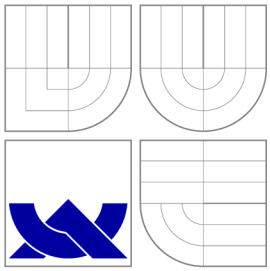
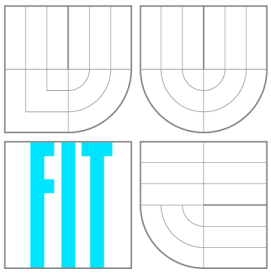


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

TVORBA DOKUMENTACE PROJEKTU Z ENGINEERING BASE

PROJECT DOCUMENTATION FROM ENGINEERING BASE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN DOMANSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2016

Zadání bakalářské práce/18832/2015/xdoman01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Domanský Martin**
Obor: Informační technologie
Téma: **Tvorba dokumentace projektu z Engineering Base**
Project Documentation from Engineering Base
Kategorie: Databáze

Pokyny:

1. Nastudujte prostředí Engineering Base a možnosti přístupu k datům pomocí webové služby.
2. Na základě získaných znalostí navrhněte program pro získání konkrétních dat z databáze Engineering Base.
3. Navrhněte přístup, kterým lze získaná data uložit do dokumentu MS Word (například jako konkrétní hodnoty v dokumentaci), aby později bylo možné tato data aktualizovat.
4. Navržený program implementujte a připravte demonstrační data.
5. Zvažte možnosti a omezení, které se mohou vyskytnout v podnikové sféře, a navrhněte pro ně alternativní řešení.

Literatura:

- Pírková, K.: Microsoft Word 2010 - Podrobná uživatelská příručka, Computer Press, 2010, ISBN 978-80-251-3033-9

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rozman Jaroslav, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá tvorbou dokumentací k elektro-technickým projektům nástroje Engineering Base. Pojednává o novém přístupu k datům z databáze Engineering Base a popisuje jeho využití. Praktickým cílem byl vývoj aplikace, která má sloužit jako základ pro komerční využití této funkcionality. Výsledná aplikace byla implementována s využitím .NET Frameworku a EB API pro Engineering Base verze 6.5.2. V práci je také popsáno testování základní aplikace a z poznatků z testování jsou vytvořeny návrhy možných rozšíření. Tyto návrhy mohou být využity pro další vývoj a řešení různých omezení.

Abstract

This bachelor thesis deals with creation of documentation for electro-technical projects based on the Engineering Base tool. It cintains the new access how to obtain data from the Engineering Base database and describes how it is used. The practical aim was to develop an application that should serve as the basis for the commercial use of this functionality. The application was implemented using the .NET Framework and EB API for Engineering Base version 6.5.2. The work also describes testing of the basic application, while the outcome of the testing provides suggestions for possible extensions. These suggestions can be used for further development and solving various restrictions.

Klíčová slova

Engineering Base, CBE, Web Services, WCS, EB API, Microsoft Word, Dokumentace

Keywords

Engineering Base, CBE, Web Services, WCS, EB API, Microsoft Word, Documentation

Citace

DOMANSKÝ, Martin. *Tvorba dokumentace projektu z Engineering Base*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.

Tvorba dokumentace projektu z Engineering Base

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Další informace mi poskytl Ing. Otakar Milink z firmy Technodat, Pouria G. Bigvand a Norbert Ott z firmy Aucotec AG. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Martin Domanský
16. května 2016

Poděkování

Rád bych poděkoval konzultantům z firem Technodat a Aucotec AG za odbornou pomoc s nástrojem Engineering Base a vedoucímu práce, Ing. Jaroslavu Rozmanovi, Ph.D., za pomoc při plnění formálních požadavků této bakalářské práce. Dále bych rád poděkoval firmě Technodat za prezentaci aplikace na veletrhu ELO SYS 2015. Za pomoc s gramatickou stránkou práce bych rád poděkoval Mgr. Ing. Radce Domanské z NKÚ v Praze a Bc. Daniele Šrubařové. Arigato gozaimasu!

Obsah

Kapitola 1

Úvod

Důležitou součástí elektrotechnických projektů je dokumentace. Ke každému projektu často náleží více dokumentací, které je nutné opakovaně aktualizovat. Aktualizace dokumentací je činnost, která je náročná na čas i lidské zdroje a může se při ní vyskytovat velké množství chyb. Mnohdy se na aktualizaci dokumentace zapomíná a tím se zvyšuje riziko chybné konstrukce projektu. Programově řízená aktualizace dokumentace omezuje výskyt chyb na minimum, je podstatně rychlejší a lze ji provádět automaticky.

Univerzální nástroj *Engineering Base* je využíván pro vytváření modelů elektrotechnických projektů. Tento komplexní nástroj se specializuje na tvorbu výkresů a dalších dokumentačních prvků. Nástroj *Engineering Base* je schopen požadavky na automaticky aktualizované dokumentační prvky zajistit, pokud se tyto prvky nachází v jeho databázi. Pro aktualizace dokumentací mimo databázi nástroje *Engineering Base*, je důležité zajistit lidské zdroje, které budou opakovaně provádět tento úkon.

Do verze 6.5.2 nástroje *Engineering Base* nebylo možné automaticky aktualizovat dokumentaci, která se nacházela mimo databázi tohoto nástroje, bez použití klasického uživatelské rozhraní. S verzí 6.5.2 bylo do jádra přidáno rozšíření, které tento přístup umožňuje.

Cílem této práce je analyzovat toto rozšíření a vytvořit program, který dokáže vytvářet a aktualizovat dokumentace s aktuálními daty mimo databázi nástroje *Engineering Base*, bez nutnosti použití klasického uživatelské rozhraní. Vytvořené dokumentace se mohou nacházet na lokálním úložisti nebo vzdáleném serveru.

Tato práce je rozčleněna do šesti kapitol. Kapitola 2 popisuje nástroj *Engineering Base* a části jeho struktury. Kapitola 3 obsahuje popis objektové reprezentace dat v nástroji *Engineering Base*. Součástí kapitoly 4 je návrh aplikace a detaily implementace. Kapitola 5 se věnuje testování a řešení problémů vzniklých při testování. V kapitole 6 jsou popsány řešení různých omezení a možná rozšíření, které je možné implmentovat v dalších verzích.

Kapitola 2

Engineering Base

Engineering Base [1, 2] je univerzální nástroj nové generace elektro-projekčních nástrojů vyvíjený německou firmou *Aucotec AG*, který svou koncepcí spadá do skupiny *Computer Based Engineering* systémů, tuto skupinu popisuji níže. *Engineering Base* je nástroj využívaný pro automatizaci sestavování složitých struktur přístrojů a svorkovnic, vytváření mnohožilových kabelů, návrh umístění jednotlivých předmětů u montážních desek či integraci *CRM*[1] prodejních dat do verzí plánovacích procesů.

Computer Based Engineering (CBE)

CBE [1, 2, 14] je přístup zpracování elektro-technických projektů, kdy jsou data uložena v hlavní relační databázi, a virtuální model projektu je vytvářen na aplikační vrstvě. Při práci s daty je možné použít grafické a nebo alfanumerické přístupy. Grafickým přístupem rozumíme například editaci výkresu, alfanumerickým například editaci dat v tabulkách. Všechna data jsou si rovnocenná a jsou zpracovávána a ukládána v grafické, tabulkové i stromové struktuře.

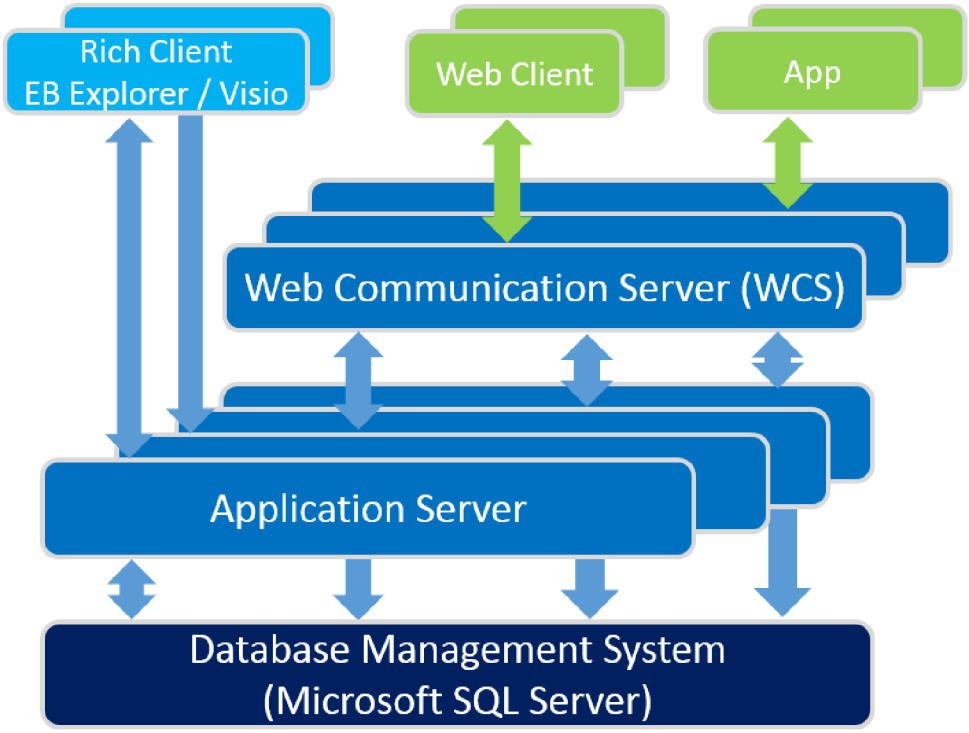
Zkratka *CBE* se používá pro označení elektro-projekčních nástrojů, které jsou založeny na obecné systémové databázi. Toto označení bylo zavedeno firmou *Aucotec AG* a vychází ze starších označení pro projekční nástroje – *CAD* a *CAE*. Typičtí představitelé skupiny *CAD* jsou *AutoCAD* a *CADELEC*, u skupiny *CAE* jsou to *ELCAD*, *RUPLAN*, *AUCOPLAN*, *EPLAN*, *SIGRAPH* a *PC Schematic*. Důležitou vlastností u všech výše uvedených projekčních systémů je vedle dostupnosti přístrojových databank (katalogy přístrojů) možnost programového přístupu k projektovým datům. Další vlastností je například doprogramování nových funkcí systému (tvorba Maker, kapitola 3.3), a to v různých programovacích jazycích: `Visual Basic`, `C++` či `C#`.

[1] CRM je zákaznicky podnikatelský přístup, který se vyznačuje aktivní tvorbou a udržováním dlouhodobě prospěšných vztahů se zákazníky. Tyto vztahy musí být prospěšné pro zákazníka i pro firmu, což vylučuje neetické chování k zákazníkům.

2.1 Struktura Engineering Base

Engineering Base je komplexní software sestavený převážně z produktů společnosti Microsoft. *Microsoft Visio* slouží pro vytváření a úpravu výkresů či schémat, *Microsoft SQL Server* slouží pro ukládání dat a *Visual Basic for Application* s *Microsoft .NET Framework* je určen pro vytváření jednoduchých maker[2]. *Engineering Base* správně pracuje pouze na operačním systému *Microsoft Windows*.

Aplikace *Engineering Base* má klasickou třívrstvou architekturu [3, 8], a to prezentační vrstvu, aplikační vrstvu a relační vrstvu. Od verze 6.5.2 je aplikační vrstva rozšířena o mezivrstvu pro komunikaci s webovými službami (kapitola 2.1.2). Jednotlivé vrstvy jsou popsány níže. Na obrázku 2.1 jsou zobrazeny hlavní části architektury *Engineering Base*. Pod prezentační vrstvu (kapitola 2.1.1) spadají všechny části, které mohou obsahovat grafické rozhraní. Jedná se o *Chytrého klienta*, *Webového klienta* a *Aplikace pro specifickou úlohu* komunikující přes webové služby. Pod aplikační vrstvu (kapitola 2.1.2) patří *Aplikační server* a *Webový komunikační server*. *Systém řízení báze dat* (*Microsoft SQL Server*) spadá do relační vrstvy (kapitola 2.1.3). Všechny hlavní části nástroje *Engineering Base* jsou podrobně popsány na další straně.

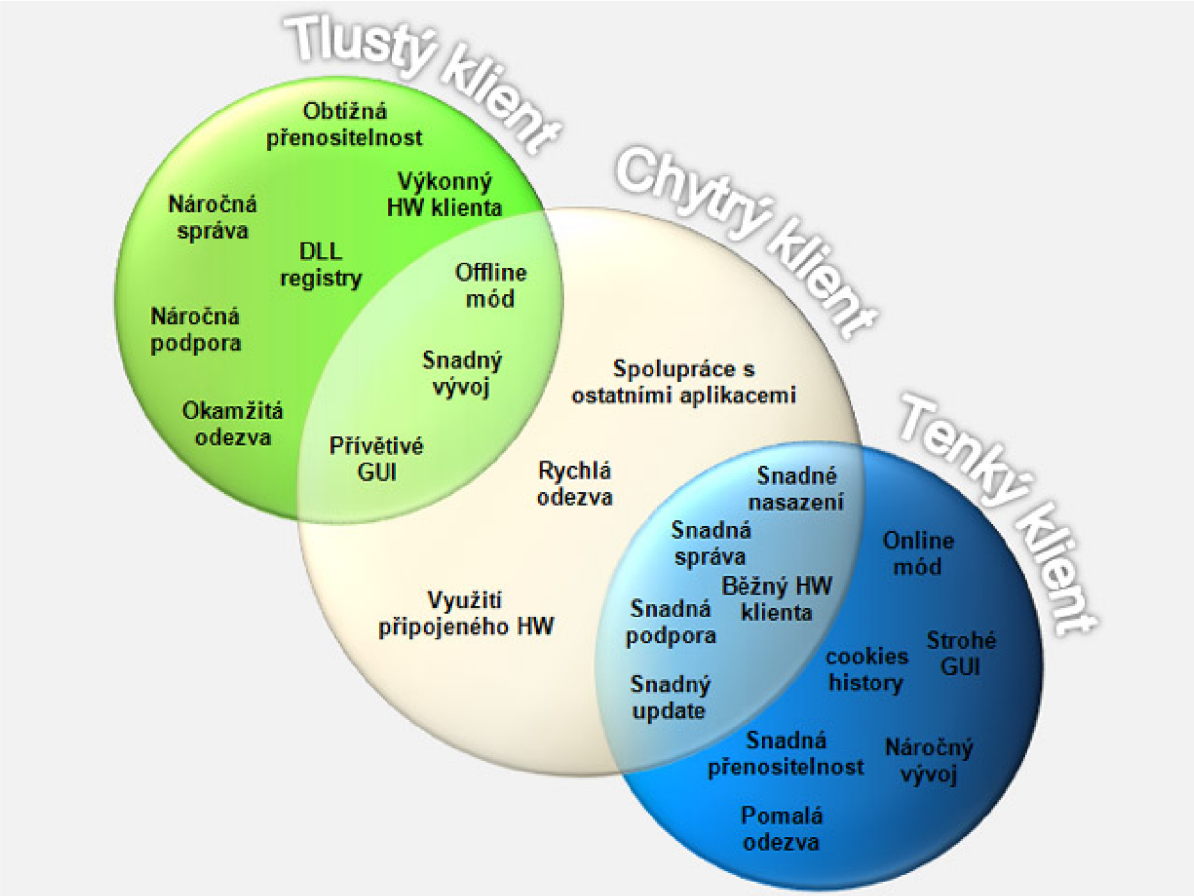


Obrázek 2.1: Hlavní části nástroje *Engineering Base*. Části nástroje nacházející se na horní části obrázku spadají do prezentační vrstvy, *Aplikační server* a *WCS* spadají do aplikační vrsty a *Systém řízení báze dat* spadá pod relační vrstvu. Části prezentační vrstvy, které komunikují s *Aplikačním serverem* skrze *WCS*, byly přidány od verze 6.5.2. [4].

[2]Makro je program zjednodušující práci v nástrojích *EB Explorer* a *Microsoft Visio*. Makra převážně komunikují pouze s aplikačním serverem, ale mohou obsahovat i grafické prostředí k interakci s uživatelem. (viz kapitola 3.3)

2.1.1 Prezentační vrstva

Prezentační vrstva [4, 7] obsahuje grafické rozhraní, které umožňuje uživateli pracovat s daty uloženými v databázi. Dále zajišťuje vstup požadavků a prezentaci výsledků. Pro práci s daty je možné použít grafické přístupy, například editaci výkresu v nástroji *Microsoft Visio*, a nebo alfanumerické přístupy, například editaci dat v tabulkách. Změny provedené na prezentační vrstvě jsou zpracovávány na aplikační vrstvě, proto jsou si oba přístupy rovnocenné. Na prezentační vrstvě se můžeme setkat s třemi hlavními částmi nástroje *Engineering Base – Chytrý klient*, *Tenký klient* a *Aplikace pro specifickou úlohu*.



Obrázek 2.2: Obecné vlastnosti jednotlivých klientů. Každý kruh obsahuje typické vlastnosti konkrétního klienta[6].

Chytrý klient (Rich client)

Chytrý klient [5] vhodně kombinuje výhody tenkého a tlustého klienta a potlačuje jejich nevýhody. Výhody a nevýhody různých klientů jsou popsány na obrázku 2.2. *Chytrý klient* obsahuje určitou logiku aplikace a lokálně si udržuje data, která se v okamžiku navázání spojení s databází synchronizují. Využívá místní i serverové systémové zdroje jako je například

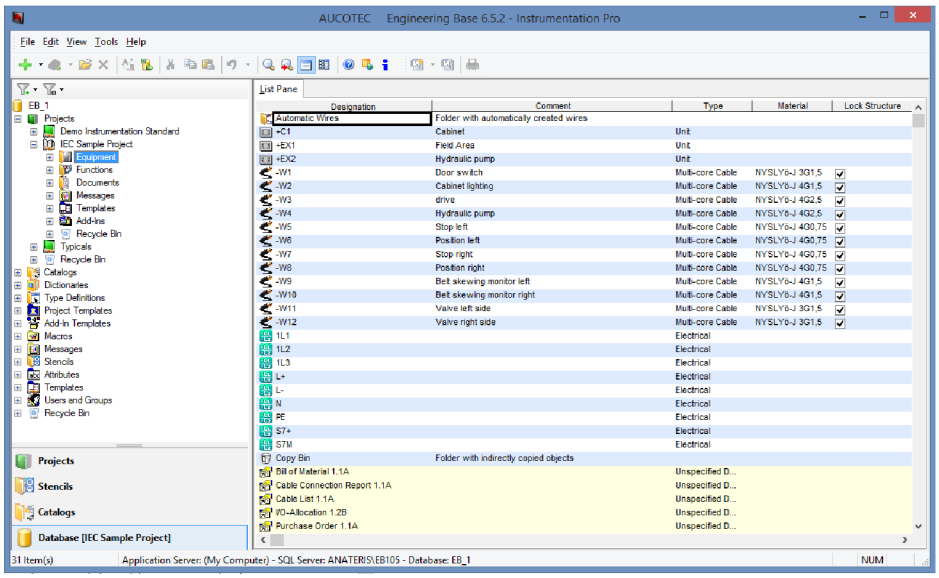
[4] Převzato a upraveno z: [2]
[6] Převzato a upraveno z: [5]

paměť, procesor a diskový prostor, ale může komunikovat a využívat i připojená zařízení. Stejně tak může využívat přítomnosti a funkcí, které nabízejí ostatní nainstalované aplikace např. *Microsoft Office*. Dále předpokládá existenci například *.NET frameworku*. Díky rozdělené zátěži mezi klienta i server, *Chytrý klient* snižuje HW nároky na pracovní stanici i server a zrychluje a zjednodušuje komunikaci. *Chytrý klient* poskytuje stejnou komunikaci s lokálním i vzdáleným serverem, a tedy umožňuje i práci v offline režimu na lokální databázi.

Fyzická přítomnost klienta na pracovní stanici umožňuje využít záznamů v registrech pracovní stanice, do kterých lze uložit nastavení pro specifické činnosti. To také zvyšuje univerzálnost použití nástroje *Engineering Base*, protože pouhou úpravou nastavení v registrech můžeme povolit nebo zakázat danou činnost. Povolení a omezení dané činnosti lze též nastavit přímo na konkrétní databázi. Tento přístup však není vhodný u společností, které aktivně pracují s více databázemi a jejich specialisté vlastní pracovní stanice pro jejich specifickou činnost[7], více v [2].

Chytrý klient umožňuje uživateli využívat:

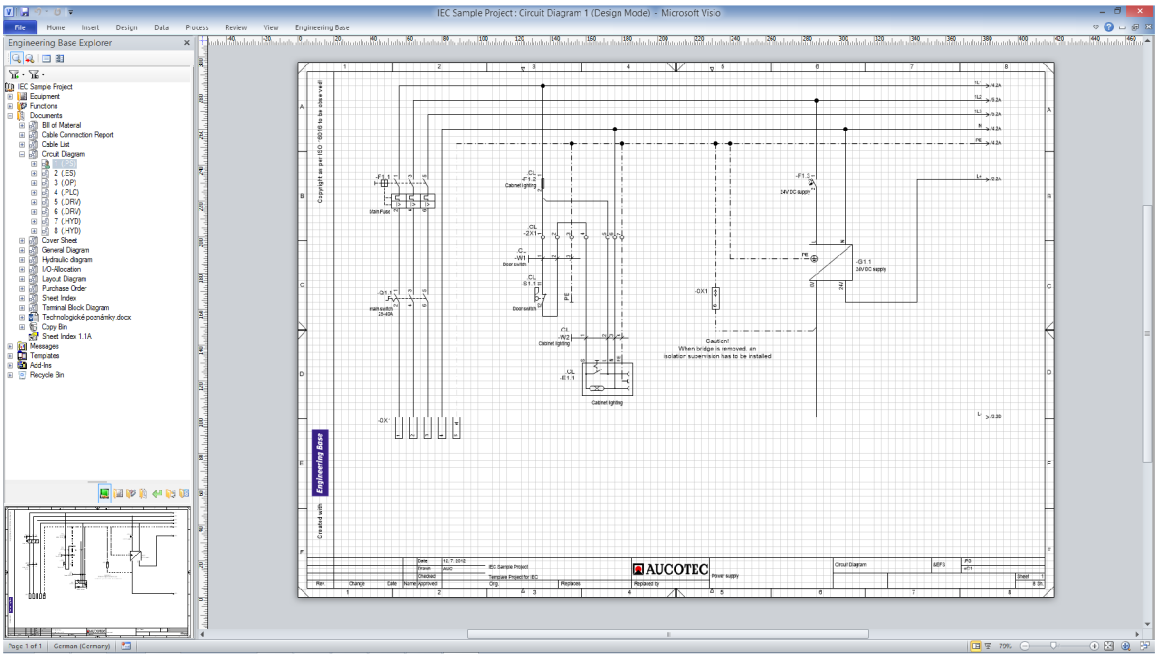
- EB Explorer – *EB Explorer* [1, 2] je nástroj, který lze využít pro ovládání celého systému a poskytuje uživateli alfanumerický přístup pro správu dat. Obsah databáze je reprezentován virtuálními objekty a je zobrazen ve stromové struktuře (tzv. databázový strom), což umožňuje uživateli lepší orientaci. Ukázka zobrazení projektu v *EB Explorer* je na obrázku 2.3. Uživatel má dále k dispozici náhled vybraného objektu a tabulkový přehled objektů pro jednodušší úpravu vlastností objektů.



Obrázek 2.3: Ukázka zobrazení projektu v *EB Explorer*. V levé části je stromová struktura databázového stromu, v pravé části je náhled obsahu označené položky.

[7] Často není potřeba, aby designér mohl měnit schéma zapojení, naopak je spíše vhodné mu omezit práva pouze pro jeho specifickou činnost, aby nemohl omylem poškodit projekt

- Microsoft Visio – *Microsoft Visio* [1, 2] je grafický nástroj k vytváření schémat a diagramů a poskytuje uživateli grafický přístup pro správu dat. Položky na výkrese mohou být propojeny s virtuálními objekty v databázovém stromě a změny provedené na výkrese se pak promítnou i do databázového stromu. *Microsoft Visio* může sloužit k vizuální dokumentaci, k návrhu nebo k porozumění obchodních procesů a systémů pomocí velkého množství diagramů. Uživatel může tvořit přehledy použitých materiálů, výpisy propojení konkrétních drátů z kabelů s konkrétním piny daných zařízení, seznamy položek vytvořených v projektu (kabely, zařízení, terminály), obecné schéma, schémata zapojení, kruhový diagram, diagram svorkovnice či diagram rozložení.



Obrázek 2.4: Ukázka zobrazení výkresu v nástroji Microsoft Visio. V levé části je stromová struktura databázového stromu a pod ní je náhled obsahu označené položky. V pravé části je zobrazen dokument, se kterým uživatel aktuálně pracuje.

Tenký klient (Web client)

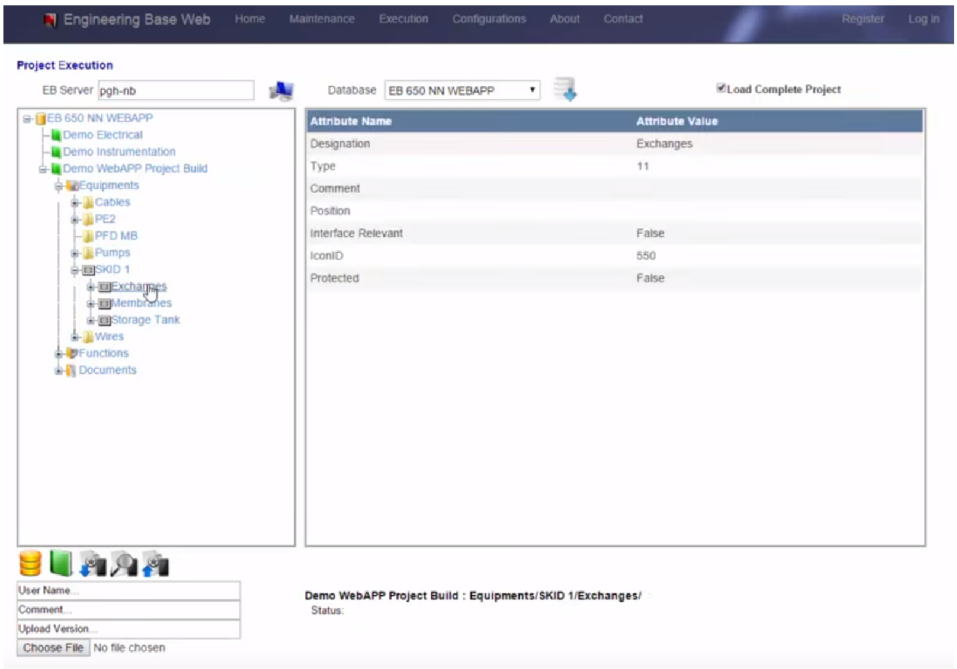
Tenký klient [5] je obvykle spouštěn přes webový prohlížeč, který zobrazuje prezentační vrstvu a komunikuje přes bezstavový `HTTP` protokol. Na *Tenkém klientovi* neprobíhá žádná rozhodovací logika, pokud nebereme v úvahu například validaci dat, zadávaných do webového formuláře. Z výše uvedené skutečnosti je zřejmé, že nároky na instalaci, konfiguraci a podporu tenkých klientů jsou minimální, protože uživateli takové aplikace stačí, když má na svém počítači nainstalován nějaký webový prohlížeč (IE, FF apod.). Naprosto tak odpadá starost o to, aby měl klient k dispozici vždy aktuální verzi, neboť ta je k dispozici všem klientům v okamžiku, kdy je nasazena na serveru. Komunikace mezi tenkým klientem a serverem je intenzivnější než v předchozím případě, ale na druhou stranu se mezi nimi přenese menší objem dat, protože k vlastnímu zpracování dat dochází na serveru a na klienta se přenáší pouze výsledek tohoto zpracování. Vývoj aplikace pro tenkého klienta je náročnější především z důvodu rozčlenění aplikace do jednotlivých vrstev. Určitou

nevýhodou je licenční politika nástroje Engineering Base, která omezuje počet aktivních připojení podle zakoupených licencí. Omezení licencí se však vztahuje pouze na aktivní připojení a po odpojení klienta se uvolní licence pro dalšího uživatele.

Tenký klient umožňuje uživateli využívat:

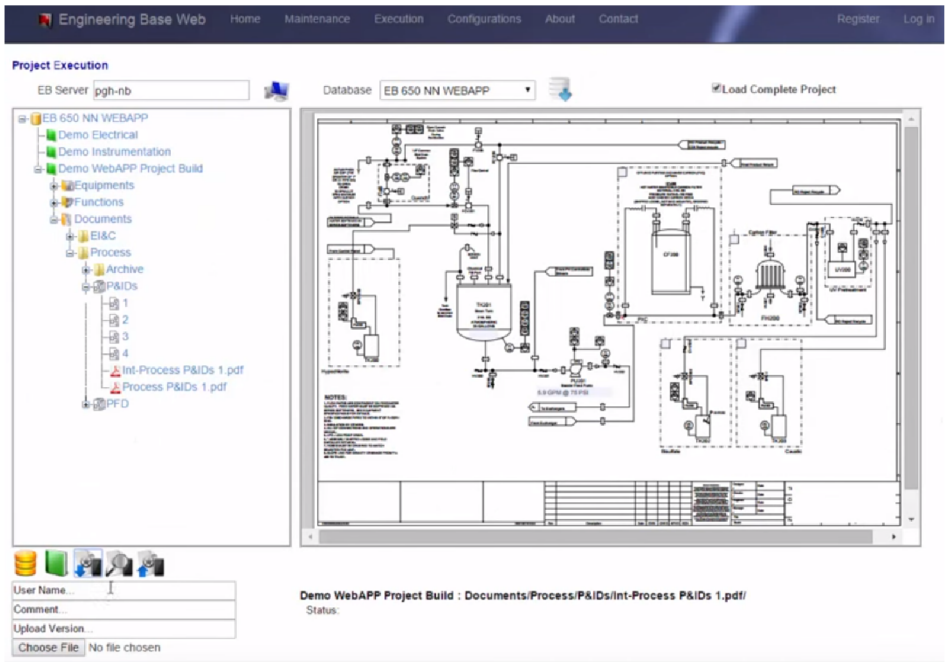
- Web Explorer - *Web Explorer* [2] je nová součást dostupná až od verze Engineering Base 6.5.2, kdy do aplikační vrstvy přibyl *Webový Komunikační Server* (WCS, kapitola 2.1.2) a je tedy možné přistupovat k datům, které jsou uloženy v databázi i bez spuštěného nástroje *EB Explorer*. V dřívějších verzích existoval pouze jediný způsob, jak pracovat s daty uloženými v databázi, a to pomocí *Chytrého klienta*. Aktuální verze *Engineering Base* 6.6.0 stále umožňuje pouze omezené vlastnosti oproti *Chytrému klientovi* a dovoluje pouze monitorování aktuálních dat objektů a vytvoření lokální kopie souborů z projektu.

  Na obrázcích 2.5 a 2.6 jde vidět zobrazení projektu a výkresu přes *Web Explorer*. Jak je vidět zobrazení projektu a výkresu pomocí *Tenkého klienta* je zjednodušeno oproti zobrazení přes *Chytrého klienta*. Použití *Tenkého klienta* je cíleho hlavně na mobilní zařízení, které nesplňují požadavky[8] pro bezproblémový běh *Chytrého klienta*.



Obrázek 2.5: Ukázka zobrazení projektu přes Web Explorer[9].

[8] Nemají dostatečný výkon nebo podporovaný operační systém.
[9] Převzato a upraveno z: [2]
[10] Převzato a upraveno z: [2]



Obrázek 2.6: Ukázka zobrazení výkresu přes Web Explorer[10].

Aplikace pro specifickou úlohu

V nástroji *Engineering Base* lze pro konkrétní a často opakované činnosti využít *Makro* (kapitola 3.3). Hlavní výhodou *Makra* je možnost provádět téměř jakoukoliv činnost. Hlavní nevýhoda *Maker* spočívá v tom, že je lze spustit pouze v rozhraní *Chytrého klienta* (jak v nástroji *EB Explorer*, tak i nástroji *Microsoft Visio*), a proto jsou zatím[11] nepoužitelné pro *Webového klienta*. Další nevýhoda spočívá v tom, že *Makro* není vždy možné spouštět periodicky, ani automaticky v reakci na specifickou událost.

Řešením pro výše zmíněné nevýhody může být *Aplikace pro specifickou úlohu* [2], která s *Aplikačním serverem* komunikuje přes *Webový Komunikační Server*. Aplikace tedy není závislá na *Chytrém klientovi* a může být spouštěna serverem periodicky[12] nebo v reakci na událost[13]. Navíc se *Aplikace pro specifickou úlohu* nemusí omezovat na využití *EB API* a programovacích jazyků (`VBA` a `C#`), které jsou podporovány nástrojem *Engineering Base*, ale mohou být vytvořeny (nebo jejich části) skoro v libovolném jazyce. Nicméně *EB API* značně zjednodušuje práci s daty v databázi nástroje *Engineering Base* a proto je výhodné jej využívat.

Podobně jako u *Webového klienta* je podpora *Aplikací pro specifickou úlohu* bez běžícího *Chytrého klienta* dostupná až od verze 6.5.2, ale na rozdíl od *Webového klienta*, byly *Aplikace pro specifické úlohy* využívány již v dřívějších verzích nástroje *Engineering Base*. *Aplikace pro specifické úlohy* se využívaly jako rozhraní (most) pro komunikaci nástroje *Engineering Base* s programy třetích stran. Tato komunikace však probíhala skrze *Chytrého klienta*.

[11] Je pravděpodobné, že v dalších verzích nástroje *Engineering Base* bude možné používat *Makra* i ve *Webovém klientovi*.

[12] Jako naplánovaná úloha na serveru.

[13] Například databázovým trigrem po aktualizaci dat v projektu.

2.1.2 Aplikační vrstva (Business vrstva)

Aplikační vrstva [3, 4, 7] je prostřední část třívrstvé architektury, ve která obsahuje aplikační logiku celého aplikace. Je to vrstva, do které by mělo být soustředěno maximum výkonného kódu, protože zajišťuje komunikaci s relační vrstvou a prezentační vrstvě poskytuje služby pro práci s daty. Tato vrstva obsahuje třídy reprezentující entity zajišťující logiku aplikace a provádí výpočty a datové manipulace na základě vstupů a uložených dat.

Aplikační server

Aplikační server [2] nástroje *Engineering Base* je implementován jako samostatná jednotka, která bývá nainstalována na pracovní stanici nebo separátním serveru. *Aplikační server* umožňuje víceuživatelský přístup, kdy více uživatelů může nezávisle na sobě pracovat nad stejnou databází. Pro každého uživatele vytvoří *Aplikační server* unikátní proces, který obsluhuje požadavky uživatele a komunikuje s příslušnou databází. Při práci více uživatelů nad stejnou databází mohou vznikat konflikty[14]. *Aplikační server* detekuje konflikty u objektů pomocí Timestamps[15] a řeší je podle pravidel, které si zvolí společnost vlastnící licenci.

Komunikace *Aplikačního serveru* s relační vrstvou je zajišťována pomocí proprietární technologie **DCOM**[16] od firmy Microsoft. Technologie **DCOM** umožňuje sdílet objekty v binární podobě i mezi vzdálenými zařízeními, více v [9, 10, 11, 13].

Webový Komunikační Server (WCS)

Webový Komunikační Server [2] je další součást aplikační vrstvy, která zajišťuje komunikaci mezi *Aplikačním serverem* a *Tenkým klientem* či *Aplikací pro specifickou úlohu*. *WCS* je realizován jako služba, která zpracovává, validuje a předává zprávy **SOAP**. Pro každé nové spojení uživatel-databáze vytvoří *WCS* proces, který obsluhuje danou komunikaci. Pro každou databázi lze definovat maximální počet aktivních spojení. Protože se jedná o bezstavovou komunikaci, *WCS* ukončí proces pro spojení uživatel-databáze ve chvíli, kdy uživatel požádá o další spojení uživatel-databáze k jiné databázi. *WCS* běží na stejném zařízení jako aplikační server, ale není vyžadován pro správnou funkci nástroje Engineering Base. *Webový Komunikační Server* je pouze rozšířením pro aplikační server a byl přidán až ve verzi 6.5.2.

Webový Komunikační Server poskytuje dva základní druhy služeb, „BasicDataService“ a „TranslationService“, pro komunikaci s aplikačním serverem a každá z nich obsahuje čtyři služby pro specifickou činnost. Komunikace probíhá pomocí dotazů a odpovědí ve formátu `XML`, které mají přesně definované struktury.

[14] Například pokud uživatel chce měnit data objektu, který aktuálně používá jiný uživatel.
[15] Časové razítko – posloupnost znaků sloužící pro rozpoznání, kdy nastala určitá událost.
[16] Distributed Component Object Model

BasicDataService

Mezi „BasicDataService“ spadají služby, které poskytují informace o databázích a jejich obsahu:

- GetDatabases - Služba poskytuje seznam databází, které jsou dostupné.
- GetObjectData - Služba prohledává databázový strom a poskytuje data o konkrétní objektu, který je definován dle OID. Lze získat jeho atributy nebo objekty, dle druhu a typu, nacházející se pod ním v databázovém stromě.
- GetProjectData - Služba poskytuje seznam dostupných projektů a katalogů z dané databáze.
- UpdateObjectData - Služba umožňuje vytvořit, upravit nebo smazat konkrétní objekt. Při modifikaci objektu je možné upravit hodnoty atributů nebo vazby mezi objekty.

TranslationService

Mezi „TranslationService“ spadají služby, které se využívají pro lokalizaci a práci se slovníkem textů.:

- GetDictionary - Služba poskytuje překlady textových řetězců z databázového slovníku. Lze vytvořit dotaz pro množinu textů i množinu jazyků. V odpovědi lze nalézt překlad do daného jazyka (pokud existuje) nebo původní řetězec.
- GetDictionaryInfo - Služba poskytuje seznam slovníků a jazyků, které poskytují, z dané databáze.
- GetTranslation - Služba poskytuje překlad referencí ze systémového slovníku nebo slovníku projektu.
- UpdateDictionary - Služba umožňuje aktualizaci databázového slovníku.

2.1.3 Relační vrstva

Relační vrstva [3, 4, 7] je nejnižší vrstva a přijímá požadavky od vyšší *Aplikační vrstvy* (kapitola 2.1.2). Zajišťuje komunikaci s databází, řídí připojení a odpojení od databázového zdroje, spravuje databázové relace (sessions), zajišťuje podporu transakcí, poskytuje **DAO**[17] objekty realizující **CRUD**[18] funkcionalitu pro jednotlivé třídy objektů. Zajišťuje dále mapování databázových struktur na objekty příslušných tříd.

Microsoft SQL Server

Microsoft SQL Server [6] (dále jen *SQL Server*) je relační databázový a analytický systém od společnosti Microsoft, více v [12]. Mezi jeho hlavní dotazovací jazyky patří `SQL` (Structured Query Language) a `TSQL` (Transact-SQL). *SQL Server* původně zahrnoval pouze databázový modul pro zpracování **OLTP**[19] a komponenty replikace určené k distribuci dat.

[17] Data Access Object - objekt pro přístup k datům, je to abstraktní objekt, který poskytuje specifické operace s daty.

[18] Create Read Update Delete - čtyři základní operace nad záznamem v databázi – vytvoření, čtení, editace, smazání.

[19] OnLine Transaction Processing

Později se *SQL Server* vyvinul do podoby komplexní datové platformy, která dokáže obsluhovat požadavky na ukládání, manipulaci a prezentaci dat v rámci celé společnosti.

Microsoft SQL Server umožňuje:

- Navrhnout, zabezpečit a spravovat relační databáze,
- Načítat data a manipulovat s nimi,
- Zabezpečit a obnovit podniková data,
- Zajistit výkon a odolnost databáze,
- Transformovat podniková data na prvky buseness intelligence.

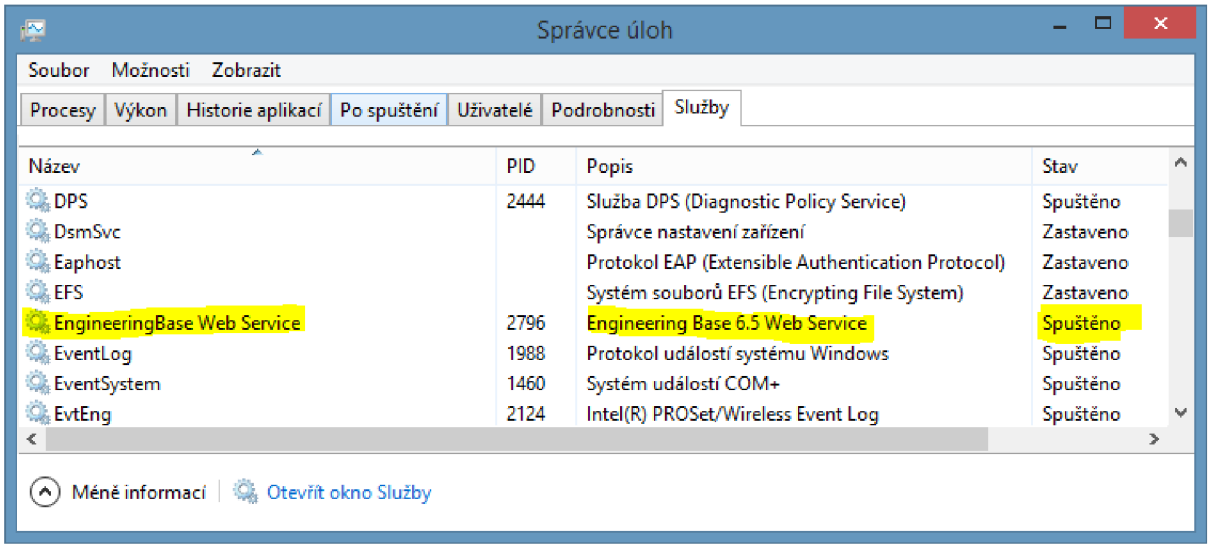
2.2 Engineering Base API (EB API)

Engineering Base obsahuje specifické/vlastní API, které umožňuje pracovat s objekty na aplikační vrstvě a poskytuje služby pro práci s daty uloženými v databázi. API je možné využít v jazycích `VBA` a `C#` a je rozděleno na služby pro klienta a pro server, více v [2].

Služby pro klienta vyžadují běh *Chytrého klienta* a využívají se primárně v implementaci *Maker*. Mohou spolupracovat s prohlížečem *EB Explorer* i nástrojem *Microsoft Visio*, a dokonce spouštět další *Makra*. Vytvořené programy lze aplikovat jako *Makra*, která zjednodušují uživateli práci, nebo jako procesy běžící na pozadí *Engineering Base*, které v reálném čase reagují na akce uživatele. Mezi nejčastěji využívanými službami pro klienta patří získání všech uživatelem označených objektů ve stromové struktuře či na výkrese, získání aktuálně otevřeného projektu a vytvoření seznamu objektů dle aplikovaného filtru.

Služby pro server oproti službám pro klienta nevyžadují běh *Chytrého klienta*, není však možné využít funkcionalit jiných maker. Jejich výhoda tkví v tom, že umožňují práci s aplikační vrstvou a databází bez spuštěného klasického klienta[20]. Díky nim lze implementovat vlastní uživatelské rozhraní s omezenou funkcionalitou, či uživatelské rozhraní pro specifické úlohy a úkony. Protože nevyžadují běh klasického klienta, nelze využívat služby pro získání instance aktuálního projektu či seznamy objektů dle filtrů. Lze získat pouze stromovou strukturu databáze (případně pouze stromovou strukturu daného projektu) nebo specifický objekt dle jeho ID. Služby pro server využívají pro komunikaci s aplikační vrstvou *Webový Komunikační Server* (kapitola 2.1.2).

[20] S daty z databáze lze pracovat i ze zařízení, které nemá dostatečný výkon plynulý běh klasického klienta nebo nemá podporovaný operační systém.



Obrázek 2.7: Služba *WCS* umožňující komunikaci s *Aplikačním serverem* bez spuštěného klasického klienta.

Kapitola 3

Objektová reprezentace v Engineering Base

Všechna data, kter8 jsou uložená v databázi, jsou pomocí *Aplikačního serveru* transformovaná na objekty. Základním objektem je objekt typu `ObjectItem`, který obsahuje atributy typu `AttributeItem`. *Aplikační server* nástroje *Engineering Base* podporuje polymorfismus a `ObjectItem` je objektem, ze kterého vycházejí všechny ostatní objekty, více v [2].

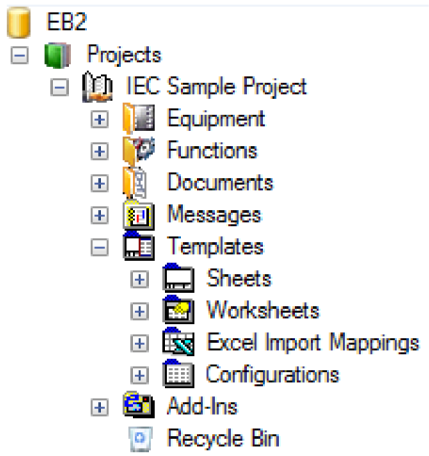
3.1 Objekty a atributy

Každý objekt v databázi je jednoznačně identifikovatelný pomocí Object ID (zkráceně OID), které je v celé databázi unikátní a ani po smazání příslušného objektu nebude OID přiděleno dalšímu objektu. OID je hexadecimální číslo o 32 pozicích a je přidělováno postupně od nejnižšího čísla[1]. Maximální počet objektů v databázi je tedy přes $1,208 \times 10^{24}$. Takového počtu objektů v jedné databázi je však v praxi nereálné dosáhnout. Kromě unikátního OID mezi základní atributy, které musí každý objekt obsahovat, patří také označení pro specifický druh (Kind nebo také CID), vlastní označení (Designation), typ (TID), ikona a komentář. Další atributy objektu jsou již specifické pro konkrétní druh a typ objektu.

Ze základního objektu `ObjectItem` je odvozeno několik specifických druhů objektů, jako například *Project*, *Macro*, *Sheet*, *Document* a další. Tyto objekty jsou unikátní pro specifické použití a bývají umístěny ve speciálních složkách. Některé z objektů jsou popsány dále v této kapitole.

[1] Vrchol databázového stromu má tedy vždy OID 00000000-0000-0000-0000-000000000001.

3.2 Projekt



Obrázek 3.1: Databázový strom, který zobrazuje ukázku struktury *Projektu* a základní části.

Projekt [2] slouží jako složka, která může obsahovat různé zařízení, funkce, dokumenty nebo šablony. Jako zařízení uvažuji soustavy kabelů, vodičů, koncovek ale i dalších elektrotechnických součástek. Každý *Projekt* má pevně definovanou strukturu složek a každý objekt projektu musí být uložen v příslušné složce podle jeho druhu. Všechny Projekty jsou imístěny ve složce „Projects“ (česky Projekty, obrázek 3.1).

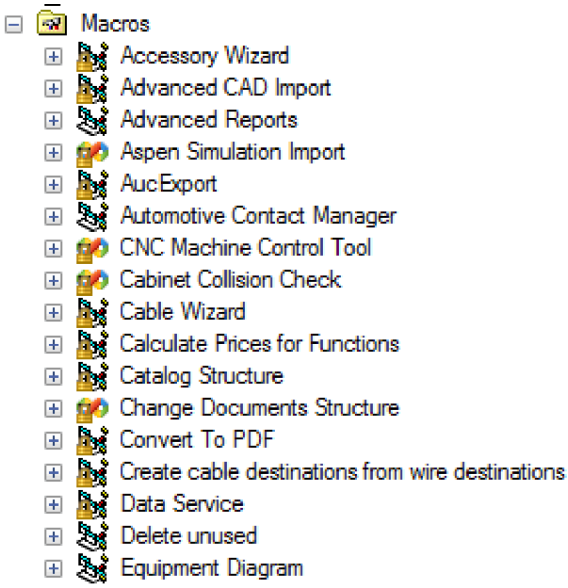
3.2.1 Aktualizace dat projektu v Engineering Base

Projekt v *Engineering Base* může obsahovat různé dokumenty a dokumentační soubory (kapitola 3.4). Objekty v *Projektu* je možné průběžně upravovat a aktualizovat a dokud se nachází v databázi *Engineering Base*, změny provedené na výkrese se promítnou i v objektech ve stromové struktuře a naopak, více v [2].

Některé změny, jako například aktualizace katalogu součástek, mohou vést ke změně uvnitř několika projektů. Důsledkem toho může uživatel, při manuálním exportu nebo při použití *Makra*[2], opomenout exportování novější dokumentace u aktualizovaných projektů. Tento stav může způsobit vážné problémy, pokud se například ve schématu začne používat novější součástka, ale konstruktér má ve schématu zapojení stále uvedenou starou a již nepoužívanou součástku. Pro eliminaci podobných situací je nutné po provedení změny automaticky vytvořit novou verzi dokumentace nebo zajistit, aby se pravidelně softwarově kontrolovala její aktuálnost a v případě změn se vytvořila nová verze.

[2]Určitá Makra mohou exportovat vybrané výkresy v projektu do specifického formátu, který není standartně podporován v *Engineering Base* (např. DWG)

3.3 Makro



Obrázek 3.2: Speciální složka obsahující *Makra* psaná v jazycích `VBA` a `C#`.

Objekt druhu *Makro* [2] je specifický objekt, který je využívaný pro zjednodušení opakujících se činností v nástroji *Engineering Base*. *Makro* obsahuje jednoduchý program vytvořený v jazycích `VBA` nebo `C#`, který je využívaný pro specifickou činnost. *Makra* mohou být také využita pro doprogramování nových funkcí a mohou také využít funkcionalit jiných, již existujících *Maker*. Při implementaci se využívá *EB API* (kapitola 2.2), které obsahuje třídy a metody zjednodušující práci s daty uloženými v databázi *Engineering Base*. Funkcionalita *Makra* není omezená pouze na prostředí *Engineering Base*. Je možné spolupracovat i s jinými aplikacemi, vytvářet a zpracovávat soubory v souborovém systému nebo využívat služby, které poskytuje operační systém.

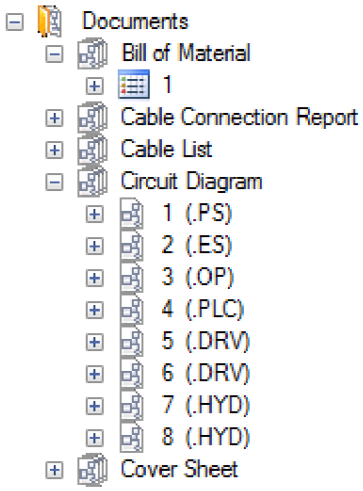
Pro větší přehlednost je každé *Makro* uloženo ve speciální složce jménem „Macros“ (česky Makra) jak jde vidět na obrázku 3.2. Každé *Makro* lze obecně spustit nad jakýmkoliv objektem v prostředí *EB Explorer* či nástroji *Microsoft Visio*. Omezení pro použití *Makra* lze nastavit ve zdrojovém kódu *Makra* nebo požadavkem na určité licence.

3.4 Dokumenty

Dokument je další specifický objekt. Stejně jako u *Maker* jsou všechny objekty druhu *Dokument* vždy umístěné ve specifické složce „Documents“ (česky Dokumenty, obrázek 3.3). Tato složka se nachází v každém *Projektu* (obrázek 3.1), protože patří mezi základní části struktury objektů druhu *Projekt*.

Jak vidíme na obrázku 3.3, mezi nejčastější *Dokumenty* patří výkresy (objekt druhu *Výkres*), které mohou obsahovat jednotlivé listy výkresu nebo různé záznamy[3]. Kromě výkresů může být objekt druhu *Dokument* jakýkoliv textový i grafický soubor a je možné, aby tyto *Dokumenty* byly využity v *Projektu* či *Makru*.

[3]Například záznamy o použitých materiálech u kabelů a dalších prvků, které jsou použity v projektu.



Obrázek 3.3: Struktura složky „Documents“ obsahující objekty druhu *Dokument*. Na obrázku jsou vidět výkresy obsahující záznamy či diagramy.

3.5 Zprávy

Zpráva je speciální druh objektu, která se využívá jako záznam události. Tyto *Zprávy* většinou obsahují popis události, komentář, kategorii[4] zprávy a čas. Každá *Zpráva* může také obsahovat odkaz na objekt, ke kterému se vztahuje. *Zprávy* jsou umístěny ve složce „Messages“ (česky Zprávy, obrázek 3.1). Dle umístění se *Zprávy* vztahují k projektu nebo databázi[5].

[4]Nejčastěji se jedná o kategorie *Information*, *Warning* a *Error*.
[5]Složka „Messages“ se nachází v databázovém stromě i mimo *Projekt*.

Kapitola 4

Návrh a implementace

V této kapitole je popsána specifikace požadavků, návrh, architektura a implmentace aplikace. Pro aplikaci jsem použil pracovní návrh „EB2Word“.

4.1 Specifikace požadavků

Hlavním požadavkem je vytvořit funkcionalitu, která umožní uživateli dohledat, přenést a uložit data konkrétních objektů uložených v databázi *Engineering Base* mimo prostředí *Engineering Base* a později tyto data aktualizovat.

Výsledná aplikace by měla být jednoduchá a snadno rozšířitelná o nové funkce. Je žádoucí, aby aplikace nevyžadovala spuštěné prostředí *EB Explorer* a nebyla závislá na specifické verzi nástroje *Engineering Base*. Dokumentace by měla být vytvořená dokumentem *Microsoft Word*.

4.2 Návrh

Před návrhem aplikace jsem se snažil zjistit, jestli existuje podobný projekt. Nicméně jsem neobjevil žádný dostupný projekt, který by se zabýval automatickou aktualizací dokumentací mimo nástroj *Engineering Base*. Inspirací pro návrh části aplikace, pro mě byl projekt *Web Services* (kapitola 4.3), na jehož vývoji jsem se podílel, a který se zabývá získáváním dat z databáze *Engineering Base* bez využití klasického rozhraní. Proto jsem se rozhodl využít projekt *Web Services* pro návrh komunikace s nástrojem *Engineering Base* pomocí *WCS*.

Abych zjednodušil další vývoj aplikace, navrhl jsem pouze základní aplikaci, která bude sloužit jako základ ke komerčnímu využití. Základní aplikace obsahuje všechny vyžadované funkcionality a je snadno rozšířitelná o další funkce. Problematikou se podrobně zabývám v kapitolách 4.4 a 4.5. Návrhy dalších funkcionalit, které jsem vytvořil na základě výsledků testování, jsou popsány v kapitole 6.

4.3 Web Services

Web Services je projekt, který slouží pro testování a prezentování *Služeb pro server*. Je implementován jako jednoduchá stand-alone aplikace s uživatelským rozhraním, navržená firmou Aucotec AG. Původně projekt *Web Services* sloužil pouze pro testování funkční komunikace s Aplikačním serverem skrze *WCS*. Z důvodu potenciální možnosti využití aplikace

k prezentačním a školícím účelům jsem navrhl a implementoval rozšíření tohoto projektu o uživatelské rozhraní a o demonstrační příklady pro jednotlivé služby, které poskytuje WCS (kapitola 2.1.2).

Web Service je aktuálně využíván jako vzorový projekt pro budoucí aplikace využívající *Služby pro server*. Obsahuje totiž vzorové příklady a postupy pro správné nastavení struktur pro dané služby. Různé metody jsou vytvořeny pro univerzální použití i pro specifické úlohy. Použitím těchto metod lze rychle a jednoduše vytvořit prototyp nové aplikace.

```
/// <summary>
/// Looks for all object under parent, defined by kindList, with attributes specified by AttrInfo
/// </summary>
/// <param name="dbName">Name of database where you search</param>
/// <param name="parentId">ID of start object</param>
/// <param name="kindList">List of kind of allowed objects</param>
/// <param name="attrInfo">List of wanted attributes</param>
/// <returns>List of found items or null</returns>
public IList<ObjectItem> GetObjectChildrensWithSpecificAttributes (string dbName, string parentId,
    List<ObjectKindInfo> kindList, List<AttributeInfo> attrInfo)
{
    var request = new ObjectDataRequest
    {
        StartObjectId = parentId,
        SearchMode = SearchMode.Deep,
        OutputAttributeList = attrInfo,
        OutputObjectKindList = kindList
    };
    return ExecuteRequest(dbName, request);
}
```

Obrázek 4.1: Příklad z ukázkové metody z projektu Web Services. Pomocí této metody lze z k konkrétní databáze `dbName` získat seznam objektů daného druhu `kindList`, které jsou potomky objektu `parentId`. Získané objekty obsahují atributy dle kolekce `attrInfo`.

4.4 Architektura

Při návrhu architektury jsem musel zohlednit skutečnost, že půjde o aplikaci pro podnikovou sféru, tudíž jsem ji musel navrhnout tak, aby ji bylo možné upravovat podle specifických požadavků konkrétního zákazníka. Vzhledem k tomu, že aplikace má spolupracovat s nástroji, u nichž se očekává další vývoj, je nezbytné zajistit, aby ji bylo možné dále dynamicky rozvíjet a upravovat.

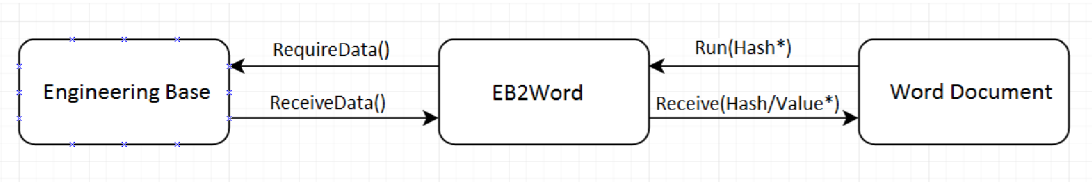
Výstupem aplikace má být dokumentace s aktuálními hodnotami z databáze *Engineering Base*. Lze předpokládat, že dokumentace bude obsahovat i grafické prvky a že každá firma bude chtít mít vlastní design výsledného dokumentu. Řešením tohoto problému by měla být šablona (kapitola 4.4.1), která bude sloužit jako předloha pro vytvořený dokument. Uživatel vytvoří pevný text a grafiku šablony v nástroji *Microsoft Word* a poté pouze označí místa, na které má být umístěna hodnota z databáze. Po doplnění hodnot se vytvoří nový dokument aktuální dokumentace a šablona zůstane v původním stavu pro další verze dokumentaci.

Aplikaci jsem rozvrhl do čtyř částí - *Vstupní rozhraní*, *Uživatelské rozhraní*, *Komunikační rozhraní* a makra nacházející se v dokumentu *Microsoft Word*. Všechna rozhraní jsou součástí jedné aplikace a musí být schopna mezi sebou navzájem komunikovat, makra v dokumentu Word pouze usnadňují uživateli správné volání aplikace EB2Word.

Jednotlivé části aplikace EB2Word:

- *Vstupní rozhraní* - Vstupní rozhraní je část aplikace, která obstarává zpracování vstupních parametrů a vrací získaná data dle požadavků. Jedná se o konzolovou část, kterou je možné spustit z příkazového řádku nebo jiného programu.
- *Uživatelské rozhraní* - Uživatelské rozhraní obsahuje dynamické uživatelské rozhraní, které lze designovat dle přání a požadavků zákazníka. Jedná se o jednoduchou okenní aplikaci, která je přizpůsobena prostředí *Windows*. Uživatelské rozhraní slouží pouze pro prohledávání databázového stromu a selekci atrubutu konkrétního objektu.
- *Komunikační rozhraní* - Komunikační rozhraní obsahuje aplikační logiku a metody pro komunikaci přes *Webovou službu*.
- *Makra* - Makra jsou součástí dokumentu Word a zjednodušují komunikaci mezi aplikací EB2Word a dokumentem Word.

Na obrázku 4.2 je vidět, jak má probíhat komunikace mezi jednotlivými programy. Z dokumentu Word se pomocí makra spustí aplikace EB2Word, které se mohou předat jako vstupní parametry textové *Hash řetězce* (popsány v kapitole 4.4.1). Tyto řetězce jednoznačně identifikují konkrétní objekt a atribut v databázi, jehož hodnota se požaduje. Pokud je aplikace EB2Word volána bez parametrů, spustí se uživatelské rozhraní, ve kterém uživatel může vyhledat objekt a atribut, který chce umístit do šablony. V závislosti na vstupních parametrech aplikace EB2Word vrací buď hodnoty atributů nebo Hash řetězec identifikující konkrétní objekt a atribut v databázi. Dle vstupních požadavků zasílá aplikace EB2Word požadavky na data službě *WCS* a ta navrací odpovědi na požadavky s daty z databáze *Engineering Base*.



Obrázek 4.2: Schéma komunikace mezi *Engineering Base*, EB2Word a dokumentem Word.

4.4.1 Šablona

Pro uživatele by bylo vhodné, aby bylo možné mít vzhled dokumentace dle požadavků společnosti nebo účelu dokumentace. Proto jsem se rozhodl, že uživatel nejprve vytvoří design dokumentace a až poté umístí specifické značky na místa, kam se nakonec umístí aktuální data z nástroje *Engineering Base*. Po doplnění hodnot se vytvoří nová dokumentace, která nahradí tu původní. Vytvoření nové dokumentace šablonu nijak nezmění a ta se bude moct používat opakovaně.

Jako specifické značky používám tzv. *Hash řetězce*. *Hash řetězec* je posloupnost znaků, která přesně definuje zvolený objekt v databázi a jeho atribut. Obsahuje název databáze, ID nadřazeného objektu, ID vybraného objektu, ID druhu objektu a ID zvoleného atributu. ID nadřazeného objektu a ID druhu objektu slouží jako ověření, že se jedná o původně zvolený objekt.

Šablona dokumentace si uchovává všechny získané a umístěné *Hash řetězce*. Hodnoty atributů těchto objektů budou použity ve vytvořené dokumentaci. Pokud by objekt daného ID v databázi neexistoval nebo nebyl správně ověřen, zůstal by ve vytvořené dokumentaci původní *Hash řetězec*.

4.5 Implemetace

Proto aby jsem mohl pro *komunikační rozhraní* využít příklady z projektu *Web Services*, bylo nutné aplikaci implmentovat v jazyce `C#`. Pro uživatelské rozhraní jsem využil technologii *Windows Presentation Foundation* (WPF), která je podmnožinou *.NET Frameworku*. Implementace aplikace proběhla ve vývojovém prostředí Microsoft Visual Studio 2013 Ultimate. Makra byla vytvořena ve *Visual Basic Editoru* nástroje *Microsoft Word*.

Zatímco při návrhu jsem postupoval metodou shora dolů, u implementace jsem se rozhodl postupovat opačně, tj. zdola nahoru. Nejprve jsem tedy implementoval jednotlivé funkce daného rozhraní, které jsem ihned testoval. Tyto funkce jsem poté sdružoval do jednotlivých rozhraní programu - vstupní, uživatelské, komunikační. Výsledkem je základní aplikace, která obsahuje požadovanou funkcionalitu. Kromě základní aplikace jsem také vytvořil tři jednoduchá makra pro *Microsoft Word*, která popisuji níže v kapitole 4.5.5.

4.5.1 Základní aplikace

Implementace základní aplikace je rozdělena do dvou samostatných projektů - „WebService“ a „WebServiceConnecting“. Projekt „WebService“ obsahuje implementaci vstupního rozhraní a projekt „WebServiceConnecting“ implementaci komunikačního a uživatelského rozhraní. Fyzické oddělení vstupního rozhraní zjednodušší jeho modifikaci v budoucnu. Komunikační a uživatelské rozhraní jsou velmi provázané a proto jejich fyzické oddělení nebylo příliš vhodné.

4.5.2 Komunikační rozhraní

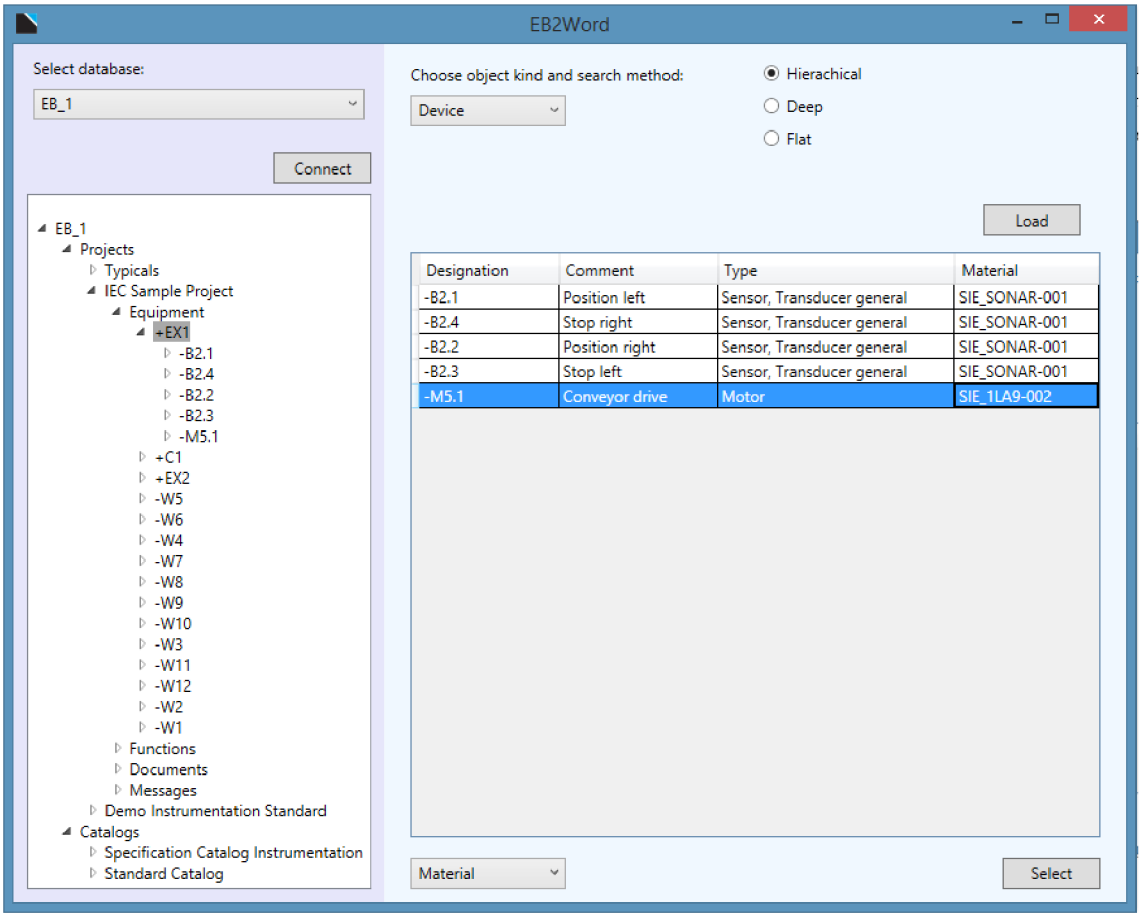
Pro implementací komunikačního rozhraní jsem se rozhodl využít existujícího projektu Web Services, ve kterém jsem již implementoval metody pro komunikaci s *Aplikačním serverem* skrze *WCS*. Protože ve specifikaci nebyla vyžadována lokalizace, nebylo potřeba využívat lokalizační služby, tj. „TranslationService“. Proto jsem se omezil pouze na využití služeb „BasicDataService“. Oba druhy služeb jsou popsány v kapitole 2.1.2.

V komunikačním rozhraní jsem implementoval dva způsoby komunikace skrze WCS. První způsob komunikace slouží pro prohledávání databázového stromu. Při této komunikaci se využívají služby GetDatabases, GetProjectData a GetObjectData pro získání dostupných databází, projektů v konkrétní databázi a objektů, které se nacházejí v projektech. Tento způsob komunikace je využíván uživatelským rozhraním, které prezentuje uživateli získaná data. Druhý způsob komunikace slouží pro získání dat konkrétního objektu. Při této komunikaci se využívá pouze služba GetObjectData. Tento způsob komunikace je využíván vstupním rozhraním pro získání hodnoty atributu konkrétního objektu.

I když je uživatelské rozhraní aplikace navrženo pouze pro procházení projektů a katalogů, rozhodl jsem se do komunikačního rozhraní základní aplikace přidat i metody pro podporu služby „UpdateObjectData“. Tato služba umožňuje provádět úpravy u konkrétního objektu (více v kapitole 2.1.2). Pokud bude budoucí zákazník vyžadovat rozšíření uživatel-

ského rozhraní o možnost editovat u objektů určitý atribut[1] (např. komentář či příznak) nebude nutné upravovat architekturu aplikace a využijí se již existující metody.

4.5.3 Uživatelské rozhraní



Obrázek 4.3: Uživatelské rozhraní základní aplikace, které je zjednodušené pro názornější prezentaci funkcionality.

Při tvorbě uživatelského rozhraní jsem se snažil, vytvořit jednoduché a přehledné prostředí. Uživatelské rozhraní slouží pouze pro procházení projektů a katalogů ve vybrané databázi a je inspirováno *Webovým klientem* nástroje *Engineering Base* (kapitola 2.1.1). Implementoval jsem jej pomocí technologie WPF, která je podmnožinou *.NET Frameworku*. Okno aplikace jsem rozdělil do dvou částí, které jsou pro větší zdůraznění odlišeny různými barvami. Vzhled a rozvržení uživatelského rozhraní vidíme na obrázku 4.3.

Uživatelské rozhraní je inicializováno a spuštěno vstupním rozhraní. Dále reaguje na akce uživatele a využívá funkce, které poskytuje komunikační rozhraní, pro získávání dat z databáze *Engineering Base*. Získaná data prezentuje uživateli. Po selekci atributu konkrétního objektu je vytvořen *Hash řetězec* (popsán v kapitole 4.4.1), který je navrácen vstupnímu rozhraní. Služba „GetObjectData“ pro *Engineering Base* verze 6.5.2 má ome-

[1]Např. že je daný objekt použitý v dokumentaci a nesmí být tedy smazán.

zení, které umožňuje získat pouze konkrétní množinu atributů objektu, nikoliv všechny jeho atributy[2]. Z tohoto důvodu jsem pro základní aplikaci omezil selekci atributů pouze na množinu čtyř základních atributů - označení, komentář, typ a materiál. Pro verzi 6.5.2 se pro každého zákazníka bude muset upravit tato množina atributů. V novějších verzích *Engineering Base* se toto omezení již nevyskytuje.

Levá část (bledě fialová) slouží pro výběr zdrojové databáze, procházení projektů či katalogů a selekci objektu, pod kterým bude uživatel hledat konkrétní objekt. Tato část obsahuje rozbalovací nabídku, která obsahuje seznam dostupných databází, ke kterým se lze připojit. Pod ní se zobrazuje omezený hierarchický databázový strom. V databázovém stromě jsou zobrazeny pouze projekty a katalogy a jejich podobjekty.

Pravá část (azurově modrá) slouží pro vyhledání specifického objektu a selekci jeho atributu. Z důvodu omezení služby „GetObjectData“, které neumožňuje získat najednou objekty různého druhu[3], je nutné, aby uživatel zvolil druh objektů, které se budou v databázi vyhledávat. Dále může uživatel zvolit jeden ze tří způsobů vyhledávání v databázi – *Hierarchical*[4], *Deep*[5] a *Flat*[6]. Požadovaný atribut konkrétního objektu může uživatel zvolit z tabulky dostupných objektů nebo pomocí rozbalovací nabídky pod tabulkou.

4.5.4 Vstupní rozhraní

Vstupní rozhraní je důležitá část aplikace, která se stará o zpracování požadavků a předávání získaných dat. Mým původním záměrem bylo integrovat vstupní rozhraní do uživatelského rozhraní, ale nakonec jsem se rozhodl jej implementovat samostatně do odděleného projektu. Při získávání aktuálních dat z databáze totiž není nutné zobrazit uživatelské rozhraní, protože hledané položky jsou definované vstupními parametry.

Pro splnění požadavků jsem implementoval dva způsoby spuštění aplikace. První způsob je čistě bez parametrů, kdy vstupní rozhraní spustí uživatelské rozhraní, ve kterém může uživatel vyhledat a vybrat požadovaný objekt. Tento způsob může také sloužit pro otestování dostupnosti požadovaných databází. Druhým způsobem je volání aplikace s parametry, které definují atributy konkrétních objektů, jejichž hodnoty se mají dohledat. Pro získání hledaných dat využívá vstupní rozhraní funkce implementované v komunikačním rozhraní.

Získaná textová data (*Hash řetězec* vybraného objektu nebo hodnoty objektů ze vstupních parametrů) jsou uložena do speciálního souboru. Umístění tohoto souboru musí být nastaveno při instalaci aplikace. Daný soubor slouží jako schránka pro předání získaných dat makru, které spustilo tuto aplikaci.

[2] V požadavku služby musí být uvedeno ID atributů, které chci získat.

[3] Toto omezení by bylo možné obejít opakovanými dotazy pro všechny druhy objektů. Tento přístup by měl však příliš velkou časovou náročnost.

[4] Prohledávání do hloubky, dokud není nalezen objekt hledaného typu. Jeho potomci sejiž neprohledávají.

[5] Prohledávání do hloubky, prohledávají se i potomci hledaného typu.

[6] Prohledávání pouze mezi přímými potomky vybraného objektu.

4.5.5 Makra pro Microsoft Word

V nástroji Microsoft Word jsem vytvořil tři makra – „Add Attribute“, „Load Attributes Value“ a „Delete Settings“. Tyto makra zjednodušují uživateli práci s aplikací. Uživatel pouze vybere požadovanou operaci a zbytek zajistí předpřipravené operace v makru.

Add Attribute

Makro „Add Attribute“ jsem vytvořil pro výběr aktributu objektu, který se později umístí na aktuální pozici kurzoru. Makro spustí aplikaci EB2Word bez parametrů a čeká na její ukončení. Pokud po ukončení aplikace získá ze speciálního souboru *Hash řetězec*, uloží ho do seznamu lokálních proměnných dokumentu a umístí na pozici kurzoru hash tag (řetězec `<$n>`, kde `n` je pozice uloženého *Hash řetězce*). Makro „Load Attributes Value“ tento hash tagy nahradí za aktuální hodnotu atributu.

Load Attributes Value

Makro „Load Attributes Value“ jsem vytvořil pro získání aktuálních hodnot z databáze a vytvoření nové dokumentace. Makro spustí aplikaci EB2Word s *Hash řetězci* ze seznamu lokálních proměnných dokumentu jako parametry a čeká na jeho ukončení. Poté umístí získané hodnoty na pozice příslušných hash tagů a nový dokument uloží do vybraného adresáře.

Delete Settings

Makro „Delete Settings“ slouží pouze pro smazání všech uložených *Hash řetězců* v dokumentu. Pokud by byla šablona vícekrát upravována a v seznamu lokálních proměnných dokumentu se tak nacházelo mnoho nepoužívaných *Hash řetězců*, pomocí tohoto makra bude tento seznam vyčištěn. Vymazání všech uložených *Hash řetězců* v dokumentu může zrychlit proces vytváření nové dokumentace.

Kapitola 5

Testování

Aplikace je realizována jako spustitelný soubor, tak aby ji bylo možné využívat v různých prostředích, avšak bezchybné chování aplikace vyžaduje správné nastavení dalších nástrojů a služeb, se kterými aplikace komunikuje. V první řadě je nutná přítomnost nástroje *Microsoft Word* s možností spustění makra. Dále je vyžadován nástroj *Engineering Base* s existující databází obsahující projekty, nainstalované rozšířené pro *Engineering Base*, *Web Service*, a běžící službu a nakonec je potřeba, aby operační systém dovolil aplikaci ukládat dočasné soubory.

5.1 Průběh testování

Vlastní testování bylo prováděno na několika zařízeních s různými verzemi Microsoft produktů. V různých kombinacích jsem testoval na OS W7 i OS W8.1, Microsoft Office 2010, 2013 a 365, procesorech od AMD i Intel, na různých typech a velikostech operační paměti RAM a na SSD i HDD. U testování se hledělo hlavně na odhalení chyb, stabilitu aplikace a vliv na výkon v různém prostředí. Dále se také zaměřovalo na intuitivnost a přehlednost uživatelského rozhraní.

U každého testování bylo žádoucí, aby testující subjekt splňoval všechny podmínky pro správný běh aplikace (počítač obsahuje správě nastavený potřebný software[1]) a *SQL Server* obsahoval funkční databázi s minimálně jedním projektem. Při testovaní bylo podmínkou vytvořit šablonu v nástroji *Microsoft Word*, která obsahovala dokumentační značky, které se později aktualizovali hodnotami z *Engineering Base*. Očekávaným výstupem byl dokument, který obsahoval na správných pozicích aktuální hodnoty atributů.

[1] Nástroj *Engineering Base* verze 6.5.2, službu „Engineering Base 6.5 Web Service“ a *Microsoft Word*.

5.2 Výsledky testování

Při testování se objevilo několik problémů, se kterými jsem v návrhu nepočítal. Ty mohu rozdělit na více a méně závažné problémy. Zajímavým výsledkem testování bylo i to, že se neprojevily problémy při testování v různých prostředích. Dokud byla dodržena funkčnost všech důležitých součástí, nemělo různé prostředí ani různé verze nástrojů vliv na správný běh aplikace. Z testů také vyplynulo, že z výkonostního hlediska je výhodnější používat SSD pro fyzické umístění databáze. Nicméně pokud šablona obsahuje odkazy na objekty z různých databází, doba pro připojení k nové databázi eliminuje rychlostní rozdíl HDD a SSD.

Méně závažné:

- Ztráta spojení – jde o nedostupnost služby, která je vyžadována pro komunikaci s aplikačním serverem. Již dle původního návrhu je uživatel upozorněn na problém s komunikací, pokud tato situace nastala. Nejčastější příčinou je chybná inicializace služby WCS. Pro odstranění tohoto problému bylo nutné reinicializovat tuto službu.

- Chybějící multi-selekce – ikdyž uživatel vytváří dokumentaci, ve které používá objekty ze stejné databáze, musí opakovaně zvolit databázi, ke které se chce připojit. V základní aplikaci jsem tento problém vyřešil tak, že si aplikace pamatuje poslední relaci a při novém spuštění se k ní opět připojí[2]. V kapitole 6.3 popisuji alternativní rozšíření, které řeší tento problém efektivněji.

- Nepřehlednost hash tagů – při úpravě šablony je někdy potřeba odkazovat na jiný objekt než doposud[3], avšak uživatel nemá možnost jak určit, na který objekt daný hash tag odkazuje. Protože hash tagy jsou v dokumentu vytvářeny jako textové řetězce, v základní aplikaci může uživatel starý hash tag smazat a na jeho místo vložit nový, který bude odkazovat na nově požadovaný objekt. Tímto přístupem však může nastat situace, kdy jsou v dokumentu udržovány hash tagy, které nejsou využity. Z tohoto důvodu jsem přidal makro „Delete Settings“ (popsáno v kapitole 4.5.5), které smaže všechny *Hash řetězce*, které jsou uloženy v šabloně. V kapitole 6.3 popisuji alternativní rozšíření pro správu hash tagů.

Více závažné:

- Nenalezen odkazovaný objekt – Situace, kdy odkazovaný objekt nelze v databázi nalézt může nastat při úpravě projektu nebo databáze. Odkazovaný objekt je vyhledáván podle unikátního OID, proto je nalezen ikdyž jej uživatel umístí na jiné místo v databázi[4]. Pokud objekt nelze najít je trvale odstraněn. Ve vytvořeném dokumentu zůstane původní hash tag. Při využití rozšíření, které popisuji v kapitole 6.3, je možné použít poslední známou hodnotu.

- Nenalezen atribut odkazovaného objektu - Uživatel většinou nemá dostatečná práva, aby mohl odstranit atribut z objektu. Pokud tato situace nastane, postup je stejný jako při situaci, kdy nebyl nalezen odkazovaný objekt.

[2] pokud se chce uživatel připojit k jiné databázi, jednoduše ze seznamu dostupných databází tu požadovanou.

[3] Např. pokud se změní struktura objektů v projektu.

[4] Pokud je objekt přesunut do koše není zcela smazán a lze jej tedy stále nalézt.

Kapitola 6

Experimenty

Přestože jsem při návrhu plánoval s nasazením aplikace v podnikové i nepodnikové sféře, nikdy není možné předpokládat a ošetřit všechny problémy a požadavky, které se mohou objevit. Na rozdíl od nepodnikové sféry má podniková sféra často pevně definovaná pravidla a omezení v rámci celé společnosti. Díky tomu lze pro konkrétní společnost provést specifické úpravy, které by měly zajistit správnou funkcionalitu v celé společnosti. Navíc existuje určitá množina omezení, která se v podnikové sféře často vyskytují. Díky tomu je možné dané omezení dopředu analyzovat a nachystat pro ně alternativní řešení. V následujících podkapitolách popíšu některé z omezeních v podnikové sféře, omezení u projektů samotného EB a také řešení, které jsem u nich zvolil.

Problémy a požadavky, které se mohou objevit v nepodnikové sféře, není tak jednoduché předpokládat oproti podnikové sféře. Tyto poznatky je nejjednodušší získat při reálném nasazení aplikace. Získané poznatky, návrhy a řešení popisuji v podkapitole 6.3.

6.1 Omezení podnikové sféry

V podnikové sféře se objevuje celá řada omezení, která mají převážně bezpečnostní charakter. V této kapitole postupně popíšu omezení, jež se vyskytla, a jejich řešení od těch nejčastějších a nejobecnější až po ty specifické.

Omezení nebo zákaz maker v nástroji Microsoft Word

Omezení a zákaz maker je jeden z nejefektivnějších způsobů, jak zabránit napadení počítače zákeřným softwarem skrze dokumenty nástrojů Microsoft Office.

Omezení maker (přesněji zákaz všech maker bez digitálního podpisu) je postup, který zabraňuje uživateli spustit makra, která nemají požadovanou certifikaci. Požadované certifikáty pak vydává specifická instituce (sama firma nebo společnost starající se o ICT). Omezení maker je sice náročnější na nastavení a správu, ale v tomto případě stačilo, abych makru obstaral potřebný certifikát. Nicméně makra není možné uložit do šablony, ale budou muset být přítomna v nástroji *Microsoft Word* na každé pracovní stanici.

Úplný zákaz maker v nástroji *Microsoft Word* je jednodušší na nastavení a správu a také mnohem účinnější opatření, protože jej nelze obejít podvrhnutím certifikátu. Toto omezení jsem se původně snažil vyřešit úpravou vstupního a uživatelského rozhraní. Úprava měla spočívat v tom, že by po spuštění aplikace uživatel vybral šablonu a následně zvolil jednu z operací, která by se provedla nad zvolenou šablonou. Buď operaci pro přidání odkazu na další objekt nebo operaci, která vytvoří novou dokumentaci s aktuálními hodnotami.

Nakonec jsem objevil, že omezení, které způsobuje zákaz maker, lze obejít, pokud potřebnou funkcionalitu implementuji jako plugin pro *Microsoft Word*.

6.2 Omezení projektů v Engineering Base

Při návrhu aplikace jsem vycházel z předpokladu, že zákazník má jednotnou databázi pro všechny uživatele nebo že má více samostatných databází, ale projekty mezi nimi nepřenáší[1].

Exportování a importování projektu způsobí, že objekty mají v nové databázi jiné OID a navíc aplikace EB2Word nemůže přesně určit, ve které z dostupných databází se hledané objekty nachází. Toto omezení lze řešit mnoha způsoby, nicméně každý způsobí velký zásah do architektury. Je možné že v dalších verzích nástroje *Engineering Base* bude existovat jednoduchý způsob na řešení tohoto problému.

6.3 Možná rozšíření

V předchozích podkapitolách jsem se zaměřil na úpravy základní verze aplikace z důvodů různých omezení, která se vyskytla při testování aplikace u potenciálních zákazníků. Aplikace je ale navržena tak, aby ji bylo možné dále rozvíjet. V této části popíšu další úpravy a rozšíření, které by mohly rozšířit použitelnost aplikace nebo naopak zjednodušit její využití. Všechna rozšíření zmíněná v této části jsem implementoval pouze jako prototyp a proto nejsou zmíněny v kapitole 4, ani implementována v přiložených zdrojových souborech.

Multi-selekce dat z Engineering Base

Dle původního návrhu aplikace lze selekci dat z databáze *Engineering Base* provádět pouze po jednotlivých prvcích[2]. Pokud chce uživatel vytvořit dokumentaci pro konkrétní projekt a všechny vyhledávané objekty se nacházejí ve stejném projektu i ve stejné databázi, bylo by vhodné, aby mohl zvolit více objektů/atributů naráz. Původně jsem tento problém řešil tím, že si aplikace ukládala záznam poslední relace. Při novém spuštění se připojila k databázi dle uloženého záznamu a načetla daný projekt. Nicméně toto řešení často zpomalovalo práci a proto je možné toto chování deaktivovat[3].

Alternativou pro multiselekci dat a zrychlení práce s aplikací proto bylo rozšíření uživatelského rozhraní. Podmínkou je, aby uživatelské rozhraní umělo vytvořit ze zvoleného objektu řetězec znaků, který přesně identifikuje vybraný objekt a atribut. Vytvořený *Hash řetězec* se uloží do kolekce, která se později předá vstupnímu rozhraní. Vstupní rozhraní poté uloží *Hash řetězce* z kolekce do souboru, ze kterého si je načte makro ve Wordu. Protože původní implementace makra vytvoří hash tagy na aktuální pozici kurzoru, musí uživatel tyto hash tagy umístit na žádané pozice.

[1] Při obnovení zálohy celé databáze se zachovávají OID objektů.

[2] Uživatel zvolí umístění pro data z *Engineering Base*, spustí aplikaci, vybere databázi, projekt, vyhledá konkrétní objekt a zvolí konkrétní atribut.

[3] Pro aktivaci a deaktivaci jednotlivých funkcionalit využívám klíče s textovými řetězci, které jsou uložené v registrech.

Makro pro správu hash tagů

Makro „Add Attribute“ po obdržení *Hash řetězce*, který identifikuje vybraný objekt a atribut, vytvoří unikátní hash tag, který vloží na pozici kurzoru. *Hash řetězec* a příslušný hash tag uloží mezi proměnné dokumentu. Aktuální hodnoty z databáze uložených řetězců jsou umístěny na místa příslušných hash tagů. Pokud uživatel požaduje hodnoty řetězců na více místech v dokumentu, stačí hash tag zkopírovat a umístit na další místa v dokumentu[4]. Při vytváření šablony je tento postup jednoduchý a praktický, avšak při pozdější úpravě šablony už vzniká problém, kdy uživatel nemá možnost identifikovat jednotlivé hash tagy.

Pro řešení zmíněného problému jsem navrhl další makro, které zajišťuje správu uložených *Hash řetězců* a příslušných hash tagů. Navíc je vhodné, aby aplikace společně s řetězci vybraných atrubutů vracela i metadata, která budou sloužit pro lepší identifikaci[5]. Samotné makro pak bude implementováno jako formulář, ve kterém se bude nacházet seznam již vybraných objektů a jejich atributů. Uživatel ve formuláři vybere požadovaný hash tag a umístí jej na pozici kurzoru v dokumentu. Toto rozšíření značně zjednoduší pozdější orientaci v šabloně a její úpravu.

Dále jsem zvažoval i funkcionalitu, která by při spuštění makra aktualizovala hodnoty uložených řetězců a metadat. Tato funkcionalita je však příliš časově náročná a její náročnost roste s počtem uložených řetězců. Navíc pokud by bylo potřeba aktualizovat data z různých databází, výsledné zpoždění by bylo na slabších strojích příliš omezující.

Náhrada nástroje Microsoft Word

Náhrada nástroje *Microsoft Word* je další rozšíření, u kterého jsem vytvořil pouze návrh a funkční prototyp, protože jeho implementace již velmi přesahuje původní požadavky. Hlavní myšlenka tohoto rozšíření vychází z předpokladu, že uživatel nejprve vytvoří vzorovou dokumentaci a až poté do ní umístí hash tagy. A protože se hash tagy většinou umísťují mezi již existující texty, stačilo by vytvořit editor, který by zobrazil texty z šablony. Mezi texty by pak uživatel umístil hash tagy.

Funkcionalita makra „Add Attribute“ by musela být přepracována do jazyka `C#` a implementována jako rozšíření uživatelského rozhraní. Následné chování (vyhledání objektu a selekce atributu) by bylo zachováno.

[4] Makro při nahrazování hash tagů za aktuální hodnoty vyhledává hash tagy jako textové řetězce, proto je možné jej přesunovat či kopírovat na jiná místa v dokumentu.

[5] Pokud je dané, že se označení objektu nebude měnit, může být vhodné jako metadata použít kombinaci „název_databáze|označení_objektu|jméno_atributu“. Pokud je možné, že se označení objektu může změnit, bude vhodnější přidat do metadat i hodnotu zvoleného atributu.

Kapitola 7

Závěr

Cílem této práce bylo analyzovat rozšíření *Webový Komunikační Server* a vytvořit program, který dokáže vytvářet a aktualizovat dokumentace s aktuálními daty mimo databázi nástroje *Engineering Base*.

V souladu s cílem jsem vytvořil aplikaci, která dokáže komunikovat s databází nástroje *Engineering Base* alternativním přístupem přes *Webový Komunikační Server*. Vytvořil jsem návrh základní aplikace, která obsahuje požadovanou funkcionalitu a slouží jako základ pro komerční využití. Dle návrhu jsem aplikaci implementoval a dále jsem navrhl několik prototypů, které je možné použít jako rozšíření při dalším vývoji. Některé rozšíření prezentuji jako řešení různých omezení (například stav, kdy jsou zakázaná makra v nástroji Microsoft Word). Pro názornější prezentaci funkcionality aplikace jsem zvolil zjednodušené uživatelského rozhraní. To spočívalo v omezení množiny atributů, které lze u objektu vybrat. Aplikace umožňuje připravit šablonu, která obsahuje vzhled výsledné dokumentace a odkazy na atributy objektů. Tyto odkazy jsou využity při vytváření dokumentace pro získání aktuálních dat. Aplikace byla otestována a je plně funkční při splnění všech požadavků, které jsou popsány v této práci.

Práci na aplikaci jsem začal v době, kdy *Webové Komunikační Rozhraní* nebylo veřejně dostupné a neexistovala žádná aplikace se stejnou funkcionalitou. Pro *Engineering Base* verze 6.5.2 je přístup, který popisuji v této práci, unikátní a ojedinělý.

Aplikace je vhodná pro všechny firmy, které využívají nástroj *Engineering Base* pro vytváření eletrotechnické dokumentace a potřebují si udržovat dokumentace projektů mimo prostředí *Engineering Base*. Aplikaci lze využít i na automatické aktualizace, které mohou být spouštěny naplánovanými úlohami pomocí serveru. Aplikace aktuálně není oficiálně dostupná, ale je možné se na ni dotázat ve firmě Technodat a objednat si její nasazení ve své firmě.

Nástroj *Engineering Base* je placený produkt, tudíž všechny podrobnější informace jsou neveřejné a také velice těžce dostupné. Na základě spolupráce s firmami Aucotec AG a Technodat, které vlastní tento nástroj, jsem získal kompletní dokumentaci k tomuto nástroji. Uvedené informace o nástroji *Engineering Base* jsou aktuální pro verzi 6.5.2 a některé implementační detaily nástroje jsou popsány pouze obecně kvůli ochraně duševního vlastnictví firmy Aucotec AG.

Vytvořená aplikace byla pod názvem „EB2Word“ prezentována 13. – 16. 10. 2015 na veletrhu „ELO SYS 2015“ v Trenčíně. Po veletrhu společnost TATRA TRUCKS vyjádřila zájem o využití aplikace na automatické aktualizace jejich dokumentací. Před dokončením bakalářské práce se o aplikaci „EB2Word“ začala zajímat i firma Aucotec AG.

Literatura

[1] *Engineering Base - Software pro tvorbu komplexní elektro dokumentace.* Technodat Elektro, s.r.o., 2010.

[2] *Engineering Base - Complete documentation.* Aucotec AG, 2015.

[3] Dresler, R.: *Vícevrstvé architektury aplikací.* [Online; navštíveno 05.05.2016].
URL http://www.robertdresler.cz/2011/04/vicevrstve-architektury-aplikaci.html

[4] Čermák, M.: *Vícevrstvá architektura: Popis vrstev.* [Online; navštíveno 05.05.2016].
URL http://www.cleverandsmart.cz/vicevrstva-architektura-popis-vrstev/

[5] Čermák, M.: *Vícevrstvá architektura: tenký, tlustý a chytrý klient.* [Online; navštíveno 05.05.2016].
URL http://www.cleverandsmart.cz/vicevrstva-architektura-tenky-tlusty-a-chytry-klient/

[6] Hotek, M.: *Microsoft SQL Server 2008 Krok za krokem.* Computer Press, a.s., 2009, ISBN 978-80-251-2466-6.

[7] Liu, L.: *Encyclopedia of database systems.* Springer, 2009, ISBN 978-038-7496-160.

[8] Manage Mentmania: *Třívrstvá architektura.* [Online; navštíveno 05.05.2016].
URL https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture

[9] Microsoft Corporation: *Component Services (COM and DCOM).* [Online; navštíveno 05.05.2016].
URL https://msdn.microsoft.com/en-us/library/ms886465.aspx

[10] Microsoft Corporation: *DCOM (Distributed Component Object Model).* [Online; navštíveno 05.05.2016].
URL https://msdn.microsoft.com/en-us/library/cc226801.aspx

[11] Microsoft Corporation: *Distributed Component Object Model.* [Online; navštíveno 05.05.2016].
URL https://technet.microsoft.com/en-us/library/cc958799.aspx

[12] Microsoft Corporation: *SQL Server 2014.* [Online; navštíveno 05.05.2016].
URL https://www.microsoft.com/cs-cz/server-cloud/products/sql-server/

[13] Rouse, M.: *DCOM (Distributed Component Object Model)*. [Online; navštíveno 05.05.2016].
URL http://whatis.techtarget.com/definition/DCOM-Distributed-Component-Object-Model

[14] Technodat Elektro, s.r.o.: *Elektro - Software od CAD přes CAE až k CBE*. [Online; navštíveno 05.05.2016].
URL http://http://www.cad.cz/strojirenstvi/38-strojirenstvi/1548-elektro-software-od-cad-pres-cae-az-k-cbe.html

Přílohy

Seznam příloh

Příloha A

Obsah CD

- `src\` – zdrojové soubory aplikace
- `src\database\` – databáze ve formátu SQL
- `doc\ibp-xdoman01.pdf` – technická zpráva ve formátu PDF
- `doc\tex\` – zdrojové soubory technické zprávy
- `doc\examples\` – soubory s informacemi o aplikaci a návody na instalaci a použití