Programovanie prídavných modulov pre Engineering Base

Programming extending modules for Engineering Base

Bc. Jaroslav Coufalík

Diplomová práca
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

nascannované zadání s. 1

nascannované zadání s. 2

ABSTRAKT

Diplomová práca je zameraná na tvorbu výukovej demonštrácie programovania prídavných modulov pre aplikáciu EngineeringBase. Cieľom práce je vytvorenie výukového materiálu, podľa ktorého by mal byt užívateľ pochopiť základy objektového modelu aplikačného rozhrania Engineering Base a mal by byť schopný vytvorenia jednoduchých makier.

Výukový materiál je spracovaný vo forme webovej prezentácie, ktorá je doplnená o CD zo zdrojovými kódmi a súbormi. Všetky príklady v sú spustiteľné a plno funkčné. Na príkladoch je vysvetlený postup vytvárania jednotlivých makier od jednoduchších po zložitejšie.

Kľúčové slova: Engineering Base, Visual Basic for Application, makro, CBE

ABSTRACT

The main purpose of the thesis was the creation of educational demonstration of how to programme additional modules for Engineering Base aplication. The aim of the work is to create educational material which will learn users the base of object model of Engineering Base application interface and user will be able to create simple wizards.

The educational material is compiled in the form of a web presentation which is supplemented with a CD with source codes and files. All examples are executable and fully working. Many examples explain the heart of programming additional modules from the easiest to the most complicated tasks.

Keywords: Engineering Base, Visual Basic for Application, makro, CBE

Poďakovanie

Chcel by som sa poďakovať Ing. Petrovi Šilhavému Ph.D., Ing. Otakarovi Milinkovi, Ing. Davidovi Masařovi za konzultácie, odborné rady a pomoc pri spracovávaní mojej diplomovej práce. Taktiež by som rád poďakoval rodičom za podporu počas môjho štúdia.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

V dnešnej dobe existuje veľa aplikácií ktoré umožňujú rozšírenie pomocou prídavých modulou. Či už sú to aplikáce veľkých firiem ako napríklad *Microsoft* so svojím kancelárskym balíkom *Microsof Office*, ktorý umožňuje vytváranie prídavných makier alebo menšie firmy ako napríklad *Aucotec* so svojou aplikáciu *Engineering Base*. Snaha zjednodušiť užívateľovi prácu a umožniť mu prispôsobiť aplikáciu jeho zvykom a nárokom je v dnešnej dobe už takmer samozrejmosťou. Príkladom môžu byť dnešné webové prehliadače *Mozilla Firefox* a *Google Chrome*, do ktorých si užívatelia môžu písať vlastné skripty, ktoré im zrýchlia a uľahčia surfovanie webu.

Písaním rozsiahlejších prídavných modulov pre technické aplikácie sa zaoberajú špecializované firmy, ktoré sa vývojom zaoberajú niekoľko rokov a sú zárukou spoľahlivosti vytvorených makier. Vytváranie makier je však dostupné aj pre začínajúce firmy a jednotlivcov, poprípade nadšencov. A práve tým je táto práca určená.

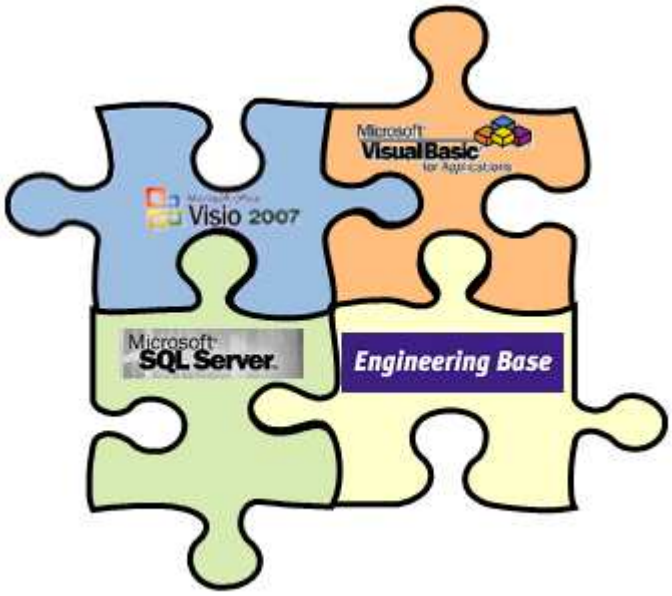
Táto diplomová práca sa teda zaoberá výukou vytvárania prídavných modulov. Programovanie je vysvetlené na množstve krátkych programov, ktoré sa zameriavajú na daný problém s postupnou zložitosťou problému a hlbším záberom do objektového modelu aplikačného rozhrania *Engineering Base*.

I. TEORETICKÁ ČASŤ

1 CHARAKTERISTIKA PRODUKTU ENGINEERING BASE

1.1 Základný popis

Slúži na tvorbu výkresovej dokumentácie v elektroprojekcii. Software je stavebnicovej koncepcie a je zostavený výhradne z produktov spoločnosti Microsoft. Spája software Microsoft Office Visio, pre jednoduché vytváranie rôznych schém s Microsoft SQL serverom, ktorý slúži ako úložisko pre dáta, tým umožňuje databázové spracovanie rozsiahlych projektov v rôznych grafických formátoch. K tomu všetkému ešte pripája Microsoft Visual Basic for Application pre tvorbu makier zefektívňujúcich prácu.



Obrázok 1. Stavebnicové zloženie aplikácie

Patrí do novej generácie elektro-projekčných nástrojov, ktoré svojou koncepciou zapadajú do skupiny CBE *(Computer Base Engineering)* systémov. [1][2]

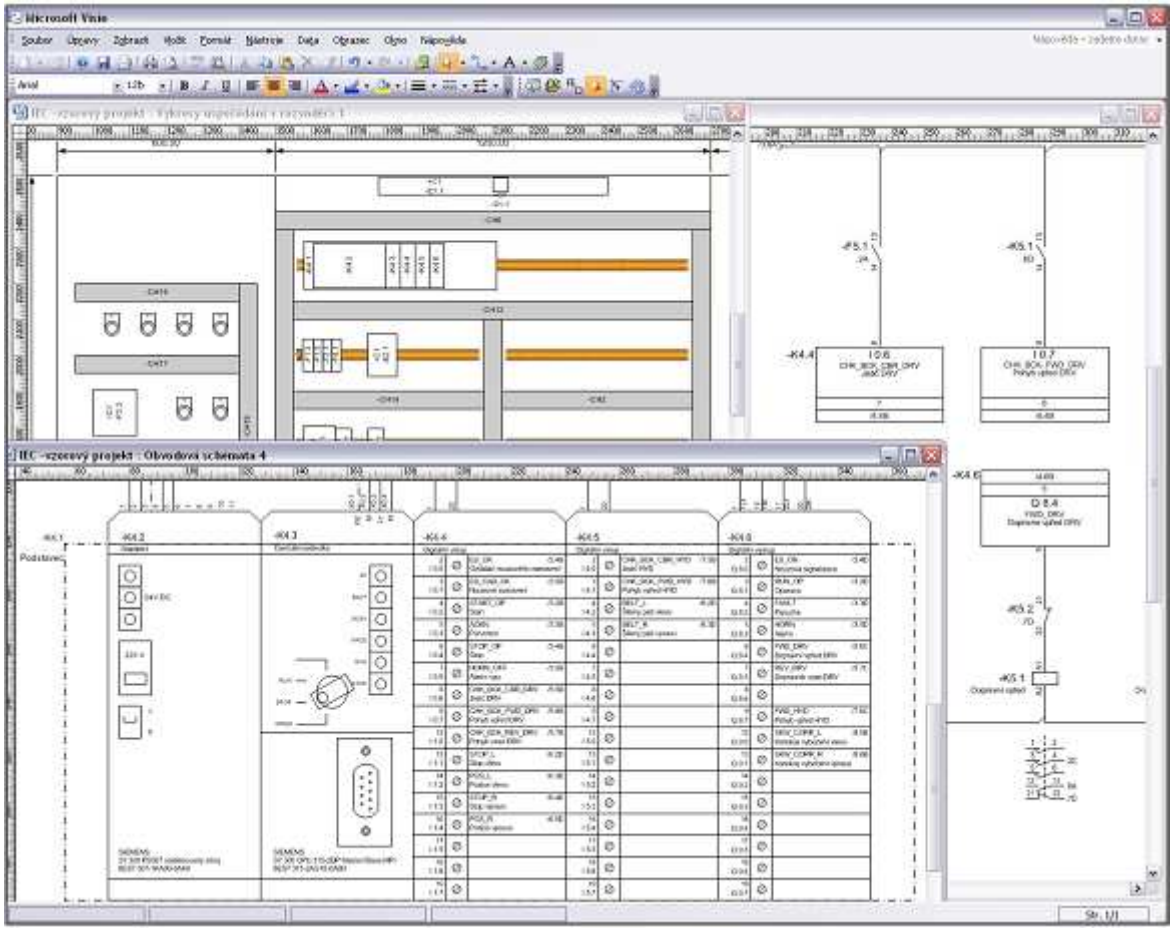
1.1.1 Computer Base Engineering

CBE je nový prístup spracovania elektronických projektov. Dáta sú uložené v hlavnej databáze, kde sa vytvára virtuálny model projektu. Pri práci s dátami môžu byť rovnocenne použité grafické alebo alfanumerické prístupy. Grafickými prístupmi rozumieme napríklad editáciu výkresu a alfanumerickými napríklad prístupy tabuľkové. Všetky dáta sú rovnocenne spracovávané a uložené v grafickej, tabuľkovej a stromovej štruktúre. [3]

1.2 Jednotlivé časti stavebnicovej štruktúry aplikácie

1.2.1 Microsoft Office Visio

Je to profesionálny nástroj na kreslenie schém, pôvodom od spoločnosti Visio Corporation, ktorú neskôr v roku 2000 kúpila spoločnosť Microsoft. Je súčasťou vyšších verzií balíku Microsoft Office alebo je dostupná ako samostatná aplikácia. [4]



Obrázok 2. Ukážka aplikácie Microsoft Office Visio

Microsoft Visio slúži na vizuálnu dokumentáciu, na návrh a porozumenie obchodných procesov a systémov pomocou veľkého množstva diagramov. Príkladom môže byť vývojový diagram procesov, sieťových diagramov, diagram pracovných postupov, databázových modelov a softwarových schém. Tieto diagramy môžu byť prepojené s podkladovými dátami, čím sa stanú užitočnejšími a získajú komplexný pohľad na proces alebo systém. [5]

Natívne súborové formáty:

- *VSD* – Diagram
- *VSS* – Stencil
- *VST* – Template
- *VDX* – Visio XML Diagram
- *VSX* – Visio XML Stencil
- *VTX* – Visio XML Template
- *VSL* – Visio add-on

[6]

Aplikácia je dostupná v dvoch verziách. *"Office Visio Standard"* a *"Office Visio Proffesional."* Tie sa líšia v množstve grafov, ktoré umožňujú vytvárať.

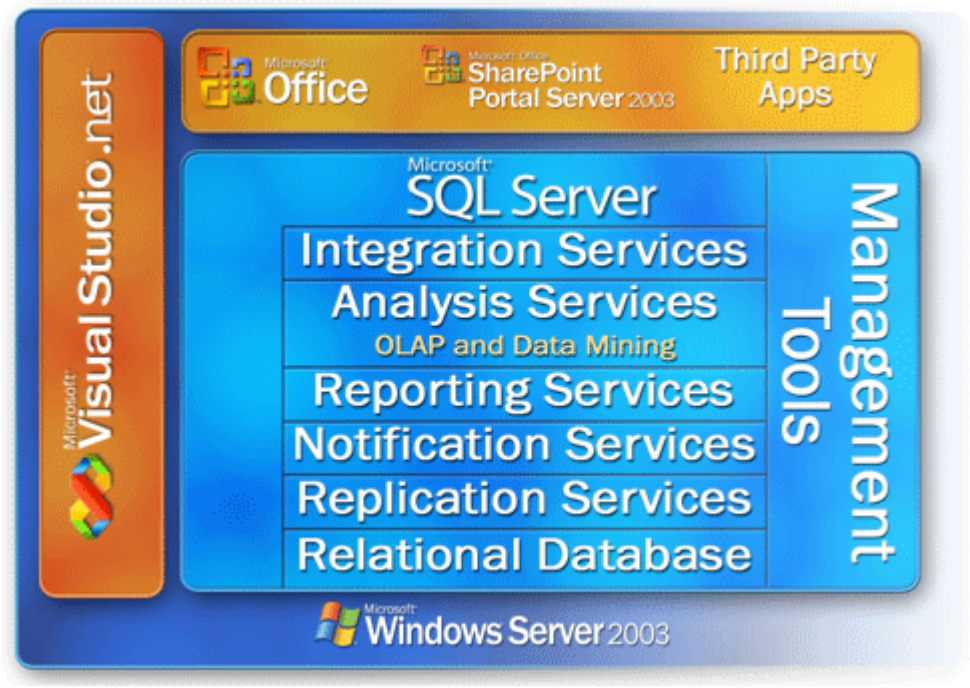
Vybrané vlastnosť a možnosti softwaru:

- *Jednoduché znázorňovanie procesov, systémov a informácií.* Obsahuje veľké množstvo šablón a symbolov. Pomocou "Sprievodcu výberom dát" je možno prepojiť diagramy s jedným alebo viacerými zdrojmi dát, napríklad s tabuľkami Microsoft Office Excel alebo databázami Microsoft Office Access.
- *Analýza komplexných informácií za účelom porozumenia.* Možnosť rozšírenia na vytváranie analýz, prechod k obchodným dátam a vytváranie viacero verzií týchto dát za účelom získania lepšieho prehľadu o podniku.
- *Funkčnosť Data Link a Data Graphic* integruje dáta s diagramami rôznorodých zdrojov pre komplexné zobrazenie vizuálnych, textových a numerických informácií za účelom poskytnutia vizuálnej súvislosti pre dáta a vytvorenia kompletného obrazu o systéme, zdrojoch a procesoch. Zobrazuje dátové polia ako poznámky vedľa obrazcov alebo ako políčka nad obrazcami alebo ako symboly, ktoré reprezentujú dáta.
- *PivotDiagram šablóny* umožňuje prechádzať hierarchicky dáta. Identifikuje kľúčové správy v dátach a vizuálne prepojiť s inými.

- *Šablóny vývoju softwaru.* Diagram vlastných softwarových riešení pomocou Microsoft Windows XP User Interface, UML šablóny, a mnoho ďalších.
- *Vytváranie rôznych technických diagramov, inžinierskych schém, databázových modelov a ďalších*
- *Information Technology Infrastructure Library (ITIL) template.* Diagram IT služieb a procesov, ktoré sú v súlade s ITIL normami.
- *Šablóny sieťových diagramov.* Diagram logických a fyzických diagramov, a diagramy *rack.*
- *Šablóny vývoju webu.* Vytvorenie mapy stránok existujúcich webov pomocou šablón a preddefinovaných obrazcov.
- *Rozširovanie aplikácie pomocou programovania.* Aplikáciu možno rozšíriť pomocou programovania alebo integráciou do iných aplikácií, aby vyhovovala situáciám špecifickým pre dané odvetvie.Taktiež je tu možnosť vyvinúť vlastné riešenia a obrazce. Zostava SDK zahrnuje podporu veľa rôznych vývojových jazykov, vrátane jazykov Microsoft Visual Basic, Visual Basic .NET, Microsoft Visual C# .NET a Microsoft Visual C++. Nástroj *„Visio Drawing Control"* umožňuje tvorbu vlastných riešení pre pripojenie dát, vloženie a programovanie prostredia výkresov aplikácie do vlastných aplikácií. [5] [7]

1.2.2 Microsoft SQL Server

Je to relačný databázový systém spoločnosti Microsoft. Medzi jeho hlavné dotazovacie jazyky patria SQL *(Structured Query Language)* a TSQL *(Transact-SQL).* [8]



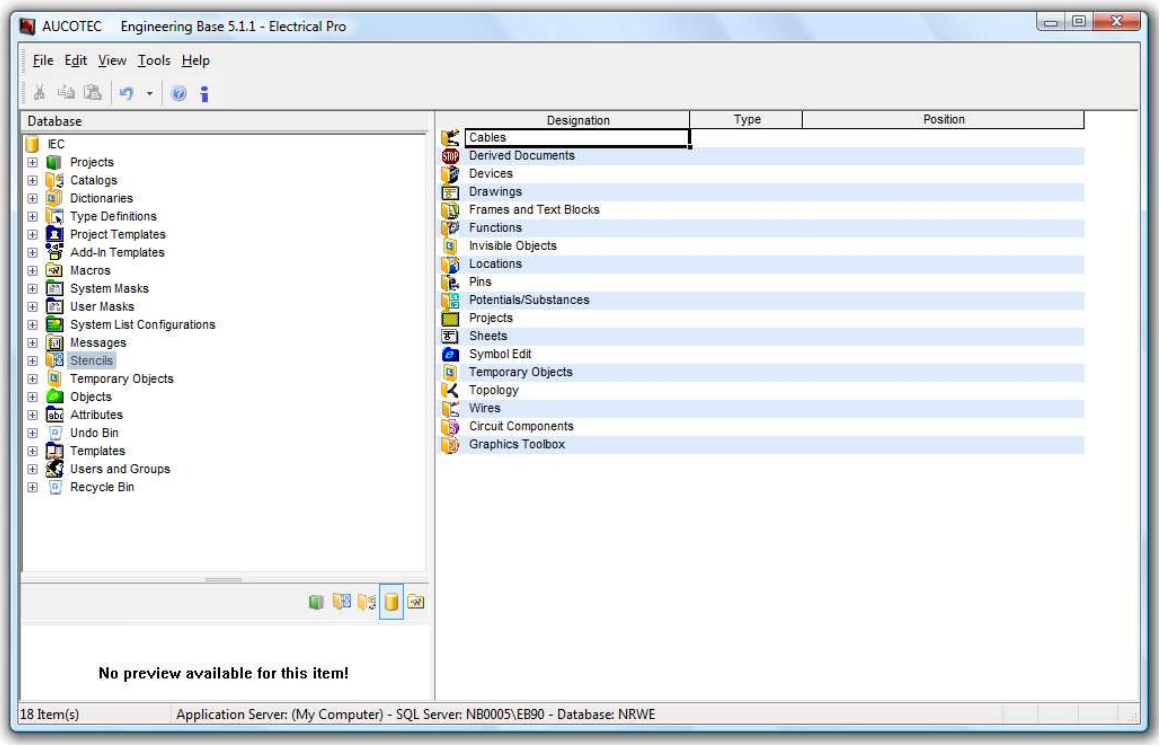
Obrázok 3. Rozloženie dátovej platformy serveru SQL Server 2005

Dátová platforma SQL Serveru obsahuje tieto nástroje:

- *Relačné databázy.* Zabezpečený škálovateľný a vysoko dostupný databázový stroj s podporou štruktúrovaných a neštruktúrovaných dát (XML).
- *Replication Services.* Replikácia dát pre aplikácie zapracovávajúce distribuované alebo mobilné dáta.
- *Notification Services.* Funkcie zasielania upozornenia pre vývoj a nasadenie škálovateľných aplikácií, ktoré môžu na pripojené a mobilné zariadenia posielať individuálnym požiadavkám prispôsobené aktuálne informácie.
- *Integration Services.* Funkcie extrakcie, transformácie a načítania dát (ETL) pre dátové sklady a integrácií dát v celom podniku.
- *Analysis Services.* Funkcie OLAP *(Online Analytical Processing)* pre rýchlu a pokročilú analýzu veľkých dátových sád s využitím viacdimenzionálnych úložísk.
- *Reporting Services.* Komplexné riešenie pre vytváranie, správu a zasielanie klasických papierových alebo interaktívnych webových sústav.

- *Nástroje pre správu.* Integrované nástroje pre správu a ladenie databáz a umožňuje integráciu s nástrojmi, ako napríklad MOM *(Microsoft Operations Manager)* a SMS *(Microsoft System Management Server)*. Obsahuje tiež integrovanú podporu webových služieb, ktoré zaisťujú vzájomnú funkčnú spoluprácu s ďalšími aplikáciami a platformami.
- *Nástroje pre vývojárov.* Integrované nástroje pre vývojárov určené pre databázový stroj, extrakciu, transformáciu a načítanie dát (ETL), funkcie OLAP a vytváranie zostáv, ktoré sú úzko integrované zo sadou Microsoft Visual Studio. [9]

1.2.3 Prieskumník



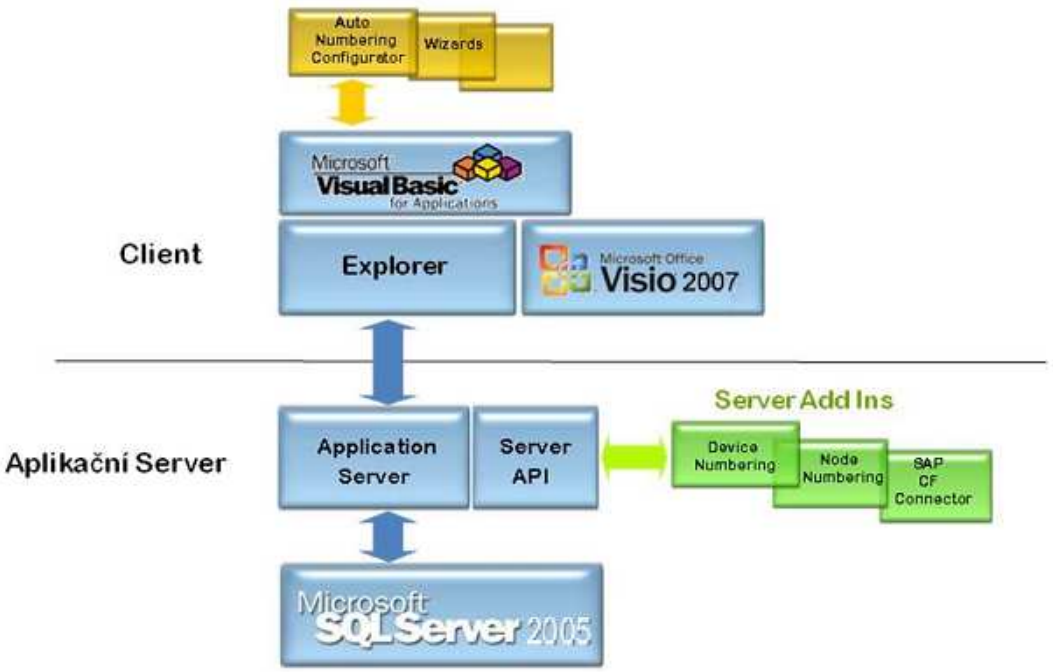
Obrázok 4. Klient – Hlavné okno aplikácie Engineering Base

Slúži na ovládanie celého systému. Skladá sa z viacerých pracovných oblastí, a súčasťou sú aj náhľadové okná, ktoré uľahčujú orientáciu pri prehľadávaní potrebných dát, alebo výkresov. Jednotlivé typy funkcií a dát sú odlíšené grafickými zástupcami kvôli zlepšeniu prehľadnosti.

1.3 Systémová štruktúra

Architektúra systému je objektovo orientovaná a skladá sa z troch hlavných vrstiev:

- *Microsoft SQL server.* Zaisťuje základu databázu aplikácie.
- *Aplikačný server.* Sú tu umiestnené projektové objekty a ich vzájomné väzby. Obsah tejto vrstvy je obsluhe sprístupnený cez okno Prieskumníka doplneným oknom grafického náhľadu a cez tabuľkové pole systému.
- *Klient.* Slúži pre spracovanie a editáciu dát. V tejto vrstve sa nachádza aj prístup pre užívateľské aplikácie spracované vo Visual Basicu a špeciálne zákaznícke riešenia.



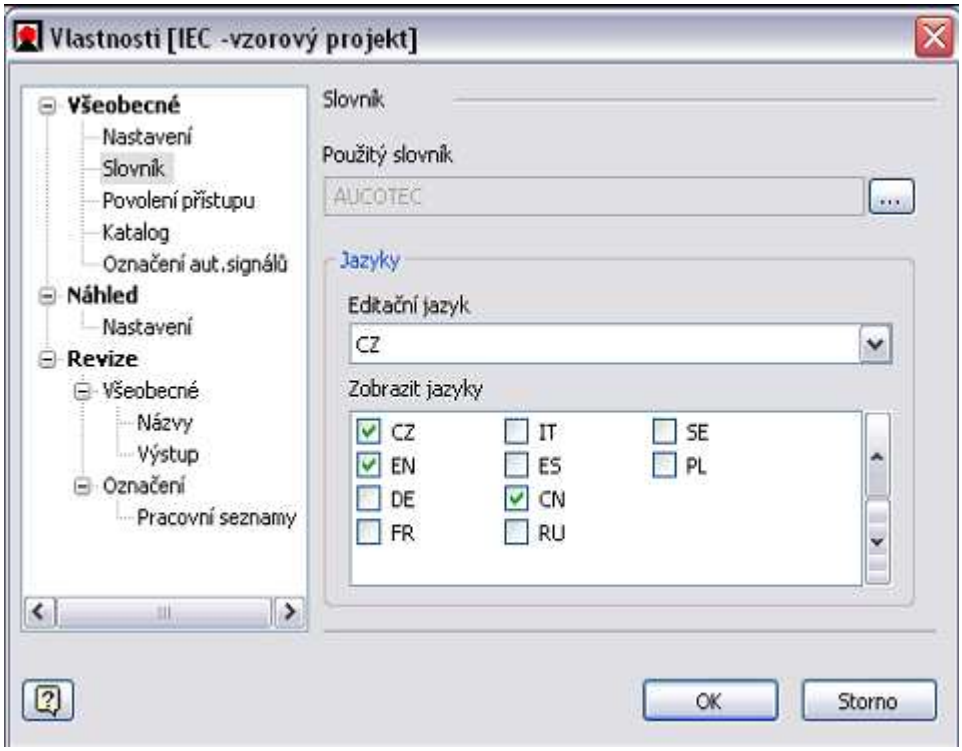
Obrázok 5. Systémová štruktúra aplikácie

Dátový model vrátane väzieb je uložený v databáze a je vytváraný a editovaný z Engineering Base. Všetky uložené objekty obsahujú základné atribúty, ktoré môžu byť rozšírené o ďalšie oborové položky. Ku každému objektu je možné pripojiť ľubovoľný počet grafických reprezentácií a doplňujúcich dokumentov. [10]

1.4 Vlastnosti a možnosti aplikácie Engineering Base

1.4.1 Slovníky

Engineering Base je určený pre medzinárodné použitie a preto obsahuje funkcie pre voľbu jazykovej verzie projektu. Obsahuje databázu textov rôznych jazykov a v prípade spracovávania viacjazykovej dokumentácie je možné zobraziť súčasne až 7 jazykových verzií. Textové znaky sú kódované pomocou Unicode, takže je popri latinštiny k dispozícii tiež azbuka, čínske znaky, atd.



Obrázok 6. Nastavenie slovníkov

1.4.2 Grafické dokumenty a schémy

V prípade CBE systémov sú schémy zapojenia len grafickými reprezentáciami informácií – modelu, ktorý je uložený v hlavnej projektovej databáze. Nie je preto potrebné popredu skladať a kresliť schémy zapojenia k tomu, aby sme mohli získať napríklad zoznamy prístrojov, atd. Schémy sú vytvárané skladaním a prepojovaním elektrických symbolov, ktoré je možné pomocou funkcie *„Drag and Drop“* priraďovať z dátového stromu objektovej databáze konkrétnemu prístroju. Visio je pre potreby priameho a rýchleho kreslenia doplnené o nové funkcie ako je napríklad *Auto-connect*, *Stay-*

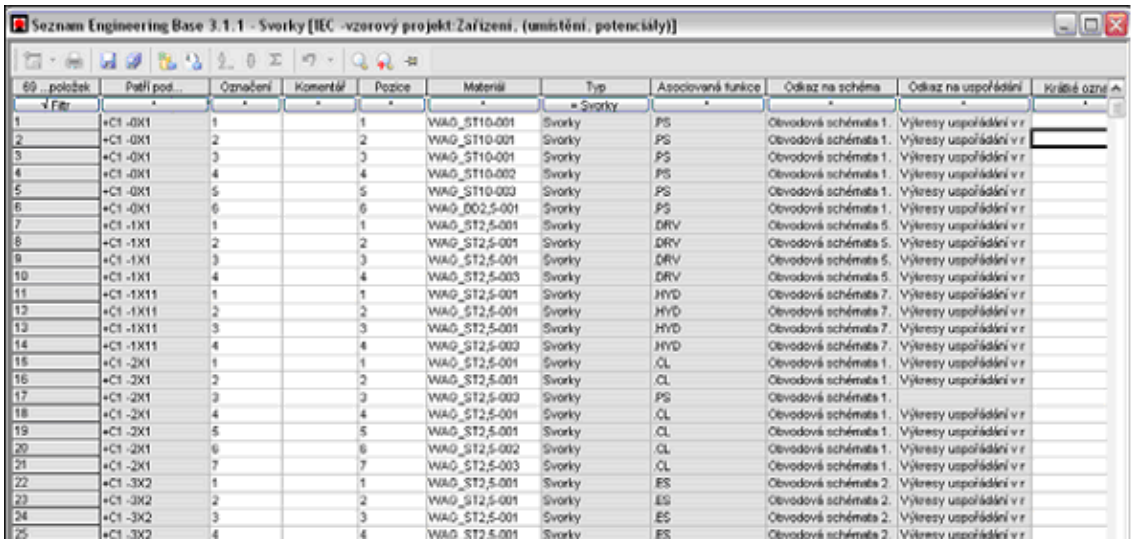
connect Mode atd. K dispozícii sú dve grafiky. Jedna pre kreslenie vodičových spojov a druhá voľna pre doplňujúce grafické informácie.

1.4.3 Zoznamy spojov a predmetov

Dôležitým dokumentom každého projektu sú zoznamy použitých predmetov. V databáze projektového modelu sú uložené všetky prvky vložené z prostredia Prieskumníka, v tabuľkovom poľu, alebo vo výkrese. Výhodou je zachovávanie on-line väzby medzi tabuľkou vo výkrese a záznamom v modelovej databáze. V tabuľkovom poli spojov a signálov sú uvedené všetky informácie o vodiči dátového modelu projektovaného zariadenia. Dáta sú graficky znázornené v obvodových schémach ako definované vodiče, zbernice, a pod. Obsah poľa je možné meniť alebo zotriediť podľa požadovaných parametrov.

1.4.4 Svorkovnice

Zostava svorkovníc obsahuje informácie o typu svoriek a ich interným a externým zapojením. Ďalej to môžu byť informácie o polohe svorky v schémach zapojení, a pod. V Engineering Base sú v svorkové dáta uložené v samostatnej časti databáze. Obsah svorkového záznamu z databáze je možné získať pomocou svorkového stromu v Prieskumníku, v tabuľkovom poli alebo v grafickej forme.

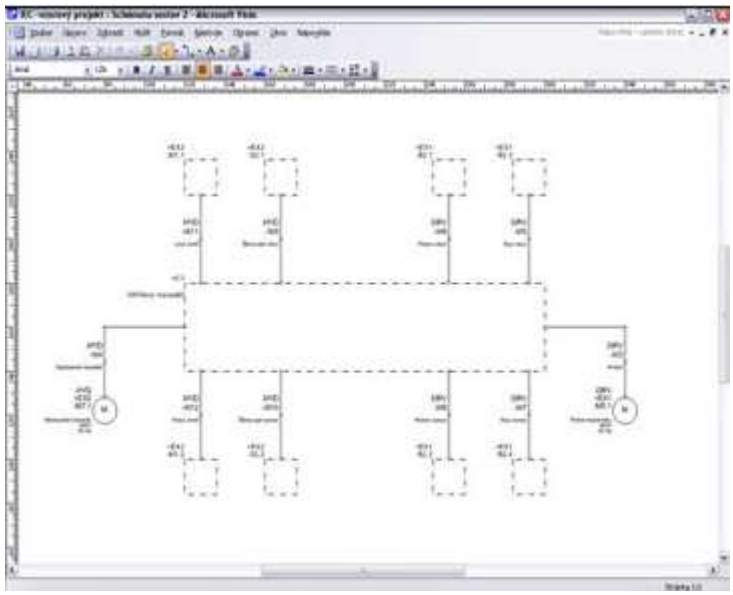


Obrázok 7. Tabuľkové pole svorkovnice

1.4.5 Kabeláže

Pre potreby spracovania dokumentácie ku kabelážnemu modelu projektového zariadenia je Engineering Base vybavený niekoľkými funkciami:

- *Funkcia Zostava káblov.* V databáze Engineering Base je niekoľko základných silových káblov. Táto funkcia slúži pre potreby rozšírenia káblovej databáze a k definícii ďalších typov káblov. Umožňuje prehľadným spôsobom definovať počet káblových žíl, ich farby, prierez a označenie.
- *Zostava káblových trias.* Slúži na prehľadné vyjadrenie jednotlivých priepojov.
- *Tabuľkové pole.* Káble znázornené v zostavách alebo definované v Prieskumníku sú systémom zaznamenávané v káblovej tabuľke polí. Tá umožňuje ľubovoľné triedenie káblových záznamov a filtrovanie. Ďalej umožňuje generovať grafické výstupy – káblové zoznamy alebo káblové XLS súbory.



Obrázok 8. Zostava zapojenia káblov

1.4.6 ADS

Ide o zákaznícku službu AUCOTEC Data Service. Cieľom služby je sprístupniť prostredníctvom internetu užívateľovi systému certifikované prístrojové databáze a umožniť zostavenie alebo rozšírenie pracovnej databáze projekčného nástroja.

1.4.7 Správa verzií

Engineering Base obsahuje dokumentovo orientovanú správu verzií. Stávajúci stav zapojenia vo výkresoch je možné uložiť ako „nultú verziu“. Nasledujúce verzie je možné uložiť až po zmene výkresu oproti poslednej vytvorenej verzii. Na jednotlivé verzie možno kedykoľvek nahliadnuť, vytlačiť ich alebo spraviť vzájomné porovnanie. [10]

1.5 Systémové verzie aplikácie Engineering Base

Aplikácia Engineergin Base má niekoľko systémových verzií, ktoré sa líšia niekoľkými vlastnosťami a sú určené na spracovanie špecifickej práce. Užívateľ si môže vybrať a nainštalovať verziu, ktorá mu vyhovuje a je prispôsobená odvetviu, v ktorej sa pohybuje.

1.5.1 Engineering Base Fluid

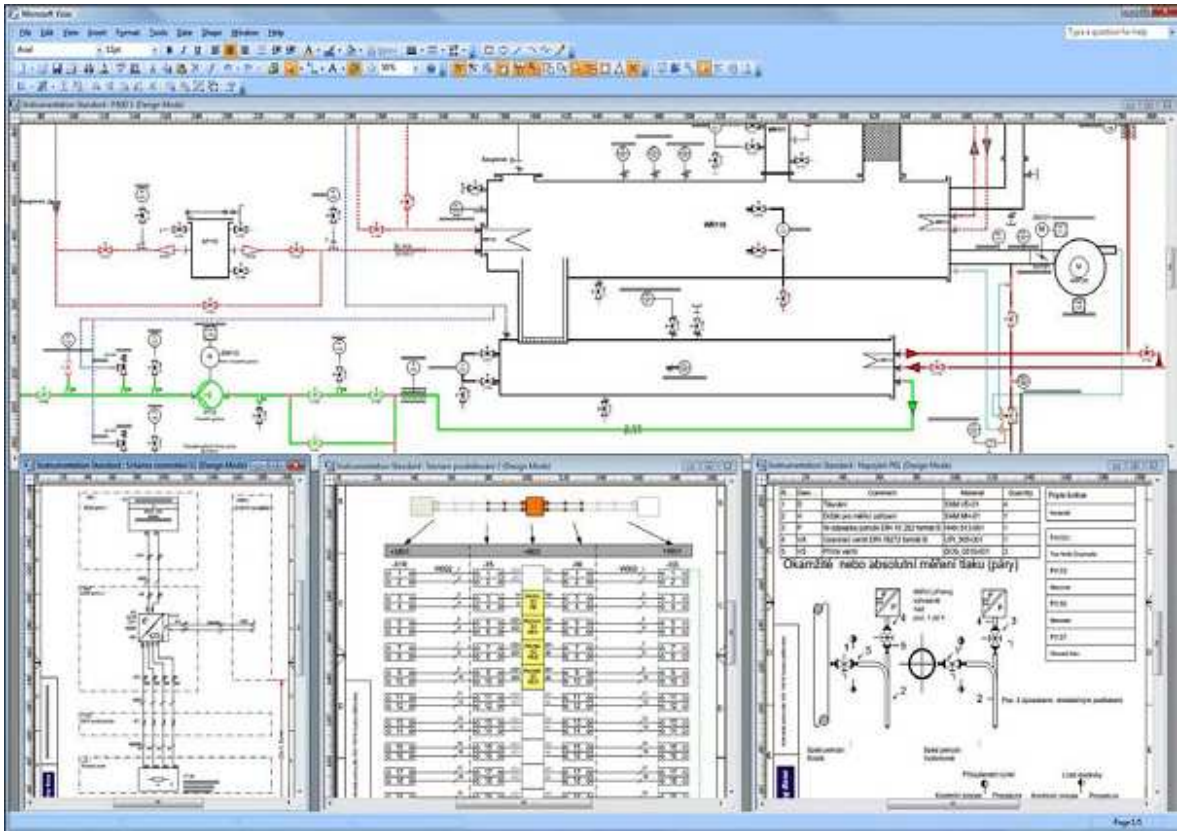
Systémová verzia Fluid je určená pre zostavovanie dokumentácie s hydraulickými a pneumatickými prvkami. Sú tu k dispozícii oborové symboly a objekty pre hydraulické a pneumatické schémy.

1.5.2 Engineering Base Cable

Systémová verzia Engineering Base Cable je určená pre projekčnú podporu elektroinštalácií, ktoré sú realizované prostredníctvom kablových zväzkov. Ide o dopravné prostriedky, vojenskú techniku, bielu techniku atd.

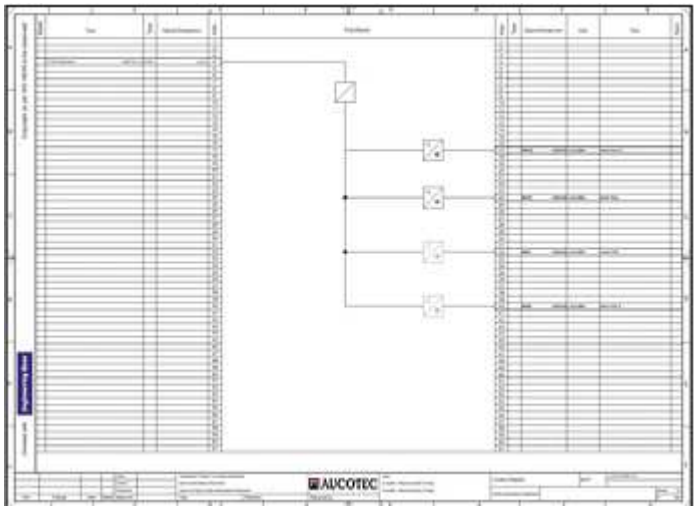
1.5.3 Engineering Base Instrumentation

Verzia je určená pre riešenia náročných projektov z oblasti inštrumentácie. Pre potreby spracovania P&I schém, Basic designu a náročného Detail designu sú pripravené tri funkčné varianty systémového riešenia.



Obrázok 9. Spracovanie MaR projektu v zostave Engineering Base – Instrumentation

Jednotná databáza a on-line dátové previazanie rôznych druhov MaR dokumentov uľahčuje a urýchľuje spracovávanie rozsiahlych projektov.

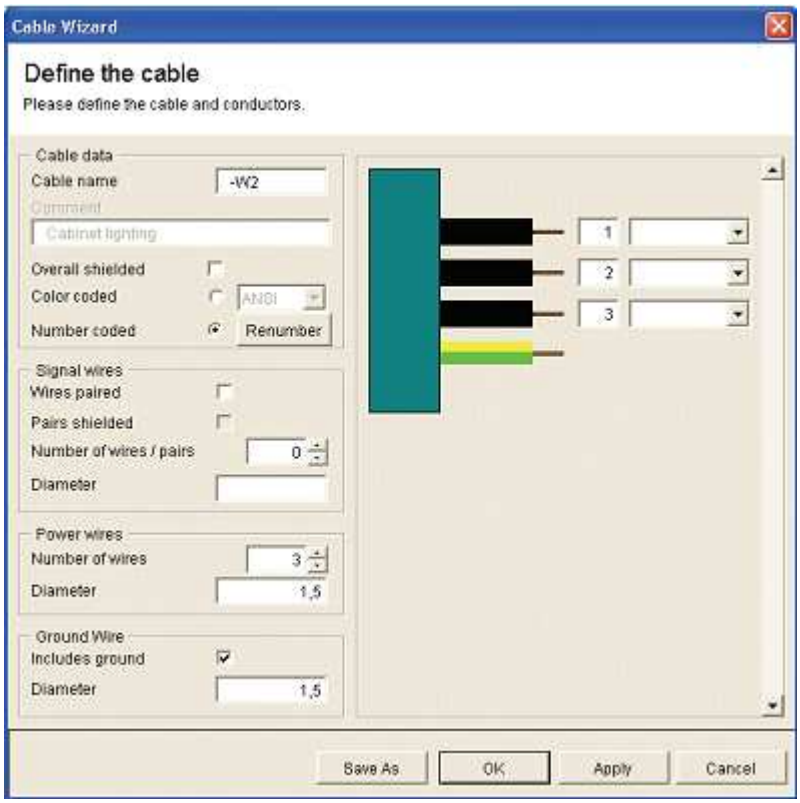


Obrázok 10. Príklad Mar dokumentu pre zostavovanie programových algoritmov

2 MOŽNOSTI TVORBY PRÍDAVNÝCH MODULOV

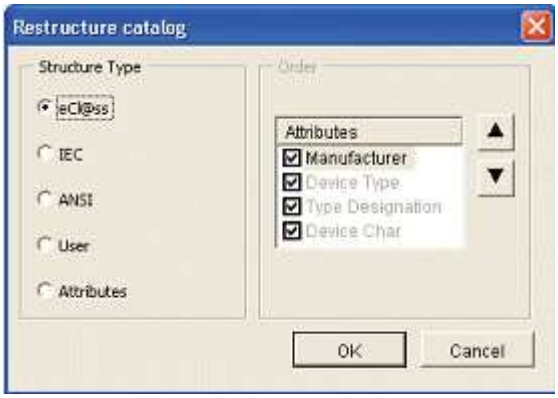
Funkcie aplikácie Engineering Base je možné ďalej užívateľom zdokonaľovať, prípadne zjednodušovať pomocou makier. Macro je vlastne séria príkazov a funkcií, ktoré sú uložené v module a môžu byť kedykoľvek spustené v prípade potreby. Na vytváranie makier je k dispozícii programovací jazyk *Visual Basic for Application*. Súčasťou dodávky systému Engineering Base je tiež niekoľko makier, ktoré môžu slúžiť zároveň aj ako príklady aplikácie programovacieho jazyka *Visual Basic for Application*:

- *Makro Cable Wizard.* Ide o VBA script, ktorý sa používa na vytváranie a úpravu káblov a ich drôtov.



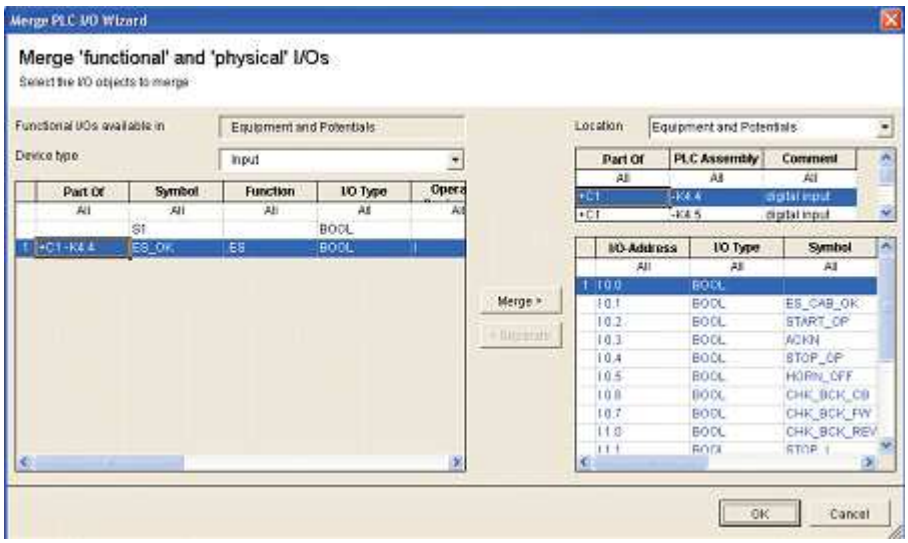
Obrázok 11. Ukážka makra Cable Wizard

- *Catalog Structure.* Používa sa na reštrukturalizáciu komponentov v katalógu podľa jedného alebo viacerých atributov.



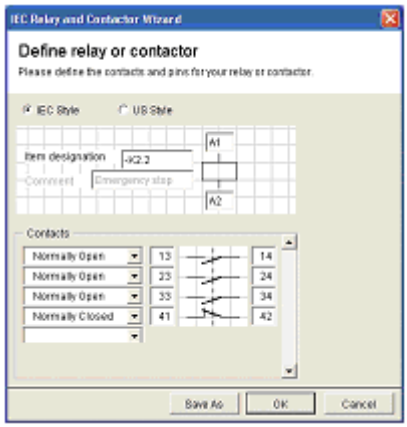
Obrázok 12. Makro Catalog Structure

- *Merge I/O Wizard.* Funkčné vstupy a výstupy, ktoré sú vytvorené v priebehu návrhu dizajnu, môžu byť spojené so vstupmi a výstupmi fyzickými z „hardware“ návrhu použitím tohto makra.



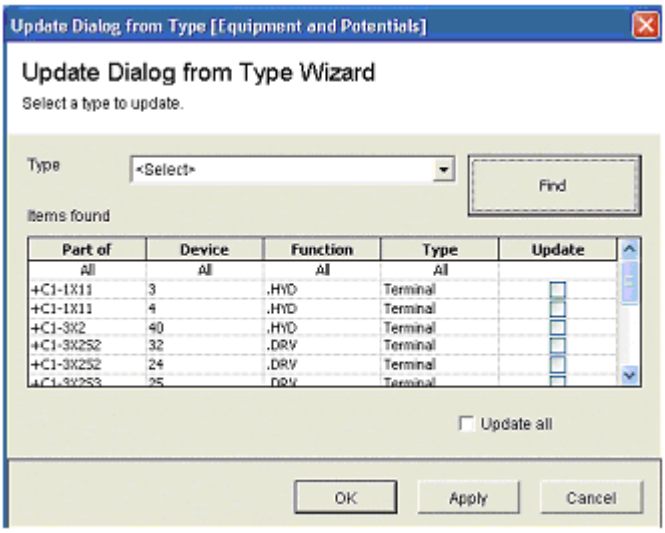
Obrázok 13. Makro Merge I/O Wizard

- *Multi Terminal Block Diagram Wizard.* S týmto markom sa môžu všetky svorkovnicové diagramy pre označené objekty vytvoriť v rovnaký čas.
- *Quality Management Tool.* Toto makro zaisťuje správny štandard aplikácie pre zvýšenie kvality vytvorenej dokumentácie. Veľký počet nastavení je pripravený prispôsobiť implementáciu individuálnym požiadavkám.
- *Relay and Contactor Wizard.* Umožňuje jednoduché a rýchle vytvorenie a kompletnú modifikáciu relé a stýkačov so všetkými kontaktmi a svorkami.



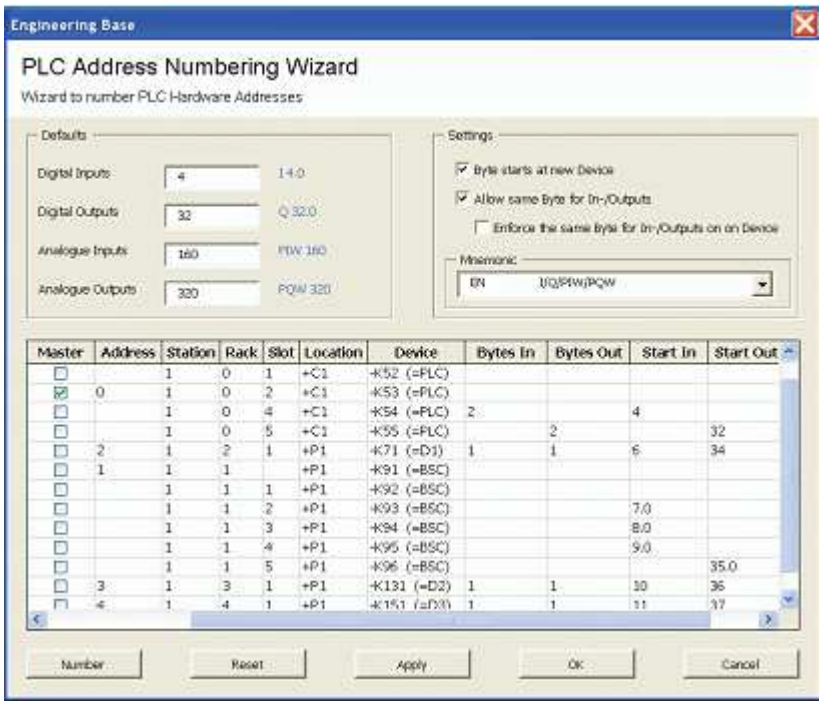
Obrázok 14. Ukážka marka Relay and Contactor Wizard

- *Terminal Block Wizard.* Môže byť použité na vytvorenie a modifikáciu terminálových blokov s ich terminálmi a segmentmi alebo soketmi.
- *Typical Select and Typical Copy.* Pre každú funkciu, typical obsahujúci výkresy a zariadenia môže byť označený pomocou makra *Typical Select*. Potom pomocou makra *Typical Copy* sú tieto funkcie skopírované so všetkými ich zariadeniami a výkresmi.
- *Update Dialog from Type Wizard.* Používa sa pre obnovenie dialógov označených objektov v projektu alebo katalógu.



Obrázok 15. Makro Update Dialog from Type Wizard

- *Update from Catalog Wizard.* Je to makro, ktoré slúži na obnovenie informácií pre vybrané zariadenia v projektu.
- *PLC Address Wizard.* Automaticky vytvorí vstupné a výstupné adresy.



Obrázok 16. PLC Address Wizard

- *PROFIBUS Configurator*. Môže byť použitý na definovanie sekvencie pre káblové zariadenie pripojené na PROFIBUS zariadenie.
- *Price Calculation.* Zaznamenáva ceny jednotlivých zariadení pre vypočítanie celkovej ceny.
- *Swap Labels.* Názvy zariadení, lokácií a funkcií môžu byť modifikované z Excelu.
- *Replace Text with Reference to Dictionary.* Nahradzuje vybraný text textom zo slovníku. Využíva sa pre dokumentáciu vo viacerých jazykoch. [11]

2.1 Visual Basic

Visual Basic je jazyk tretej generácie, čo znamená, že je užívateľsky veľmi prívetivý. Vyvinula ho spoločnosť *Microsoft*. *Visual Basic* je odvodený z jazyka *BASIC* a umožňuje efektívny vývoj *GUI (Grafické užívateľské rozhranie)* aplikácií, prístupu do

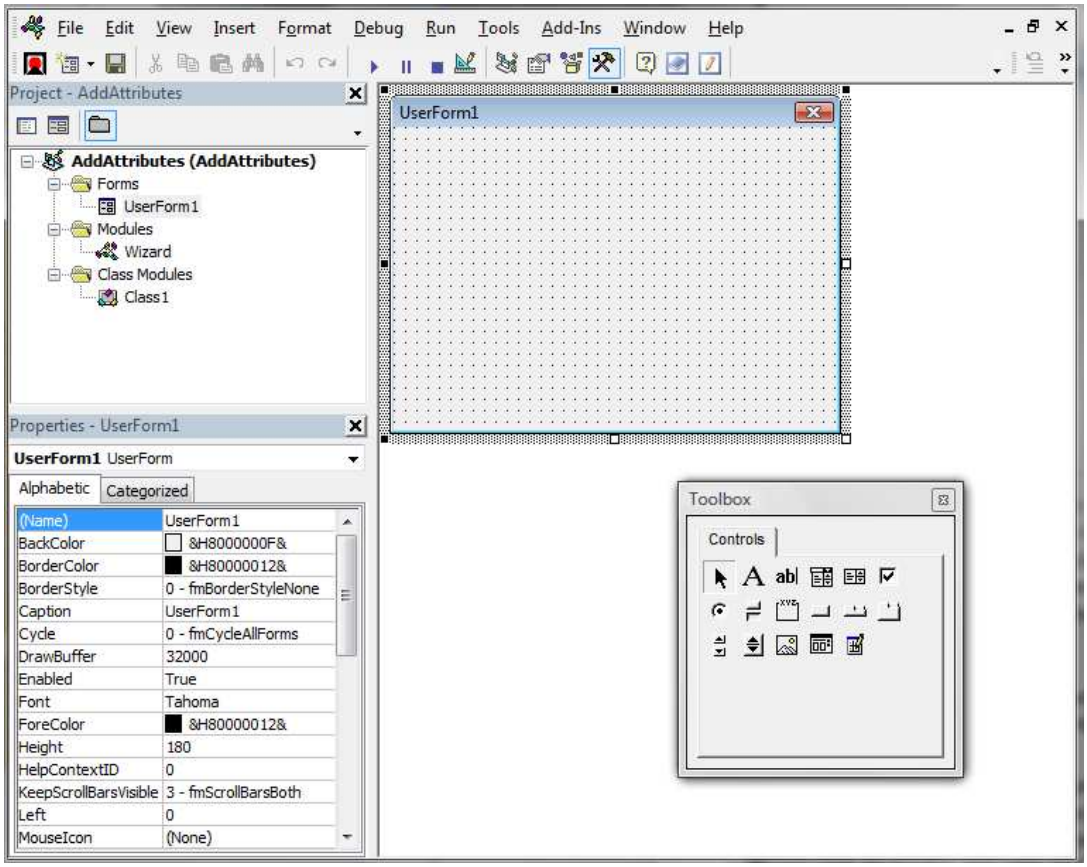
databázy použitím *DAO (Data Access Objects), RDO (Remote Data Objects), ADO (ActiveX Data Objects)* a tvorbu *ActiveX* ovládacích prvkov a objektov. Je to udalosťami riadený programovací jazyk. Programy písané v tomto jazyku môžu používať *Windows API*, ale k tomu potrebujú deklaráciu externých funkcií. Posledná verzia je *Visual Basic 6* z roku 1998. *Microsoft* v marci 2008 ukončil podporu *Visual Basicu* a jeho nástupcom sa stal *Visual Basic .NET*. Visual Basic bol navrhnutý s ohľadom na rýchle pochopenie a naučenie a používanie začiatočnými programátormi. Umožňuje však nielen vytváranie jednoduchých programov ale je predurčený aj pre vývoj komplených aplikácií. Programovanie vo *VB* je kombinácia vizuálneho rozmiestnenia komponentov a riadiacich prvkov na formulári, špecifikácii vlastností a správaní týchto komponentov, a písaní kódu pre zväčšenie a spresnenie funkcionality.

```
1 Private Sub Form_Load()
2     ' execute a simple message box
3     MsgBox "Hello, World!"
4 End Sub
```

[12]

2.2 Visual Basic for Application

Ako vyplýva z jeho názvu, ide o podmnožinu programovacieho jazyka *Visual Basic*. *Visual Basic for Application* je programovací jazyk spoločnosti *Microsoft*. Medzi hlavné rozdiely patrí nemožnosť vytvoriť nezávislý „**.exe*“ program. Môže pritom používať *(Acitve/COM) DLL* knižnice, avšak vytvárať ich nedokáže. *VBA* je implementovaný vo viacerých produktov spoločnosti *Microsoft*, ako napríklad *Microsoft Office Excel*, *Microsoft Office Word*, *Microsoft Office Power Point*, *Microsoft Office Visio* a ďalších, pre programovanie pomocných makier. Pre ich vývoj sa používa prostredie *Microsoft Visual Basic Editor*. [13]



Obrázok 17. Editor jazyka Visual Basic for Application

2.3 Objektový model aplikačného rozhrania Engineering Base

Objektový model aplikačného rozhrania Engineering Base sa skladá z množstva tried z ktorých asi najdôležitejšie sú triedy *Application* a *ObjectItem*. Vo väčšine makier sa pomocou triedy *Application* získa aktuálne vybraný projekt alebo prvok v *Engineering Base* ktorý je typu *ObjectItem* s ktorým sa potom ďalej pracuje.

Obrázok 18. Základný strom objektového modelu aplikačného rozhrania Engineering Base

2.4 Trieda Application

Trieda *Application* reprezentuje aplikáciu *Engineering Base*. Používa sa na získanie informácií o aplikácii. Zistíme pomocou nej napríklad používanú verziu *Engineering Base*, aktuálne používaný jazyk, a iné. Neslúži však len na zisťovanie vlastností spustenej aplikácie. Slúži takisto napríklad na zistenie práve aktívneho projektu, vráti ukazatel na práve označený prvok v strome objektov Prieskumníka aplikácie.

Triedu *Application* by sme umožňuje niekoľko oblastí prístupu:

- *Informácie o aplikácii* - ako už bolo spomenuté, môžeme pomocou nej získať základné informácie o spustenej aplikácii.
- *Globálne metódy* – patrí sem napríklad metóda *WaitDialog*, ktorá slúži na zobrazenie okna s priebehom. Metóda *Settings* umožňuje čítanie a zapisovanie do

registrov, zistenie cestu k zložke s dočasnými súbormi aplikácie, vráti užívateľské meno užívateľa, jeho skrátenú verziu a iné.

- *Informácie o dianí vo vnútri aplikácii* – výraz *Application.ActiveProject* nám vráti ukazateľ na práve aktívny projekt na ktorom uživateľ aktuálne pracuje, teda má v Prieskumníku aktívny. Pomocou výrazu *Selection* sa pre zmenu vracia aktuálne označené objekty v Prieskumníku.

2.5 Trieda ObjectItem

Trieda *ObjectItem* reprezentuje prvok zo stromu *Prieskumníka* aplikácie *Engineering Base*. To znamená, že každá reprezentácia prvku, či už je to projekt, výkres, materiál alebo funkcia je odvodený zo základnej triedy *ObjectItem*.

Vybrané vlastnosti triedy:

- *Name* – vlastnosť ktorá v sebe nesie pomenovanie prvku, ktorý trieda reprezentuje.
- *FullName* – je pomenovanie prvku vrátane jeho nadradených prvkov zo stromu *EB*.
- *Kind* – každý objekt v rozhraní je určitého druhu. Pomocou vlastnosti *Kind* rozoznávame či ide napríklad o materiál, projekt, výkres atd.
- *TypeID* – na rozdiel od vlastnosti *Kind*, určuje bližšie o aký typ objektu ide.
- *Attributes* – vráti kolekciu vlastností prvku. V tejto kolekcii sa nachádzajú všetky vlastnosti prvku, ktoré sú danému typu objektu definované v databáze. Nachádzajú sa tu napríklad názov prvku, asociovaná funkcia, farba objektu, ktorý prvok reprezentuje, jeho výška a iné.

Vybrané metódy triedy:

- *Store* – po jej zavolaní sa zmeny prevedené nad objektom zapíšu do databáze.
- *Delete* – vymaže prvok z databáze.
- *CopyTo* – skopíruje prvok.

II. PRAKTICKÁ ČASŤ

3 CIEĽ DIPLOMOVEJ PRÁCE, PRIORITY JEJ VÝSLEDKU

Hlavnou úlohou tejto diplomovej práce je návrh série príkladov pre pochopenie a osvojenie programovania prídavných modulov pre aplikáciu Engineering Base. Príklady sú volené postupne podľa zložitosti problému a hĺbky zanorenia do objektového modelu rozhrania aplikácie.

Cieľom je zoznámenie užívateľa s možnosťou tvorby prídavných modulov, vysvetlením základných pojmov pre tento druh aplikácie a zoznámenie užívateľa s objektovým modelom Engineering Base. Táto práca predpokladá minimálne vedomosti užívateľa s programovaním vo Visual Basicu, prípadne v inom jazyku. Všetky príklady sú podrobne popísané, takže by ich mal zvládnuť aj začiatočník.

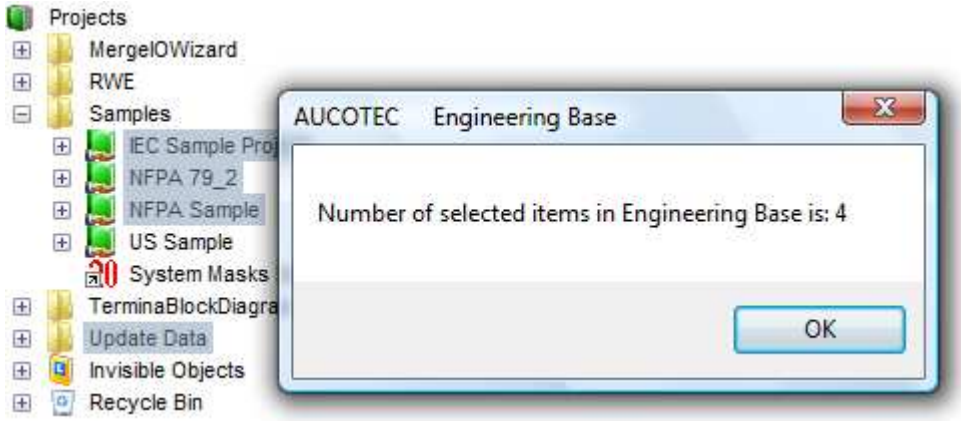
Ďalším cieľom je vybrané príklady uviesť v čo najjednoduchšej forme a zamerať sa pritom na konkrétny problém. Nechcená je situácia, kedy by sa mal užívateľ zaoberať inou časťou kódu, aká je v danej časti potrebná. Vysoký dôraz je kladený aj na náväznosť jednotlivých príkladov, tak aby bol užívateľ schopný postupne plniť jednotlivé príklady.

4 ZÁKLADY OBJEKTOVÉHO MODELU ENGINEERING BASE

4.1 Application.Selection

4.1.1 Popis problému

Úlohou prvého príkladu je zoznámiť užívateľa so základnou triedou *Application*. V tomto príklade bude použitá na získanie označených prvkov v strome. Tento postup je využívaný vo veľkom počtu makier. Umožňuje totiž identifikovať objekt respektíve objekty, na ktorých je makro spúšťané. Ďalej použije textový dialóg, pomocou ktorého zobrazí počet označených objektov.



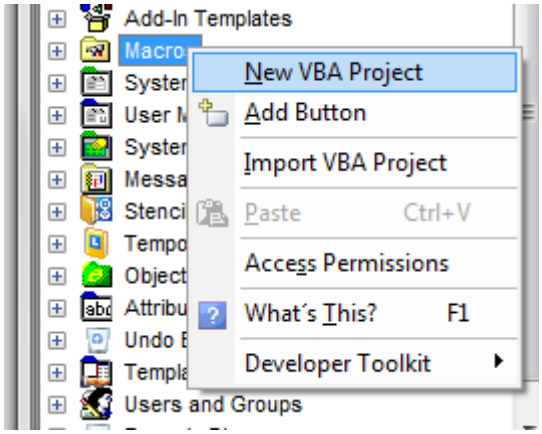
Obrázok 19. Okno správ oznamujúce počet označených prvkov v strome

Prvá úloha zároveň poslúži aj na oboznámenie užívateľa s rozhraním pre vytváranie makier v Engineering Base. Pomocou nej sa užívateľ naučí nové makro založiť a spustiť. Ide teda o jednoduchý vstupný príklad, ktorý nie je ani tak zameraný na oboznámenie s EB

Užívateľ má teda za úlohu zistiť s využitím triedy *Application* počet označených objektov v stromovej štruktúre prieskumníka aplikácie a zobraziť výsledok v textovom dialógu.

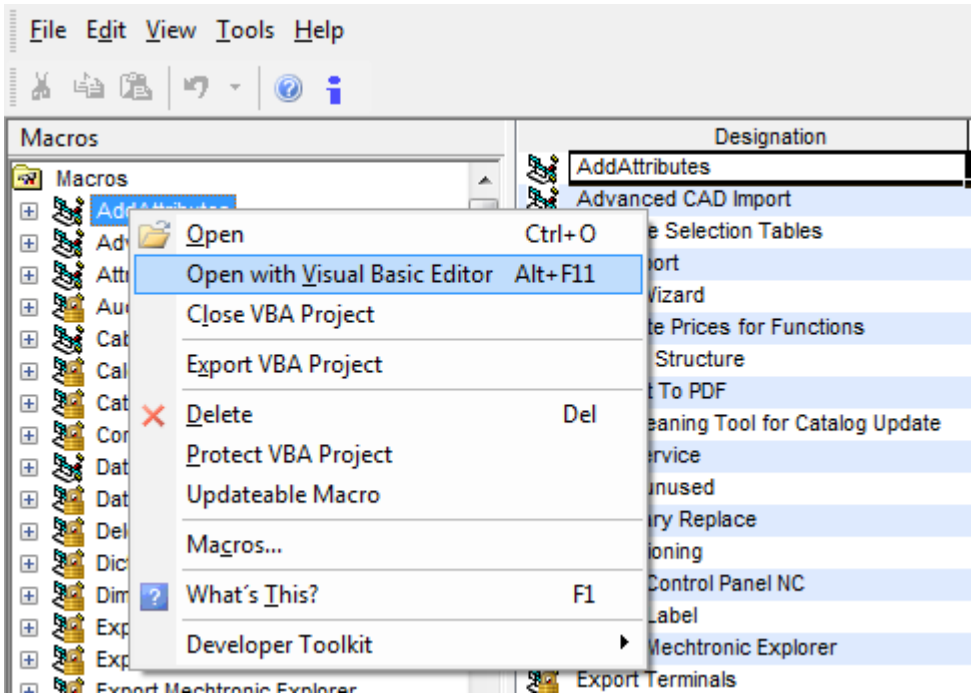
4.1.2 Vytvorenie nového makra

V Engineering Base sa nové makro vytvorí vybraním položky *New VBA Project* v kontextovom menu vyvolaným pri označení uzlu *Macros* v strome prieskumníka. Odkaz na novo založené makro sa uloží pod tento uzol.



Obrázok 20. Založenie nového makra

Pre editovanie novo založeného makra je potrebné odkaz naň vyhľadať a zvoliť položku *Open with Visual Basic Editor* kontextového menu.



Obrázok 21. Otvorenie makra pre editáciu vo Visual Basic Editoru

4.1.3 Použité triedy, vlastnosti a metódy

Pre nasledujúcu úlohu je treba využiť triedu *Application* a jej vlastnosť *Selection*, ktorá vracia kolekciu označených objektov v strome prieskumníka aplikácie. Vlastnosť *Count*, ktorá je vlastnosťou získanej kolekcie sa získa počet práve označených objektov.

Výpis použitých tried:

- *Application* - je to jedna zo základných tried objektového modelu aplikačného rozhrania aplikácie *Engineering Base*. Umožňuje získať základné informácie o spustenej aplikácii, ako napr. verziu, používaný jazyk a iné. Ďalej poskytuje globálne metódy napr. *WaitDialog*, ktorá slúži pre zobrazenie dialógového okna s lištou priebehu a animáciami. Taktiež sa používa práve na zistenie čo v aplikácii prebieha, teda aktívny projekt, aktuálne označené objekty v strome. V tabuľke (Tabuľka 1) sú zobrazené vybrané metódy a vlastnosti triedy.

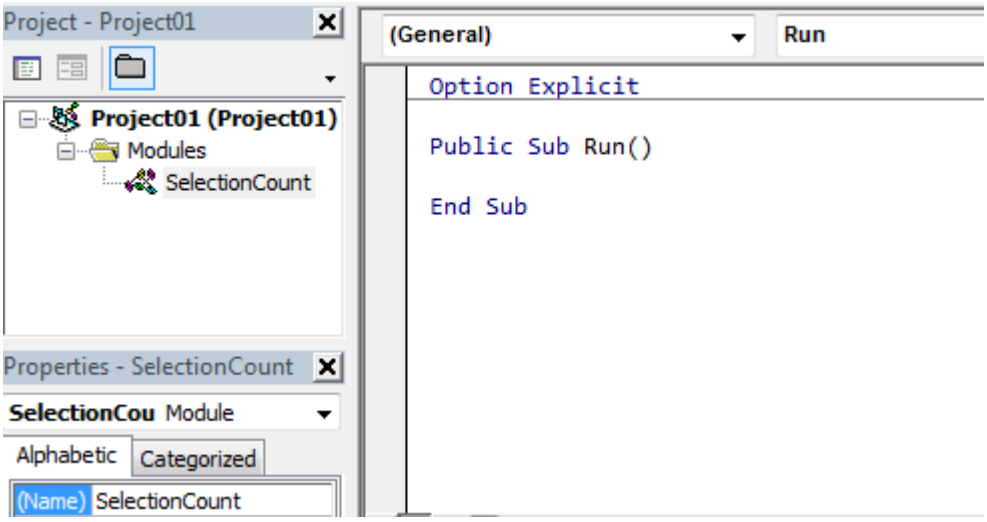
Tabuľka 1. Vybrané verejné metódy a vlastnosti triedy Application

Verejné metódy	Popis
Refresh	Obnoví dáta aktualizované v aplikácií
Verejné vlastnosti	Popis
ActiveProject	Vráti referenciu na aktívny projekt
DBVersion	Vráti číslo verzie databáze
Language	Vráti ID použitého jazyka
Name	Vráti názov aplikácie
Selection	Vráti kolekciu objektov označených v strome objektov alebo v dynamickom liste
Settings	Vráti objekt pomocou ktorého je možné získať napr. cestu k dočasnému úložisku aplikácie, meno prihláseného užívateľa a iné
Undo	Vráti objekt pre vyvolanie akcie „Späť“
WaitDialog	Vráti objekt pre vyvolanie okna s lištou priebehu

4.1.4 Vysvetlenie zdrojových kódov

Po otvorení makra pomocou kontextového menu sa zobrazí okno editoru pre *Visual*

Basic. Vo *VBA* je možné pracovať s *Modulom, Class Modulom a UserForm*. Pre vytvorenie tejto úlohy bude stačiť založenie jedného *Modulu*.



Obrázok 22. Okno editoru – spúšťacia metóda Run

Aby bolo makro z aplikácie EB spustiteľné, musí obsahovať základný modul a v ňom vytvorenú globálnu metódu bez vstupných parametrov. Kód potrebný na splnenie tejto úlohy pozostáva z trochu riadkov:

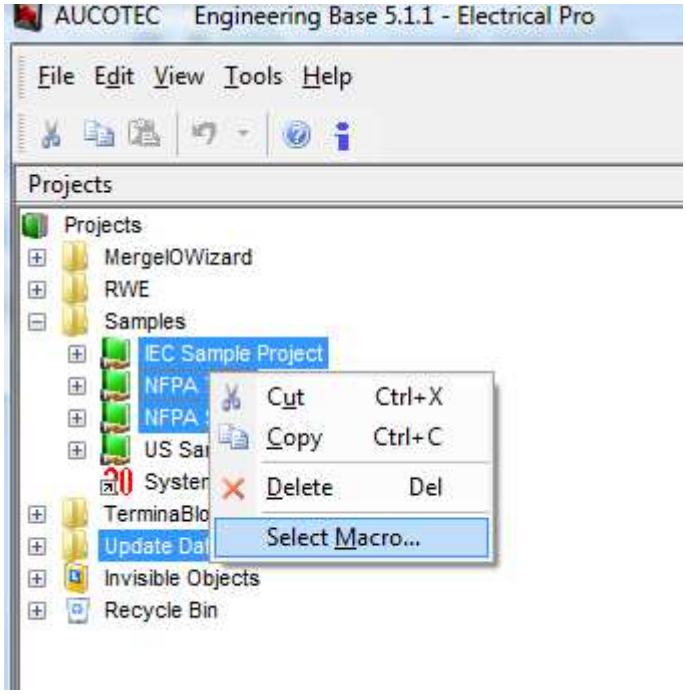
```
5 Public Sub Run()
6 End Sub
```

Komentár: Riadky ohraničujú globálnu metódu, ktorá je pomenovaná *Run* a slúži na spustenie makra.

```
7   Call MsgBox("Number of selected items in Engineering Base is: "
  & Application.Selection.Count)
```

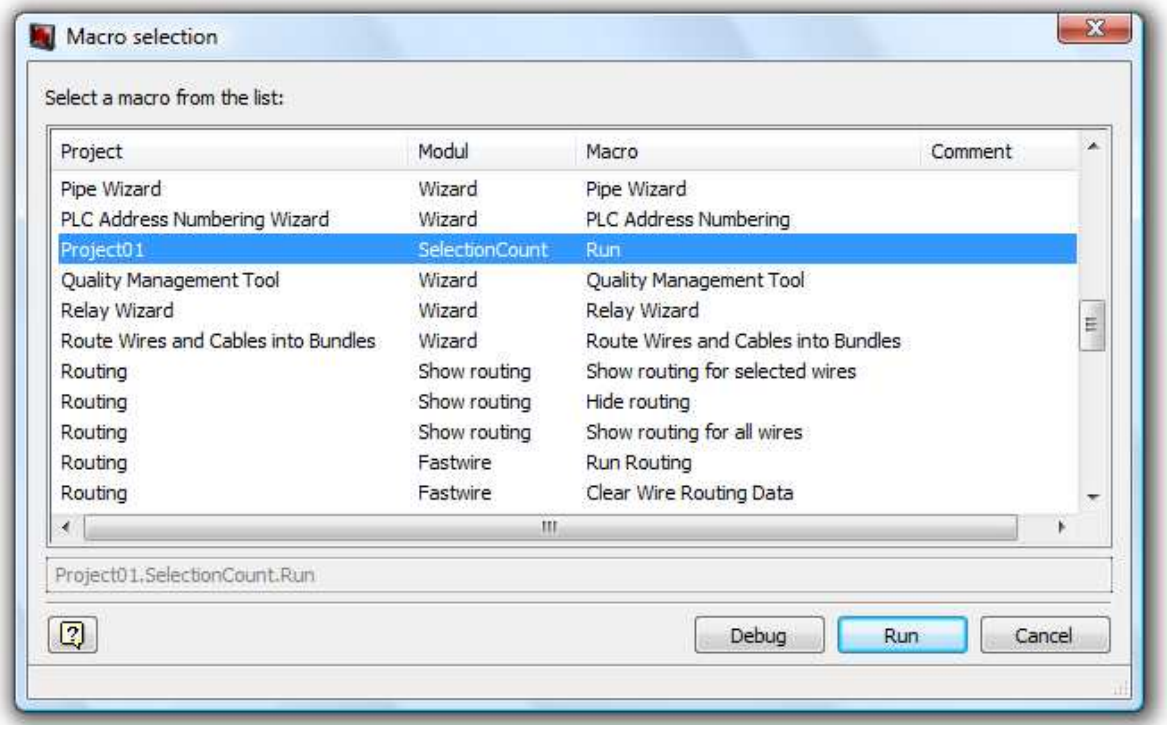
Komentár: Pomocou metódy *MsgBox* sa vyvolá okno správ s predaným reťazcom a počtom označených objektov v stromovej štruktúre prieskumníka. Počet zistíme vlastnosťou *Selection.Count.*

Makro sa spustí z okna prieskumníka aplikácie. Nad označenými objektmi sa vyvolá kontextové menu a v ňom sa vyberie prvok „*Select Macro...*“.



Obrázok 23. Spustenie makra – označenie prvkov

Následne sa zobrazí okno so zoznamom existujúcich makier, v ktorom sa nachádza novo vytvorené makro. Po označení a stlačení tlačítka *Run* sa makro spustí.

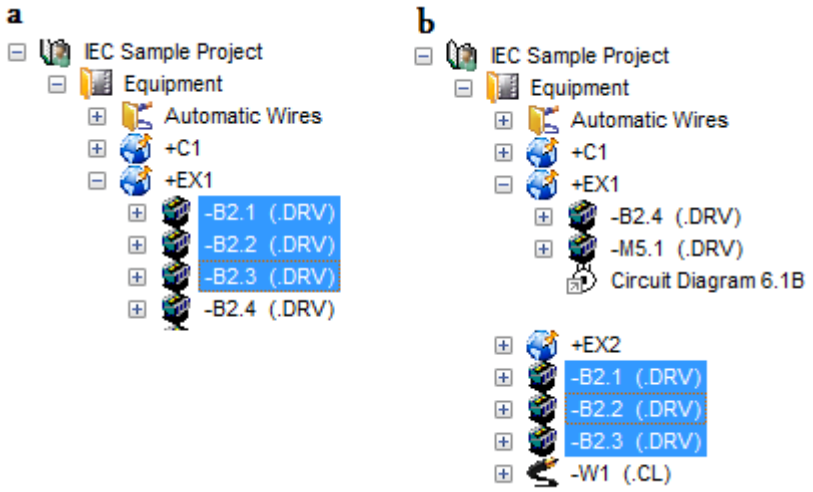


Obrázok 24. Spustenie makra – vybranie zo zoznamu makier

4.2 ObjectItem

4.2.1 Popis problému

Prvá úloha ukazuje ako získať označené objekty v strome. Úlohou tohto príkladu je pristúpiť k týmto objektom a presunúť ich do určenej lokácie. V tomto prípade objektu *Equipment*.



Obrázok 25. Ukážka premiestnenia prvokov v strome. Pred makrom (a), po marku (b)

Úlohou je teda presunúť všetky objekty z vybranej lokácie priamo do zložky *Equipment*.

4.2.2 Použité triedy, vlastnosti a metódy

Trieda *ObjectItem* je základnou triedou pre všetky prvky zo stromu prieskumníka. Preto je dobré aby sa s ňou programátor zoznámil hneď zo začiatku.

Výpis použitých tried:

- *ObjectItem* – reprezentuje objekt v strome objektov. Je to jedna zo základných tried z ktorej je ďalej odvodených množstvo podtried, medzi ktoré patria napríklad tried: *Sheet, Cable, Project,* a iné. V tabuľke (Tabuľka 2) sú uvedené vybrané metódy a vlastnosti tejto triedy.

Tabuľka 2. Vybrané verejné metódy a vlastnosti triedy ObjectItem

Verejné metódy	Popis
Store	Uloží kolekciu atributov objektu od databáze.
Delete	Vymaže konkrétny objekt.
CopyTo	Skopíruje objekt pod nového rodiča.
MoveTo	Presunie objekt a jeho deti pod nového rodiča.
Verejné vlastnosti	Popis
Name	Vráti meno objektu.
FullName	Vráti referenčné meno objektu.
Kind	Vráti typ objektu.
Attributes	Vráti kolekciu objektov atribútov objektu.
Parent	Vráti alebo nastaví rodiča objektu.
Project	Vráti odpovedajúci projekt objektu.

- *Application.*– Použije sa je vlastnosť *Selection.* Trieda *Application* už bola popísaná v predchádzajúcej úlohe.
- *Project* – je odvodenou triedou z triedy *ObjectItem*. Pomocou jej metódy sa dostaneme k objektu *EquipmentFolder*. Trieda *Project* bude podrobnejšie rozobraná v neskorších kapitolách.

4.2.3 Vysvetlenie zdrojových kódov

Najprv je potrebné si nadeklarovať premennú triedy *ObjectItem*. Do tejto premennej sa neskôr priradí objekt zo stromu.

```
8   Dim Item As ObjectItem
```

Komentár: Deklarácia premennej *Item* ako typ *ObjectItem*.

```
9   For Each Item In Application.Selection
```

Komentár: Vlastnosť *Application.Selection* vracia kolekciu označených objektov v strome. Na prechádzanie prvkov tejto kolekcie je použitý cyklus *For Each*.

```
10      Call Item.MoveTo(Item.Project.EquipmentFolder)
```

Komentár: Na presunutie objektov z jednej lokácie sa použije verejná metóda *MoveTo*, ktorej ako parameter predáme odkaz na objekt *Equipment*, ktorý sme si získali pomocou metódy *EquipmentFolder* z triedy z.

4.3 Vlastnosť Attributes

4.3.1 Popis problému

Vlastnosť *Attributes* je vlastnosťou triedy *ObjectItem*. Ide o kolekciu prvkov triedy *Attribute*. Pomocou nej je možné sa dostať po vlastnosti jednotlivých objektov aplikácie Engineering Base. Prechádzanie a upravovanie atribútov jednotlivých objektov je častá operácia pri vytváraní makier. V podstate neexistuje väčšie makro, kde by sa táto trieda nepoužívala. Napríklad pri vyhľadávaní prvkov v databáze podľa určitých požiadavkou alebo pri upravovaní, spojovaní, vytváraní objektov je potrebné k vlastnostiam objektu pristupovať. Na tomto príklade sa užívateľ zoznámi s touto triedou.

Úlohou je vytvoriť metódu na skopírovanie všetkých atribútov jedného objektu do objektu druhého.

4.3.2 Použité triedy, vlastnosti a metódy

Na zvládnutie tejto úlohy je potrebné použiť triedy z minulých príkladov na prístup k objektom zo stromu a pomocou vlastnosti Attributes k objektom typu *Aucotec.Attribute*, pomocou ktorých sa pre zmenu pristupuje atribútom získaných inštancii objektov. Ďalej sa tu využije metódy *ObjectItem.Store* pre uloženie prevedených zmien nad objektmi do databáze aplikácie.

Výpis použitých tried:

- *ObjectItem* – Ide hlavne o vlastnosť *ObjectItem.Attributes*, ktorá vracia kolekciu objektov typu *Aucotec.Attribute*. Táto vlastnosť je používaná pri práci s atribútmi objektu, pretože dokáže pridávať inštancii objektu nové atribúty, vymazať nepotrebné, vyhľadať konkrétny atribút, atd. Popis vybraných vlastností sa nachádza v tabuľke (Tabuľka 3).

Tabuľka 3. Vybrané verejné metódy a vlastnosti triedy Aucotec. Attributes

Verejné metódy	Popis
Add	Pridá atribút k vybranej inštancii objektu.
Find	Nájde a vráti prvok z kolekcie atribútov podľa ID atribútu alebo názvu.
Item	Vráti prvok z kolekcie.
ItemByID	Vráti prvok z kolekcie podľa ID atribútu.
Remove	Vymaže atribút z inštancie objektu.
Verejné vlastnosti	Popis
Count	Vráti počet atribútov.

- *Aucotec.Attribute* – Je to trieda ktorá reprezentuje prvky kolekcii atribútov objektov typu *ObjectItem*. Je potrebné pred typom *Attribute* uviesť prefix *Aucotec*. Vysvetlením je, že bez prefixu je typ *Attribute* kľúčové slovo jazyka VBA. Najčastejšia vlastnosť ktorá sa používa je vlastnosť *Value*, ktorá vracia hodnotu vybraného atribútu danej inštancie objektu. V tomto príklade to bude práve táto vlastnosť, ktorá sa využije pri získaní a zápise hodnoty atribútu. Tabuľka (Tabuľka 4) obsahuje vybrané vlastnosti. Mnohé z nich sú rovnako často používané ako vlastnosť *Value*.

Tabuľka 4. Vybrané verejné vlastnosti triedy Aucotec.Attribute

Verejné vlastnosti	Popis
FromCatalog	Vráti alebo nastaví atribútu príznak, že daná hodnota atribútu je z katalógu.
ID	Vráti ID daného atribútu.
IsDeleted	Vráti príznak, či je atribút vymazaný.
Name	Vráti meno atribútu.
ReadOnly	Vráti príznak, či je atribút prepisovateľný.
Translated	Vráti príznak, či je hodnota atribútu zo slovníku.
TranslateValue	Vráti hodnotu alebo preloží hodnotu, ktorá je zo slovníku.
Type	Vráti typ hodnoty daného atribútu.
Value	Vráti alebo nastaví hodnotu atribútu.

4.3.3 Vysvetlenie zdrojových kódov

Tentoraz vytvárame metódu, ktorá obsahuje dva vstupné parametry typu ObjectItem. Metóda potom skopíruje hodnoty atribútov z prvej inštancie objektu do inštancie objektu, na ktorý odkazuje druhý parameter.

```
11 Public Sub CopyAttributes(oSource As ObjectItem, oDestination As
   ObjectItem)
```

Komentár: Metóda je globálna a obsahuje dva vstupné argumenty.

```
12 Dim oAttribute As Aucotec.Attribute
```

Komentár: Je potrebné najprv nadeklarovať premennú typu *Aucotec.Attribute*.

```
13 For Each oAttribute In oSource.Attributes
```

Komentár: Pomocou cyklu *For Each* sa postupne získajú jednotlivé atribúty inštancie objektu.

```
14 Set oDestAttribute = oDestination.Attributes.Find(oAttribute.ID)
15 If Not oDestAttribute Is Nothing Then
```

Komentár: Dôležitá časť tejto metódy. Je potrebné overiť, či vôbec atribút s konkrétnym identifikačným číslom obsahuje aj inštancia cieľového objektu. Na získanie identifikačného čísla atribútu poslúži vlastnosť *ID* triedy *Aucotec.Attriubte*. Vlastnosť *Find* kolekcie *ObjectItem.Attributes* potom pomocou získaného identifikačného čísla vráti požadovaný atribút cieľovej inštancie objektu.

Pre prístup k jednotlivým atribútom sa veľmi často používajú metódy *Item*, *ItemByID* a *ItemByEClass* vlastnosti *Attributes* triedy *ObjectItem*. Prvá metóda pritom môže preberať viac typov argumentov a umožňuje tak viacero možností prístupu.

- Metóda *Attributes.Item* – umožňuje prístup na základe dvoch typov argumentu:
    - Podľa *indexu* – poradie atribútov nie je dané napevno. To znamená že sa nedá dopredu nikdy presne určiť aký atribút bude na určitom poradí kolekcie atribútov objektu.
    - Podľa *mena* – nevýhoda toho prístupu spočíva v odlišnostiach názvoch u jednotlivých jazykových verzií Engineering Base.
- Metóda *Attributes.ItemByID* – prístup podľa identifikačného čísla. Asi najpoužívanejšia možnosť pri prístupu pomocou tejto metódy. Identifikačné čísla sú pre všetky jazykové verzie rovnaké, problém však nastáva pri prístupu k užívateľom vytvorených atribútom. Tie nemajú pevne dané identifikačné číslo a v rôznych databázach sa ich hodnoty môžu líšiť.
- Metóda *Attribute.ItemByEClass* – v podstate ľubovoľne definovaná identifikácia. Nevýhodou však je, že nie je vždy vyplnená.

```
16 If Not oDestAttribute.ReadOnly Then
```

Komentár: Po získaní cieľového atribútu je potrebné ešte overiť či nie je nastavený príznak atribútu proti zápisu.

```
17 oDestAttribute.Value = oAttribute.Value
```

Komentár: Na tomto riadku sa pomocou vlastnosti *Value* získa zo zdrojovej inštancie objektu hodnota konkrétneho atribútu a tou istou vlastnosťou sa do cieľového atribútu získaná hodnota zapíše.

```
18 Call oDestination.Store
```

Komentár: Prevedené zmeny v atribútoch cieľového objektu je potrebné ešte uložiť do databáze. V opačnom prípade sa po skončení funkcie zmeny nezapíšu. K tomu slúži metóda *Store* triedy *ObjectItem*, ktorá sa zavolá nad inštanciou cieľového objektu.

4.4 Pridávanie a odoberanie atribútov

4.4.1 Popis problému

Nasledujúci príklad ukazuje akým spôsobom funguje správa atribútov v aplikácii Engineering Base. V predošlom príklade bolo vidieť ako pristúpiť k jednotlivým atribútom vybraného objektu. Pomocou metód *Add* a *Remove* je možné tento atribút vymazať alebo pridať nový. V Engineering Base však existuje pre každý objekt *definícia dialógu*. Tá umožňuje určitému objektu definovať atribúty. V tejto úlohe bude treba pomocou tohto dialógu ručne pridať dvom objektov ľubovoľný nový atribút a u jedného vyplniť jeho hodnotu a u druhého nechať hodnotu nevyplnenú. Oba tieto objekty pritom budú jedného typu. Potom sa upraví definícia dialógu tohto typu pridaním ďalšieho iného atribútu. Pomocou makra sa nad oboma objektmi zavolá metóda *MergeTypeAttributes*, ktorá spojí zoznamy atribútov vybraných objektov s aktuálnou množinou atribútov typu týchto zariadení.

Úlohou je teda ručne upraviť definíciu dialógu pre motory *M1* a *M2*, ku každému pridať nový atribút. Vyplniť tento atribút u *M1* a nevyplniť ho u *M2*. Upraviť definíciu dialógu pre typ *Motor* – pridať iný atribút. Zavolať metódu *MergeTypeAttributes* nad motorom *M1* a *M2*. Porovnať výsledky.

4.4.2 Použité triedy, vlastnosti a metódy

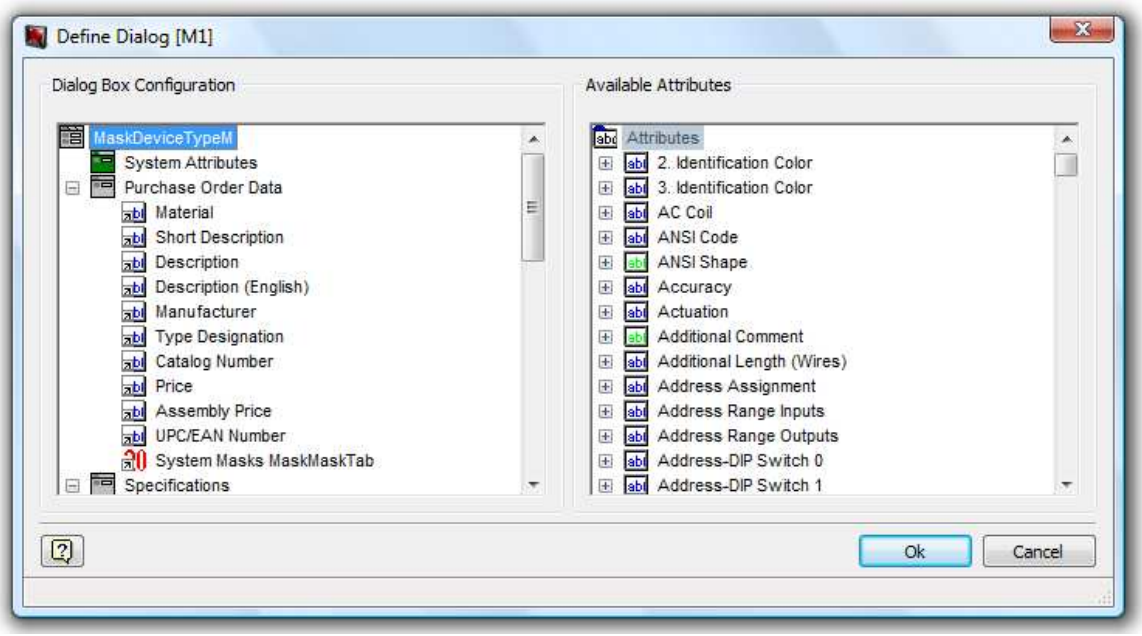
V tejto úlohe sa vyskytujú rovnaké triedy ako v predchádzajúcich príkladoch. Nová je tu metóda *MergeTypeAttributes* triedy *ObjectItem*. Tá sa volá bez vstupných argumentov.

4.4.3 Vysvetlenie zdrojových kódov

Najprv je potrebné upraviť definíciu dialógu konkrétneho prvku a neskôr aj daného typu objektu. V menu aplikácie Engineering Base po označení žiadaného prvku v strome, sa v podmenu *Edit* nachádza prvok *Define Dialog*. Po jeho vybraní sa zobrazí okno na

úpravu definície dialógu označeného prvku v prieskumníku.

4.4.4 Úprava definície dialógu



Obrázok 26. Okno pre úpravu definície dialógu

Toto okno obsahuje dva zoznamy. Ľavý zoznam, je zoznam aktuálnych atribútov daného prvku. V druhom zozname sa nachádza výpis možných atribútov, ktoré je možné objektu priradiť.

Tento postup je nutné pre splnenie tejto úlohy opakovať aj pre druhé zariadenie. Pre pridanie atribútu k typu *Motor* treba otvoriť *Define Dialog* nad prvkom v databáze, ktorý tento typ reprezentuje. Ten sa nachádza pod uzlom *Type Definitions*.

4.4.5 Vysvetlenie zdrojových kódov

```
19 Dim Motor As ObjectItem
```

Komentár: Deklarácia premennej *Motor* typu *ObjectItem*.

```
20   For Each Motor In Application.Selection
21     Call Motor.MergeTypeAttributes
```

Komentár: Pre všetky označené, v tomto prípade dva, prvky v prieskumníku sa zavolá metóda *MergeTypeAttributes*, ktorá spojí zoznam atribútov objektu so zoznamom atribútov typu objektu.

4.4.6 Porovnanie výsledkov

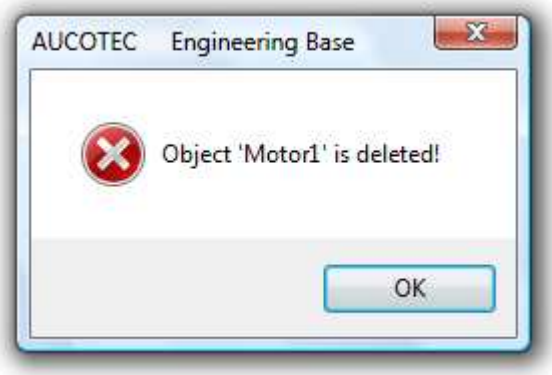
Po porovnaní výsledkov je zrejmé, že vyplnený atribút v prvom objekte ostal zachovaný, pričom ten istý atribút v druhom objekte bol odstránený, pretože jeho hodnota bola prázdna. Ďalej pribudol obom objektom nový atribút, ktorý bol pridaný do zoznamu atribútov u typu *Motor*. Z toho vyplýva, že metóda *MergeTypeAttribute* pridá existujúcim objektom všetky atribúty, ktoré sú v zozname atribútov definované danému typu objektu, a daný objekt ich neobsahuje a odoberie všetky nevyplnené atribúty, ktoré sa nenachádzajú v zozname atribútov tohto typu objektu.

4.5 Vymazávanie objektov

4.5.1 Popis problému

V tomto príklade je úlohou vymazať získaný objekt a následne nad ním zavolať vlastnosť *IsDeleted*. Vlastnosť *IsDeleted* je vlastnosťou triedy *ObjectItem* a je často využívaná na overenie, či daný objekt nebol pri spracovávaní makra alebo pri práci s objektmi vo viac užívateľskom prístupe v danej lokácii vymazaný.

Úlohou teda je zobraziť meno vymazaného objektu.



Obrázok 27. Okno správ oznamujúce, že objekt je vymazaný.

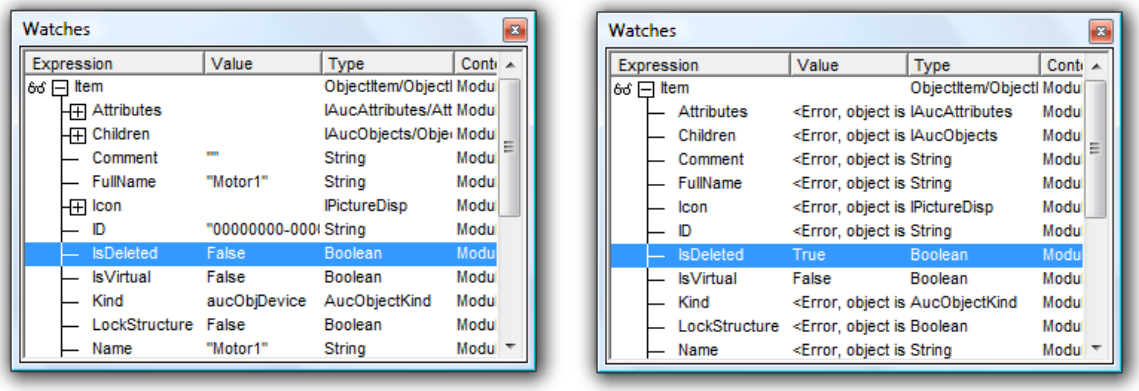
4.5.2 Vysvetlenie zdrojových kódov

Kód tohto príkladu nie je vôbec ťažký. Ide tu hlavne o preskúšanie vlastnosti IsDeleted a pochopenie tak problematiky programovania napr. pri viac užívateľskom prístupe,

kedy práve spracovávaný objekt môže byť vymazaný iným užívateľom. Táto vlastnosť sa však používa aj v makrách, kde sa objekty postupne vymazávajú, na ošetrenie stavu, kedy sa makro snaží pracovať s už vymazaným objektom v predchádzajúcom spracovaní objektu.

```
22      Name = Item.Name
23      Call Item.Delete
```

Komentár: Problém spočíva v tom, že sa po zavolaní metódy *Delete* nad objektom premenná nenastaví na *Nothing*, ako by sa mohlo predpokladať. Objekt však bude mať vlastnosť *IsDeleted* nastavenú na hodnotu *True*. Napríklad pri pokuse o prístup k vlastnosti *Attributes* daného objektu nastane pri neošetrení chyba. Preto je potrebné v tomto príklade popredu uložiť hodnotu vlastnosti *Name*, inak by už nebolo možné meno objektu po vymazaní zobraziť.



Obrázok 28. Watch window existujúceho prvku (a), Watch window vymazaného prvku(b).

```
24      If Item.IsDeleted Then
25        Call MsgBox("Object '" & Name & "' is deleted!", vbCritical)
26      End If
```

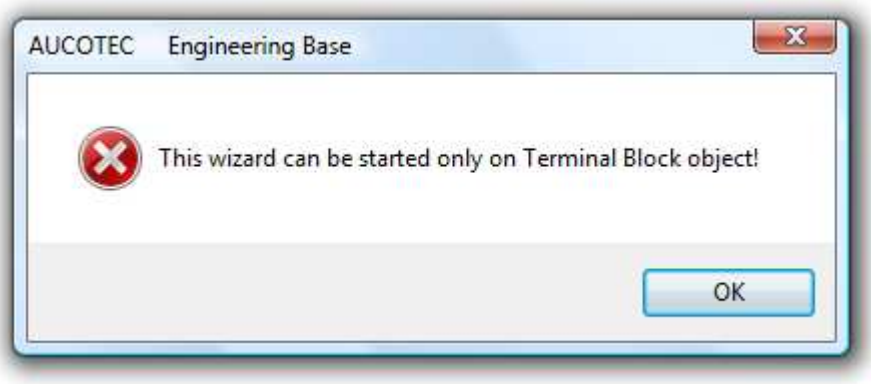
Komentár: Po vymazaní objektu z databáze Engineering Base a overení či objekt je vymazaný alebo nie je, sa volá funkcia *MsgBox*, ktorá zobrazí názov vymazaného objektu.

4.6 Vytváranie objektov

4.6.1 Popis problému

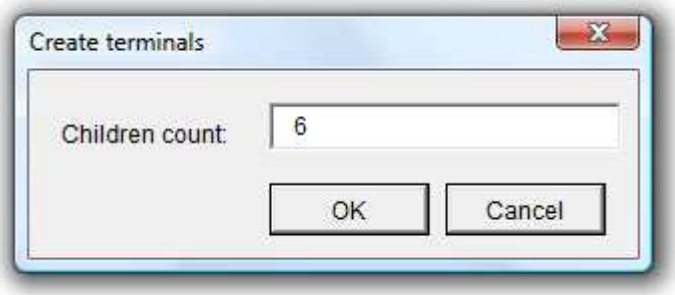
V predchádzajúcom príklade sa objekty vymazávali a overovalo sa, či vymaznaný objekt

v databáze existuje. V tejto úlohe sa budú objekty pre zmenu vytvárať. Makro sa bude spúšťať nad svorkovnicou a po spustení sa zobrazí okno s textovým polom pre vloženie počtu svoriek, ktoré makro vytvorí pod príslušným objektom. Hneď na začiatku však treba overiť, či je makro spustené na správnym objektom, v opačnom prípade vyhodí okno so správou o chybnom spustení.



Obrázok 29. Okno správ – zle označený prvok

Úlohou je teda vytvoriť makro, ktoré sa bude spúšťať nad svorkovnicou (typ *Terminal Block*), zobrazí *Input Box* pre vloženie počtu svoriek (typ *Terminal*) a potom vytvorí príslušný počet svoriek.



Obrázok 30. Okno na vloženie počtu objektov, ktoré sa majú vytvoriť

4.6.2 Použité triedy, vlastnosti a metódy

Tento príklad využíva metódy a vlastnosti triedy *ObjectItem*. Konkrétne ide o metódu *NewChild*, ktorá vytvorí pod príslušným objektom nový objekt a priradí tento nový objekt do kolekcie *Children* objektu, nad ktorým bola táto vlastnosť volaná. Metóda *TypeID* je využitá pre získanie typu objektu.

4.6.3 Vysvetlenie zdrojových kódov

Na začiatku treba získať objekt označený v prieskumníku aplikácie a overiť jeho typ.

```
27 Private Function moGetSelected() As ObjectItem
28   On Error Resume Next
29   Set moGetSelected = Application.Selection(1)
30 End Function
```

Komentár: Táto funkcia vracia inštanciu objektu triedy ObjectItem. Riadok s kódom *On Error Resume Next* spôsobí, že v prípade kedy nebude vlastnosť *Selection* obsahovať žiaden prvok, spustené makro nevyhodí výnimku pri snahe získať neexistujúci prvok z kolekcie.

```
31   Set oItem = moGetSelected
32   If Not oItem Is Nothing Then
33     If oItem.TypeID = aucDevTerminalBlock Then
```

Komentár: Je potrebné ošetriť stav, ak funkcia *moGetSelected* vrátila *Nothing*. Potom už môže prebehnúť porovnanie typu inštancie objektu.

```
34 Call frmTerminalsCount.gShow(oItem)
```

Komentár: Po splnení podmienky sa zavolá verejná metóda *gShow* z vytvoreného formulára *frmTerminalsCount* a predá sa ako argument získaný objekt.

```
35 Call MsgBox("This wizard can be started only on Terminal Block
   object!", vbCritical)
```

Komentár: Zobrazenie správy pri nesplnení podmienky.

```
36 Public Sub gShow(ByRef oTerminal As ObjectItem)
37   Set moTerminal = oTerminal
38   Call Show(vbModal)
39 End Sub
```

Komentár: Verejná metóda *gShow* s jedným argumentom. Táto metóda je volaná z hlavného modulu a umožňuje predanie ukazatela na získaný objekt. Premenná *moTerminal* je globálna premenná formulára. Funkcia *Show* zobrazí vytvorený formulár. Ako argument jej bol predaný *vbModal*, čo spôsobí modálne otvorenie formulára.

```
40   lCount = Val(txtTerminalCount.Text)
41   For i = 1 To lCount
42     Call moTerminal.NewChild(aucObjDevice, aucDevTerminal)
43   Next i
```

Komentár: Po stlačení tlačítka *Ok* sa získa hodnota z textového pola. Funkcia *Val* vráti

hodnotu číselného typu. Ak užívateľ do textového pola vložil písmeno alebo ho nechal prázdne, funkcia *Val* vráti hodnotu *0*. V cykle sa vytvorí pomocou funkcie *NewChild* objekt typu *Terminal*. Funkcia *NewChild* má dva argumenty. Prvý argument je druh objektu dátového typu *AucObjectKind*. Druhý je typ objektu dátového typu *AucType*.

```
44 Call Unload(Me)
```

Komentár: Tento príkaz vymaže formulár na ktorý ukazuje ukazateľ *Me*. V tomto prípade je to teda spustený formulár.

4.7 Vyhľadávanie objektov pomocou metódy FindObjects

4.7.1 Popis problému

Na vyhľadávanie špecifických objektov v databáze aplikácie sa využíva metóda *FindObjects* triedy *ObjectItem*. Výhodou tohto vyhľadávania je že sa celá operácia odohráva na serveru a späť sa pošlú len získané dáta. Pomocou tejto metódy je užívateľ schopný získať napríklad všetky objekty v projekte typu *Motor*, ktoré majú v atribúte *Name* hodnotu *M1*.

Úlohou je spracovať všetky motory v aktívnom projekte, ktorých názov začína výrazom „1M“ a nahradiť tento výraz výrazom „2M“.

4.7.2 Použité triedy, vlastnosti a metódy

Metóda *FindObjects* vracia kolekciu objektov a má množstvo vstupných parametrov, ktoré bližšie špecifikujú hľadané objekty pod daným prvkom.

Parametre metódy *FindObjects*:

- Parameter *Kind* typu *AucObjectKind* – ide o dátový typ enum a určuje druh hľadaného objektu. V tabuľke (Tabuľka 5) sú uvedené najpoužívanejšie hodnoty tohto typu.

Tabuľka 5. Vybrané hodnoty typu AucObjectKind

Hodnota	Popis
aucObjUnspecified	Bližšie nešpecifikovaný objekt.
aucObjProject	Na vyhľadanie projektov.
aucObjFunction	Na vyhľadanie funkcií.
aucObjCatalog	Na vyhľadanie katalógov.
aucObjUserAttribute	Na vyhľadanie užívateľom definovaného atribútu.
aucObjFolder	Na vyhľadanie zložky.
aucObjSheet	Na vyhľadanie listu.
aucObjWire	Na vyhľadanie drôtu.
aucObjCable	Na vyhľadanie káblu.

- Parameter *searchType* typu *AucVbSearchType* – je parameter, ktorý určuje typ vyhľadávania. Môže nadobúdať tri hodnoty, pričom každá z nich poskytuje iný prístup k vyhľadávaniu:
    - hodnota *aucSearchFlat* – metóda prehľadáva objekty len v prevej úrovni detí daného objektu.
    - hodnota *aucSearchHierarchical* – pri vložení tejto hodnoty prehľadáva metóda FindObject postupne všetky úrovne detí daného objektu, pričom sa neprehľadávajú úrovne pod objektmi, ktoré splňujú podmienky hľadania.
    - hodnota *aucSearchDeep* – prehľadávajú sa úplne všetky úrovne detí daného objektu.
- Parameter *TID* typu *AucType* – paremeter, ktorý bližšie určuje typ hľadaného objektu. V tabuľke (Tabuľka 6) sú uvedené jeho vybrané hodnoty.

Tabuľka 6. Vybrané hodnoty typu AucType

Hodnota	Popis
aucTypeUnspecified	Bližšie nešpecifikovaný typ objektu.
aucDevMotor	Používa sa pri hľadaní objektu typu motor.
aucDevTerminal	Používa sa pri hľadaní objektu typu terminál.
aucWireSignal	Používa sa pri hľadaní signálneho drôtu.
aucDevCable	Používa sa pri hľadaní objektu typu kábel.
aucPrjUnspecified	Používa sa pri hľadaní projektu.

- Parameter *attrID* typu *AucAttribute* – tento parameter rozširuje prehľadávanie objektov. Je pomocou neho možné získať určitý atribút objektu a porovnať jeho hodnotu so žiadanou hodnotou. V tabuľke (Tabuľka 7) sú uvedené najpoužívanejšie hodnoty tohto parametru.

Tabuľka 7. Vybrané hodnoty typu AucAttribute

Hodnota	Popis
aucAttrDesignation	Získa atribút *designation*, ktorý nesie hodnotu názvu objektu.
aucAttrComment	Komentár daného objektu.
aucAttrPosition	Atribút *position* určuje pozíciu objektu.
aucAttrCatalogNumber	Atribút číslo katalógu.
aucAttrOrigin	Získa atribút origin.

- Parameter *attrCondition* typu *AucVbFindCondition* – tento paramter určuje typ porovnávania získaného atribútu objektu. Nadobúda šesť hodnôt:
    - *aucCondEqual* – hodnota parametru je rovnaká ako získaná hodnota atribútu daného objektu.

- *aucCondNotEqual* – hodnota paramteru nie je rovnaká ako získaná hodnota atribútu daného objektu.
- *aucCondLess* – hodnota parametru je menšia ako získaná hodnota atribútu daného objektu.
- *aucCondLessEqual* – hodnota parametru je menšia alebo rovná ako získaná hodnota atribútu daného objektu.
- *aucCondGreater* – hodnota parametru je väčšia ako získaná hodnota atribútu daného objektu.
- *aucCondCreaterEqual* – hodnota parametru je väčšia alebo rovná ako získaná hodnota atribútu daného objektu.

- Parameter *attrValue* typu *Variant* – určuje hodnotu atribútu, podľa ktorej bude metóda získaný atribút porovnávať.
- Parameter *LoadWithAttributes* typu *Boolean* – tento parameter určuje, či sa majú nájdené objekty načítať aj s ich atribútmi alebo nie.

4.7.3 Vysvetlenie zdrojových kódov

Vďaka metóde *FindObjects* je výsledné makro otázkou niekoľkých riadkov kódu. O celé prehľadávanie objektov a porovnávanie hodnôt ich atribútov sa postará práve metóda *FindObjects*. V podstate stačí predať správne argumenty metóde a potom obslúžiť získané objekty.

```
45   For Each Motor In
   ActivePject.EquipmentFolder.FindObjects(aucObjDevice,
   aucSearchDeep, aucDevMotor, aucAttrDesignation, aucCondEqual,
   "1M*", True)
```

Komentár: Vlastnosťou *ActiveProject*, ktorá je vlastnosťou triedy *Application*, sa získa ukazateľ na aktívny projekt. Teda na projekt, v ktorom sa nachádza označený objekt. Táto vlastnosť vracia objekt typu *Project* a pomocou jej vlastnosti *EquipmentFolder*, ktorá má návratovú hodnotu typu *ObjectItem* sa získa referencia na objekt *Equipment*. Pod týmto objektom sa nachádzajú všetky zariadenia daného projektu, teda aj hľadané motory. Nad touto triedou sa zavolá metóda *FindObject*. Ako typ objektu sa predá hodnota *aucDevMotor* a pretože objekt motor je zariadenie, predá sa ako argument druhu objektu hodnota *aucObjDevice*. Prehľadávanie sa nastaví na *aucSearchDeep*, pretože

jednotlivé motory môžu byť rodičmi ďalším motorom a za úlohu je nájdenie všetkých motorov. Hodnota *aucAttrDesignation* predaná argumentu *attrID* zaistí, že sa pre každý motor porovná hodnota atribútu *Designation*. Ako porovnávaná hodnota je predaný výraz „1M*“. Znak „*“ sa postará o to, že sa vyberú nielen objekty s názvom „1M“, ale aj tie, ktoré týmto výrazom len začínajú.

```
46      With Motor.Attributes.ItemByID(aucAttrDesignation)
47        .Value = Replace(.Value, "1M", "2M", 1, 1, vbTextCompare)
48      End With
49      Call Motor.Store
```

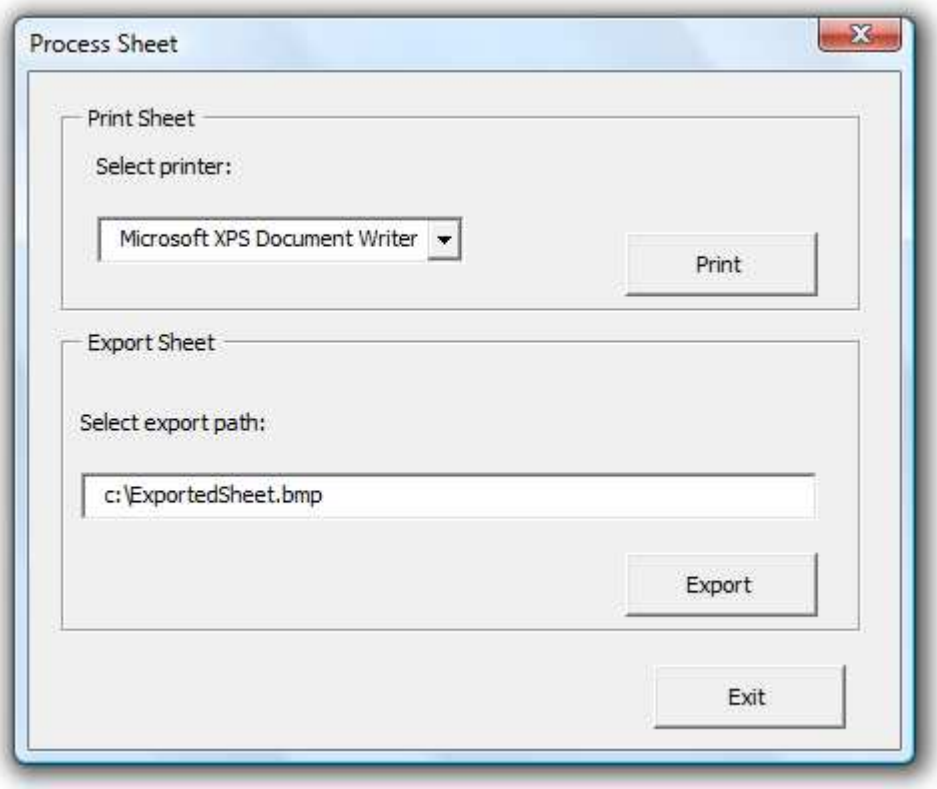
Komentár: V tejto časti kódu sa pomocou funkcie *Replace* zmení nájdený reťazec „1M“ v atribúte na „2M“. Po zmenení je potrebné ešte zavolať metódu *Store*, aby sa zmenený atribút zapísal do databázy.

5 ŠPECIALIZOVANÉ OBJEKTY ENGINEERING BASE

5.1 Trieda Sheet

5.1.1 Popis problému

Trieda *Sheet* je potomkom triedy *ObjecItem* a obsahuje špeciálne vlastnosti a metódy, ktoré sú určené na spracovávanie objektov typu *sheet*, teda listov v *Engineering Base*. V tejto úlohe sa využijú práve tieto vlastnosti a metódy na otvorenie, vyexportovanie na pevný disk a vytlačenie práve označeného listu. Na zvládnutie niektorých častí tejto úlohy bude potrebné využitie triedy *Visio.Application*.



Obrázok 31. Okno na výber tlačiarne a zložky pre export listu

Úlohou teda je vytvoriť makro, ktoré vyexportuje vybraný list vo formátu *BMP* do zložky, ktorá bude zadaná užívateľom a vytlačí pomocou vybranej tlačiarne, ktorú bude možné určiť pomocou *combo-boxu*.

5.1.2 Použité triedy, vlastnosti a metódy

Použité sú dve nové triedy. Trieda *Sheet*, ktorá je potomkom triedy *ObjectItem* a trieda

Visio.Application, pre použitie ktorej je potrebné pridať referenciu na knižnicu *Microsoft Visio Type Library*.

Výpis použitých tried:

- Trieda *Sheet* – pretože je zdedená z triedy *ObjectItem*, preberá od nej väčšinu metód a vlastností. Obsahuje však špeciálne, ktoré sú určené pre objekty typu *sheet*. V tabuľke (Tabuľka 8) sú uvedené vybrané a metódy tejto triedy.

  Tabuľka 8. Vybrané verejné metódy triedy Sheet

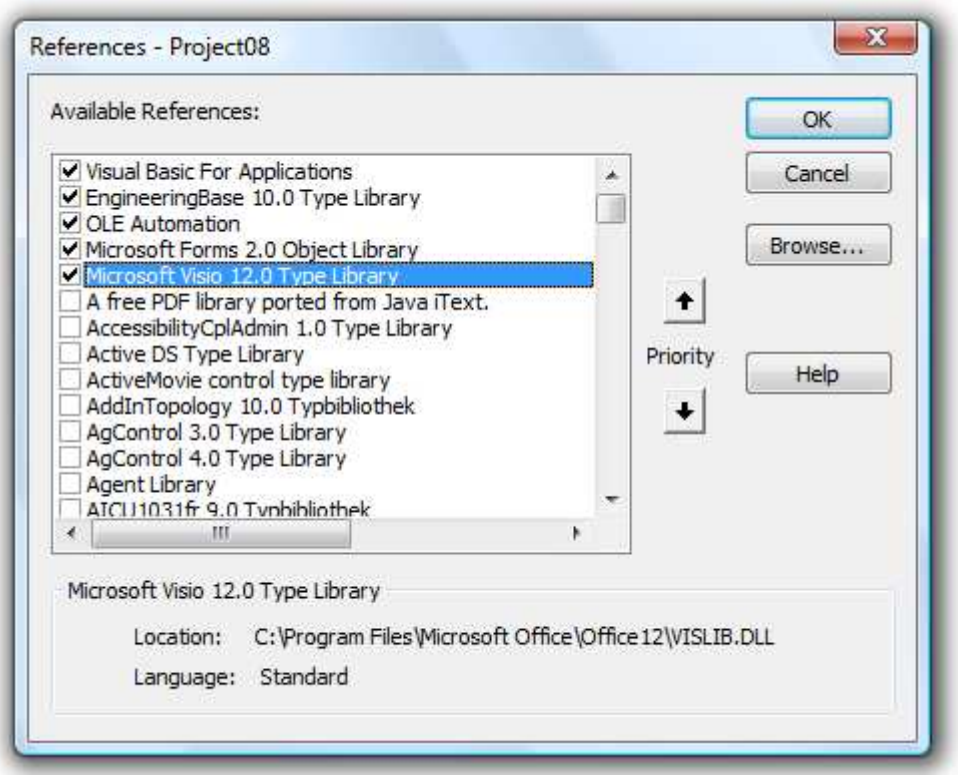
Verejné metódy	Popis
Open	Otvorí list v *MS Visiu*.
Close	Zatvorí list.
CopyTo	Skopíruje objekt s alebo bez jeho detí pod iný objekt
Export	Vyexportuje list do špecifického súboru v špecifickom formáte.
Activate	Aktivuje list.

- *Visio.Application* – využitá je jej vlastnosť *AvailablePrinters* a metóda *PrintOut*.
  - *AvailablePrintes* – je vlastnosť, ktorá vráti názvy nájdených tlačiarní ako pole prvkov typu *String*.
  - *PrintOut* – metóda, ktorá pošle požiadavok na vytlačenie listu. Obsahuje veľa parametrov, ktoré umožňujú napríklad zadať názov tlačiarne, ktorá má daný list vytlačiť. Vybrané parametre metódy:
    - *PrintRange* typu *VisPrintOutRange* – tento parameter určuje, ktoré listy sa majú aktuálne vytlačiť.
    - *FromPage* typu *Long* – parameter, ktorý určuje počiatočnú stranu, ktorá sa má vytlačiť.
    - *ToPage* typu *Long* – parameter, ktorý určuje koncovú stranu tlačenia.

- *PrinterName* typu *String* – parameter, ktorý určí tlačiareň, ktorá vykoná tlač.
- *Copies* typu *Long* – parameter, ktorý umožňuje vytlačenie viac kópii listov.

5.1.3 Pridanie referencie na knižnicu vo VBA editore

Kvôli potrebe použitia triedy *Visio.Application*, je potrebné pridať referenciu na túto knižnicu. Po zvolení položky menu *Tools – References* vyskočí okno s referenciami aktívneho makra. V ňom sa nachádza zoznam všetkých referencií, ktoré je možné k makru pridať.



Obrázok 32. Okno s referenciami projektu

Referencia sa vyberie označením check-boxu. Po vybraní požadovaných knižníc a stlačení tlačítka *OK* sa všetky vybrané knižnice pridajú k projektu.

5.1.4 Vysvetlenie zdrojových kódov

Takmer celý kód je napísaný vo formulári ako reakcia na stlačenie tlačítok. Hlavný modul tu slúži najmä na umožnenie spustiteľnosti makra z prieskumníka aplikácie *Engine-*

ering Base.

```
50 Public Sub Run()
51   Call PrintSheet.showFrame(Application.Selection(1))
```

Komentár: V hlavnom module sa získa označený list zo stromu prieskumníka a odošle sa verejnej metóde formulára.

```
52   For i = 1 To UBound(VisioApp.AvailablePrinters)
53     Call comboPrinters.AddItem(VisioApp.AvailablePrinters()(i))
54   Next i
55   comboPrinters.ListIndex = 0
```

Komentár: V tejto časti kódu sa prechádzajú všetky možné tlačiarne a pridávajú sa do *combo-boxu*. Potom sa prednastaví prvá tlačiareň z tohto zoznamu.

```
56 Private Sub buttonExport_Click()
57   Call selectedSheet.Export(textboxExportPath.Text, aucExportBMP)
58 End Sub
```

Komentár: Po stlačení tlačítka *Export*, sa označený list vyexportuje pomocou metódy *Export* do zložky podľa vloženého argumentu, ktorý obsahuje text z textového poľa. Druhému parametru metódy sa predá argument s hodnotou *aucExportBMP*, ktorý sa postará o to, aby vyexportovaný súbor bol formátu *BMP*.

```
59   Call selectedSheet.Open(0)
60   Call VisioApp.ActiveDocument.PrintOut(visPrintAll, , , ,
   comboPrinters.Text)
61   Call selectedSheet.Close
```

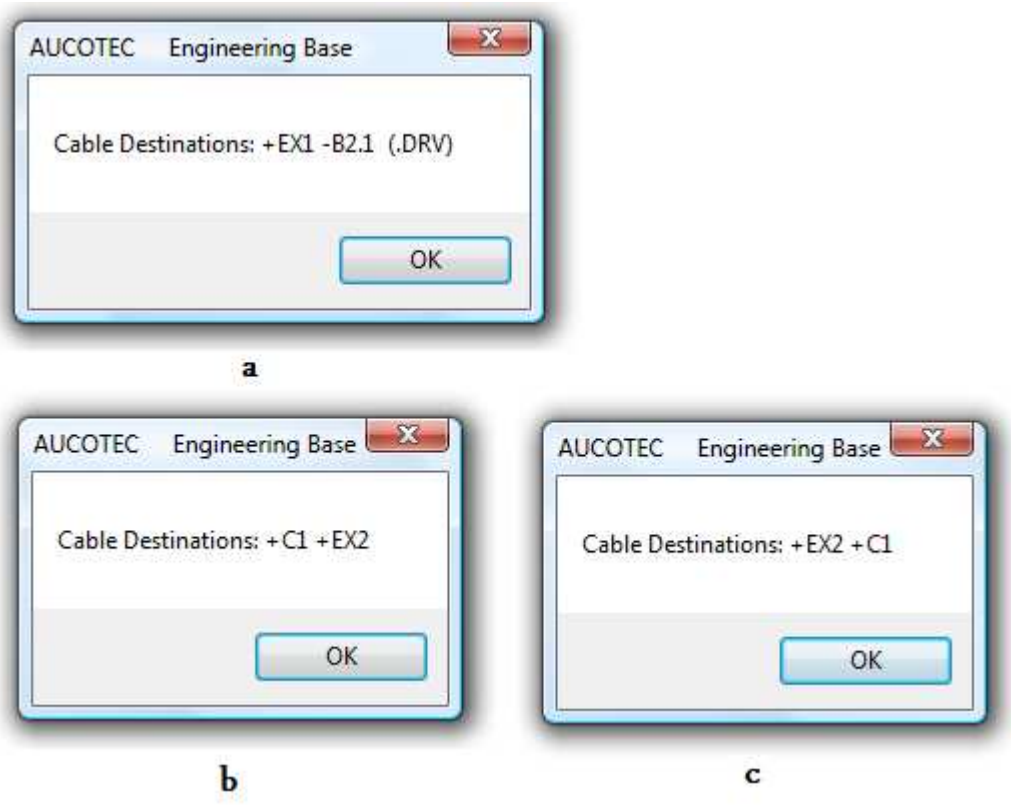
Komentár: Najprv sa otvorí aktuálny list pomocou metódy *Open* a potom sa pomocou metódy *PrintOut* a vloženého argumentu *visPrintAll* vytlačia všetky otvorené listy. Parametru *PrinterName* sa predá hodnota vybraná v *combo-boxe*, ktorá nesie meno tlačiarne, ktorá list vytlačí. Po odoslaní požiadavku na tlačenie sa otvorený list môže zatvoriť.

5.2 Trieda Cable

5.2.1 Popis problému

Trieda *Cable* sa používa pri práci s káblami, teda objektmi typu *Cable*. Pri spracovávaní týchto objektov je často krát potrebné zistiť koncové ciele káblu. Tieto ciele sa získajú pomocou vlastnosti *Destination*, ktorá vracia ich kolekciu. V *Engineering Base* je možné počet cieľov káblu ľubovoľne nastaviť, najčastejšie sa však počet cieľov pohybuje

okolo jedného alebo dvoch. V tejto úlohe bude treba zistiť ciele vybraného káblu, zobraziť ich pomocou okna správ a zmeniť ich. Metódou *Swap* ich napokon prehodiť.



Obrázok 33. Ciele káblu pred makrom (a), ciele káblu po zmenení (b), ciele káblu po prehodení (c)

Úlohou je teda zobraziť ciele vybraného káblu a zmeniť ich na ľubovoľne zvolené lokácie a po zmenení ich medzi sebou prehodiť.

5.2.2 Použité triedy, vlastnosti a metódy

V tomto makru je použitá nová trieda *Cable*. Táto trieda bola vytvorená pre prácu s objektmi typu *Cable*, ktoré majú špecifické vlastnosti. Jednou s týchto špecifických vlastností je napríklad vlastnosť *Destinations*, ktorá vracia kolekciu cieľov kábla. A pretože kábel obsahuje väčšinou drôty, má trieda *Cable* metódu na overenie, či všetky jej drôty odpovedajú cieľom káblu.

Výpis použitých tried:

- Trieda *Cable* – táto trieda je potomkom triedy *ObjectItem* a preto po nej preberá aj väčšinu vlastností a metód. Obsahuje však nové špecifické vlastnosti a metódy určené na prácu s káblami. V tabuľke (Tabuľka 9) sú uvedené vybrané metódy a vlastnosti tejto triedy.

Tabuľka 9. Vybrané metódy a vlastnosti triedy Cable

Verejné metódy	Popis
Store	Uloží kolekciu atribútov káblu od databáze.
VerifyDestination	Overí cieľové dáta káblu.
Verejné vlastnosti	Popis
Name	Vráti meno objektu.
Attributes	Vráti kolekciu objektov atribútov objektu.
Destinations	Vráti kolekciu objektov typu *CableDestination* cieľov káblu.

- Kolekcia *CableDestinations* – táto kolekcia sa získa pomocou vlastnosti *Destinations* triedy *Cable*. Jednotlivé objekty, ktoré obsahujú sú typu *CableDestination*. V tabuľke (Tabuľka 10) sú vybrané metódy a vlastnosti.

Tabuľka 10. Vybrané metódy a vlastnosti kolekcie CableDestinations

Verejné metódy	Popis
AddNew	Pridá nový cieľ do kolekcie.
Store	Uloží zmeny cieľových objektov.
Swap	Prehodí cieľové objekty.
Verejné vlastnosti	Popis
Count	Vráti počet prvkov v kolekcii.

- Trieda *CableDestination* – objekty tejto triedy sa nachádzajú v kolekcii cieľov *CableDestinations*. Jej vlastnosť *RelatedObject* sa využíva na získanie objektu, ktorý je cieľom pre konkrétny drôt.

5.2.3 Vysvetlenie zdrojových kódov

V tomto príklade sa celý kód nachádza v hlavnom modulu. Jediným grafickým výstupom je okno správ, ktoré sa volá niekoľkokrát a nesie informáciu o cieľoch vybraného drôtu.

```
62 Private Sub ShowCableDestinations(objectCable As Cable)
63   Call MsgBox("Cable Destinations: " _
64     & objectCable.Destinations(1).RelatedObject.Name _
65     & " " & objectCable.Destinations(2).RelatedObject.Name)
66 End Sub
```

Komentár: Definícia metódy *ShowCableDestinations*. Parametrom je objekt typu *Cable*. Táto metóda zobrazuje získané názvy cieľov káblu. Mená cieľov sa získajú pomocou vlastnosti *Name*, ktorá je volaná nad získanou triedou *ObjectItem*, ktorú vracia vlastnosť *RelatedObject* triedy *CableDestination*.

```
67   Set objectCable = Application.Selection(1)
68   Call ShowCableDestinations(objectCable)
```

Komentár: Získaný kábel sa predá ako argument metóde *ShowCableDestinations* na zobrazenie cieľov káblu.

```
69   With ActiveProject.EquipmentFolder
70     Set objectCable.Destinations(1).RelatedObject =
   .Children("+C1")
71     Set objectCable.Destinations(2).RelatedObject =
   .Children("+EX2")
72   End With
```

Komentár: Vlastnosť triedy *Cable* vráti kolekciu cieľových objektov typu *CableDestination*. Nad prvým a druhým objektov sa pomocou vlastnosti *RelatedObject* získa objekt typu *ObjectItem*, ktorý je cieľom pre daný kábel. Do tejto vlastnosti sa napokon priradí iný objekt vybraný zo zložky *Equipment*.

```
73 Call objectCable.Destinations.Swap(1, 2)
```

Komentár: Pomocou metódy *Swap* kolekcie *CableDestinations* sa prehodí prvý prvok s druhým.

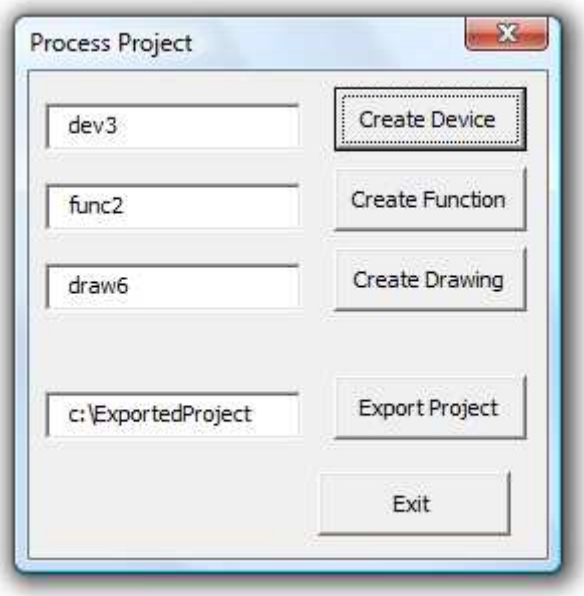
```
74 Call objectCable.Destinations.Store
```

Komentár: Po prevedených zmenách je potrebné ešte zavolať verejnú metódu *Store*, ktorá uloží zmeny do databáze.

5.3 Trieda Project

5.3.1 Popis problému

Úlohou tohto príkladu je oboznámiť užívateľa s triedou *Project*. Táto trieda sa používa pri práci s projektmi, pri ich exportovaní, importovaní, pri rýchlom prístupe k zložkám projektu. Často sa používa práve získaní koreňovej zložky zariadení, resp. funkcií. Napríklad pri hľadaní funkcie sa metóda *FindObject* spúšťa nad zložkou *Function*, ktorá sa rýchlo získa metódou *Project.FunctionFolder*.



Obrázok 34. Okno pre vytváranie zariadenia, funkcie, výkresu a pre zadanie cesty exportu projektu.

Úlohou je vytvoriť makro, ktoré vytvorí nový projekt a umožní v ňom vytvoriť nové zariadenia, funkcie a výkresy. Po spracovaní projektu umožní tento projekt vyexportovať na pevný disk do zadanej zložky.

5.3.2 Použité triedy, vlastnosti a metódy

V tomto príklade je použitá trieda *Project* a jej vlastnosti *EquipmentFolder, FunctionsFolder, DrawingsFolder* a metóda *Export.*

Výpis použitých tried:

- Trieda *Project* – táto trieda je potomkom triedy *ObjectItem* a preberá od nej veľa metód a vlastností. Dôležitá je jej vlastnosť *State*, ktorá nastavuje projekt do rôznych stavov, ako napríklad exkluzívny stav, stav *Read Only*. Prednosťou tejto triedy sú vlastnosti *EquipmentFolder*, *FunctionsFolder*, *DrawingsFolder*, *MessagesFolder*, ktoré umožňujú priamy prístup napríklad ku koreňovému objektu všetkých zariadení projektu, alebo všetkých funkcií atd. Ďalšou dôležitou metódou tejto triedy je metóda *Export*, ktorá umožňuje vyexportovanie daného projektu.

5.3.3 Vysvetlenie zdrojových kódov

Celý kód je umiestnený v jednom formulári, pričom z hlavného modulu sa iba zavolá verejná metóda tohto formuláru.

```
75 Set objectProject =
   Application.Folders.Projects.NewChild(aucObjProject)
```

Komentár: Po spustení makra sa vytvorí v zložke *Projects* nový projekt.

```
76 Private Sub buttonCreateDevice_Click()
77   Dim objectDevice As ObjectItem
78
79   Set objectDevice =
   objectProject.EquipmentFolder.NewChild(aucObjDevice)
80   objectDevice.Attributes.ItemByID(aucAttrDesignation).Value =
   DeviceName.Text
81   Call objectDevice.Store
82 End Sub
```

Komentár: Tento kód sa prevedie po stlačení tlačítka *Create Device*. Následne sa vytvorí nové zariadenie v zložke *Equipment* pomocou metódy *NewChild*. Novému objektu sa priradí meno podľa vyplneného textu v príslušnom textovom poli. Po zmenení mena zariadeniu, je samozrejme potrebné zavolať metódu *Store*, kvôli zápisu do databáze.

```
83   Set objectDrawing =
   objectProject.DrawingsFolder.NewChild(aucObjDrawing)
```

Komentár: Po stlačení tlačítka *Create Drawing* sa vytvorí nový výkres v zložke *Drawings*. Pre prístup ku koreňovej zložke výkresov sa využije vlastnosti *DrawingFolder* triedy *Project*. Pomocou metódy *NewChild* sa výkres vytvorí.

```
84 Set objectFunction =
   objectProject.FunctionsFolder.NewChild(aucObjFunction)
```

Komentár: Nová funkcia sa vytvorí v zložke *Functions*. Odkaz na túto zložku vráti vlastnosť *FunctionsFolder* triedy *Project*.

```
85 Private Sub buttonExportProject_Click()
86   Call objectProject.Export(ProjectPath.Text)
87 End Sub
```

Komentár: Po stlačení tlačítka *Export Project* sa zavolá metóda *Export* triedy *Project*. Ako argument sa predá cesta, ktorá bola vložená do príslušného textového pola.

ZÁVER

Cieľom tejto práce bolo stručne oboznámiť užívateľa so aplikáciou *Engineering Base* a uviesť ho do objektového modelu aplikačného rozhrania *Engineering Base* a potom navrhnúť príklady demonštrujúce možnosti rozšírenia systému *Engineering Base*.

Teoretická časť oboznamuje užívateľa s aplikáciou Engineering Base od firmy Aucotec. Na začiatku je vysvetlené k čomu táto aplikácia slúži a v niekoľkých bodoch je tu popísaná štruktúra tejto aplikácie. Rozpísané sú tu niektoré už vytvorené makrá, ktoré sú súčasťou aplikácii po inštalácii a ktoré môže užívateľ používať. V závere teoretickej časti je stručne načrtnutý objektový model rozhrania.

V praktickej časti sú rozpísané jednotlivé navrhnuté príklady. Každý príklad je tu podrobne rozpísaný tak, aby užívateľ nemal problémy s jeho pochopením a mohol zadaný úkol splniť. Príklady sú navrhnuté so snahou, čo najjednoduchšie vysvetliť daný problém a pritom zbytočne nezaťažovať užívateľa nepodstatnými komplikáciami, ktoré sa často v programovaní vyskytujú, pretože táto práca je zameraná skôr na rozhranie Engineering Base, ako na programovanie vo Visual Basicu.

Prvé príklady sa orientujú na základné triedy rozhrania a na prácu s nimi. Je tu vysvetlené ako získať označený objekt v klientovi aplikácie a ako s ním pracovať. Postupne v ďalších príkladoch sa preniká hlbšie do objektového modelu a nakoniec sa práca zaoberá špeciálnymi triedami, ktoré sú určené pre jednotlivé typy objektov.

RESULT

Theoretical part is talking about basics of Engineering Base application from Aucotec company. There is an explanation the purpose of this application and also application structure is described in couple of bullets. There are also described some of the prepared macros which are the part of application after installation and which can be used by user. In the end of theoretical part is a description of object model of the interface.

The practical part of thesis is about individual designed examples. Each of them is described in detail, so user shouldn't have a problem with understanding and can fulfill his issues. Examples are design to explain issues with no complications so user don't have to think about unimportant complications which are really common in programmers work because their work is especially aimed on Engineering Base instead of programming in Visual Basic.

First examples are oriented to the basic classes of the interface and to the work with them. There is an explanation of how to obtain indexed object in client application and how to work with this object. I next examples is deeper explanation of object model and in the end is thesis talking about special classes which are designated for individual object types.

ZOZNAM POUŽITEJ LITERATÚRY

[1] Technodat Elektro, s.r.o. Engineering Base - software pro tvorbu komplexní elektro dokumentace [online]. 2010 [cit. 2010-04-07]. Představení systému. Dostupné z WWW: <http://www.engineeringbase.cz/system-eb-predstaveni-systemu>.

[2] Engineering Base [online]. [s.l.] : [s.n.], 2010 [cit. 2010-04-07]. Technický popis softwaru, s.

[3] Technodat Elektro, s.r.o. Engineering Base - software pro tvorbu komplexní elektro dokumentace [online]. 2010 [cit. 2010-04-07]. Technický popis systému. Dostupné z WWW: <http://www.engineeringbase.cz/system-eb-technicky-popis-systemu>.

[4] Microsoft Visio. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 2010-04-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Microsoft_Visio>.

[5] Microsoft Office [online]. 2006 [cit. 2010-04-012]. Přehled aplikace Microsoft Office Visio 2007. Dostupné z WWW: <http://www.microsoft.com/cze/office/programs/visio/overview.mspx>.

[6] Microsoft Visio. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 2010-04-12]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Microsoft_Visio>.

[7] Microsoft. Microsoft Office Online [online]. 2010 [cit. 2010-04-20]. Microsoft Office Visio 2007 edition comparison. Dostupné z WWW: <http://office.microsoft.com/en-us/visio/FX101757911033.aspx>.

[8] Microsoft SQL Server. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 2010-04-20]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Microsoft_SQL_Server>.

[9] Microsoft SQL Server [online]. 2005 [cit. 2010-05-12]. Přehled produktu SQL Server 2005. Dostupné z WWW: <http://www.microsoft.com/cze/windowsserversystem/sql/prodinfo/overview/default.mspx>.

[10] Engineering Base [online]. [s.l.] : [s.n.], 2010 [cit. 2010-05-12]. Technický popis Engineering Base, s.

[11] Engineering Base [online]. [s.l.] : [s.n.], 2010 [cit. 2010-05-16]. EB 3-2 Technical Overview EN (ss), s.

[12] Visual Basic. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2003 [cit. 2010-05-21]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Visual_Basic>.

[13] Visual Basic for Applications. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2005 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Visual_Basic_for_Applications>.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

CBE Computer Base Engineering

VB Visual Basic

VBA Visual Basic for Application

DLL Dynamic Linked Library

EB Engineering Base

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha P I: CD s webovou prezentáciou, zdrojovými kódmi a súbormi

PRÍLOHA P I: CD S WEBOVOU PREZENTÁCIOU, ZDROJOVÝMI KÓDMI A SÚBORMI