rozhraní a jeho vlastnosti jsou ostatními rozhraními děděny.[8]



Obr. 1.1: Rozhraní objektu COM

1.1.2 Rozhraní IUnknown

IUnknown musí podle [4] vždy obsahovat tyto metody:

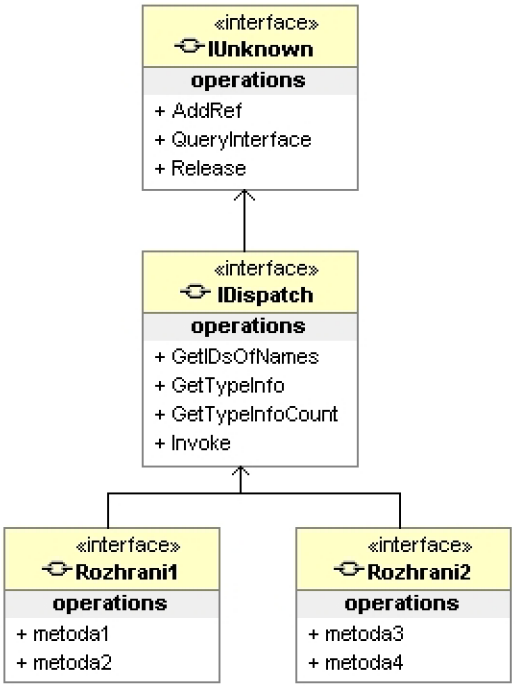
- QueryInterface - slouží k zjištění zda komponenta poskytuje požadované rozhraní, pokud ano vrátí ukazatel na toto rozhraní
- AddRef - volá se při vytvoření nového odkazu na komponentu, slouží k počítání referencí na komponentu - v těle metody musíme tento čítač zvýšit

- Release - volá se při zrušení odkazu na komponentu, slouží k počítání referencí na komponentu - v těle metody musíme tento čítač snížit, tuto metodu by měl provádět klient

Postup komunikace klienta s komponentou COM: Klient vytvoří komponentu a dostane ukazatele na IUnknown. Chce použít určité rozhraní, proto zavolá metodu: ukazatel_na_IUnknown.QueryInterface(požadované_rozhraní). Pokud komponenta toto rozhraní poskytuje, vrátí klientovi ukazatele na toto rozhraní. Nyní může klient volat metody rozhraní příkazem ukazatel_na_rozhraní.metoda(). Pokud už klient toto rozhraní nechce dále využívat, zavolá metodu ukazatel_na_rozhraní.Release().

1.1.3 Rozhraní IDispatch

Od rozhraní IUnknown je odvozeno rozhraní IDispatch, které je používáno skriptovacími jazyky.



Obr. 1.2: Zobrazení vícenásobné dědičnosti u rozhraní IDispatch

Podle [7] obsahuje minimálně tyto metody:

- GetIDsOfNames - přijímá ID objektu ke zpracování
- Invoke - zprostředkovává přístup k vlastnostem a metodám, nalezeným klientem
- GetTypeInfoCount - zjišťuje číslo typového rozhraní

- GetTypeInfo - získá pro objekt typové informace

V tomto případě je postup následující: skriptovací interpret dostane ukazatele na základní rozhraní IDispatch. Metodou IDispatch.GetIDsOfName interpret zjistí dispid metody, kterou požaduje. Po obdržení dispid volá metodu Idispatch.Invoke() s parametrem dispid_metody a polem, které obsahuje parametry volané metody. [6]

Pokud k dané komponentě skriptovací jazyky nepřistupují, IDispatch se neimplementuje. Pokud ale má být COM objekt přístupný přes OLE Automation (programové řízení - viz níže), musí implementovat obě rozhraní (tzv. duální rozhraní). Potom první tři vstupy patří IUnknown, další čtyři vstupy IDispatch a zbytek vstupů patří duálnímu rozhraní. Jednotlivé objekty však ještě implementují rozsáhlejší rozhraní tzv. dispatch interface (dispinterface), v němž je každé vlastnosti a metodě přidělen jednoznačný 32 bitový identifikátor DISPID. Každému objektu, rozhraní nebo knihovně v modelu COM je přiřazen 128 bitový identifikátor GUID (Globally Unique Identifier), který je pro něj na všech počítačích na světě unikátní.[8]

1.1.4 Typy COM serverů

V technologii COM se rozeznávají dva typy serverů. In-process a Out-of-process servery. In-process servery běží ve stejném adresovém prostoru jako klientská aplikace a je zkompilován do dll knihovny. Na požádání klienta vytvoří objekt přímo v jeho adresovém prostoru. Klient již potom může volat metody tohoto objektu, protože zná jeho adresu. Naproti tomu Out-of-process je spuštěn vně adresového prostoru klienta. Je to spustitelný soubor, který dokáže na požádání vytvářet COM objekty. S klientem je spojen přes zástupné objekty proxy a stub. Proxy objekt i stub jsou vytvořeny v adresovém prostoru klientské aplikace a nahrazují chybějící COM objekt. Klientovi se proxy objekt jeví jako požadovaný COM objekt, má stejnou sadu rozhraní, ale neumí provádět jeho metody. V případě, že klient zavolá některou metodu COM objektu, proxy objekt předá parametry volání proxy objektu stub, který se chová jako COM klient. Stub parametry převezme a zavolá metodu rozhraní skutečného COM objektu. Návratové hodnoty potom odešlě zpět proxy objektu a ten je předá klientské aplikaci. Podle umístění klienta se tyto servery dělí na local server (komponenta i klient jsou spuštěny na stejném počítači) a remote server (komponenta a klient jsou na různých počítačích).[5]

1.2 Dynamic Data Exchange (DDE)

DDE je protokol, který slouží ke komunikaci mezi programy. Aplikace, které sdílí data, si zasílají speciální zprávy a data si vyměňují přes sdílenou paměť. Je to komunikace typu server - klient. Aplikaci, která zahájí konverzaci se říká klient a aplikaci, která odpoví na požadavek server. Pro řízení konverzace a zjednodušení práce s DDE se využívá Dynamic data exchange management library (DDEML). Každá DDE konverzace je unikátně definovaná jménem aplikace a tématem konverzace, které jsou určeny na začátku konverzace. Po zahájení konverzace může klient obsadit jeden z několika trvalých datových odkazů na serveru. Tímto trvalým odkazem server oznamuje klientovi každou změnu dat. Trvalý odkaz je zrušen po ukončení konverzace. Podle [7] existují dva typy trvalých odkazů:

- warm (teplý) - server informuje klienta o změně hodnoty
- hot (horký) - server klientovi ihned zašle změněná data

Server může mít navázáno více konverzací s klienty a klienti zase více konverzací se servery. Když přichází zprávy z více zdrojů najednou, příjemce je zpracovává synchronně a může přepínat z jedné konverzace na jinou. [7]

1.3 Technologie OLE

1.3.1 Úvod

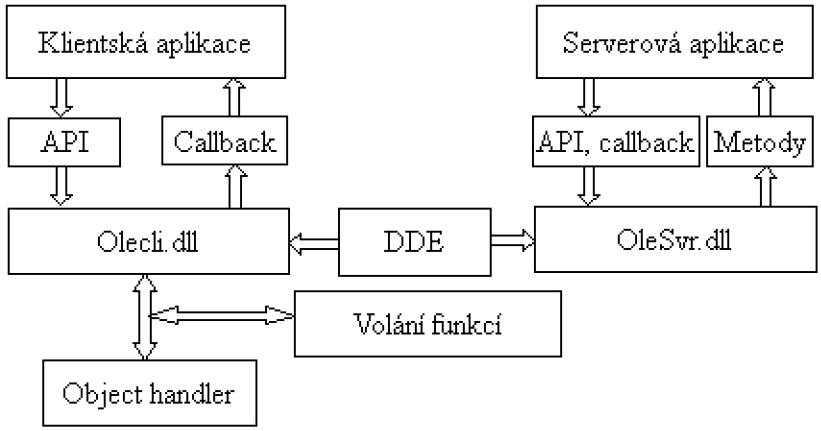
Zkratka OLE je zkratka z anglického „Object Linking and Embedding“, což česky znamená „Vkládání a propojování objektů“. Tato technologie rozšiřuje starší model DDE. OLE ke své práci využívá tzv. servery OLE, což jsou aplikace, které jsou schopny poskytnout jiným aplikacím své služby. S tímto mechanismem můžeme vytvořit aplikaci, která bude využívat některý dostupný server OLE k provedení operace, kterou sama provést nedokáže. Máme-li například server Microsoft Word, který zobrazuje dokumenty ve formátu .doc, můžeme tento server zavolat a v aplikaci zobrazit dokument v tomto formátu bez vytváření vlastních funkcí pro jeho zobrazení.[3]

Mechanismus OLE využívá podle [7] tyto knihovny:

- Olecli.dll (OLE client library) - Aplikace využívající OLE (klientská aplikace) volá funkce pro vytvoření, uložení nebo načtení dokumentu z klientské knihovny. Návratové hodnoty těchto funkcí vysílané OLE serverem jdou zpět opět přes tuto knihovnu a informují klientskou aplikaci o tom, zda byla požadovaná akce vykonána.

- Olesvr.dll (OLE server library) - Serverová aplikace volá funkce ze serverové knihovny, pro svoji registraci, že je nebo není k dispozici a pro indikaci uložení nebo přejmenování dokumentu. Knihovna se serverovou aplikací komunikuje přes sadu 27 aplikací definovaných callback funkcí. Těmito funkcemi jsou volány požadavky pro specifické akce nebo pro informaci, že byla provedena určitá událost.

- Shell.dll - Většina aplikací využívá i tuto knihovnu. Poskytuje API funkce, které umožňují číst a modifikovat registry Windows. V registrech je uložen seznam OLE serverů, objektových tříd a podporovaných klíčových slov. Knihovna také podporuje manipulaci se soubory pomocí „drag and drop“ (dokument je otevřen přetažením do okna klientské aplikace).



Obr. 1.3: Blokové schéma technologie OLE [2]

Kromě serveru a klienta existuje ještě třetí typ OLE modulu, který se nazývá Object handler. Je to dynamicky linkovaná knihovna, která může omezeně využívat funkcí serverové aplikace. Obsahuje funkce pro podporu objektových tříd serverové aplikace. Když klient zavolá klíčové slovo některého objektu, volání může zpracovat object handler. Ten je načten do paměti, zpracuje volání a paměť uvolní bez pomoci hlavního serveru. Object handler představuje výkonnější způsob zpracování, než

serverová aplikace, protože je obvykle menší a je snadněji a rychleji načten a smazán z paměti. Tento modul také jako server využívá OLE server library.[7]

1.3.2 Použití kontejneru OLE

Kontejner OLE se používá pro zobrazení OLE objektu uvnitř kontejnerové aplikace. Kontejnerová aplikace se vytvoří umístěním komponenty OleContainer do formuláře. Pro zobrazení některých OLE objektů (například dokumentů MS Word) používá bitmapový obrázek. Po aktivaci je možné objekt editovat. Existují dva typy umístění objektu OLE do kontejneru [3]:

- Vložení objektu - do aplikace jsou ze svého původního umístění data objektu zkopírována. S daty se zároveň přenesou informace o serveru, aby se mohla serverová aplikace pro úpravu objektu aktivovat z kontejneru. U některých objektů (například dokumenty MS Office, obrázky atd...) je možné objekt měnit v okně kontejneru. V tomto případě se nabídky a panely nástrojů aplikace serveru a kontejneru spojí. Data tohoto objektu ukládá kontejnerová aplikace do vlastních souborů.

- Propojení objektu - do aplikace není objekt zkopírován, ale je v ní vytvořen pouze odkaz na data a na informace o serveru. Soubor nebude vložen do kontejnerového okna jako v případě vložení, ale bude otevřen v samostatném okně kontejnerové aplikace. Data se ukládají do původních souborů.

Důležité vlastnosti komponenty OleContainer:

- AllowActiveDocs - podpora aktivních dokumentů (vložení jednoho dokumentu do druhého).

- AllowInPlace - nastavuje, zda se objekt otevře v okně kontejneru nebo v novém okně.

- AutoActivate - nastavuje, jak se provede aktivace objektu v kontejneru (dvojklikem, zavoláním funkce atd...).

- AutoVerbMenu - pokud je nastavena, automaticky se zobrazuje kontextová nabídka objektu a případně nahrazuje ručně vytvořenou nabídku pro OLE kontejner.

- Linked - pokud je nastavena, je objekt propojen, jinak je vložen.

- OleObject - reprezentuje objekt v OLE kontejneru. Touto vlastností se programově ovládá OLE server.

Metody komponenty OleContainer:

- Close - ukončení serveru OLE, změny provedené v objektu uživatelem se automaticky uloží
- CreateLinkToFile - propojení OLE kontejneru s fyzickým souborem.
- CreateObjectFromFile - vložení objektu do kontejneru
- DestroyObject - ukončení objektu v kontejneru, změny provedené v objektu se neuloží
- DoVerb - provedení požadované akce v OLE objektu
- LoadFromFile - vložení existujícího OLE objektu
- Copy - zkopíruje objekt OLE do schránky
- Paste - pokud je ve schránce objekt OLE, je vložen do kontejneru
- SaveAsDocument - uložení objektu ve formátu, který používá aplikace poskytující OLE server. Pokud se touto metodou uloží dokument MS Word, bude po spuštění v MS Word normálně čitelný.
- SaveToFile - uložení OLE objektu. Po uložení ho lze otevřít pouze jako objekt metodou LoadFromFile. Pokud se tak například uloží dokument v MS Word, potom když se otevře v MS Wordu, není čitelný.
- UpdateObject - znovu načte zdroj dat

Příklad použití:
OleContainer1.CreateObjectFromFile(ExpandFileName('test.doc'),false);

Do komponenty OleContainer1 je vložen dokument test.doc. Metoda ExpandFileName převádí relativní adresu na absolutní, hodnotou false je nastaveno, že se objekt v kontejneru nemá zobrazovat jako ikona.

1.3.3 OLE Automation

Úvod

Pomocí mechanismu OLE je možné do vytvářené aplikace programově vložit nějaký objekt z cizí aplikace. Je možné objekt vytvořit, otevřít, uložit nebo zavřít a umožňuje uživateli měnit vlastnosti objektu (například v Microsoft Wordu je možné měnit text, formátování atd...). Pokud ale chceme objekt ovládat plně programově

(měnit vlastnosti objektu, které v technologii OLE může měnit pouze uživatel), je potřeba využít technologii OLE Automation. Automation objekt je COM objekt, který má implementováno rozhraní IDispatch. Metody volání serveru jsou tyto [1]:

- S využitím rozhraní IDispatch. Názvy metod se předávají v řetězci pomocí proměnné typu variant (do tohoto typu proměnné se dají ukládat různé datové typy a v době překladu nemusí být známo, jaký datový typ je v proměnné umístěn). U této techniky není potřeba typových informací o serveru. Použití je jednoduché, překladač může zkompilovat kód, i když neví nic o metodách serveru. Při kompilaci překladač nekontroluje typy proměnných - kontrolují se až za běhu programu. Proto je tato metoda nejpomalejší.

- S využitím rozhraní dispinterface, které k volání metod serveru využívá DISPID. Před použitím tohoto způsobu je nutné importovat typové knihovny serveru, které obsahují popis všech objektů, rozhraní a dalších typových informací, které daný typ serveru poskytuje. Při použití tohoto způsobu překladač kontroluje typy parametrů a objeví chyby již při překladu. Aplikace je však omezena jen na komunikaci s typem serveru, ze kterého byly importovány typové knihovny.

- Voláním rozhraní pomocí VTBL tabulky. Tento způsob je nejrychlejší. S rozhraním se zachází jako s objektem COM.

Využití rozhraní IDispatch k ovládání programu MS Word

Aplikace MS Word se programově otevírá metodou CreateOleObject('Word.Application'), jejíž výsledek, který na spuštěnou aplikaci ukazuje, se přiřazuje proměnné typu variant. Prostřednictvím této proměnné je potom možno volat metody a objekty aplikace. Metoda CreateOleObject() zavolá metodu CoCreateInstance, která voláním metody CoGetClassObject vytvoří neinicializovaný objekt určený číslem CLSID. Metoda CoGetClassObject zavolá COM service Control Manager, který pomocí CLSID hledá knihovnu asociovaného COM serveru, načte jí do paměti a zavolá metodu DllGetClassobject. Tato metoda vytvoří objekt COM třídy ClassFactory (jehož jediným úkolem je vytvořit jiný objekt), následně zavolá metodu CreateInstance, která s pomocí objektu ClassFactory vytvoří objekt COM a zavolá metodu QueryInterface, která vrátí ukazatel na rozhraní, jenž potřebujeme.

Následující kousek kódu se pokusí použít spuštěnou instanci Wordu metodou GetActiveOleObject, pokud není spuštěna, spustí ji a vytvoří nový dokument.

```
Try
   WordAplikace := GetActiveOleObject('Word.Application');
Except
Begin
   WordAplikace:=CreateOleObject('Word.Application');
   WordAplikace.Visible:=True;
   WordAplikace.DisplayAlerts:=True;
End;
WordAplikace.Documents.Add;
```

Nejdůležitější metody objektu Document:

- Add - vytvoření nového dokumentu
- Open('Název dokumentu') - otevření existujícího dokumentu
- Save - uložení všech otevřených dokumentů
- Item(1).SaveAs('Název dokumentu') - uložení nového dokumentu, do závorek je možno psát pouze název dokumentu nebo celou cestu
- Item(1).Save - uložení dokumentu
- Item(1).Close - zavření dokumentu
- Item(1).Activate - aktivace dokumentu
- Item(1).Select - označení textu celého dokumentu
- Item(1).Undo - krok zpět
- Item(1).Redo - krok vpřed

Některé potřebné metody objektu Selection (objekt Selection pracuje s označeným textem v dokumentu):

- TypeText(Text:= 'Nějaký text') - na místo kurzoru se vloží text: Nějaký text
- Font.Bold:= True - nastavení vybraného textu na tučný (místo Bold může být Italic - kurzíva, Underline - podtržené, ...)
- Font.Size:= velikost - nastavení velikosti písma
- TypeParagraph - vytvoří nový odstavec

- Font.Name:= 'Název písma' - nastavení písma na písmo, které je udáno položkou Název písma (např. Times New Roman, Arial,...)
- EndKey() - nastavení kurzoru na konec řádku
- HomeKey() - nastavení kurzoru na začátek řádku
- Move(x,y) - provede se posuv v dokumentu x o y znaků

Aby bylo možno s objektem Selection pracovat, musí se nejprve označit oblast, kterou chceme formátovat. K tomu se používá metoda Range objektu Document. Vlastnosti Start a End určují začátek a konec oblasti, kterou chceme označit. Pokud jsou hodnoty Start a End stejné, na místo určené těmito hodnotami se přesune kurzor a nic se neoznačí. Například takto se označí první řádek textu a provede se jeho podtržení:

```
WordAplikace.Documents.Item(1).Range(Start:=0,
End:=WordAplikace.Selection.EndKey()).Underline:=True;
```

Využití rozhraní dispinterface k ovládání programu MS Word

Po naimportování typových knihoven do vývojového prostředí se do palety komponent přidají další komponenty pro práci s MS Word. Jedna z komponent je komponenta WordApplication. Názvy metod jsou podobné jako při použití rozhraní IDispatch (pro MS Office 2000):

- WordApplication1.Documents.Add() - vytvoření nového dokumentu
- WordApplication1.Documents.Save() - uložení dokumentu
- WordApplication1.Documents.Open() - otevření dokumentu
- WordApplication1.Selection.SetRange(n, m) - označení oblasti, začínající na n-tém znaku a končící na m-tém znaku
- WordApplication1.Selection.Font.Underline:=wdUnderlineSingle - změna formátu označeného písma na podtržené
- WordApplication1.Selection.Font.UnderlineColor:=clRed - nastavení barvy podtržení
- WordApplication1.Selection.Font.Color:=clRed - nastavení barvy písma
- WordApplication1.Selection.TypeParagraph - nový odstavec

- WordApplication1.Selection.TypeText('Nějaký text') - vložení textu na místo kurzoru

V tomto případě se MS Word programově spouští takto:

```
WordApplication1.Connect;
WordApplication1.Visible := True;
WordApplication1.Documents.Add(EmptyParam,
  EmptyParam, EmptyParam, EmptyParam);
```

První parametr metody Add je název šablony, která bude použita, druhý je název nové šablony, která se vytvoří, třetí je typ vytvořeného dokumentu a čtvrtý je viditelnost dokumentu. Pokud jsou všechny parametry EmptyParam, otevře se obyčejný nový dokument.

1.4 Použití háků

Háky slouží k monitorování systémových zpráv, ve kterých se přenášejí stisknuté klávesy, pohyby myší, pohyby okna programu atd. Používají se k provedení určité akce, kdykoliv nastane událost určená ukazatelem na callback funkci, tzv. hákovou proceduru uloženou v aktuálním hákovém řetězci. Hákový řetězec je seznam ukazatelů na aplikací definované hákové procedury. Pokud je hákem zachycena událost, jsou postupně volány jednotlivé procedury obsažené v řetězci. Hákové procedury mohou monitorovat zprávy, některé jsou schopny je měnit nebo zastavit jejich zpracování. Vstupními parametry hákové procedury jsou:

- hákový kód - závisí na typu háku a slouží pro určení akce, kterou má procedura provést
- informace o zprávě, která byla poslána

Háky se dělí na globální háky a lokální háky. Globální háky monitorují zprávy pro všechny programy, zatímco lokální jen pro jeden určený program. Při práci s nimi je hlavní rozdíl v tom, že funkce pracující s lokální háky mohou být uvnitř programu, který je volá. Funkce globálních háků musí být umístěny v oddělené dll knihovně. Zde je seznam hákových procedur [7]:

- WH_CBT - je volána před aktivací, vytvořením, zničením, minimalizací, maximalizací, posunutím nebo změnou rozměrů okna.
- WH_DEBUG - je volána před voláním jiné hákové procedury.

- WH_FOREGROUNDIDLE - je volána, když se vlákno aplikace stane nečinným.
- WH_GETMESSAGE - monitoruje zprávy ze vstupu klávesnice a myši, které jsou posílány do fronty zpráv.
- WH_JOURNALPLAYBACK - umožňuje získání série zpráv ze vstupu myši a klávesnice, které byly zachyceny při spuštěné proceduře WH_JOURNALRECORD. Dokud je tato procedura spuštěna, jsou vstupy klávesnice a myši vypnuty.
- WH_JOURNALRECORD - umožňuje monitorovat a nahrávat vstupní události myši a klávesnice, které je možné později zpracovat pomocí procedury WH_JOURNALPLAYBACK.
- WH_KEYBOARD - monitoruje zprávy WM_KEYDOWN a WM_KEYUP posílané do fronty zpráv.
- WH_MOUSE - monitoruje zprávy myši posílané do fronty zpráv.
- WH_MSGFILTER - umožňuje monitorovat zprávy z hlavního menu, scroll baru, message boxu, dialog boxu a při aktivaci okna pomocí klávesových zkratek ALT+TAB nebo ALT+ESC. To vše dokáže monitorovat pouze u aplikace, která tuto hákovou proceduru spustila.
- WH_SYSMSGFILTER - monitoruje to samé, co předchozí procedura, ale dokáže monitorovat okna všech aplikací.
- WH_SHELL - tímto hákem přijímá shellová aplikace důležitá oznámení.

Funkce potřebné pro práci s hákovými procedurami [7]:

- SetWindowsHookEx - spustí hákovou proceduru. Jejími parametry jsou: typ hákové procedury reprezentované číslem, ukazatel na proceduru, ukazatel na dll (pokud je hák lokální, tak je zde nula) a ID programu nebo vlákna, které se má monitorovat (pokud je hák globální, je zde nula). Vrací ukazatel na spouštěný hák, pomocí něhož se nakonec hák uvolňuje.
- CallNextHookEx - volá následující hákovou proceduru v řetězci a jako parametry jí předává informace z aktuální hákové procedury.
- UnhookWindowsHookEx - uvolňuje hákovou proceduru z hákového řetězce, jejím parametrem je ukazatel na hákovou proceduru, která má být uvolněna.

- RegisterWindowMessage - definuje jméno okna, které je v celém systému unikátní.
- CreateFileMapping - otevře, případně vytvoří mapování určeného souboru. Parametry této funkce jsou: ukazatel na soubor pro mapování, dědičnost procesů potomků, přístupová práva k souboru, maximální velikost souboru, která bude mapována a název namapovaného objektu.
- MapViewOfFile - mapuje zobrazení souboru v adresovém prostoru volajícího procesu. Parametry této funkce jsou: ukazatel na vytvořený mapovaný objekt, přístupová práva k objektu, offset, kde má mapování začít a počet bajtů, které se mají namapovat.
- UnmapViewOfFile - zruší namapovaný pohled ze souboru z adresového prostoru procesu.
- CreateToolhelp32Snapshot - zachycení určitého procesu v systému. Parametry této funkce jsou: co se bude zachytávat (moduly, procesy, vlákna...) a identifikátor procesu, který má být zachycen.
- Process32First - získává informace o prvním zachyceném procesu systému.
- Process32Next - získává informace o následujícím procesu systému.
- SendMessageTimeout - posílá zprávu jednomu nebo více oknům. Jejími parametry jsou: ukazatel na okno, pro které je zpráva (pokud je parametrem HWND_BROADCAST, posílá se zpráva všem oknům), druh zprávy, dodatečné informace, způsob odeslání a čas vypršení.

Příklad použití hákové procedury WH_KEYBOARD (Do komponenty Memo1 se zapisují kódy stisknutých kláves) :

```
//odkaz na funkce obsažené v knihovně dll:

function GetHookMsg: DWORD; stdcall; external 'HookLib.dll';
function RemoveHook: Boolean; stdcall; external 'HookLib.dll';
function SetHook(ThreadID: DWORD): Boolean; stdcall;
  external 'HookLib.dll';

function FindProcessName(PID: DWORD): String;
var
   SnapProcHandle: THandle;
```

```
   ProcEntry: TProcessEntry32;
   NextProc: Boolean;
begin
   Result := ' ';
   SnapProcHandle := CreateToolhelp32Snapshot(
     TH32CS_SNAPPROCESS, 0);
   if  SnapProcHandle <> THandle(-1) then
   begin
      ProcEntry.dwSize := Sizeof(ProcEntry);
      NextProc := Process32First(SnapProcHandle, ProcEntry);
      while NextProc do
      begin
         if ProcEntry.th32ProcessID = PID then
         begin
            //uložení ukazatele na spustitelný soubor
            //do proměnné Result:
            Result := ProcEntry.szExeFile;
            break;
         end;
         NextProc := Process32Next(SnapProcHandle, ProcEntry);
      end;
      //uvolní ukazatele na proces:
      CloseHandle(SnapProcHandle);
   end;
end;

procedure TForm1.WndProc(var Msg: TMessage);
begin
   if Msg.Msg = GetHookMsg then
   begin
      //do komponenty memo1 jsou vypisovány kódy kláves:
      Form1.Memo1.Lines.Add(Format('%.8x %.4x %s', [Msg.LParam,
      Msg.WParam, FindProcessName(Msg.LParam)]));
      Msg.Result := 1;
   end
   else
      inherited;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
   //pokud nebyl hák nastaven, zobrazí se okno s chybou:
   if not SetHook(0) then ShowMessage('Chyba');
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
   RemoveHook;
end;
```

Kód knihovny dll:

```
function GetHookMsg: DWORD; stdcall;
function RemoveHook: Boolean; stdcall;
function SetHook(ThreadID: DWORD): Boolean; stdcall;

const
   HookLibIdName = 'WindowsHookLib';

type
   PSharedData = ^TSharedData;
   TSharedData = record
      HookHandle: HHOOK;
      HookMessage: DWORD;
   end;  //struktura pro uložení háků

var
   MapFileHandle: THandle;
   SharedData: PSharedData;

function HookProc(Code: UINT; WParam: WPARAM; LParam: LPARAM):
LRESULT; stdcall;
var
   Res: DWORD;
begin
   Result := 0;
   if Code = HC_ACTION then
      SendMessageTimeout(HWND_BROADCAST, SharedData^.HookMessage,
        Wparam, GetCurrentProcessId, SMTO_ABORTIFHUNG, 200, Res)
```

```
   else
   if Integer(Code) < 0 then
      Result := CallNextHookEx(SharedData^.HookHandle, Code,
        WParam, LParam);
end;

function GetHookMsg: DWORD; stdcall;
begin
   Result := SharedData^.HookMessage;
end;

function RemoveHook: Boolean; stdcall;
begin
   Result := UnhookWindowsHookEx(SharedData^.HookHandle);
   if Result then SharedData^.HookHandle := 0;
end;

function SetHook(ThreadID: DWORD): Boolean; stdcall;
begin
   if (MapFileHandle <> 0) and (SharedData^.HookHandle = 0) then
   begin
       SharedData^.HookMessage := RegisterWindowMessage(
         HookLibIdName);
       SharedData^.HookHandle := SetWindowsHookEx(WH_KEYBOARD,
       @HookProc, HInstance, ThreadID);
       Result := SharedData^.HookHandle <> 0;
   end
   else
      Result := False;
end;

//následující kód se provede při spuštění:

initialization
   MapFileHandle := CreateFileMapping ($FFFFFFFF, null,
     PAGE_READWRITE, 0, Sizeof(SharedData^), HookLibIdName);
   SharedData := MapViewOfFile(MapFileHandle, FILE_MAP_WRITE,
   0, 0, Sizeof(SharedData^));
```

```
finalization  //tento kód se provede před ukončením
   UnmapViewOfFile(SharedData);
   CloseHandle(MapFileHandle);
end.
```

1.5 Práce s XML

1.5.1 Úvod

XML je značkovací jazyk, to znamená, že označuje svůj obsah pomocí symbolů, které se nazývají značky v (identifikátory v lomených závorkách).

Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. XML umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML.

Na začátku dokumentu XML může být a nemusí XML deklarace, která je uzavřená mezi značkami <? a ?> a v atributu version obsahuje verzi standardu XML. Tato deklarace vypadá takto: <?xml version=”1.0”?>. Dokument musí obsahovat jeden kořenový element, což je element, ve kterém jsou uzavřeny další elementy a text. Elementy jsou vymezeny počátečním a koncovým tagem. Uvnitř mohou mít další obsah, ale mohou zůstat prázdné. Názvu elementu, který je uzavřen mezi značkami <a> se říká uzel. Uzlu, který je umístěn uvnitř jiného uzlu se říká potomek. V následujícím příkladu je potomek uzlu <vlastnosti> uzel <hlavicka>, <teloDokumentu> a <zapati>.

Příklad XML dokumentu:

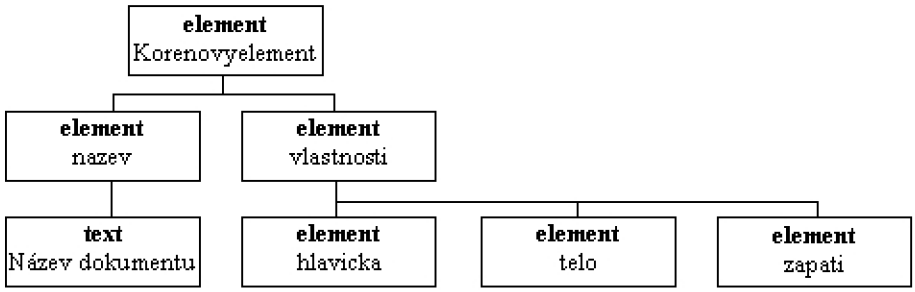
```
<?xml version="1.0"?>
<korenovyElement>
   <nazev>Název dokumentu</nazev>
   <vlastnosti>
      <hlavicka>Hlavička dokumentu</hlavicka>
      <teloDokumentu></teloDokumentu>
      <zapati></zapati>
   </vlastnosti>
</korenovyElement>
```

Podle [1] existují dva základní přístupy k práci s dokumenty XML:

- rozhraní modelu DOM (Document Object Model) - načte celý dokument do hierarchického stromu uzlů a umožní dokumenty číst a upravovat. DOM se používá, pokud chceme procházet a upravovat strukturu XML v paměti nebo vytvářet nové dokumenty.
- rozhraní SAX (Simple API for XML) - analyzuje dokument a pro každý jeho element spouští událost. Nevytváří při tom žádnou strukturu v paměti. Je mnohem rychlejší než tvorba stromu v DOM. SAX se hodí pro jedno čtení dokumentu nebo pro hledání konkrétní části dokumentu.

1.5.2 Model DOM

Struktura dokumentu načteného v paměti



Obr. 1.4: Příklad stromové struktury načteného dokumentu v paměti

Metody pro používání modelu DOM

Pro práci s dokumenty XML je potřeba implementace modelu DOM, která je dostupná jako server COM a ze které je potřeba naimportovat typové knihovny. Nejpoužívanější implementace je MSXML SDK od Microsoftu, která je součástí Internet Exploreru. Po importu je do panelu komponent přidána komponenta DOMDocument, která se používá k práci s XML dokumenty.[1]

Nejdůležitější metody pro práci s XML (při použití typových knihoven MSXML):

- load('umístění') - načte existující XML dokument.
- documentElement.getElementsByTagName('název_uzlu') - vrací seznam hodnot, které jsou umístěny mezi uzly určenými parametrem název_uzlu. Výsledek je možno procházet a hledat určitou hodnotu.

- documentElement.createElement('název_uzlu') - vytvoří uzel a vrací ukazatel na tento uzel.
- documentElement.appendChild(ukazatel na uzel); - Přidá uzel na konec listu potomků.
- documentElement.insertBefore(ukazatel na uzel,ukazatel na uzel před ním) - Nový uzel je vložen za uzel. určený v druhém parametru metody. Pokud je parametrem hodnota null, je nový uzel vložen na konec.
- documentElement.createTextNode('text') - vloží do uzlu text.
- documentElement.createAttribute('atribut') - vytvoří atribut s názvem, který je uveden jako parametr.
- save('umístění') - uloží XML dokument.

Příklad použití modelu DOM:

```
ixml:=Domdocument1.DefaultInterface;
iroot:=ixml.appendChild(ixml.createElement('xml'));
inode:=iroot.appendChild(ixml.createElement('dokument'));
ichild:=inode.appendChild(ixml.createElement('nazev'));
ichild.appendChild(ixml.createTextNode('Nějaký text'));
ichild2:=inode.appendChild(ixml.createElement('hodnota'));
ichild2.appendChild(ixml.createTextNode('Nějaká hodnota'));
DomDocument1.save('umístění');
```

Uvedený kód vytvoří tento dokument XML:

```
<xml>
   <dokument>
      <nazev>Nějaký text</nazev>
      <hodnota>Nějaká hodnota</hodnota>
   </dokument>
</xml>
```

1.5.3 Technologie API SAX

SAX při průchodu XML dokumentem vyvolává tyto události:

- StartDocument - na začátku dokumentu
- EndDocument - na konci dokumentu

- StartElement - na začátku každého uzlu
- EndElement - na konci každého uzlu
- Characters - pro text v uzlech

Pro práci s dokumenty XML pomocí technologie SAX je také potřeba naimportovat typové knihovny serveru COM. Nejpoužívanější implementace je MSXML od Microsoftu. Využívá se jeho rozhraní IVBSAXXMLReader. Na začátku je nutné vytvořit COM objekt s tímto rozhraním. Potom se musí vytvořit odvozená třída od rozhraní IDispatch a IVBSAXContentHandler, které sleduje elementy při průchodu dokumentem, a upravit těla metod, které mají něco dělat (např. zjistit hodnotu určitého uzlu). Pro start analýzy dokumentu se použije příkaz parseURL('umístění').[1]

Následující příklad prochází dokument a hledá uzly se jménem nazev. Když takový uzel najde, uloží do seznamu typu string text tohoto uzlu (pokud nějaký text obsahuje). Nakonec se všechny řetězce překopírují do komponenty Memo1.

```
//Ve třídě TMySAXHandler, odvozené od IVBSAXContentHandler,
//se vytvoří metody Characters a StartElement:

var
   Log: TStrings;
   Spravne: Boolean;
procedure TMySaxHandler.characters(var strChar: WideString);
begin
   inherited;
   if (Spravne) then
      if (strChar <> ' ') then
         Log.Add ('Text: ' + strChar);
end;

procedure TMySaxHandler.startElement(var strNamespaceURI,
   strLocalName, strQName: WideString;
   const oAttributes: IVBSAXAttributes);
begin
   inherited;
   Spravne:=(strLocalName='nazev');
end;

//Potom se ošetří událost po stisku tlačítka button1.
//Aktivuje se analyzátor SAX voláním metody parseURL
```

```
//(s parametrem analyzovaného souboru) po přiřazení obsluhy obsahu:

procedure TForm1.Button1Click(Sender: TObject);
begin
   Memo1.Clear;
   sax.ContentHandler := TMySaxHandler.Create;
   sax.parseURL('c:\temp\pokus.xml');
end;

//V události po vytvoření formuláře se proměnná sax
//inicializuje objektem COM:

procedure TForm1.FormCreate(Sender: TObject);
begin
   sax:=CreateComObject(CLASS_SAXXMLReader) as IVBSAXXMLReader;
end;
```

2 ŘEŠENÍ

2.1 Popis řešení

Pro programové ovládání MS Word jsem vybral metodu s využitím rozhraní IDispatch, které pracuje v tzv. pozdní vazbě, což znamená, že překladač kontrolu syntaxe neprovádí a ta je kontrolována až za běhu programu. Běh programu je potom sice pomalejší, ale program bude fungovat ve všech verzích MS Office. Pokud bych využil dispinterface a importoval typové knihovny, program by fungoval jen pro tu verzi MS Office, ze které by byly importovány typové knihovny.

Pro zachytávání systémových zpráv obsahujících kódy stisknutých kláves využívám hákovou proceduru WH_KEYBOARD a pro detekci aktivace okna a jeho zavření proceduru WH_CBT. Tyto procedury jsou (spolu s procedurou pro zavedení háku a uvolnění háku) umístěny v připojené dll knihovně, aby bylo možno zachytávat zprávy globálně v celém systému.

Pro načítání údajů o podtrženém řádku a velikosti a poloze oken z XML využívám API SAX. Tato technologie je rychlá a jednoduchá, avšak není v ní možno dokument modifikovat. Pro ukládání údajů používám model DOM, který zápis do XML umožňuje, ale čtení z dokumentu je složitější než u API SAX.

2.2 Popis jednotlivých procedur a funkcí

2.2.1 Hlavní program

- FindProcessName - Tato funkce zachytává zprávy od všech procesů.
- uloz_hodnoty - Nejprve zkontroluje, zda byl naposledy skok na další řádek nebo stránku. Pokud ne, prohodí proměnné mez1 a mez2. Potom načte XML dokument a nastaví polohu na uzel dokumenty. Pokud již uzel s atributem, který chce vytvořit, existuje, nejprve ho smaže a potom vytvoří nový uzel s atributem cesty k překládanému souboru. Tento uzel je rodičem uzlů: radek (kolikátý řádek), mez1 (na kolikátém znaku je začátek řádku), mez2 (na kolikátém znaku je konec řádku), delka (počet znaků v dokumentu) a preklad (cesta k překladu).
- uloz_polohu - Pracuje podobně jako předchozí, ale vytváří uzel nastaveni_orig nebo nastaveni_prekl podle toho, jaký je vstupní parametr funkce. Jeho potomky jsou: sirka, vyska, left, top.

- WndProc - Testuje kódy zpráv, zachycených dll knihovnou. Pokud zjistí, že byla stisknuta některá z klávesových zkratek, spustí časovač, který po vypršení spustí příslušnou proceduru. Pokud zachytí zprávu aktivace okna MS Wordu v okamžiku, kdy bylo kliknuto na tlačítko otevřít, provede se uložení PID procesu, aby bylo možné proces identifikovat až zachytí zprávu uzavření překládaného dokumentu.

- TMySaxHandler.characters - Ukládá textové uzly, které načte z XML, do proměnných.

- TMySaxHandler.endDocument - Po dosažení konce dokumentu XML, nastaví rozměry okna překládaného dokumentu.

- TMySaxHandler.startElement - Nastavuje pomocné proměnné na True, aby se dalo testovat, v jakém uzlu se právě nachází.

- TMySaxHandler.endElement - Nastavuje pomocné proměnné na False.

- TlOtevreniOriginalu - Vytvoření OLE objektu Word.Application, otevření požadovaného souboru a spuštění XML parseru, který hledá v XML dokumentu uzel se správným atributem.

- TlPredchoziRadek - Nejprve přesune kurzor na začátek dokumentu a potom skočí na předchozí pozici (kdyby uživatel kliknul do překládaného dokumentu). Označí řádek, na kterém se kurzor nachází. Na tomto řádku zruší podtržení. Kurzor se posune na předchozí řádek. Program zjistí, kolik je na řádku znaků a pro tento počet znaků provede podtržení.

- TlKonec - Pokud je otevřeno okno s překládaným dokumentem, uloží polohu v dokumentu a polohu okna a okno zavře. Pokud je otevřeno okno s překladem, uloží polohu okna a okno zavře. Pokud není otevřeno ani jedno okno, ukončí program.

- TlDalsiRadek - Pokud byl naposledy skok na předchozí řádek, prohodí hodnoty mez1 a mez2. Pokud je to první řádek, zjistí délku celého dokumentu, aby věděl, kdy dojde na konec. Potom přesune kurzor na předchozí pozici a zruší podtržení řádku, na kterém se kurzor nachází. Poté se přesune na následující řádek a řádek podtrhne. Nakonec ještě uloží cestu k překladu, aby ho bylo možné automaticky otevřít při příštím spuštění překládaného souboru.

- FormCreate - Pokud proceduru spustí Form1, zavedou se háky a otevře se soubor s hodnotami polohy okna hlavního programu. Tato procedura se spouští

také po každém kliku na tlačítko „Otevřít originál“. Následně se nastaví proměnné na výchozí hodnoty.

- TlDalsiStranka - Nejprve se zruší podtržení předchozího řádku. Poté 49 krát posune kurzor na další řádek. Pokud v dokumentu není dalších 50 řádků, provede se opět podtržení předchozího řádku. Nakonec zavolá proceduru TlDalsiRadek, která provede podtržení následujícího řádku.
- FormDestroy - Odinstaluje háky a uvolní proměnnou ukazující na knihovnu dll.
- FormClose - Podobná jako procedura TlKonec. Navíc ukládá údaje o poloze okna hlavního okna programu.
- TlPredchoziStranka - Zjistí, jestli je nad podtrženým řádkem 50 nebo více řádků. Pokud ano, zruší podtržení aktuálního řádku a posune kurzor o 49 řádků nahoru. Potom zavolá proceduru TlPredchoziRadek, která provede podtržení řádku.
- TlNovyPreklad - Vytvoří nový objekt OLE typu Word.Application, vytvoří nový dokument a nastaví rozměry a polohu okna překladu.
- TlOtevreniPrekladu - Pokud proceduru spustí tlačítko, otevře se dialog, kde je možné vybrat překlad. Pokud ji ale spustí program automaticky po otevření překládaného souboru, otevře se soubor, jehož cesta je uložena v proměnné prekladovy_soubor. Poté se vytvoří objekt OLE a nastaví se rozměry a poloha okna.

2.2.2 Knihovna dll

- HookKeyb - Filtruje zachycené kódy stisknutých kláves. Pokud jsou to klávesy z klávesových zkratek programu, přepošle je všem oknům.
- HookCBT - Filtruje zachycené kódy procedury WH_CBT. Pokud je zachycena aktivace okna nebo zavření, je kód zprávy přeposlán všem oknům.
- TKnihovna.GetHookMsg - Získává zprávy systému.
- TKnihovna.RemoveHook - Odinstaluje háky.
- TKnihovna.SetHook - Zařadí hákové procedury do hákového řetězce.
- Knihovna_export - Vytvoření třídy TKnihovna pro export.

2.3 Funkce programu

Při spuštění programu se zavedou hákové procedury. Po kliknutí na tlačítko „Otevření originálu“ se vytvoří nový objekt aplikace MS Word. Po výběru dokumentu program okno aktivuje, aby mohl hák WH_CBT zachytit zprávu s kódem HCBT_ACTIVATE a uložit si z ní PID procesu dokumentu. Poté se spustí XML parser, který hledá v XML dokumentu uzly s hodnotami pro otevřený dokument. Tyto hodnoty se uloží do proměnných. Pokud je v uzlu „preklad“ cesta k existujícímu souboru, je soubor automaticky načten. Podobnou funkci jako tlačítko „Otevření originálu“ mají i tlačítka „Nový překlad“ a „Otevřít překlad“ (kromě spuštění XML parseru). Po otevření okna se deaktivuje příslušné tlačítko. Po stisku některé z těchto klávesových zkratek:

- ALT + PgUp - Posun o jeden řádek zpět
- ALT + PgDown - Posun o jeden řádek vpřed
- ALT + Home - Posun o stránku vzad
- ALT + End - Posun o stránku vpřed

se spustí časovač (pokud se časovač nepoužije, MS Word hlásí chybu). Ten po vypršení aktivuje proceduru, která je přiřazena do události OnClick u příslušného tlačítka. Po kliknutí na tlačítko „Konec“ se uloží údaje o podtrženém řádku, cestě k překladu a pozici a velikosti jednotlivých oken do XML dokumentu s následující strukturou (pokud již uzel se stejnou cestou existuje, bude nahrazen):

```
<dokumenty>
   <dokument nazev="Cesta k originálu 1.doc">
      <radek>7</radek>
      <mez1>473</mez1>
      <mez2>564</mez2>
      <delka>877</delka>
      <preklad>Cesta k prekladu 1.doc</preklad>
   </dokument>
   <dokument nazev="Cesta k originálu 2.doc">
      <radek>5</radek>
      <mez1>198</mez1>
      <mez2>290</mez2>
      <delka>7702</delka>
      <preklad>0</preklad>
   </dokument>
```

```
   <nastaveni_orig>
      <sirka>672</sirka>
      <vyska>301</vyska>
      <left>186</left>
      <top>1</top>
   </nastaveni_orig>
   <nastaveni_prekl>
      <sirka>672</sirka>
      <vyska>301</vyska>
      <left>185</left>
      <top>307</top>
   </nastaveni_prekl>
</dokumenty>
```

Po opětovném kliknutí na tlačítko „Konec“ je program ukončen. Pokud uživatel zavře některé z oken MS Wordu, je zachycena zpráva a tlačítko se opět aktivuje (jestliže zpráva obsahuje PID překládaného dokumentu, je pozice v dokumentu uložena do XML dokumentu). V tomto případě nedojde k uložení pozic a rozměrů oken, protože v okamžiku zachycení zprávy je příslušné okno MS Word již zavřeno a nelze zjistit jeho rozměry.

Překládaný dokument nesmí obsahovat obrázky, tabulky a další vložené soubory. Velikost písma by měla být dvanáct (MS Word nemá metodu, která by posunula kurzor o stránku a vrátila počet znaků, o kolik se posunul. Proto se provádí posun o padesát řádků – při velikosti písma dvanáct je na stránce padesát řádků).

3 ZÁVĚR

Cílem této práce bylo vytvořit Systém pro podporu práce překladatelů. V první části jsem rozebral technologie potřebné pro jeho vytvoření. Popsal jsem model COM, který umožňuje komunikaci mezi jednotlivými procesy. Dále OLE a OLE Automation, které jsou založeny na COM a umožňují programu přístup k OLE serverům jednotlivých aplikací, díky nimž může program využívat některých jejich služeb. Potom jsem popsal činnost háků a funkce pro práci s nimi. Na konci první části jsem rozebral API SAX a model DOM pro práci s XML dokumenty.

Na začátku druhé části jsem z možných řešení vybral ty nejvhodnější. MS Word je ovládán pomocí rozhraní IDispatch v pozdní vazbě. API SAX využívám pro hledání a čtení z XML dokumentu a model DOM pro zápis do XML dokumentu. Pro zachytávání zpráv Windows program volá hákovou proceduru WH_KEYBOARD pro zachytávání kódů stisknutých kláves a WH_CBT pro zachytávání kódu aktualizace a zavření okna. Poté jsem vypsal seznam procedur a funkcí, obsažených ve zdrojovém kódu programu s popisem jejich funkce. Nakonec jsem popsal funkci vytvořeného programu. Program byl testován s MS Office 2000, XP a 2007.

LITERATURA

[1] CANTÚ, Marco. *Myslíme v jazyku Delphi 7.* Olga Hanušová; Jiří Hynek. 1. vyd. Praha : Grada, 2003. 580 s. Myslíme v.... ISBN 80-247-0694-6.

[2] Int21 Corporation. *Int21* [online]. 1996- [cit. 2007-12-17]. Dostupný z WWW: <http://www.int21.co.jp/pcdn/vb/noriolib/vbmag/6/oleabs/oleabs.html>.

[3] KADLEC, Václav. *Delphi : Hotová řešení.* 1. vyd. Brno : Computer Press, 2003. 312 s., 1 CD-ROM. ISBN 80-251-0017-0.

[4] KADLEC, Václav. Pod pokličku modelu COM . *Umíme to s Delphi* [online]. 2005, č. 167 [cit. 2005-10-19]. Dostupný z WWW: <http://www.zive.cz/Clanky/Umime-to-s-Delphi-167-dil–pod-podklicku-modelu-COM/sc-3-a-127146/default.aspx>.

[5] KADLEC, Václav. Popis modelu COM, dokončení. *Umíme to s Delphi* [online]. 2005, č. 168 [cit. 2005-10-26]. Dostupný z WWW: <http://www.zive.cz/Clanky/Umime-to-s-Delphi-168-dil–popis-modelu-COM-dokonceni/sc-3-a-127262/default.aspx>.

[6] LY, Binh. *Techvanguards.com* [online]. 1999-2005 , 1/23/2005 [cit. 2007-11-20]. Dostupný z WWW: <http://www.techvanguards.com/com/>.

[7] Microsoft. *Microsoft Developer Network* [online]. 2007 [cit. 2007-11-20]. Dostupný z WWW: <http://www.msdn.microsoft.com>.

[8] ROFAIL, Ash, SHOHOUD, Yasser. *Mastering COM and COM+.* 1st edition. [s.l.] : Sybex, Incorporated, 1999. 848 s. ISBN 0782123848.

[9] WOOD, Keith. *Delphi developer's guide to XML.* [s.l.] : Wordware Publishing, Inc, 2001. 530 s. ISBN 1-55622-812-0.

SEZNAM SYMBOLŮ A ZKRATEK

API rozhraní pro programování aplikací – Application Programming Interface

COM objektový model – Component Object Model

DDE dynamická výměna dat – Dynamic Data Exchange

DDEML knihovna řízení dynamické výměny dat – Dynamic Data Exchange Management Library

DLL dynamicky připojená knihovna – Dynamically Linked Library

DOM objektový model dokumentu – Document Object Model

GUID globálně unikátní identifikátor – Globally Unique Identifier

OLE vkládání a propojování objektů – Object Linking and Embedding

SAX jednoduché API pro XML – Simple API for XML

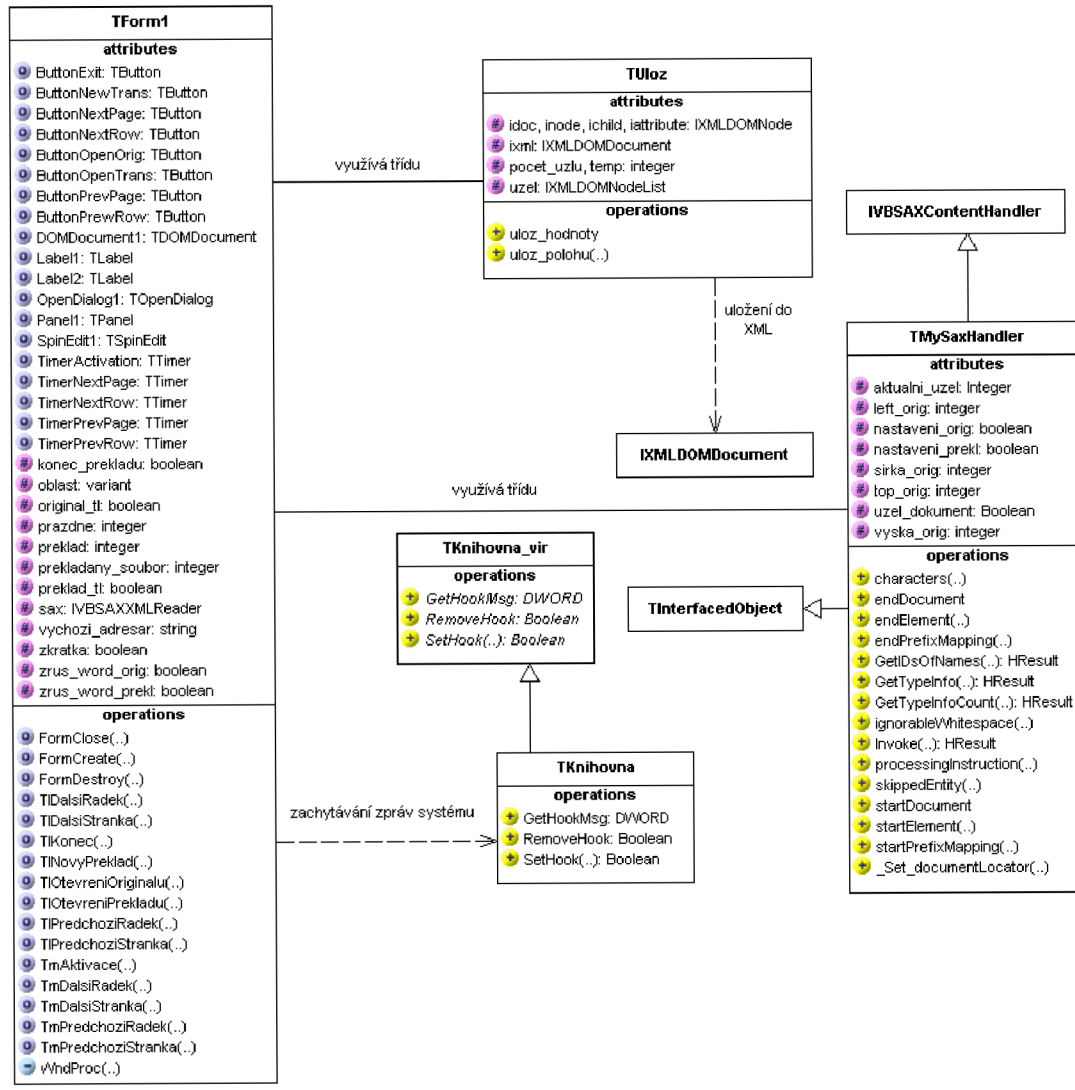
VTBL virtuální tabulka funkcí – Virtual Function Table

XML rozšiřitelný značkovací jazyk – eXtensible Markup Language

SEZNAM PŘÍLOH

A PŘÍLOHY

A.1 Diagram tříd



Obr. A.1: Diagram tříd

A.2 Obsah přiloženého CD

- Bakalářská práce ve formátu pdf
- Licenční smlouva ve formátu pdf
- Soubor s metadaty
- adresář Hlavní program: zdrojové kódy hlavního programu, spustitelný soubor Project1.exe
- adresář knihovna: zdrojové kódy knihovny dll