VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PŘEVOD VIZUALIZACE WINCC DO MES SYSTÉMU COMES
CONVERSION OF WINCC SCREENS TO MANUFACTURING SYSTEM COMES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jiří Kárník**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Pásek, CSc.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Jiří Kárník
Ročník: 2

ID: 134517
Akademický rok: 2015/16

NÁZEV TÉMATU:

Převod vizualizace WinCC do MES systému COMES

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření automatického nástroje pro převod vizualizačních obrazovek z WinCC do systému COMES. V rámci diplomové práce proveďte:
1. Seznamte se se systémy COMES a WinCC.
2. Zvolte vhodnou techniku pro vykreslování grafických objektů na webové technologii (VML, SVG, Canvas)
3. Vytvořte script pro export definic viziualizačních obrazovek WinCC do XML souborů.
4. Realizujte vlastní řešení nástroje pro převedení vyexportovaných XML souborů do prostředí COMES.
5. Otestujte správnost funkčností vizualizačních obrazovek v COMES.

DOPORUČENÁ LITERATURA:

1. SIEMENS AG. Simatic HMI - WinCC V7.2 WinCC, Scripting (VBS, ANSI-C, VBA), System Manual. 2013.
2. BRÁZDA, Roman. COMPAS AUTOMATIZACE SPOL. S R.O.Výrobní informační systém COMES verze 3: Modul CCI. Žďár nad Sázavou, 2009.
3. JOHNSON, Glenn. Programming in HTML5 with JavaScript and CSS3, Training Guide.

Termín zadání: 8. 2. 2016

Termín odevzdání: 16.5.2016

Vedoucí práce: Ing. Jan Pásek, CSc.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Abstrakt

Tato diplomová práce se zabývá vizualizací průmyslových aplikací, konkrétně převodem vizualizačních dat z programu Siemens Simatic WinCC do formátu XML s využitím vestavěného editoru skriptů.

V dalším kroku řeší převod těchto dat a jejich implementaci do MES systému COMES. K tomu je využito programovacího jazyka C#.

Práce dále obsahuje rozbor obou použitých průmyslových systémů a výběr grafického formátu, který bude využit pro zobrazení obrazových dat v systému COMES.

Klíčová slova

Siemens Simatic WinCC, výrobní informační systém, MES, COMES, VML, SVG, XML, HTML 5 Canvas, převod vizualizace, exportní balíček, C#, Visual Basic

Abstract

Presented thesis describes visualization of industrial processes. It focuses especially on the conversion of visualized data between Siemens Simatic WinCC and XML format. Build-in script editor is used to do aforementioned operations.

Conversion and implementation of the data to the MES system COMES is solved in the next section of the thesis.

Thesis also contains analysis of both used industrial systems and solves graphic format selection, which will be used to display image data in system COMES

Keywords

Siemens Simatic WinCC, manufacturing execution system, MES, COMES, VML, SVG, XML, HTML 5 Canvas, conversion of visualization, export package, C#, Visual Basic

Bibliografická citace:

KÁRNÍK, J. *Převod vizualizace WinCC do systému COMES*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 85s. Vedoucí diplomové práce byl Ing. Jan Pásek,CSc.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Převod vizualizace WinCC do systému COMES jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **16. května 2016** ……
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Janu Páskovi, CSc. a konzultantovi Ing. Aleši Stehnovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **16. května 2016** ……

podpis autora

Obsah

Seznam obrázků

1 ÚVOD

Průmyslová výroba se od jejího vzniku do současnosti neustále vyvíjela a vyvíjí. V průběhu času se díky technologickému vývoji nasazovaly různé technologie, k výrobě byla přiřazována různá další odvětví, měnilo se složení profesí zastoupených v průmyslové výrobě a měnila se i struktura podniku jako celku. Všechen tento vývoj byl zapříčiněn snahou výrobu zefektivnit, zlevnit a zvýšit bezpečnost. Jedním z důležitých odvětví je prezentace dat z výroby pro potřeby jejího řízení nebo pro řízení celého podniku[1].

Prezentace dat o výrobě byla vždy velice důležitým prvkem průmyslové výroby, díky kterému obsluha měla a má přehled o stavu výrobků, zařízení nebo i celé technologie. V počátcích průmyslové sériové výroby byla data prezentována pomocí mechanických, případně analogových indikátorů nebo dvoustavových signálek. Dalším prvkem pro prezentaci a archivaci dat byly mechanické a elektromechanické zapisovače, které byly schopné zaznamenávat data například na papír navinutý na bubnu. Velkým problémem v této době byla v porovnání s dnešní dobou nepřehlednost a také nemožnost data prezentovat a využít automatizovaně k jiným účelům na více místech výroby a podniku.

S postupem času, kdy byly do výroby zaváděny automatizační prvky a řídicí systémy, se samozřejmě vyvíjely i možnosti sběru a prezentace dat. Ze systémů pro řízení procesů (PCS) byla data prezentována nejprve pomocí segmentových displejů, zobrazujících potřebná data i s náležitým popisem. Tento prvek značně usnadnil orientaci v naměřených datech a umožnil data z řízení zobrazovat centralizovaně na jednom přístroji. S vývojem řídicích počítačů a komunikačních sítí vznikl nový prvek pro sběr a prezentaci dat a ovládání výroby. Tímto novým prvkem byly takzvané systémy SCADA. Tyto systémy umožnily data zobrazovat graficky, například pomocí vizualizovaných schémat technologie. Přesto, že se jejich vznik datuje do 70. let minulého století, jejich dramatický rozvoj a nasazení koreluje s nasazováním PC do systémů řízení technologických procesů.

S vývojem vizualizačních softwarových systémů SDACA nastala možnost zobrazovat data na takzvaných operátorských panelech (OP), které je možno montovat přímo do místních ovládacích skříní v procesní úrovni nebo do rozváděčů v místních velínech. Tyto systémy se ve zkratce označují HMI z anglického Human Machine Interface. Přidanou hodnotou systémů SCADA/HMI byla možnost odstranění mnoha jednoúčelových prezentačních prvků z technologie a tím i zjednodušení montáže a zvýšení přehlednosti celého zařízení z hlediska konstrukce elektrických rozvodů v rozvodných skříních.

Systémy SCADA/HMI jsou implementovány do PC, nebo OP, které tvoří komplementární dvojici s řídicími automaty PLC a umožňují tak sbírat, uchovávat, zpracovávat a prezentovat data z více technologických zařízení a zároveň tato zařízení ovládat s využitím pouze jednoho PC s komunikačními kartami umožňujícími

[1] Tato kapitola čerpá [20]

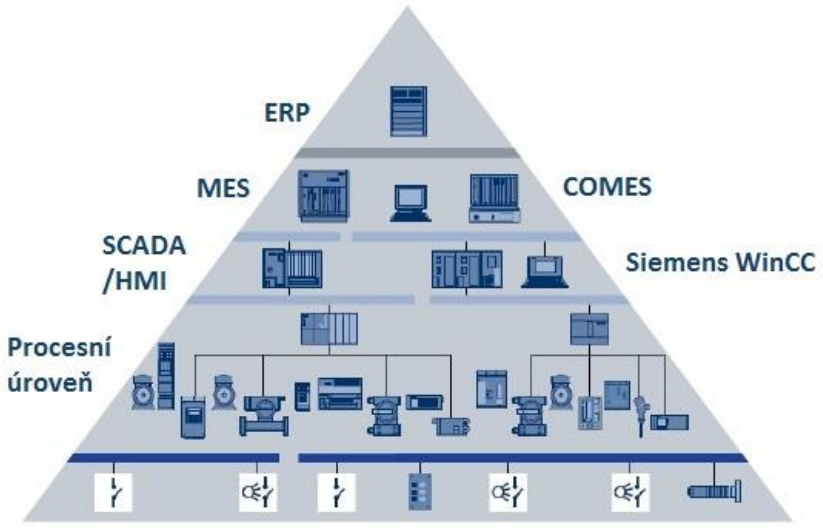
komunikaci po konkrétní sběrnici. Jejich hlavními výhodami je možnost zcela vyřadit lidský faktor z procesu sběru a zálohování dat a umožnit pracovníkům přístup k řízení a datům technologie z místa, které je umístěné tak, aby byly minimalizovány negativní vlivy technologie. Nevýhodou těchto systémů je nemožnost automatizovaného využití dat pro jiné účely výrobního závodu, například objednávání materiálu, nebo prezentaci dat vedení podniku. Tyto funkce zůstávaly i nadále v kompetenci člověka.

Nastala situace, kdy na jedné straně byla dosti slušně automatizována výroba na procesní úrovni, jejímž supervizorem je člověk – operátor a na straně druhé se úspěšně rozvíjely IT obchodně ekonomické systémy pro top management podniku - ERP (z anglického Enterprise Resource Planning). Existovaly dva nekonzistentní systémy, mezi kterými, jako interface, fungovala četa lidí nějakým způsobem zpracovávajících a předávajících informace. Nevýhody takového „rozhraní“ odstraňují systémy vyvinuté primárně pro práci s výrobními daty. Nazývají se výrobní informační systémy, zkráceně MES (z anglického Manufacturing Execution System). Jejich principem je sběr dat, jejich zpracování, rozšířené možnosti jejich prezentace a ukládání na serveru a jejich využití pro následné řízení. V těchto systémech má každý pracovník nastaveny práva pro přístup do systému a podle jeho funkce mu je umožněn přístup k určitým datům prezentovaným z technologie. Díky tomu je opět minimalizován lidský faktor. Výhod těchto systémů je nespočet, ale je nutné zmínit pro tuto práci hlavní výhodu, kterou je využití webových technologií. Tato vlastnost rozšiřuje možnosti přizpůsobení těchto systémů, zjednodušuje ovládání a úpravy pro potřeby zákazníka a také ulehčuje vývoj, protože webové technologie jsou standardizované a mezi programátory všeobecně známé. Velkým problémem při použití těchto systémů je fakt, že díky jednoduché práci s daty dochází v podnicích ke sběru všech různých dat s myšlenkou, že by mohly být v budoucnu použitelné. Vznikají tak obrovské databáze, plné důležitých i bezcenných dat, ve kterých se často nejsou schopni orientovat ani sami zaměstnanci podniku. Tento jev se v odborné literatuře nazývá big data.

V dnešní době se výrobci snaží o vytvoření komplexních systémů, zahrnujících funkčnosti všech úrovní, čímž je usnadněn chod v celém podniku. Vznikají další systémy a moduly systémů, zabývající se například správou životního cyklu výrobku (PLM) nebo řízením dodavatelských řetězců (SCM). Díky této koncepci jsou možnosti prezentace dat sjednoceny pod jeden systém, což zjednodušuje práci pracovníků uvnitř podniku. Tímto směrem vývoje se vydala i firma Compas automatizace.

Vzhledem k velkému pokroku v oblasti vývoje mobilních zařízení a možností dříve jmenovaných systémů se možnosti prezentace posouvají opět výše. Již není nutné, aby pracovníci k přístupu k datům využívali stanovená pracovní místa, ale k práci a řešení problémů mohou využívat moderních zařízení, díky kterým mají přístup k datům neustále u sebe v jakémkoli místě podniku. Není třeba speciálních verzí mobilních zařízení, protože s webovými technologiemi využitými v podnikových řídicích systémech umí pracovat všechna tato zařízení. Je pravděpodobné, že mobilní technologie budou společně se stále se vyvíjejícími podnikovými řídicími systémy tvořit stále větší zastoupení v práci s podnikovými daty.

Je tedy patrné, že moderní technologie usnadňují řízení výroby a zjednodušují práci zaměstnancům na všech úrovních. S příchodem nových technologií však vznikají také určité problémy, které jsou v hojné míře zastoupeny při vyvíjení podnikových aplikací. Pro přenos dat mezi různými typy systémů byla vytvořena standardní rozhraní a protokoly, které umožňují bezpečný přenos technologických dat. K problémům však dochází při přenosu vizualizační části aplikace. Tato část průmyslové automatizace není standardizována. Každý výrobce si pro zobrazení dat vytváří vlastní prvky, jejichž parametry jsou závislé na požadavcích na daný systém, ale také na technologii, na které je systém založen. Vzniká tedy závislost na tom, zda se jedná o program určený pro PC, pro operátorské panely nebo pro využití webových technologií. Převod grafických dat si tedy řeší výrobce mezi svými systémy a dochází tak ke zpomalení vývoje aplikace, složené ze systémů více výrobců. Jako příklad lze uvést téma této práce. Ze systému WinCC firmy Siemens, jehož úkolem je zprostředkovat ovládání aplikace operátorovi, budou prvky vizualizace převáděny do systému COMES firmy Compas automatizace, který využívá webových technologií a je tedy přístupný z různých zařízení, napojených na podnikovou síť. Systém WinCC bude dále sloužit jako operátorský terminál a nebude nahrazen. Data budou pouze předávána do systému vyšší úrovně, tedy do systému COMES. Systém COMES obsahuje několik dále popsaných modulů, které jsou neustále vyvíjeny, ale z výše popsaných důvodů není možné, aby obsahoval modul pro převod grafických dat, dokud nebude definován jejich jednotný formát. Proto je mým úkolem vytvořit převodní program, který tuto vzájemnou nekompatibilitu řeší. V dalších kapitolách této práce bude detailněji popsána realizace převodu dat, mezi těmito dvěma systémy.



Obr. 1: Úrovně automatizačních zařízení v podniku [25]

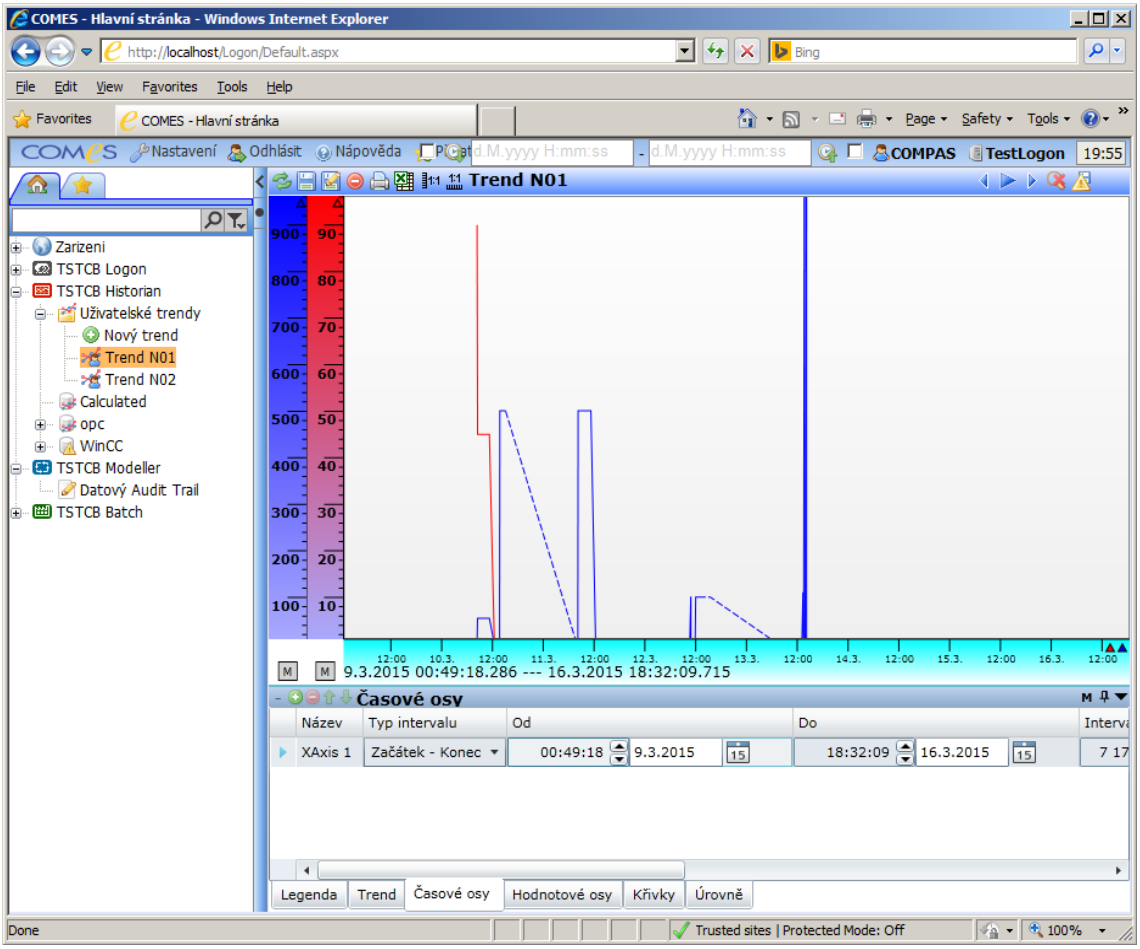
2 VÝROBNÍ INFORMAČNÍ SYSTÉM COMES

Jak již bylo řečeno v předchozí kapitole, v dnešní době se ve výrobě velkou měrou prosazují výrobní informační systémy a celopodnikové řídicí systémy. Na tento trend zareagovala česká firma COMPAS automatizace, spol. s.r.o. a vytvořila vlastní výrobní informační systém nazývaný COMES. Tento systém, který je neustále vyvíjen, je vytvořen jako modulární a multijazyčný. Integruje do sebe funkčnosti výrobních informačních systémů a některé funkčnosti ERP systémů. V době psaní této práce je vyvíjena verze 3.12. Obsahuje 6 modulů – COMES Logon, COMES Historian, COMES Modeller, COMES Traceability, COMES Batch a COMES CCI. Díky tomu, že systém COMES vychází z mezinárodních standardů MESA INTERNATIONAL, INSI/ISA S95, ANSI/ISA S88 a že je vyvíjen a testován v souladu s principy GEP (Good Engineering Practice) a GAMP (Good Automated Practice), které určují, že se jedná o ověřený produkt v požadované kvalitě a že splňuje pokyny pro výrobce systémů pro farmaceutický průmysl, je možné tento systém využít v celé řadě odvětví průmyslu. Systém je nasazován ve farmacii, potravinářství, v chemickém průmyslu, ve sklářství a také v automobilovém průmyslu[2].

COMES je postaven na technologiích firmy Microsoft. Z produktů této firmy je využit Windows Server, IIS, Microsoft SQL Server, Internet Explorer, NET Framework, ASP. NET, C# a Silverlight. Windows server je využit jako operační systém na straně serveru, IIS slouží jako webový server systému, Microsoft SQL Server zajišťuje správu dat a tvorbu systémových databází, Internet Explorer je využit pro přístup klienta do systému a ostatní technologie jsou využity při vývoji systému, nebo k jeho úpravám pro konkrétního zákazníka. Právě Internet Explorer je klíčovým prvkem pro prezentaci dat, protože jeho podpora různých webových technologií a formátů určuje, jaké technologie je možné využívat také v systému COMES. Jak již bylo řečeno, systém je neustále vyvíjen, jsou rozšiřovány dříve jmenované moduly a optimalizována jejich funkčnost. Tyto moduly budou popsány v následujících kapitolách.

Pro uživatele je vytvořen jednotný grafický styl, díky kterému se pracovnici mohou v systému lépe orientovat. Po přihlášení je zobrazena úvodní obrazovka s posledním zobrazením, jak lze vidět na Obr. 2.

[2] Tato kapitola čerpá z [7][22]

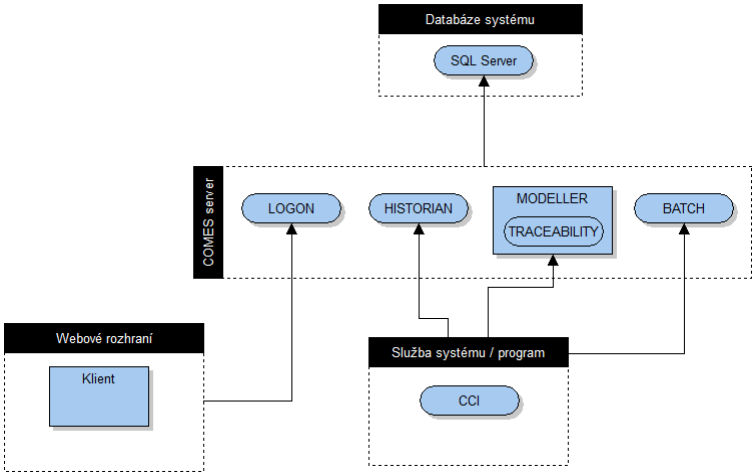


Obr. 2: Úvodní obrazovka systému COMES

2.1 Moduly systému COMES

V této kapitole budou podrobněji rozebrány moduly systému COMES. Systém COMES nemá možnost připojit moduly od jiného výrobce. Systém je tedy v tomto směru uzavřený. Zákazník však má možnost určit si, které moduly budou pro jeho aplikaci využitelné a které chce pro svoji aplikaci do systému připojit[3]. Struktura celého systému je zobrazena na blokovém schématu na Obr. 3.

[3] Tato kapitola a následné podkapitoly čerpají z [7][8][9][10][11][12][23][24]



Obr. 3: Blokové schéma systému COMES [12]

2.1.1 COMES Logon

Prvním modulem je modul COMES Logon. Tento modul je základním kamenem systému a je v systému integrován vždy. Je určen hlavně pro správu uživatelů, správu přístupu do systému, diagnostiku modulů, správu událostí a správu číselníků pro celý systém.

Jednotliví uživatelé jsou zařazováni do tzv. uživatelských skupin. Těmto skupinám je možné nastavovat přístupová práva do jednotlivých částí systému a umožnit tak variabilní uzpůsobení vzhledu systému pro jakoukoli skupinu pracovníků od manažerů, přes vývojové pracovníky, techniky až po správu budovy a ostrahu. Je možné nastavit taková práva, aby někteří pracovníci mohli vidět například pracovní plán, ale nebyla jim umožněna editace.

Pro přístup pracovníků do systému je možné využít například vstupních čipů, případně se připojit i skrze WinCC. Důležitými prvky modulu je správa licencí, které omezují počet uživatelů a dlouhodobá záloha systému, ve které je možné nastavit zálohování jednotlivých modulů a jejich součástí.

V modulu lze také vytvořit model zařízení. Tímto modelem je částečně definováno menu, kde je možné přiřadit formuláře a další prvky systému do jednotlivých částí stromové hierarchie modelu a vytvořit tak intuitivní ovládání celého systému.

2.1.2 COMES Historian

Modul COMES Historian v systému slouží pro sběr dat z technologie a jejich archivaci, prezentaci a analýzu dat. Umožňuje tvorbu trendů a alarmových hlášení. Trendy lze upravovat podle potřeby a zobrazit v nich různé proměnné v čase. Lze je také využít přímo ve formulářích modulu COMES Modeller. Pomocí webové služby lze naopak data získávat z modulu COMES Modeller, nebo aplikací třetích stran. Pro sbírání dat a jejich následnou prezentaci pomocí trendů a alarmových hlášení je nutné mít vytvořené interní, nebo externí tagy v tzv. zdroji tagů a napojit se tak například na PLC. S daty ze zdrojů tagů lze pracovat ve skriptech. To přináší možnost různých matematických výpočtů.

2.1.3 COMES CCI

Modul CCI slouží jako komunikační rozhraní se systémy připojenými k systému COMES. Propojuje různé vrstvy řízení podniku a shromažďuje pro systém COMES data z nižších vrstev technologie, vstupních zaměstnaneckých čipů a ERP systémů. Skládá se z klienta a serveru. CCI server zajišťuje pravidelné čtení dat a jejich zprostředkování klientům. Data jsou do tohoto programu přijímána přes protokol TCP/IP. Spouštět jej lze jako běžný program, nebo automaticky jako služba. Pomocí modulu COMES CCI jsou načítány definované tagy ze zdroje tagů a poté lze v systému COMES pracovat s jejich hodnotami.

2.1.4 COMES Batch

Modul Batch slouží k obsluze dávkových procesů. Modul umožňuje vytvořit a spravovat receptury a předpisy životního cyklu výrobku, řídit a spravovat fáze výroby a další operace. Umožňuje plánování výrobních dávek, jejich řízení a zaznamenávání historie výrobní dávky. Z těchto funkčností lze vygenerovat protokol. Modul vychází ze standardu ANSI/ISA 88.

2.1.5 COMES Traceability

Modul COMES Traceability v systému slouží pro tvorbu rodokmenů, ke sledování stavů materiálů v technologických zařízeních a k ukládání procesních a kvalitativních parametrů k materiálových položek.

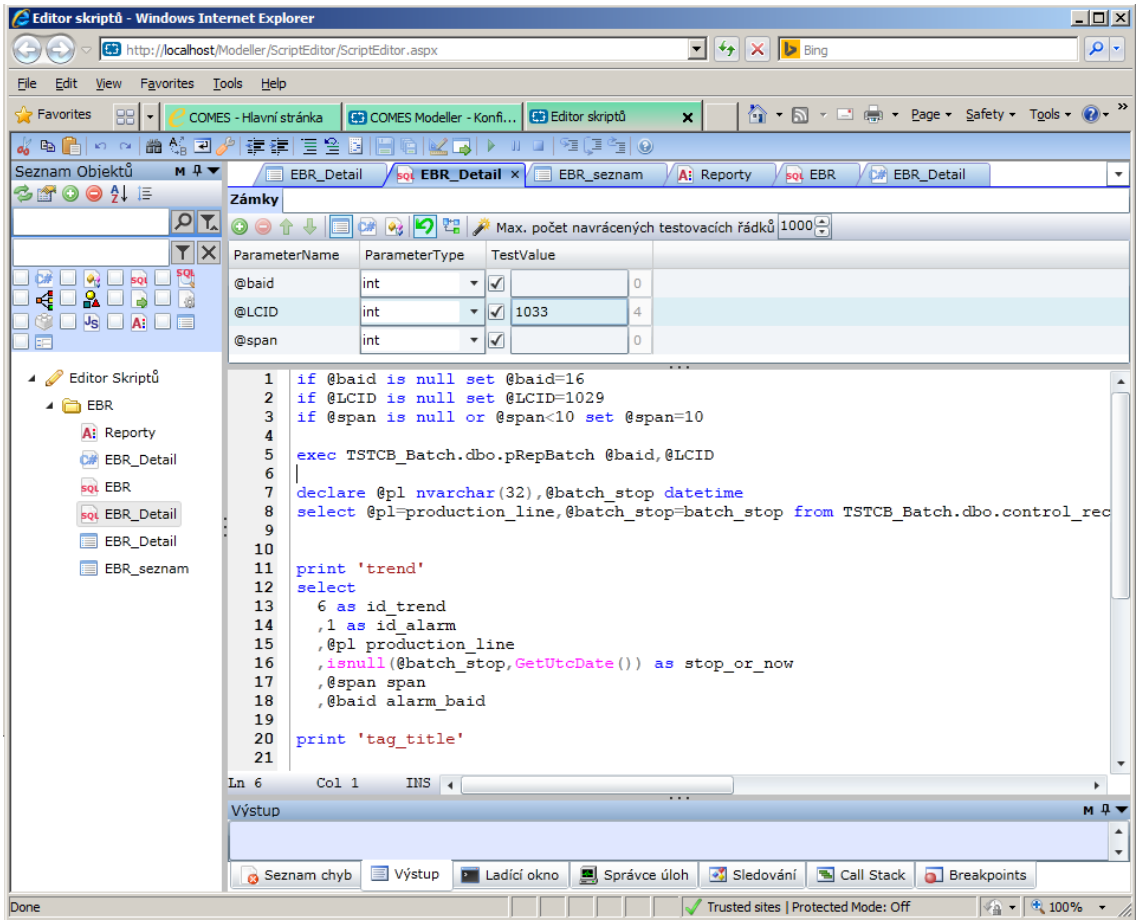
2.1.6 COMES Modeller

Tento modul je pro tuto práci stěžejním modulem systému COMES. Slouží k tvorbě zákaznických aplikací, což znamená, že umožňuje tvorby formulářů, protokolů, reportů a různých výpočtů, dále identifikaci objektů a označování výrobků například čárovými kódy. Vzhledem k tomu, že COMES Traceability běží na stejném základu, jako COMES Modeller, umožňuje COMES Modeller společně s modulem COMES Traceability sledování výrobků a materiálu. Dále umožňuje tvorbu operátorských terminálů a přehledové vizualizace. Právě tato funkčnost je důležitá pro tuto práci, protože data získaná z vizualizace systému WinCC budou převáděna do modulu COMES Modeller. Proto budou některé prvky sloužící k prezentaci dat tohoto modulu rozebrány více.

2.1.6.1 Editor skriptů

Modul COMES Modeller obsahuje zabudovaný editor skriptů, díky kterému je možné přímo v okně webového prohlížeče vytvářet uživatelskou aplikaci (Obr. 4). Editor umožňuje otestovat funkčnost skriptu. Typy skriptů, které je možné vytvořit, jsou: C# skript, C# třída, JavaScript, Css, sql procedura, sql pohled, strukturovaný datový typ,

globální proměnná, webová služba, ASPX stránka, Animace a Formulář. Právě poslední tři jmenované typy skriptů mohou být využity pro vizuální prezentaci dat.



Obr. 4: Editor skriptů systému COMES

2.1.6.2 ASPX stránka

ASPX stránka se vyznačuje dvěma typy kódu. Hlavní kód, kterým je vytvářen vzhled uživatelské aplikace. Tento kód je tvořen pomocí jazyka HTML a lze využít CSS stylů a JavaScriptu. Každý prvek, který má být ovládán programově druhým typem kódu, musí obsahovat parametr runat = „server“. Druhý typ kódu je výkonný kód, tzv. code behind. V něm se nachází kód v jazyce C#, který zajišťuje ovládání prvků vykreslovaných v kódu vzhledu, například změnu barvy určitého prvku se změnou hodnoty tagu. Zobrazená data je možné vkládat pomocí SQL procedur, které zajistí přenos dat z databáze.

2.1.6.3 Animační objekt

Animační objekt je dalším důležitým prvkem pro vizualizaci technologie. Podobně jako v případě ASPX stránky obsahuje animační objekt kód vzhledu a výkonný kód, tzv. code behind. Takto naprogramovaný prvek může být opakovaně vkládán do formulářů a ASPX stránek, což je velice důležitá vlastnost právě při tvorbě průmyslové vizualizace, kde se konkrétní typy grafických objektů mohou mnohokrát opakovat. Jako příklad lze uvést vizualizaci ventilů, jejichž četnost je v technologii často velká a funkčnost většiny z nich

stejná. Lze tak vytvořit jednotné grafické zobrazení například pro dříve jmenovaný ventil a pomocí výkonného kódu dát tomuto objektu schopnost měnit vzhled podle stavu technologie (hodnoty konkrétní proměnné, navázané na animační objekt), případně k vizualizaci tohoto prvku zobrazovat hodnotu veličiny, související s konkrétním prvkem. Lze jej využít také k programování drobných funkcí formuláře, například ke vložení nového řádku do tabulky zobrazené ve formuláři.

Do uživatelských stránek a formulářů je vkládán pomocí tagu *iframe*. Tento tag definuje rozhraní, mezi proměnnými definovanými v animačním objektu a proměnnými nebo konstantami, definovanými vně objektu. V tagu *iframe* musí být uveden atribut *animation*, jehož hodnota definuje, který animační objekt bude volán. Pro nastavení hodnot, důležitých pro animační objekt lze definovat proměnné uvnitř tohoto objektu. Při napojení těchto proměnných lze rozlišit dva případy. Prvním případem je vložení konstanty, tedy hodnoty, která se nebude měnit. Tohoto typu definice lze využít například pro definici požadované velikosti animačního objektu, který je definován univerzálně tak, aby se jeho velikost mohla měnit:

```
<iframe animation="CFP31_AQM_pump_down" style="display:block;"
Inp_Width="0" Inp_Height="0"></iframe>
```

Druhým typem je navázání proměnných na tagy (komunikační proměnné) systému COMES. V tomto případě je nutné zadat název tagu (proměnné), název CCI serveru, a dobu obnovení v milisekundách:

```
<iframe animation="CFP31_AQM_pump_down" Inp_CollectValue_Tag =
"CFP31/AQM01.GRP_STAT" Inp_CollectValue_CCI = "OPCDAW"
Inp_CollectValue_Sample="1000"></iframe>
```

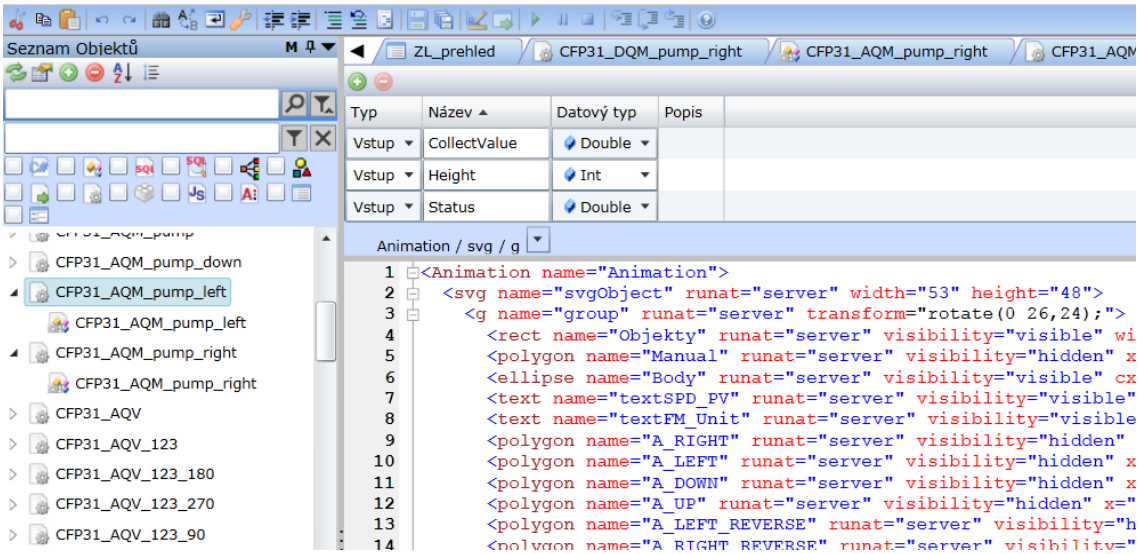
Vstupně výstupní proměnné lze definovat v různých datových typech dle potřeby, čímž je zajištěna univerzálnost použití. V této práci je využit hlavně pro převod animačních objektů, popsaných v kapitole 3.2.1, jakými jsou například grafické displeje, vizualizace motorů, čerpadel a mnohých dalších prvků. Pro vstupní a vstupně výstupní proměnné lze využít následujících datových typů:

- Bool
- DateTime
- Decimal
- Double
- Int
- Long
- Object
- String
- Timespan

Další možností, kterou animační objekt umožňuje, je obsluha událostí. V definici animačního objektu se mezi proměnné, sloužící jako rozhraní animačního objektu nadefinují názvy těchto událostí, jejichž obsluha je poté naprogramována do výkonného

kódu objektu. Událost je podobně jako komunikační proměnné navázána na zbytek systému definicí v tagu *iframe*:

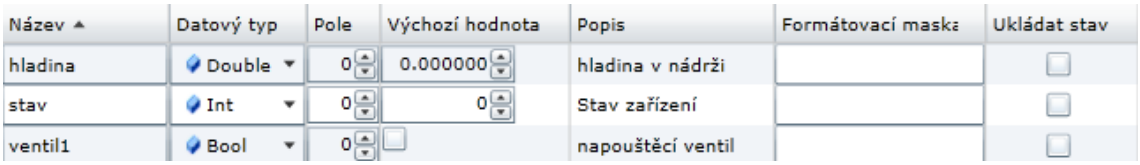
```
<iframe style="width: 700px; height: 30px;" Evnt_Zapsat =
"Material_VadaAdd" animation="AddDefect"></iframe>
```



Obr. 5: Obrazovka editoru skriptů s definicí vzhledu animačního objektu a jeho proměnných

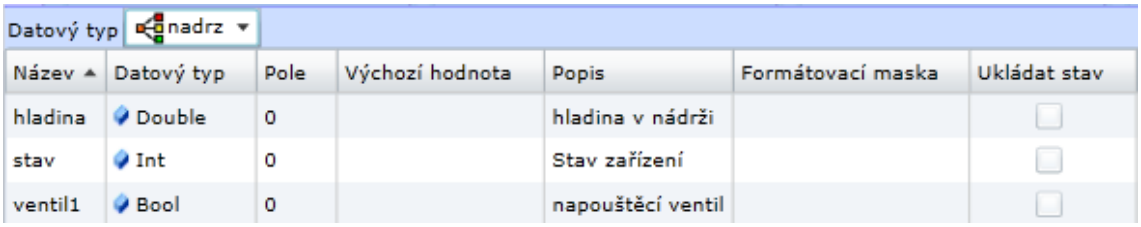
2.1.6.4 Uživatelský formulář

Nejčastějším prvkem, který se v systému COMES využívá k prezentaci dat a tvorbě vizualizací, jsou uživatelské formuláře. Tvoří se pomocí HTML kódu a je možné využívat JavaScriptu, CSS, C# skriptů pro úpravu zobrazovaných dat a C# skriptů spouštěných po uložení dat z formuláře. Pro přístup k datům může být vytvořena SQL procedura nebo SQL pohled, pomocí kterých budou data do formulářů vkládána a případně měněna uživatelem a zpětně uložena. Další možností je navázat na konkrétní prvek formuláře tag systému COMES a zobrazovat jeho hodnotu přímo, nebo pomocí enumerace hodnotu odpovídající konkrétní hodnotě proměnné. Poslední možností je přístup k datům pomocí globálních proměnných systému COMES. Globální proměnná je v podstatě strukturou, která může zapouzdřovat více proměnných s různými datovými typy. Je tedy možné definovat globální proměnnou obsahující například proměnnou datového typu *Double*, jednu proměnnou datového typu *Integer* a podobně. Strukturu si určuje programátor sám vytvořením tzv. strukturovaného datového typu. Ten definuje vnitřní strukturu globální proměnné, tedy názvy vnitřních proměnných a jejich datové typy (Obr. 6).



Obr. 6: Definice strukturovaného datového typu s názvem Nadrz

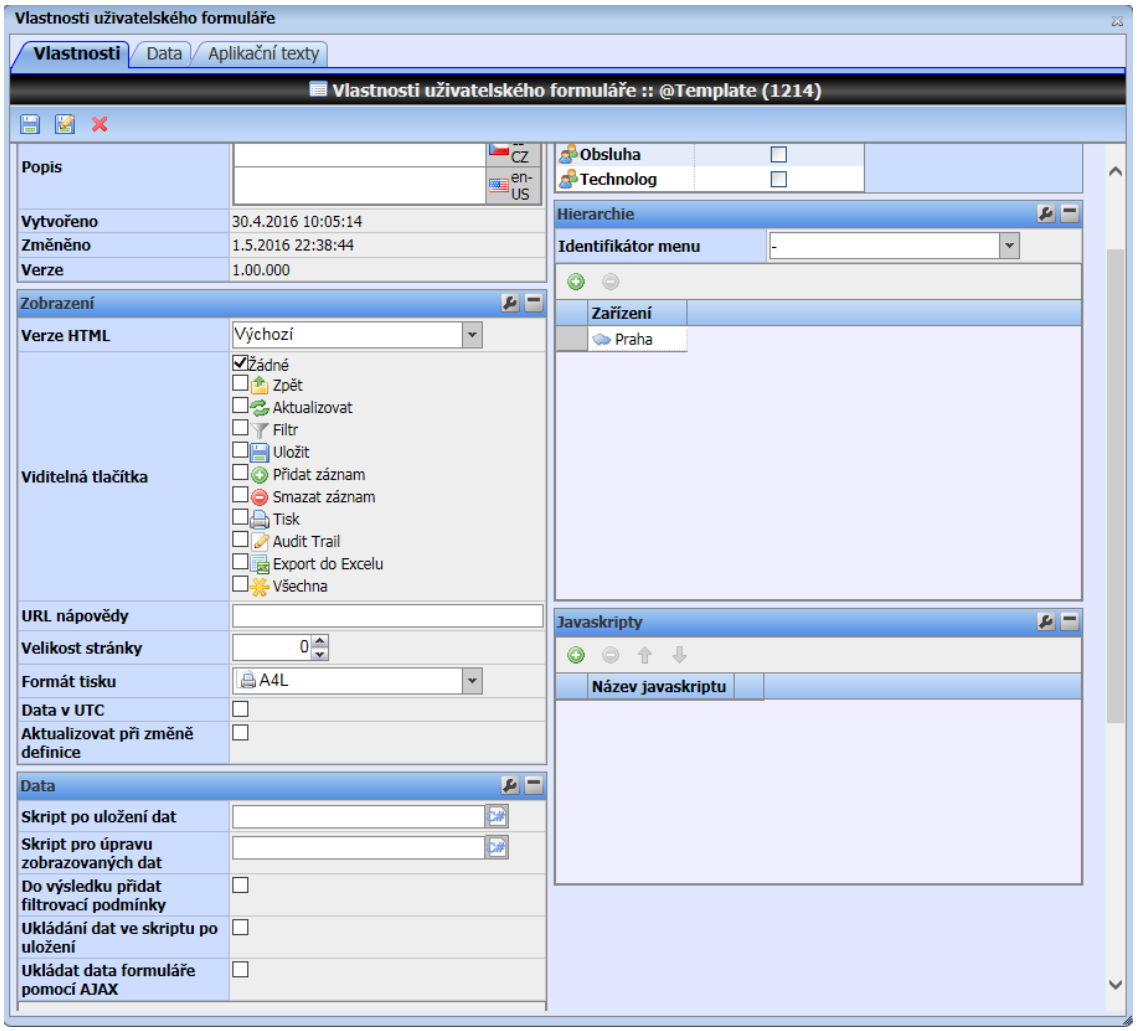
Vytvořením globální proměnné programátor tedy vytvoří v podstatě instanci strukturovaného datového typu s definovaným názvem (Obr. 7).



Název	Datový typ	Pole	Výchozí hodnota	Popis	Formátovací maska	Ukládat stav
hladina	Double	0		hladina v nádrži		
stav	Int	0		Stav zařízení		
ventil1	Bool	0		napouštěcí ventil		

Obr. 7: Definice globální proměnné s názvem *Nadrz1*, která je datového typu *Nadrz*, jehož struktura je uvedena na Obr. 6

Data lze ve formulářích zobrazovat tak, aby bylo možné jejich hodnoty měnit a upravené následně ukládat na původní místo. Do formulářů lze vložit animační objekt pomocí HTML tagu *iframe* nebo zakomponovat do celkového vzhledu formuláře například trendy z modulu COMES Historian. Stejně jako v případě ASPX stránky je možné pro formulář definovat jazykovou lokalizaci jednotlivých prvků v závislosti na nastavení jazyka systému. Formuláře je možné nastavit tak, aby po přihlášení do systému byly zobrazeny pouze uživatelům s dostatečnými právy pro přístup (Obr. 8).



Obr. 8: Část okna pro nastavení formuláře. Jak je patrné, lze nastavit základní vzhled a další parametry. V dalších záložkách lze definovat automatické filtry a jejich parametry a aplikační texty pro konkrétní formulář

3 SIEMENS SIMATIC WINCC

SIMATIC WinCC je systém pro procesní vizualizaci s výkonnými funkcemi pro sledování automatizovaných procesů, vyvinutý firmou Siemens. Pomocí tohoto systému je možné vytvářet jak vizualizaci pro PC, tak pro operátorské panely. Systém je plně škálovatelný. Integruje v sobě všechny funkčnosti systému SCADA s možností rozšíření, která přidávají funkčnosti ze systémů MES a ERP. Umožňuje tvořit systém s jedním, nebo více uživateli. Umožňuje využití redundantních serverů[4].

Existuje ve dvou základních verzích. První verzí je WinCC complete package, označovaná také RC (Runtime and Configuring). Druhou je WinCC runtime package, též RT. Tyto verze se dále dělí podle počtu tak zvaných PowerTags (od 128 až po 256 000), které je možné v systému využít. Lze je nazývat také procesními tagy. Propojení s řídicím systémem je řešeno přes komunikační kanál WinCC. Systém obsahuje množství nástrojů, které budou popsány v následujících kapitolách. Další nástroje lze do WinCC dodatečně přikoupit, ale vzhledem ke skutečnosti, že se jimi tato práce nezabývá, nebudou dále rozebírány.

3.1 Nástroje systému SIMATIC WinCC

Systém SIMATIC WinCC v sobě integruje mnoho nástrojů. Tyto nástroje lze rozdělit do několika kategorií, přičemž každá z kategorií obsahuje několik různých nástrojů z dané oblasti. Jednotlivé kategorie a jejich prvky budou popsány v této kapitole.

3.1.1 Škálovatelnost řízení výroby

Tato kategorie obsahuje nástroje pro rozšíření podniku podle způsobu a velikosti řízení. Tyto nástroje posunují systém WinCC nad rámec funkčností systémů SCADA a přibližují jej ke skupině MES systémů.

3.1.1.1 WinCC/Server

Nástroj WinCC/Server umožňuje změnu systému z jednouživatelského systému na systém klient-server s možností až 32 klientů a 12 redundantních serverů.

3.1.1.2 WinCC/Central Archive Server (CAS)

CAS slouží jako server pro centralizovanou archivaci procesních dat a zpráv z technologie v reálném čase. Umožňuje nastavení redundance ukládání dat. Je postaven na Microsoft SQL Server. Nástroj WinCC/Central Archive Server je v novějších verzích nahrazen nástrojem SIMATIC Process Historian.

[4] Tato kapitola a následné podkapitoly čerpají z [13][14][15][16]

3.1.1.3 WinCC/WebNavigator

Tento nástroj umožňuje obsluhu a monitorování závodů pomocí webového prohlížeče. Na zařízení připojeném do podnikové sítě. Pro zobrazení je možné využít Microsoft internet Explorer, nebo prohlížeč integrovaný do WinCC, pojmenovaný WinCC Web Viewer. Tento nástroj by bylo možné využít pro export grafických dat do formátu podporovaného webovými prohlížeči, ale problém nastává při využití animačního objektu vytvořeného firmou Compas. Tento objekt není kompatibilní s nástrojem WebNavigator.

3.1.1.4 WinCC/WebUX

WebUX je nástroj, který umožnil rozšíření použití systému na mobilní zařízení, jako jsou chytré telefony a tablety a další zařízení podporující HTML5. Díky tomuto nástroji je možno sledovat chod výrobního podniku z kteréhokoli místa s přístupem na Internet. Další výhodou tohoto nástroje je, že není třeba složitá konfigurace systému. Nástroj umožňuje komunikovat pomocí zabezpečeného protokolu HTTPS.

3.1.1.5 WinCC/TeleControl

Nástrojem TeleControl je umožněno napojit se a zahrnout do vizualizace komponenty řady SIMATIC, které jsou připojeny do sítě. Využití nachází například ve vodárenství, kde jsou komponenty rozesety na obrovské ploše.

3.1.2 Zpracování dat, konektivita a integrace

Tato kategorie obsahuje nástroje pro práci s daty a jejich analýzu. Umožňují propojení systému s ostatními nástroji a podnikovým software.

3.1.2.1 WinCC/DataMonitor

DataMonitor zajišťuje zobrazení, analýzu a distribuci procesních a historických dat z databáze procesu. Umožňuje tak přenos těchto dat do všech úrovní řízení podniku. K distribuci využívá Microsoft Internet Explorer nebo Microsoft Excel. Informace o procesu lze též rozesílat pomocí e-mailových zpráv.

3.1.2.2 WinCC/DowntimeMonitor

Jak již název napovídá, tento nástroj umožňuje detekci a analýzu prostojů jednotlivých zařízení centrálně. Pro konkrétní zařízení ve výrobním řetězci je schopen vypočítávat statistické parametry, jako OEE (celková efektivita zařízení), MTBF (Střední doba mezi poruchami), MTFF (střední doba první poruchy) a další. Od verze WinCC V7.2 je nahrazen nástrojem WinCC/PerformanceMonitor, obsahujícím stejné funkčnosti, jako WinCC/DowntimeMonitor.

3.1.2.3 WinCC/ConnectivityPack

Tímto nástrojem je umožněno propojení a přístup do archivu WinCC pomocí standardů OPC XML DA, OPC HDA, OPC A&E, WinCC OLE-DB. Další možností využití tohoto nástroje je přístup z WinCC do jiných aplikací. Nástroj vytváří rozhraní pro propojení s jiným software. Díky tomu je možné se napojit do systémů typu ERP, MES, nebo Microsoft Excel).

3.1.2.4 WinCC/IndustrialDataBridge

Pro vytvoření datového spojení mezi WinCC průmyslovými systémy od jiných výrobců se využívá nástroje IndustrialDataBridge. Nástroj zajišťuje správu chyb a bezpečný přenos dat mezi těmito systémy. Je možné připojit například Microsoft SQL Server, Microsoft Excel, Oracle MySQL Server, nebo data z WinCC archivů. Data je možné spravovat a měnit, a to i pomocí webového rozhraní. Nástroj umí také tvorbu reportů.

3.1.2.5 SIMATIC Information Server

Tento nástroj je určený výhradně pro tvorbu reportů na základě analýzy historických dat z WinCC. Využívá technologie Microsoft Reporting Services. Reporty vytvořené tímto nástrojem jsou centrálně spravovány a jsou přístupné pomocí webového rozhraní, e-mailu. Nástroj využívá technologií Microsoft Office. Nástroj je určený i pro systém SIMATIC PCS 7.

3.1.3 Rozšíření dosažitelnosti

Nástroje z této kategorie mají za úkol kontrolu chyb a zdrojů dat.

3.1.3.1 WinCC/Redundancy

Nástroj redundanci zajišťuje vzájemné monitorování a propojení redundantních stanic, serverů a spojení. Umožňuje tak dosáhnout integrity dat a ochraňuje systém proti výpadku některého členu řetězce přenosový kanál – stanice – server. V případě chodu bez poruchy má každá stanice, nebo server své vlastní připojení. V případě výpadku a jeho následného obnovení jsou data z funkčního serveru nebo stanice přenesena na pozadí na poškozený prvek. V případě, že jsou data přenesena, systém je opět redundantní.

3.1.3.2 WinCC/ProAgent

Tento nástroj umožňuje rychlou diagnostiku chyb jednotlivých strojů, nebo výrobních zařízení. Tuto funkčnost umožňuje díky napojení na nástroje systému SIMATIC. Využívá nástrojů řídicích systémů SIMATIC S7.

3.1.3.3 SIMATIC Maintenance Station

SIMATIC Maintenance Station je komponenta systému SIMATIC PCS 7. Jedná se o distribuovaný systém řízení procesů. Nástroj SIMATIC Maintenance Station umožňuje

zobrazit data o celé technologii a má za úkol maximalizovat efektivitu a redukovat prostoje technologie.

3.1.4 Validace a trasování

V kapitole validace a trasování jsou uvedeny nástroje pro správu životního cyklu výrobků a sledování změn procesu. Tyto funkčnosti jsou pro průmyslový systém velice důležité a umožňují rychlé řešení problémů způsobujících problémy ve výrobní technologii.

3.1.4.1 WinCC/Audit

Nástroj Audit slouží k monitorování aktivit operátorů v procesu řízení a k záznamu změn aktuálně běžícího řídicího programu. Změny jsou ukládány do zabezpečené databáze, nazývané Audit Trail. Díky tomuto nástroji je umožněno dohledávat chyby způsobené zásahy operátorem a urychluje tak řešení chyb.

3.1.4.2 WinCC/ChangeControl

ChangeControl je nástrojem pro monitorování projektu. Oproti WinCC Audit ale sleduje změny konfigurace celé aplikace, například změnu správy tagů, uživatelských skupin a uživatelských souboru a dokumentů. Nástroj umožňuje porovnávání verzí uložených položek a umožňuje tak podobně jako v případě WinCC/Audit porovnávat a dohledávat chyby.

3.1.5 Rozšíření SCADA funkčností

Tato kapitola obsahuje rozšíření funkcí systému WinCC nad rámec obecně známých funkčností SCADA systémů.

3.1.5.1 WinCC/User Archives

Nástroj User Archives umožňuje připojení do datových archivů uživatelské aplikace mimo systémy firmy Siemens a výměnu a případné změny v datech v těchto archivech. Pro tuto funkčnost obsahuje jednotné uživatelské rozhraní, umožňující snadnou obsluhu. Dále umožňuje import a export funkcí pro další zpracování dat například mezi WinCC a Microsoft Excel.

3.1.5.2 WinCC/Calendar Scheduler and WinCC/Event Notifier

Pomocí Calendar Scheduler jsou do systému integrovány funkce kalendáře, mezi které patří plánování časového rozvrhu a možnost ověření časové platnosti. Vzhledem k tomu, že některé akce je nutné vázat na datum a čas, je pomocí tohoto nástroje možné vytvářet spouštění těchto časově závislých událostí. Nástroj WinCC/Event Notifier umožňuje pomocí funkcí kalendáře informovat konkrétní uživatele o chování technologie. Ke kontaktování uživatelů může být využita e-mailová zpráva, nebo SMS. Nastavení těchto

nástrojů je realizováno přes grafický editor systému WinCC a pomocí editoru skriptů, který je implementován do grafického editoru.

3.1.5.3 WinCC/SES

Nástroj WinCC/SES (Sequence Execution System) je nástroj určený pro řízení sekvenčních procesů a procesů založených na recepturách, například v chemickém průmyslu. Nástroj obsahuje předpřipravené moduly, proto je velice jednoduše aplikovatelný.

3.1.6 Systémová rozšíření

3.1.6.1 WinCC/ODK

V případě, že nástroje integrované ve WinCC a nástroje, kterými je systém možné po dokoupení rozšířit, neplní požadavky zákazníka, je možné si pomocí WinCC/ODK (Open Development Kit) vytvořit vlastní nástroj splňující tyto požadavky. WinCC/ODK specifikuje rozhraní, pomocí kterých je možné přistupovat k datům a funkcím WinCC a zákaznické aplikace. Tvorba těchto aplikací je umožněna pomocí programovacích jazyků C++,C#, nebo VB.NET.

3.1.7 Nastavení spotřeby energie

Nástroje v této kategorii pomáhají optimalizovat provoz z hlediska spotřeby energie a snížit tak náklady na provoz celé technologie.

3.1.7.1 SIMATIC powerrate

SIMATIC powerrate je dalším modulem, který je společný pro systém WinCC a SIMATIC PCS 7. Nástroj sbírá data týkající se spotřeby energie ve výrobním podniku a zprostředkovává tyto data pomocí vizualizace. Zároveň jsou data archivována.

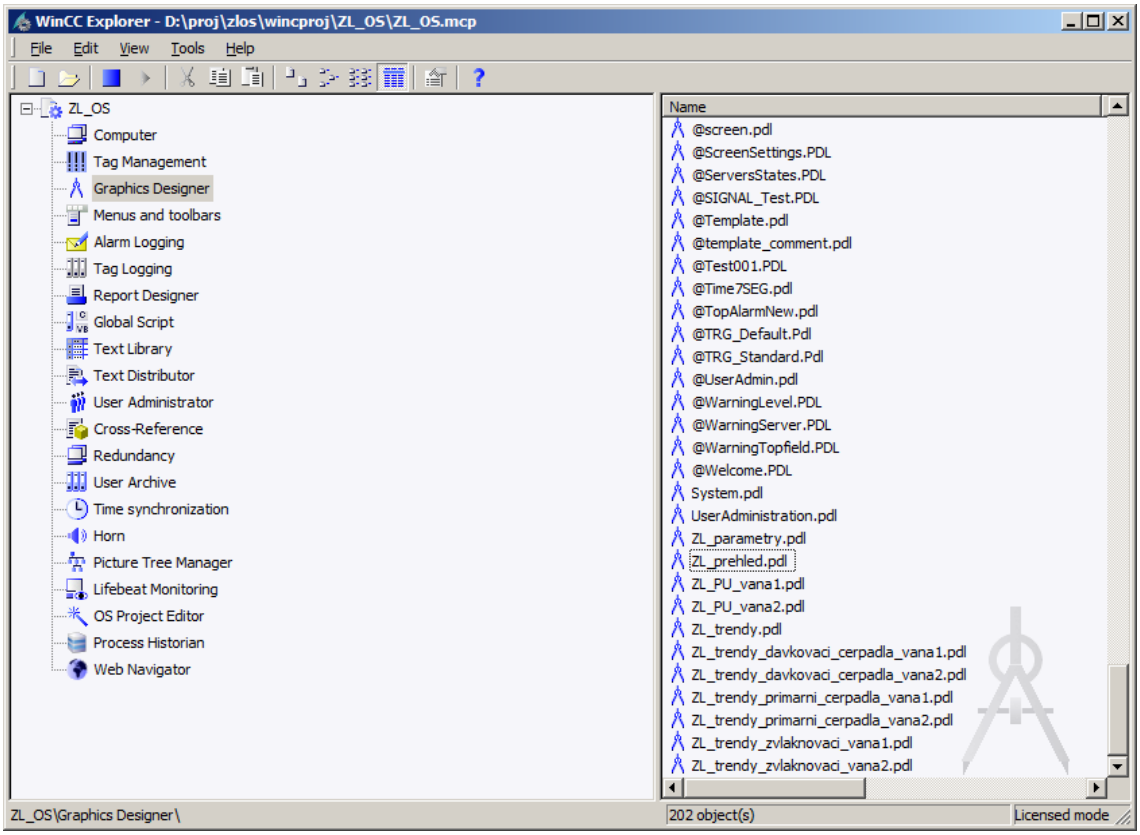
3.1.7.2 WinCC/B.Data

Jednou z možností zvýšení zisků a efektivity výroby je maximalizace úspory energie. Proto byl do WinCC přidán nástroj B. Data, který se stará o správu energie továrny. Nástroj minimalizuje energetické náklady na provoz podniku pomocí řízení spotřeby, plánování a řízení nákupu energie.

3.2 Grafický editor WinCC

Vzhledem k tomu, že je tato práce zaměřena na převod grafických dat z WinCC, je důležité, aby byl uveden nástroj, ve kterém jsou vizualizační obrazovky ve WinCC tvořeny a jaké má tento nástroj možnosti.

Editor umožňuje tvorbu jednotlivých vizualizačních obrazovek a práci s grafickými objekty a jejich vlastnostmi (Obr. 10). Patří mezi základní nástroje systému WinCC. Přístup k tomuto nástroji je realizován pomocí programu WinCC Explorer, ve kterém je uveden seznam dostupných nástrojů systému WinCC (Obr. 9). Tvorba vizualizace je realizována následujícím způsobem. Jednotlivé obrazovky jsou vytvářeny zvlášť v grafickém editoru. Pro správný běh vizualizace je třeba určit, která obrazovka má být využita jako startovací. Do jednotlivých obrazovek můžeme přidávat grafické objekty, ať už statické, například potrubí nebo dynamicky se měnící na základě hodnot, čtených z PLC skrze definované tagy. Příkladem lze uvést ventily, motory a další prvky, jejichž aktuální stav je pro obsluhu důležitý. Tagy mohou být přiřazovány grafickým objektům. Tím můžeme dosáhnout změny vzhledu objektu (například zobrazení výšky hladiny v tanku) nebo vypsání hodnoty tagu, například do textového pole.

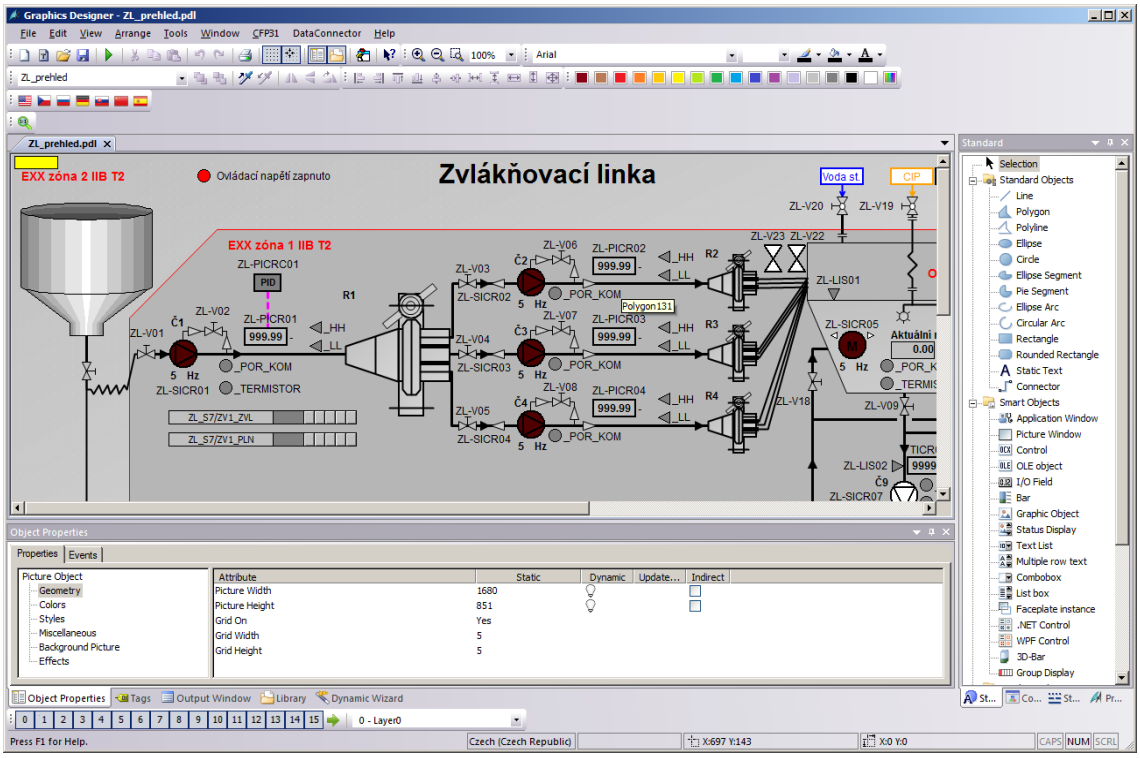


Obr. 9: Program WinCC Explorer se zobrazenými jednotlivými obrazovkami vizualizace projektu

Mezi přidávané objekty lze zařadit základní grafické objekty typu obdélník, čára, tlačítko a další. Složitější objekty lze nalézt v knihovnách. Některé knihovny jsou již integrovány do WinCC. Mezi ně patří knihovna Global library, která obsahuje velké množství grafických objektů z různých odvětví. Příkladem lze uvést velké množství zobrazení tanků, motorů a dalších objektů. WinCC také umožňuje vytvářet vlastní složitější grafické objekty kombinací libovolného počtu základních grafických objektů a vkládat je do vlastních knihoven.

Některé knihovny je třeba pouze přidat, ale jsou již součástí instalace. Mezi takovéto knihovny patří Siemens HMI symbol library, která značně rozšiřuje paletu použitelných grafických prvků. Tato knihovna je také využívána firmou Compas při tvorbě vizualizací pro zákazníka. Vyznačuje se tím, že je napsaná jako prvek ActiveX control. Tyto prvky se obecně vyznačují tím, že po integraci do systému je možné jejich využití ve více programech. Byly vyvinuty společností Microsoft. Jedná se o komponenty, které nejsou samostatně využitelné – vždy musí být přidruženy nějakému programu.

Někdy je nezbytné, aby byly grafické objekty vytvořeny jiným způsobem, než jak jsou definovány v knihovnách společnosti Siemens. Jednou z možností je animační objekt, vytvořený společností Compas. V další podkapitole bude tento objekt popsán včetně definice a důvodů pro jeho vytvoření.



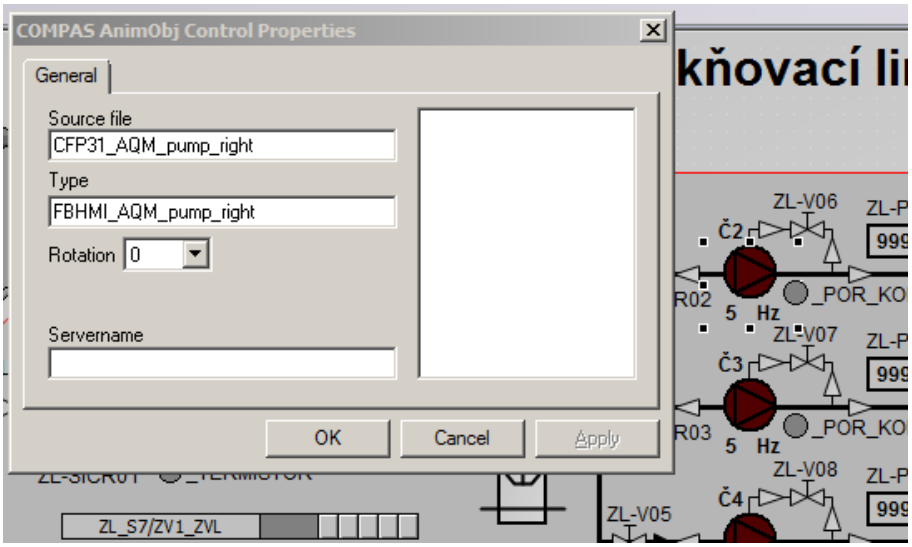
Obr. 10: Grafický editor systému Siemens Simatic WinCC s obrazovkou testovacího projektu

3.2.1 Animační objekt

Podobným způsobem, jako je vložena do WinCC knihovna HMI symbol library, je vložen i animační objekt firmy Compas. Také využívá technologie ActiveX a ukazuje, jak je možné upravovat zákaznickou aplikaci v prostředí WinCC. Animační objekt je prostředkem, který umožňuje definovat vzhled i animaci v jednom definičním souboru a vytvořit tak vzhled výsledného grafického objektu. Je definován externím CSV souborem, který popisuje, jak bude výsledný grafický objekt vyobrazen a animován. Je proto možné vytvářet mnoho grafických objektů. Animační objekt byl vytvořen z toho důvodu, že umožňuje mnohem jednodušší vytváření uživatelem definovaných grafických objektů, přenosnosti do jiných uživatelských aplikací a dále také z důvodu mnohem lepší

rychlosti překreslení v případě, že obrazovka obsahuje mnoho cyklicky překreslovaných prvků. Tento objekt je taky jedním z důvodů, proč nebylo možné pro export grafiky do formátu vhodného pro web využít nástroje WinCC/WebNavigator, protože tento nástroj nepodporuje animační objekt firmy Compas[5].

Do aktuální obrazovky musí být přidán objekt s názvem AnimObj (jak již bylo řečeno, jedná se o objekt ActiveX, definovaný v souboru AnimObj.ocx). Lze jej nalézt v záložce *Controls* grafického designéru systému WinCC společně s dalšími ActiveX objekty umožňujícími vložení do vizualizační obrazovky. V nastavení tohoto objektu je nutné definovat název CSV souboru, ve kterém se nachází kompletní definice výsledného grafického objektu (Obr. 11).



Obr. 11: Definice animačního objektu. Pole „Source file“ udává soubor definující konkrétní animační objekt

Struktura CSV souboru je následující: První částí je definice grafického objektu, uvozeným v první buňce tabulky zápisem *[Objects]*. Jedná se o část definiční tabulky, která specifikuje, z jakých elementů se celý animační objekt skládá a jaké mají tyto elementy vlastnosti. Její vzhled je možné vidět na Obr. 12.

[Objects]														
#Name	Type	Visible	Left	Top	Width	Height	BackColor	BackFlashColor	FillColor	FillStyle	BorderColor	BorderFlashColor	BorderWidth	Borc
Objekty	Frame	1			21	21								
Body	Ellipse	1	3	2	15	15	0xFF00		0xFF00	0	0x0		1	
Sim	Text	1	0	10	11	11	0xFFFF		0xFF00	0	0x0		1	
Bypass	Text	1	10	10	11	11	0xFFFF		0xFF00	0	0x0		1	
ComErr	Text	1	10	10	11	11	0xFFFFFF		0xFF00	0	0x0		1	
BadConf	Text	0	10	10	11	11	0xA1FF		0xFF00	0	0x0		1	
Occ	Text	0	0	0	11	11	0xFFFFFF		0xFF00	0	0x0		1	

Obr. 12: Příklad části definičního souboru animačního objektu, definující elementy, ze kterých se výsledný animační objekt skládá

[5] Tato kapitola čerpá z [17]

Tabulku je možné vyexportovat i přímo z WinCC pomocí nabídky přidané firmou Compas. Tabulka obsahuje tyto vlastnosti:

- Název elementu
- Typ elementu (dostupné jsou typy Rectangle, Ellipse, Polygon, Polyline, Text, Frame)
- Viditelnost elementu
- X, Y souřadnice levého horního bodu elementu – každý z parametrů má vlastní buňku
- Šířka, Výška elementu – každý z parametrů má vlastní buňku
- Barva pozadí elementu
- Barva pozadí pro blikání
- Způsob vyplňování (zda bude objekt s výplní – hodnota 0, nebo bez – hodnota 65536)
- Barva čáry, nebo okraje
- Barva čáry, nebo okraje pro blikání
- Tloušťka čáry nebo okraje
- Seznam bodů čáry nebo polygonu. Body jsou zadávány dvěma čísly, oddělenými dvojtečkou. V buňce je kompletní seznam bodů, tedy například: 0:0 0:10 10:10
- Barva textu
- Barva textu pro blikání
- Název fontu pro text
- Velikost fontu
- Volba tučného písma
- Volba písma typu kurzíva
- Volba potrženého písma
- Textový řetězec
- Horizontální zarovnání textu
- Vertikální zarovnání textu
- Vypnutí transformací

Barvy jsou zadávány ve formě kódu RGB v šestnáctkové soustavě následujícím způsobem – *0xBBGGRR*, tedy například *0x09ABFF*. Tato tabulka je od následujících oddělena jedním prázdným řádkem.

Další částí specifikace jsou definice animací objektu. Definují, které elementy se budou měnit a jakým způsobem. Je možné vytvořit tři typy definic. Všechny jsou shodně uvedeny zápisem *[Table]* v první buňce této tabulky. Příklad definice jednoho typu tabulek je možné vidět na Obr. 13.

[Table]								
Status	Mask							
Mask	Value	Occ.Visible						
0x01000000	0x01000000	1						# Occupied active
0x0	0x0	0						
[Table]								
CollectValue	Mask							
Mask	Value	Body.BorderColor	Body.BorderFlashColor					
0x40000000	0x40000000	0x0000FF	0x000000					#new error
0x00800000	0x00800000	0x00FFFF	0x000000					#new warning
0x20000000	0x20000000	0x0000FF						#error
0x00400000	0x00400000	0x00FFFF						#warning
0x0	0x0	0x0						#no error

Obr. 13: Část definice animačního objektu pro definování animací jednotlivých elementů v závislosti na hodnotách definovaných proměnných

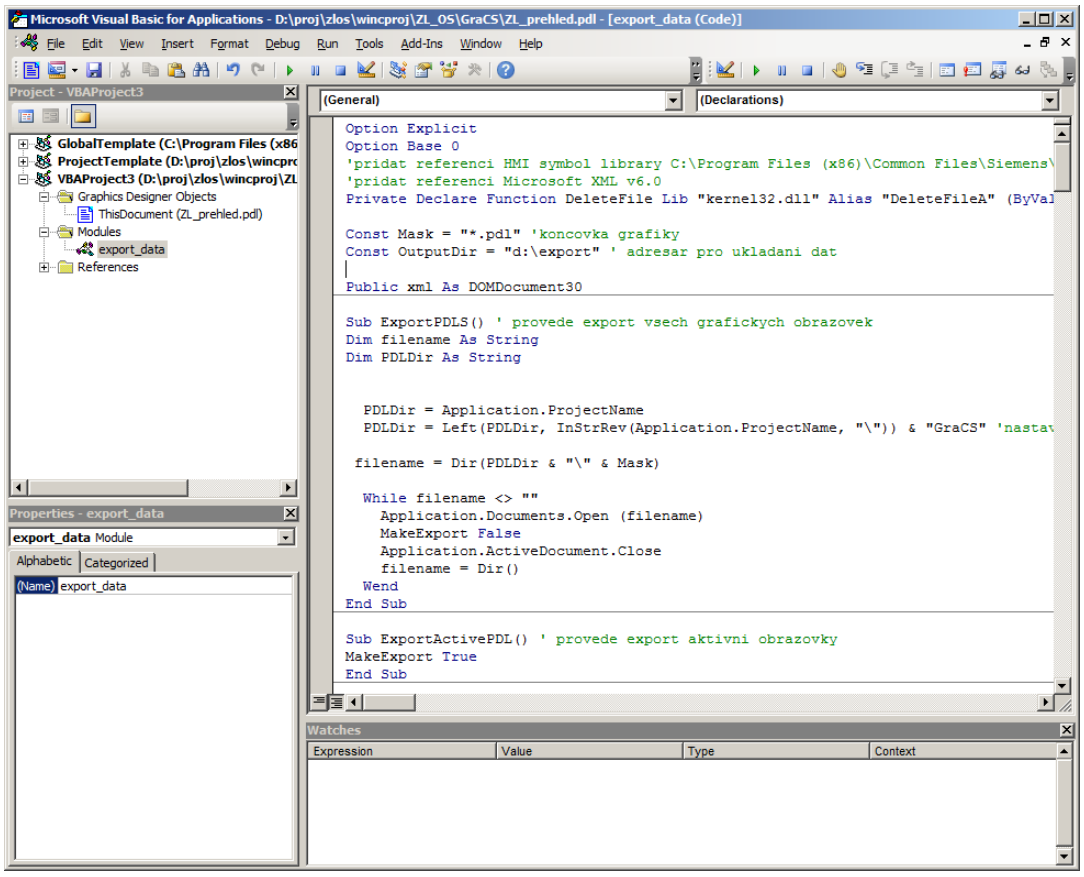
V buňce pod tímto zápisem je uvedena proměnná, do které je ve WinCC namapován tag, podle něhož se budou měnit parametry uvedené v tabulce. Ve vedlejším sloupci je uveden typ tabulky animací. Těchto tabulek je možné definovat libovolné množství podle potřeby a složitosti objektu. Od dalších tabulek animací je vždy oddělena posledním prázdným řádkem a novým uvozením tabulky pomocí buňky *[Table]*. Prvním typem tabulky animací je typ *Mask*. Tato tabulka obsahuje v prvním sloupci hodnoty masky, pomocí které je vstupní hodnota bitově maskována. Ve druhém sloupci se nacházejí možné hodnoty. V hlavičce dalších sloupců jsou vypsány elementy a jejich parametry, které budou měněny. Využívá se tečková konvence, tedy pokud je v tabulce elementů (Obr. 12) uveden element s názvem *Body* a je nutné měnit barvu pozadí, zápis v tabulce animací (Obr. 13) bude *Line1.BackColor*. Pod touto hlavičkou jsou uvedeny hodnoty, které budou nastaveny do této vlastnosti. Druhým typem tabulky animací je typ *Compare*. Hodnota proměnné je porovnávána s hodnotami v prvním sloupci této tabulky. Pokud je menší nebo rovna, tak se nastaví vlastnosti definované opět v hlavičkách v dalších sloupcích. Třetím typem tabulky je *Format*. Je určen pro převod reálné, nebo textové hodnoty na text pomocí standardního formátovacího řetězce v jazyku C. U této tabulky je v prvním sloupci místo reálných hodnot uvedena na prvním řádku vlastnost elementu – například *Text1.Text*. Ve druhém řádku je poté formátovací řetězec, například *%s*.

Tím specifikace animačního objektu končí. Mezi velké výhody jistě patří kromě vyšší plynulosti překreslování obrazovek díky využití těchto objektů oproti objektům společnosti Siemens také jednoduchá tvorba a díky specifikaci v externích souborech také možnost přenosnosti do jiných aplikací, kde je tento ActiveX objekt vložen.

4 PŘEVOD VIZUALIZACE Z WINCC DO XML

V předchozí kapitole byl rozebrán grafický editor a jeho možnosti pro tvorbu uživatelské grafiky pomocí uživatelského rozhraní. Tento editor obsahuje ještě jeden velice užitečný nástroj, díky kterému je možné grafické objekty do obrazovek přidávat, upravovat (i za běhu vizualizace) nebo exportovat jejich parametry do externích souborů pomocí skriptů. Tím nástrojem je Visual Basic editor (Obr. 14). Jak již název napovídá, lze v něm vytvářet skripty s použitím programovacího jazyka Visual Basic. Pomocí jednoduchých skriptů lze editovat aktuálně otevřenou obrazovku vizualizace i ostatní, neaktivní okna v celém projektu. Skripty mohou být spouštěny různými způsoby. Lze je spouštět manuálně z VB editoru, pomocí tlačítka přidaného do vizualizace, nebo automaticky, jako reakci na událost vzniklou v běžící vizualizaci[6].

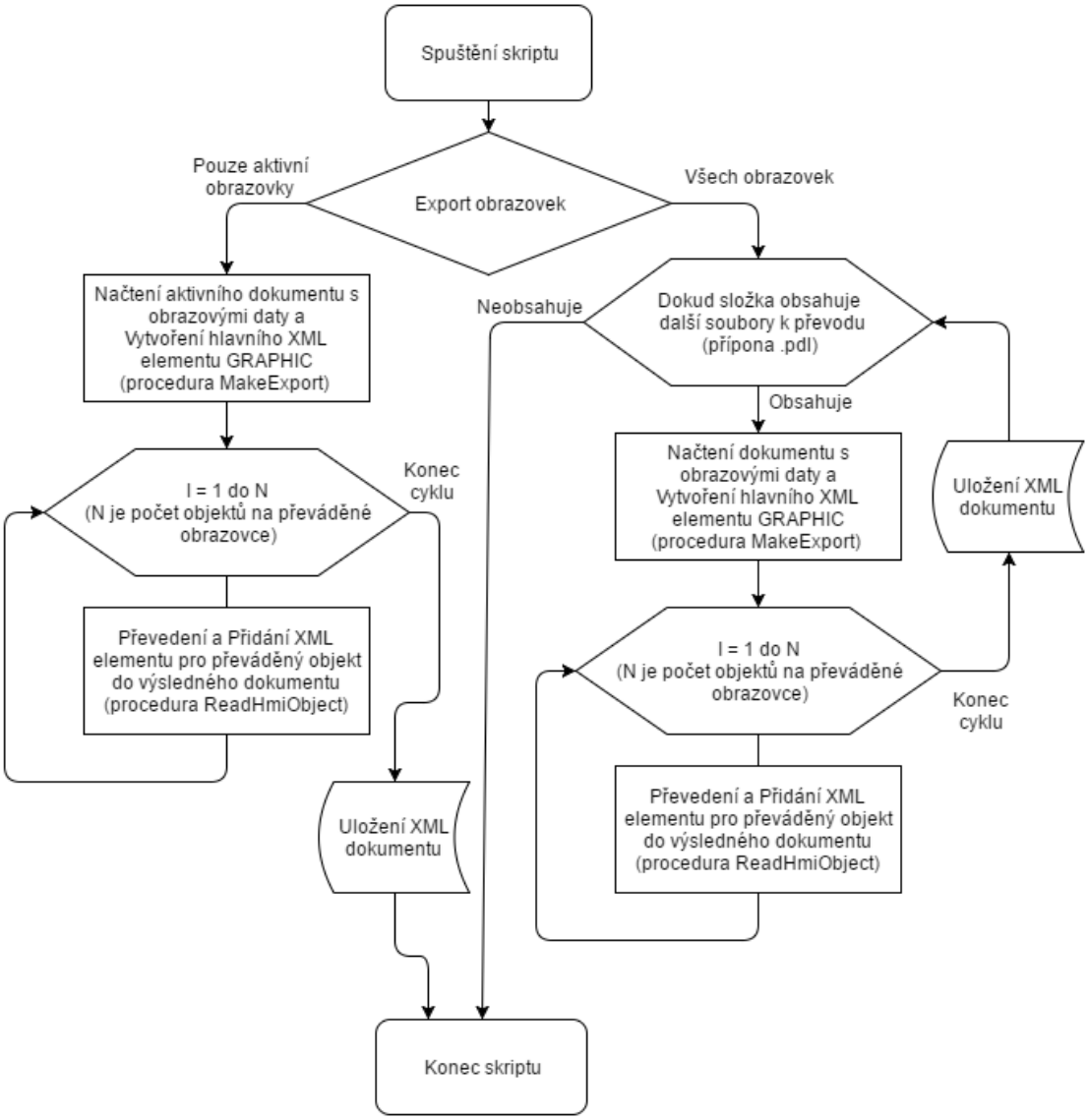
Ve VB editoru lze vytvořit tři typy kódu – *module*, *class module* a *user form*. První jmenovaný umožňuje psát skripty a vkládat do nich funkce a procedury. Druhý jmenovaný vytváří třídu – umožňuje vytvořit vlastní objekt s vlastnostmi a metodami. Posledním jmenovaným je user form, tedy uživatelský formulář pro vkládání dat.



Obr. 14: VB editor, zabudovaný v grafickém editoru programu Siemens Simatic WinCC

[6] Tato kapitola čerpá z [16]

Převod, který jsem vytvořil ve VB editoru, lze využít pro převod vizualizace kompletního projektu, nebo pouze aktuálně otevřeného okna. Kód je napsán jako skriptový modul, tedy výše uvedený *module* a je tvořen souborem s příponou *.bas*. Tento modul musí být po otevření editoru skriptů pro určitou obrazovku importován. Postup převodu je nastíněn ve schématu na Obr. 15.



Obr. 15: Vývojový diagram skriptu pro převod grafických dat do formátu XML

Je schopen převést vlastnosti základních grafických objektů, složitějších objektů z knihovny *HMI symbol library* i vlastnosti grafických objektů tvořených na základě animačního objektu firmy Compas. Knihovna *HMI symbol library* se vyznačuje tím, že lze přistupovat pouze k obecným vlastnostem a oproti základní knihovně prvků v programu WinCC tak nelze zjistit podrobné informace, díky kterým by grafiku bylo možné dále převést. Nelze například zjistit definice základních grafických elementů, ze kterých je výsledný grafický objekt složen a jejich umístění na konkrétní pozici vykreslovací plochy objektu. Ve stejném duchu je zajištěn i převod animačního objektu

společnosti Compas. V tomto případě ale jsou dostupné specifikace zobrazených objektů, tudíž je možné objekty opět vykreslit s využitím externích definičních souborů a s využitím vlastností vyexportovaných mým skriptem správně umístit do výsledné vizualizace s přiřazenými akcemi.

Skript má několik částí, realizujících export. Výsledkem tohoto exportu jsou soubory ve formátu XML, obsahující definice grafických objektů, obsažených ve vizualizačních obrazovkách a jejich vlastností pomocí vnořených XML elementů. Hlavní dvě procedury se jmenují *ExportPDLS* a *ExportActivePDL*. První jmenovaná realizuje export celého projektu a druhá pouze aktivní, v grafickém editoru právě otevřené obrazovky. Po spuštění skriptu je zobrazena nabídka a uživatel si vybere, kterou z těchto dvou procedur využít. Tyto procedury poté volají další pomocné procedury, realizující převod.

První z nich je procedury *MakeExport*, realizující převod obrazovky jako celku, včetně uložení XML dokumentu do definovaného umístění.

Další důležitou funkcí je procedura *ReadHmiObject*. Tato procedura realizuje čtení objektů na vizualizační obrazovce, včetně jejích parametrů a zajišťuje uložení definičních dat těchto objektů do výsledného XML souboru. Každý grafický objekt je vložen jako element do výsledného XML dokumentu a nese název *HMIObject*. Jedná se o nejobsáhlejší proceduru, realizující nejdůležitější část převodu.

Následují pouze pomocné procedury pro přidání vlastnosti do XML elementu (procedura *AddAttribute*) a procedura *ReadPoints*, realizující čtení bodů pro objekty definované body, například lomené čáry.

Pro správnou funkci skriptu je nutné přidat do VB editoru referenci na knihovnu *HMI symbol library* a na Microsoft XML, v6.0. V případě použití dalších knihoven musí být přidány reference i na tyto knihovny, aby nedošlo k chybě při provádění skriptu. To zajistí, že skript, jehož výstupem je XML soubor, bude funkční. Převod funguje tak, že jsou do složky nastavené ve skriptu („*D:\export*“) přidány XML soubory s vyexportovanými objekty jednotlivých obrazovek. Soubory jsou přidány v případě exportu všech obrazovek. V případě exportu aktivní obrazovky je přidán pouze jeden soubor, popisující aktivní obrazovku. Hlavním elementem XML souboru je element s názvem *GRAPHIC*. Tento element obsahuje název původního souboru, tedy název konkrétní vizualizační obrazovky, její rozměry, barvu pozadí a rychlost obnovování. V tomto elementu jsou vnořeny XML elementy s názvem *HMIObject*, které popisují jednotlivé objekty na obrazovce. Každý element této úrovně obsahuje informaci o tom, o jaký typ objektu se jedná a jaký měl tento objekt název v původní vizualizaci. Do každého elementu *HMIObject* jsou vnořeny další XML elementy s názvem *Property*. V posledním jmenovaném jsou jako atributy uvedeny jednotlivé vlastnosti objektů a není tedy přesně daný jejich počet ani obsah – liší se podle typu objektu, pro který je tato vlastnost určena. Jedná se například o umístění objektu, jeho velikost, popis tvaru, navázání na komunikační tagy, počáteční a koncové body v případě přímek, vlastnosti fontu u textových objektů a další. Příklad části vyexportované definice objektu je patrná na následujícím příkladu.

```
<GRAPHIC UpdateCycle="4" BackColor="11974326" Height="851"
Width="1680" name="ZL_prehled.pdl">
    <HMIObject ObjectName="Polygon131" Type="HMIPolygon">
          <Property IsDynamicable="False" IsPublished="False"
          DynamicStateType="0" Dynamic="" DisplayName="Object Name"
          value="Polygon131" Name="ObjectName"/>
          <Property IsDynamicable="False" IsPublished="False"
          DynamicStateType="0" Dynamic="" DisplayName="Layer"
          value="0" Name="Layer"/>
```

Některým typům objektů bylo nutné věnovat zvláštní pozornost, protože některé vlastnosti nebylo jednoduše možné vyexportovat. Lze uvést například různé kruhové a eliptické výseče, u nichž bylo nutné kromě obecně exportovaných vlastností přidat navíc export bodů, sloužících pro napojení dalších objektů.

Dalším příkladem mohou být objekty typu zaškrtávacích polí. Objekt tohoto typu, anglicky nazývaný *checkbox*, má definován počet zaškrtávacích polí, ale navíc také jejich popis. Tento popis je pouze obyčejný text. Vyexportován může být ale pouze tak, že se v cyklu do objektu tohoto typu nastavuje index vybraného zaškrtávacího pole a tím se zpřístupní hodnota textového popisku. Objekt totiž neobsahuje seznam popisků jako celek. Podobné problémy byly se všemi výběrovými listy, například comboboxy, listboxy i prvky typu option group.

Nejsložitěji převáděným objekt je objekt typu *HMICustomizedObject*, což je v podstatě objekt, který seskupuje více objektů do skupiny s možností definice vstupních a výstupních parametrů. Protože je definice složena i z těchto objektů, je nutné, aby byla metoda zpracování volána rekurzivně. Protože objekt typu *HMICustomizedObject* v sobě další objekty tohoto typu obsahovat nemůže, je tím zajištěno, že se doba vykonání skriptu nijak zásadně nezpomalí.

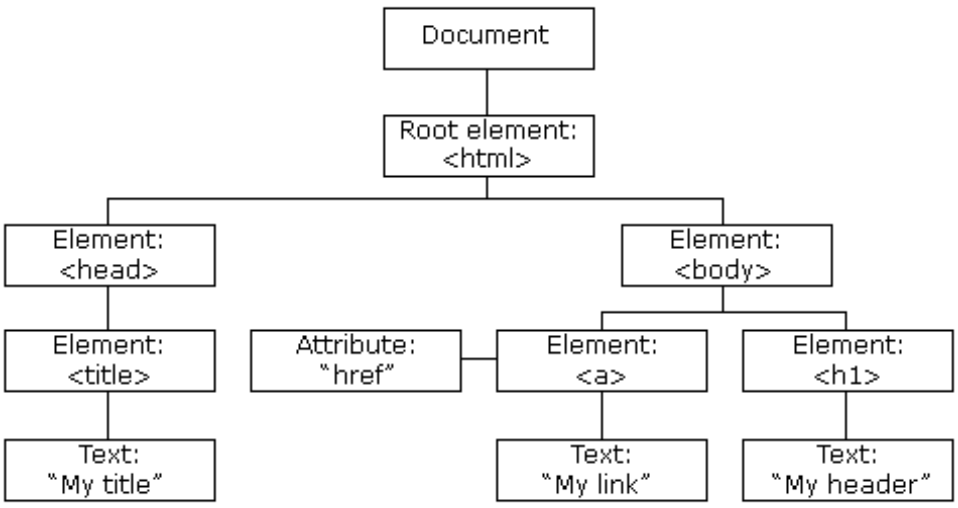
Nakonec je nutné zmínit, že skript byl vytvářen ve WinCC verze 7.3 a testován na testovacím projektu firmy Compas.

5 TECHNOLOGIE PRO VYKRESLOVÁNÍ GRAFICKÝCH PRVKŮ NA WEBU

V této kapitole budou podrobně rozebrány formáty souborů uvedené v zadání této práce, rozdíly mezi nimi a popis práce s těmito objekty. Všechny formáty souborů uvedené v následujících kapitolách jsou v případě využití na webu zahrnuty v takzvaném objektovém modelu dokumentu (DOM – Document Object Model). Protože je toto důležitá součást tvorby webové grafiky a tím pádem i vizualizace, bude v první podkapitole tento model rozebrán podrobněji.

5.1 Document object model (DOM)

Objektový model dokumentu reprezentuje všechny objekty v XML nebo HTML dokumentu (Obr. 16). Umožňuje, aby byly jednotlivé objekty na webové stránce měněny i po vykreslení ve webovém prohlížeči. V minulosti byla možnost přístupu k objektům HTML dokumentů pro různé prohlížeče rozdílná. Takový přístup se nazývá Intermediate DOM (přechodný DOM). K jednotlivým elementům se přistupovalo s využitím JavaScriptu pomocí specifického rozhraní každého prohlížeče. Tato vzájemná nekompatibilita byla pro další vývoj webových aplikací omezující, a proto byl mezinárodním konsorciem *W3C* (World Wide Web Consorcium), zabývajícím se vyvíjením webových standardů, vyvinuta specifikace DOM. Tato specifikace popisuje model nezávislý na jazyce a platformě. Umožňuje přístup k dokumentu jako ke stromové struktuře, což je shodný přístup jako v případě HTML a XML dokumentů. K prvkům stránky lze díky tomuto modelu přistupovat pomocí JavaScriptu[7].



Obr. 16: Příklad stromové struktury objektů jazyka HTML s využitím DOM modelu [21]

[7] Tato kapitola čerpá z [5][18][19]

5.2 VML

Vector Markup Language, zkráceně VML, je značkovací jazyk. Jedná se o praktickou aplikaci formátu XML 1.0 (Extensible Markup Language), pomocí které lze tvořit vektorové grafické objekty. Jedná se o jazyk vyvinutý firmou Microsoft v roce 1998 používaný pro zobrazování grafických prvků na webu. Díky tomu, že se jedná o značkovací jazyk, lze objekty vytvořené pomocí tohoto jazyka jednoduše upravovat a to jak manuálně, tak strojově pomocí skriptů. Díky této vlastnosti je možné vytvářet animované objekty, jejichž vzhled se mění podle předem nastavených parametrů a událostí. V závislosti na událostech je možné měnit v podstatě každou součást těchto grafických prvků a tím vytvářet webové hry, animace nebo vizualizaci průmyslových provozů. Další obrovskou předností jazyka VML je obrovská míra komprese a omezení přenosu dat při změnách pouze určité části celého grafického obrazce. Oproti bitmapovému obrazu, kde je nutné při jeho změně pracovat s celým obrazcem, v případě VML lze upravit pouze konkrétní část, nezávisle na velikosti celku[8].

VML, podobně jako jazyk HTML, využívaný pro tvorbu webových stránek, využívá kaskádových stylů pro definici šablony vykreslování. V případě VML se jedná konkrétně o verzi CSS 2. Díky tomuto propojení, mezi oběma technologiemi lze využít stejných postupů pro vykreslení textu v případě HTML a grafiky v případě VML. Hlavní rozdíl v případě zpracování dat z obou těchto formátů se týká generování dat důležitých pro vykreslení v průběhu zpracování konkrétní technologie. V případě HTML jsou generovány údaje o pozici a další informace o vykreslení textu jsou sbírány přímo z operačního systému. V případě VML je tento postup odlišný a v průběhu zpracování dat zadaných pomocí VML jsou generovány data jednak určující pozici, ale zároveň další data, například pro vykreslení křivek a dalších objektů, jako jsou bitmapové obrazce.

Díky tomu, že VML podporuje objektový model dokumentu (DOM) je možné jednotlivé prvky měnit. Pro úpravu a změnu statických objektů v animaci je nutné využití JavaScriptu. VML tedy samo o sobě animaci neumožňuje.

Podpora formátu VML byla v případě použití prohlížečů jiných výrobců, než je společnost Microsoft, v podstatě nulová. V současných a nových verzích webových prohlížečů již není a nebude tento grafický formát podporován. Přednost je dávána formátům SVG a HTML 5 CANVAS.

5.2.1 Struktura VML souboru

Jak již bylo uvedeno v předchozím textu, formát VML již není podporován žádným prohlížečem. Proto bude pouze stručně popsána struktura, pomocí které je grafika tvořena a to z toho důvodu, že se VML soubor může i nadále vyskytovat ve starších realizacích průmyslové vizualizace. Aby bylo možné zobrazit grafiku ve formátu VML uvnitř HTML stránky, je nutné, aby kód splňoval jisté náležitosti. Základní element HTML souboru

[8] Tato kapitola čerpá z [1][2]

(element *html*) musí být nahrazen specifickým, který v prohlížeči spustí vykreslování VML souborů. Stránka je tedy uvozena následujícím elementem:

```
<html xmlns:v="urn:schemas-microsoft-com:vml">
```

Protože se názvy VML elementů dost často zapisují s prefixem *v:*, je nutné, aby o tom byl vyrozuměn i prohlížeč. Proto se do hlavičky HTML souboru ohraničené elementem *head* musí umístit následující zápis:

```
<style> v\:* { behavior: url(#default#VML); }</style >
```

Jednotlivé grafické objekty jsou poté zapisovány do těla HTML dokumentu. Pozicování může být relativní nebo absolutní a umístění vztažného bodu je závislé na natočení objektu. VML má sadu předdefinovaných grafických elementů. Pro seskupování objektů do skupin slouží element *group*.

5.2.1.1 Základní geometrické tvary

- Kruhová výseč – Tvořena elementem *arc*. Je tvořená výsečí elementu *oval*. Je určena startovacím a koncovým úhlem výseče.
- Kvadratická beziérova křivka – Tvořena elementem *curve*. Je definována pomocí startovního a koncového bodu a parametry control1 a control2, definujícími prohnutí křivky
- Úsečka – tvořena elementem *line*. Určena je počátečním a koncovým bodem
- Elipsa – tvořena elementem *oval*. Je definována pomocí rozměrů – šířkou a výškou.
- Lomená čára – tvořena elementem *polyline*. Je definována pomocí bodů čáry.
- Obdélník – tvořen elementem *rect*. Je definován šířkou a výškou.
- Obdélník se zaoblenými rohy – tvořen elementem *roundrect*. Je definován šířkou, výškou a velikostí radiusu zaoblení.

5.2.1.2 Obecné křivky

Další skupinou je kreslení obecných křivek. V tomto případě se využívá elementu *shape* pro určení křivky a *shapetype* pro parametry křivky. Element *shapetype* lze využít u více *shape* elementů a definuje například barvy a další parametry křivek.

5.2.1.3 Ostatní elementy

Posledním typem základních elementů jsou obrázky, přidávané pomocí elementu *image*. Ostatní elementy jsou víceméně podružné a umožňují například psaní vzorců, tvoření textů a textových polí.

5.2.1.4 Příklad zápisu elementů

Psaní jednotlivých elementů v konstrukci popsané výše může být provedeno například následujícím způsobem:

```
<v:shapetype id="master" fillcolor="red" path="M 0,0 L 100,100,200,0 X
E"></v:shapetype>
<v:shape type="#master" style="width:50px;  height:50px"/>
<v:rect style="width:100;height:100 fillcolor="blue"
strokecolor="red"></v:rect>
<v:oval style="width:150;height:100">
<v:fill color2="#00FF00" type="gradient"></v:fill>
</v:oval>
```

5.3 SVG

SVG (Scalable Vector Graphics) je formát pro popis dvourozměrné vektorové grafiky. Poprvé byl tento formát uvolněn v roce 2001. Vyvinut byl konsorciem W3C, zmiňovaným už u objektového modelu dokumentu. Je založen, stejně jako v případě VML, na formátu XML a podporuje také kaskádové styly CSS. Díky těmto vlastnostem je jednoduché jej vytvořit pomocí základních nástrojů, jako je textový editor a webový prohlížeč[9].

SVG formát je možné použít jako samostatné soubory s příponou *.svg*, nebo v případě použití ve webové stránce vložit přímo do těla HTML stránky. Díky tomu, že je SVG také součástí DOM, lze u něj opět, jako v případě VML, upravovat jednotlivé prvky pomocí Javascriptu. Animování SVG souborů je možné pomocí JavaScriptu, nebo na rozdíl od VML přímo pomocí elementu specifikujícího animaci. Formát SVG má oproti VML velkou výhodu v podpoře vykreslování všemi prohlížeči. Formát VML byl podporován prakticky pouze prohlížečem Microsoft Internet Explorer.

5.3.1 Struktura SVG souboru

Na začátku a na konci SVG dokumentu musí být uveden tzv. fragment SVG. Jedná se o kořenový párový element dokumentu, obalující všechna obrazová data SVG dokumentu. Tyto elementy mohou být zanořeny, ale je nutné, aby byl tento element vždy kořenovým elementem. Tento element má některé důležité parametry. Vždy musí být uveden implicitní jmenný prostor, zapsaný v atributu *xmlns*. Díky tomuto atributu webový prohlížeč rozpozná, že se jedná o grafický objekt a ne o XML soubor. Dalšími parametry jsou rozměry kreslícího plátna. Na úvod je dobré uvést, že souřadnice pro tvorbu grafických objektů mají nulovou hodnotu v levém horním rohu. Tato vlastnost platí ale pouze do doby, kdy s objektem nebude rotováno. Poté zůstává tento bod na stejném místě a objekt rotuje kolem něj. Příklad kořenového elementu je patrný zde:

```
<svg xmlns ="http://www.w3.org/2000/svg" width = "100" height =
"200">
</svg>
```

SVG soubor obsahuje 3 základní typy grafických objektů – základní geometrické tvary (basic shapes), cesty (path) a texty (texts). Grafické objekty lze seskupovat do skupin

[9] Tato kapitola čerpá z [3][4]

pomocí elementů *g*, měnit jejich styly a transformovat jejich vzhled a rozměry. Tyto grafické objekty mají množství parametrů, které lze vyčíst ve specifikaci formátu SVG a z nichž některé jsou pro tyto tři typy objektů společné. Zde bude uveden alespoň stručný popis těchto základních objektů. Elementy SVG dokumentu jsou vždy párové.

5.3.1.1 Základní geometrické tvary

- Úsečka – První z uvedených grafických primitiv je úsečka. Element pro tvorbu úsečky má název *line* a povinnými parametry jsou pouze souřadnice počátečních a koncových bodů.
- Obdélník – Dalším ze základních tvarů je obdélník. Element s názvem *rect* má povinné parametry výšku, šířku a souřadnice levého horního rohu.
- Kružnice – Kružnice má 3 povinné parametry – souřadnice středu a poloměr r. Zapisuje se pomocí elementu *circle.*
- Elipsa – Elipsa má 4 povinné parametry – souřadnice středu a poloměry os r_x a r_y. Zapisuje se pomocí elementu *ellipse*.
- Lomená čára – Tento grafický objekt obsahuje libovolný počet na sebe navazujících úseček. Zapisuje se do elementu polyline a má jediný povinný parametr points, který obsahuje neomezený počet dvojic bodů, určujících počáteční a konečné body úseček. Tato lomenná čára nemusí být tvořena jako uzavřená křivka.
- Uzavřený polygon – posledním ze základních geometrických tvarů je uzavřený polygon. Je zapisován do elementu polygon a jediným povinným parametrem je stejně jako u polyline parametr *points*. Na rozdíl od lomené čáry se do tohoto parametru nemusí uvádět koncový bod.

5.3.1.2 Cesty (path)

Pro jednoduchou grafiku jsou základní geometrické tvary dostačující, ale v případě, že je nutné vytvořit složitější grafiku, je nutné použít elementu *path*. Pro vykreslení nějakého tvaru je nutné zadat parametr *d*, do kterého se vkládají příkazy pro vykreslení objektu. Příkazy se zadávají písmeny. Pokud je příkaz zadán velkým písmenem, jsou souřadnice brány jako absolutní, pokud je zapsán malým písmenem, jedná se o relativní souřadnice. Příkazy slouží k určení, jakým způsobem (po jaké křivce) mají být zadané body propojeny. Využívá se zde podobného principu jako při tisku na plotteru, kdy je zadán jeden příkaz a k němu potřebné souřadnice. Lze tak vytvářet křivky s body propojenými úsečkami, beziérovými křivkami nebo jen přesouvat kreslící bod na jiné souřadnice. Příkazů je celkem deset a každý z nich vyžaduje jiný počet a typ souřadnic. Příkazy a parametry lze nalézt ve specifikaci SVG formátu.

5.3.1.3 Texty

V případě SVG je text definován párovým elementem *text*. Jelikož je text v SVG brán jako plnohodnotný uzavřený objekt, využívá se pro nastavení parametrů shodných

s předchozími dvěma skupinami grafických objektů, jakými jsou například barva výplně nebo tloušťka obrysů, stejných postupů. To velice zjednodušuje případné zpracování grafických objektů ve skriptech. Pro text lze nastavit velké množství parametrů, jako je font, počáteční souřadnice a další.

5.3.1.4 Příklad zápisu elementů

Formát SVG je tvořen podobně, jako formát VML. Do párového elementu *svg* jsou vkládány elementy jednotlivých grafických obrazců. Definice elementů je zapisována buď párovým elementem, nebo lze využít zkrácený zápis, který však není podporován všemi prohlížeči. Může vypadat například takto:

```
<g transform="translate(50,0) rotate(30)">
     <rect stroke="red" stroke-linejoin="miter" x="200" y=20"
     width="100" height="60"/>
     <text fill="red" x="15" y="50" font-family="SanSerif" font-
     size="50">testovaci text</text>
</g>
```

5.4 HTML 5 CANVAS

HTML 5 CANVAS, jak už název napovídá, je založen na poslední verzi technologie HTML (HyperText Markup Language). Poslední verze tohoto jazyka značně rozšířila nástroje, které jsou pro tvorbu moderních webových stránek důležité. Jedním z rozšíření bylo přidání elementu s názvem *canvas,* který vytváří obdélníkovou pracovní plochu pro tvorbu bitmapových obrazců. Tyto obrazce jsou poté tvořeny pomocí jazyka JavaScript s použitím rozhraní Canvas API. Pomocí tohoto rozhraní je při každém překreslení okna celý bitmapový obrazec kompletně překreslen. Úkolem programátora je vytvořit takový kód, který vytvoří vzhled obrazce ještě před jeho vykreslením v prohlížeči tak, aby byly vykresleny jen údaje, které chceme zobrazit[10].

Oproti SVG a VML, které fungují tak, že jsou řízeny údaji uvedenými přímo v kódu obrazců, je objekt vytvořený pomocí Canvas zpracováván kompletně při opětovném překreslení. Výhoda tohoto principu je v široké možnosti ovlivnění tohoto vykreslení. Lze říci, že SVG a VML jsou dále od těchto nízkoúrovňových operací, čímž se zjednodušuje jejich kód, ale omezují možnosti. Velkou nevýhodou tohoto přístupu je fakt, že jednotlivé elementy obrazu nejsou přístupny skrze objektový model dokumentu. Jediným, co je přístupné skrze objektový model je celé pracovní plátno uzavřené v elementu *canvas*. HTML 5 Canvas umožňuje pracovat v různých kontextech. Nejčastějším je 2D kontext. Existují i další, např. 3D kontext, které umožňují využití rozhraní OpenGL pro vykreslení grafických dat.

[10] Tato kapitola čerpá z [5][6]

5.4.1 Struktura grafických objektů v HTML 5 CANVAS

HTML 5 Canvas umožňuje kreslení křivek, renderování textu a zobrazování bitmapových obrazů přímo v definovaném místě pracovní plochy vymezené v atributech párového elementu *canvas*. Díky tomu, že je k vykreslení využíváno JavaScriptu, je možné tvořit velké množství animací, závislých například na čase, na stisknutí určitého tlačítka na klávesnici, nebo na dalších vlivech, které lze pomocí JavaScriptu zpracovat, naopak nevýhodou je složitost zápisu jednotlivých elementů oproti SVG i VML.

Element *canvas* umístěný v těle dokumentu definuje plochu následujícím způsobem:

```
<canvas id="canvasTest" width="400" height="400">
Prohlížeč nepodporuje HTML 5 Canvas
</canvas>
```

Jak je patrné, dovnitř elementu Canvas je možné vložit text, který se zobrazí v případě, že prohlížeč nepodporuje grafiku pomocí HTML 5 Canvas. Dalším krokem je využití DOM modelu dokumentu. Pomocí JavaScriptu je nutné nalézt element canvas. Tato část se již nebude zapisovat do těla HTML dokumentu (element *body*), ale do hlavičky dokumentu (element *head*), konkrétně dovnitř elementu vymezujícího skripty – *script*.

var canvasElm = document.getElementById("canvasTest");

Za tuto část kódu je nutné vložit navíc test, zda element existuje a zda jsou dostupné metody a vlastnosti pro práci s grafickými objekty. Následuje nastavení kontextu, který určuje rozhraní pro práci s grafickými objekty

```
var context = canvasElm.getContext("2d");
```

Získáním kontextu byl získán prostředek ke tvoření grafiky. Nyní již následuje vkládání grafických elementů. Grafické prvky, které lze vložit ve 2D kontextu, jsou podobné předchozím dvěma jazykům. Pro kreslení těchto prvků se však využívá zápisu kódu v JavaScriptu, takže se od dříve uvedených VML a SVG liší zápisem. Pro ukázku bude uvedena tvorba čáry z bodu (0,20), navazující na předchozí příklady.

```
context.moveTo(0,20);
context.lineTo(100,100);
```

Lze tvořit tyto prvky:

- Čára – viz Předchozí příklad. Pro čáru lze nastavit mnoho parametrů, včetně zakončení čar při jejich spojení.
- Obdélník – lze tvořit klasické obdélníky, bez výplně, se zaoblenými rohy a obdélníkové výřezy.
- Kruhové výseče – lze tvořit kruhy, nebo části kruhů. Je zde možnost také vytvořit kruhovou výseč mezi dvěma tečnami.
- Křivky – lze tvořit obecné, Bézierovy, nebo kvadratické Bézierovy křivky.
- Text – stejně jako u předchozích jazyků, lze vytvářet textové řetězce. Lze nastavit font, velikost a další grafické prvky textu.

- Obrazová data – lze vložit obrázek s upravenou velikostí, nebo pouze výřez obrázku.
- Ostatní prvky – Oproti VML a SVG umožňuje Canvas manipulovat s jednotlivými pixely, nebo měnit průhlednost vytvořených objektů.

5.5 Shrnutí kapitoly 5

V páté kapitole jsou uvedeny grafické formáty, které byly, v současné době jsou, nebo v budoucnu mohou být použity pro tvorbu vizualizace průmyslové výroby ve webových prohlížečích. Každý z těchto tří formátů má svá specifika použití. V případě VML se jedná spíše o minulost, neboť podpora tohoto formátu skončila a v současné době bývá ve velké míře nahrazován formátem SVG, což se vzhledem k podobnosti obou těchto formátů přímo nabízí. V případě HTML 5 Canvas lze předpokládat, že použití v průmyslových aplikacích bude z hlediska přístupu k vykreslování a tvorbě spíše omezené. Pro budoucí využití v této práci jsem tedy zvolil právě formát SVG, do něhož budou převáděna data z XML souborů, generovaných skriptem, popsaným v kapitole 4. Vybral jsem jej z následujících důvodů. Kód je oproti Technologii Canvas přehledný a je v něm jasně dáno, co patří ke konkrétnímu grafickému prvku. SVG podporuje objektový model dokumentu, takže je možné přistupovat k jednotlivým grafickým komponentům pomocí skriptů, což v případě HTML 5 Canvas není možné. Posledním důvodem bylo to, že SVG je vektorový formát, což je dle mého názoru výhoda do budoucna, kdy se vizualizace bude upravovat pro různá zařízení s různou velikostí displeje.

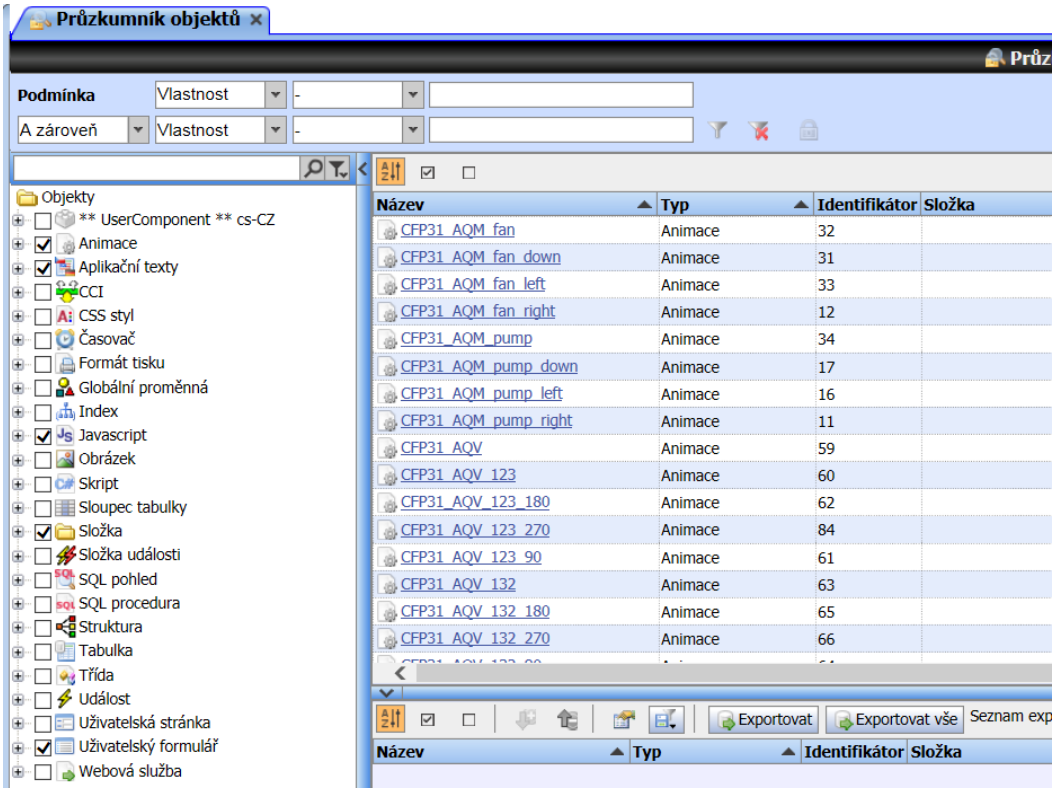
6 PŘEVOD DAT Z FORMÁTU XML DO SYSTÉMU COMES

Převod je realizován pomocí programu v programovacím jazyce C#. Data vyexportovaná ze systému WinCC jsou načtena do dále popsaného programu, který slouží k převedení grafických objektů a jejich vlastností. Je určen především pro převod grafiky a částečně také funkčnosti jednotlivých grafických prvků. Kompletní funkčnost zaručena není a to hlavně z důvodu, že prvkům, které jsou převáděny přímo do jazyka HTML, není v systému COMES možné nastavit stejné parametry, jako v případě systému WinCC. Stoprocentní funkčnost je zaručena pouze u animačního objektu firmy Compas, kde jsou všechna data popisující objekt přístupná a tyto objekty mají jednodušší strukturu vlastností.

6.1 Exportní balíček

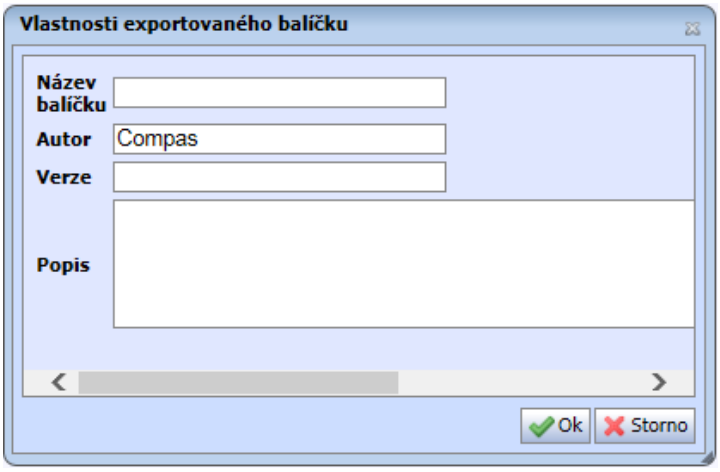
Aby čtenář mohl pochopit, jakým způsobem pracuje program, převádějící grafické objekty pro systém COMES, je nutné vysvětlit, jakým způsobem budou tato data do systému nahrávána. Do systému COMES se definiční data uživatelské aplikace vkládají pomocí tzv. exportního balíčku. Tento balíček lze vytvořit pomocí systému COMES a jeho smyslem je přenášení definičních souborů uživatelské aplikace z jedné instalace na instalaci jinou – lze tedy využít již jednou napsaných zdrojových kódů, nebo tímto způsobem provést jejich zálohu. Export dat je možné provést v modulu COMES Modeller v záložce průzkumník objektů (Obr. 17).

V levé části obrazovky se nachází panel, kde si uživatel může vybrat, jaké typy souborů a jejich vlastností požaduje vyexportovat. V horní části obrazovky lze nastavit doplňkové filtry a aplikovat filtry s vybranými typy souborů. Vpravo od panelu pro výběr typu objektů se nachází obrazovka, kde se po nastavení a aplikaci filtrů zobrazí soubory odpovídající požadavkům. Pod tímto oknem se nachází poslední část průzkumníku objektů. V této části jsou vypsány konkrétní objekty, určené k exportu pomocí exportního balíčku. Nachází se zde panel s ovládacími prvky s možností vyexportování vybraných souborů, vyexportování všech souborů, nebo uložením seznamu souborů určených k exportu. Po stisknutí tlačítka *Exportovat,* nebo *Exportovat vše* je zobrazen formulář pro nastavení vlastností exportovaného balíčku. Lze nastavit název balíčku, autora, verzi a popis balíčku. Hodnoty nastavených vlastností se projeví při načtení balíčku při opětovném použití v systému COMES.



Obr. 17: Okno průzkumníku objektů, pomocí kterého je možné vytvořit exportní balíček ze souborů systému COMES.

V mé práci je struktura exportního balíčku využita, ale na rozdíl od běžného použití není soubor s grafickými daty vytvářen pomocí systému COMES, nýbrž pomocí mnou vytvořeného programu, který realizuje převod grafických dat. Mým programem vytvořený soubor vypadá ve výsledné podobě stejně, jako soubor exportovaný ze systému COMES. Vlastnosti tohoto balíčku při exportu ze systému COMES lze nastavit ve formuláři na Obr. 18. Tato data jsou poté uložena do souboru *package.info* a slouží pro identifikaci balíčku při jeho načtení do systému COMES. Stejný soubor obsahuje i balíček, exportovaný mým programem.



Obr. 18: Formulář pro nastavení vlastností exportního balíčku.

6.1.1 Struktura exportního balíčku

Exportní balíček jako takový je v podstatě pouze archiv ve formátu *zip*, který v sobě obsahuje pevně danou strukturu souborů a složek. Tato struktura se liší podle toho, jaké typy souborů jsou exportovány. Může obsahovat složku s Javaskriptovými soubory, složku s obrázky a mnoho dalších. V mé aplikaci bude ale jeho struktura omezena pouze na soubory definující uživatelské formuláře, animační objekty a volitelně složky. Exportní balíček pro tuto konkrétní aplikaci zahrnuje následující soubory:

- XML soubor s vlastnostmi exportního balíčku
- XML soubor s definicí vlastností animačních objektů
- XML soubor s definicí vlastností uživatelských formulářů
- XML soubor s definicí vlastností složek
- Složku se soubory referenčních vlastnosti všech typů souborů
- XML soubor referenčních vlastností animačních objektů
- XML soubor referenčních vlastností uživatelských formulářů
- XML soubor referenčních vlastností složek

6.1.1.1 Struktura souboru s vlastnostmi exportního balíčku

Definiční soubor vlastností exportního balíčku obsahuje informace zadané ve formuláři na Obr. 18. jedná se o XML soubor s názvem *package.info*. Díky informacím z tohoto souboru je možné zjistit datum vytvoření, verzi balíčku a další informace uvedené v popisu. Přesná struktura je definována následujícím způsobem:

```
<?xml version="1.0" encoding="utf-8"?>
<package>
  <Name>testPackage</Name>
  <Description> </Description>
  <Author>Compas</Author>
  <PackageVersion>3.11.000</PackageVersion>
  <CreateTime>2016-03-30T22:45:17.7521416+02:00</CreateTime>
  <ModuleVersion>3.11.000</ModuleVersion>
  <ModuleType>Modeller</ModuleType>
</package>
```

Parametr ModuleType určuje, ze kterého modulu byl balíček exportován. V mé práci je nastaven stejně, jako v případě exportu ze systému COMES. Data uvedená v tomto souboru jsou shodná s daty, nastavenými při exportu ze systému COMES pomocí formuláře z Obr. 18.

6.1.1.2 Struktura souboru s definičními vlastnostmi animačních objektů

Soubor s názvem *Animation.xml* obsahuje definici aplikačních vlastností všech animačních objektů, které jsou v balíčku obsaženy. Vlastnosti konkrétního animačního objektu jsou ohraničeny párovým elementem *Animation*. Všechny tyto elementy jsou součástí párového elementu *StaticProperties*. Jsou definovány následující vlastnosti:

- Id – identifikační číslo souboru v systému COMES z doby exportu
- Name – jméno animačního objektu – pod tímto názvem lze objekt identifikovat v editoru skriptů
- Version – verze animačního objektu
- ModifyTime – čas poslední změny
- CreateTime – čas vzniku souboru
- Description – popis v českém a anglickém jazyce
- Code – kód vzhledu animačního objektu
- CsCode – výkonný kód objektu
- Parameters – část kódu definující komunikační proměnné mezi animačním objektem a zbytkem systému COMES.

Ukázka definice souboru s jedním animačním objektem a jedním vstupem následuje na dalších řádcích:

```
<?xml version="1.0" encoding="utf-8"?>
<StaticProperties>
  <Animation>
    <Id>197</Id>
    <Name>CFP31_GR_DISPLAY_MENU</Name>
    <Version>1.00.000</Version>
    <ModifyTime>635949747038871119</ModifyTime>
    <CreateTime>635949747038871119</CreateTime>
    <Description>
      <Value Key="1029">
      </Value>
      <Value Key="1033">
      </Value>
    </Description>
    <Code>&lt;Animation name="Animation"&gt;
  &lt;svg name="svgObject" runat="server" width="97" height="20"&gt;
      &lt;rect name="Objekty" runat="server" visibility="visible"
width="97" height="20" fill="none"&gt; &lt;/rect&gt;
  &lt;/svg&gt;
&lt;/Animation&gt;</Code>
    <CsCode>
</CsCode>
    <Name>CFP31_GR_DISPLAY_MENU</Name>
    <Parameters>&lt;?xml version="1.0" encoding="utf - 16"?&gt;
&lt;Parameters&gt;
&lt;Input Name ="CollectValue" Description = "" Output = "0"
DataType ="Double" /&gt;
&lt;/Parameters&gt;</Parameters>
  </Animation>
```

6.1.1.3 Struktura souboru s definičními vlastnostmi uživatelských formulářů

Podobně jako v případě animačních objektů soubor s názvem *UserForm.xml* obsahuje definici aplikačních vlastností všech uživatelských formulářů, které jsou v balíčku obsaženy. Vlastnosti konkrétního formuláře jsou ohraničeny párovým elementem *UserForm*. Všechny tyto elementy jsou součástí párového elementu s názvem *StaticProperties*. Jsou definovány následující vlastnosti:

- Id – identifikační číslo souboru v systému COMES z doby exportu
- Name – jméno animačního objektu – pod tímto názvem lze objekt identifikovat v editoru skriptů
- Version – verze animačního objektu
- ModifyTime – čas poslední změny
- CreateTime – čas vzniku souboru
- Description – popis v českém a anglickém jazyce
- AddFilterConditionsToResult – povoluje aplikaci filtrů na data formuláře
- AuthorizeUser – zajišťuje aplikaci formuláře
- Code – kód definující vzhled formuláře
- DataInUtcTime – atribut určující, zda bude čas zapisován v UTC
- DisplayName – zobrazovaná název
- Filter – filtry definující stránku
- HelpUrl – URL adresa nápovědy ke stránce
- HtmlVersion – určuje verzi stránky – hodnota je definována pomocí číselného kódu, který definuje položku ve výběrovém menu okna pro nastavení vlastností uživatelských formulářů
- PageSize – definuje velikost stránky
- RefreshOnDefinitionChange – určuje, zda bude stránka automaticky obnovena při definici stránky
- SaveInSaveScript – určuje, zda budou data ukládána pomocí C# skriptu po stisku tlačítka pro uložení
- UseAjax – atribut určující využití technologie Ajax ve formuláři – v této práci nevyužit, proto je nastaven na false
- VisibleButtons – Atribut definující tlačítka zobrazená na horní liště – definice je tvořena pomocí číselného kódu – exportovací program tlačítka nenastavuje a to z toho důvodu, že není jasné, jak bude stránka použita a jaké parametry jsou zde důležité – nastavení tlačítek musí být v případě potřeby provedeno manuálně ve vlastnostech formuláře v okně editoru skriptů

Definice uživatelského formuláře se od definice animačního objektu liší použitím jiných parametrů. Aby bylo možné nastavit všechny parametry formulářů, je nutné vytvořit definiční soubor pro rozšířené vlastnosti uživatelského formuláře. Tyto vlastnosti nejsou v mé práci využity, a proto v této práci tento soubor vytvořen nebude. Ukázka kódu tohoto souboru je z důvodu délky přiložena pouze na DVD v exportním balíčku.

6.1.1.4 Struktura souboru s definičními vlastnostmi složek

Definiční soubor vlastností složek je oproti předchozím dvěma souborům jednodušší, protože složka slouží pouze pro zpřehlednění tvorby aplikace. Jmenuje se *Folder.xml* a Definice obsahuje následující parametry:

- Id – identifikační číslo složky v systému COMES z doby exportu
- Name – jméno animačního objektu – pod tímto názvem lze objekt identifikovat v editoru skriptů
- Version – verze animačního objektu
- ModifyTime – čas poslední změny
- CreateTime – čas vzniku souboru
- Description – popis složky v českém a anglickém jazyce

Soubory definičních vlastností definují pouze konkrétní soubory jako takové. Ostatní parametry jsou uvedeny v definičních souborech referenčních vlastností, popsané v následujících kapitolách.

6.1.1.5 Struktura souborů s referenčními vlastnostmi

Soubory referenčních vlastností jsou si u všech typů objektů velice podobné, proto budou popsány pro všechny použité objekty souhrnně v jedné kapitole. Názvy souborů jsou shodné, jako v případě souborů s definičními vlastnostmi, pouze se nachází ve složce *References*. Referenční soubory obsahují propojení s ostatními soubory uživatelské aplikace. Kostra definice se u souborů neliší. Definice všech souborů je podobně jako u souborů definičních vlastností uvozena elementem *ReferencedProperties*. Každý objekt je uvozen elementem podle typu souboru. V případě animací se jedná o element *Animation*, v případě uživatelských formulářů *UserForm* a v případě složek *Folder*. Soubor *package.info* referenční část neobsahuje. Definice kostry obsahuje následující elementy:

- Id – identifikační číslo složky v systému COMES z doby exportu – pomocí hodnoty uvedené v tomto elementu je definice objektu spojena
- Rights – element, definující, které skupiny uživatelů mohou přistupovat ke konkrétnímu souboru. Pokud má být k objektu povolen přístup určité skupině, musí být v tomto elementu uveden pomocí subelementu *value*.
- FkColumns – element, který obsahuje rozdílné subelementy podle typu souborů.

Na následujících řádcích bude popsán vnitřní obsah elementu *FkColumns* pro jednotlivé typy souborů. V případě Animace element může obsahovat odkaz na složku, ve které má být tento soubor zahrnut. Tento odkaz je tvořen subelementem *FolderId* s atributem *Name*, udávajícím jméno složky. Text obsažený v elementu pak definuje číslo Id složky, do které má být objekt zahrnut. V následujícím příkladu je patrný zápis.

```
<FkColumns>
    <FolderId Name="Test">2</FolderId>
</FkColumns>
```

V případě definice pro objekt složky se zápis téměř neliší. Protože je možné složky zanořovat, je zde opět pouze definice popisující nadřazenou složku. V tomto případě je definice uvozena elementem *ParentId*:

```
<FkColumns>
    <ParentId Name="Test">2</ParentId>
</FkColumns>
```

Uživatelský formulář má ze tří použitých typů objektů nejsložitější definici. Je to způsobeno tím, že principiálně může pro svou funkci potřebovat nejvíce externích souborů. Opět obsahuje subelement definující příslušnost ke konkrétní složce, stejně jako v případě animace nazvaný *FolderId*, který je zanořen v elementu *FkColumns*. Navíc ale obsahuje několik dalších subelementů. Prvním z nich je *MenuId*, obsahující umístění v menu. Tento element vždy obsahuje vnořený subelement *Null*. Dalším elementem, zanořeným do elementu *FkColumns* je element *PrintFormatId*. Tento element definuje velikost stránky při tisku a obsahuje vlastnost *Name*, do které je pevně nastavena hodnota "A4L" a do elementu vložen text „*1*" označující číslo tohoto výběru v roletovém menu v nastavení formuláře. Dále jsou zde subelementy *SaveScriptID* pro odkaz na skript volaný po uložení formuláře, *SqlProcedureId* s odkazem na Sql proceduru, který získává data z databáze a *ViewScriptId* pro odkaz na skript, který má být volán při zobrazení formuláře. Všechny tyto elementy obsahují subelement *Null* a nejsou tedy využity, jak lze vidět na následujícím příkladu elementu *FkColumns* pro uživatelský formulář:

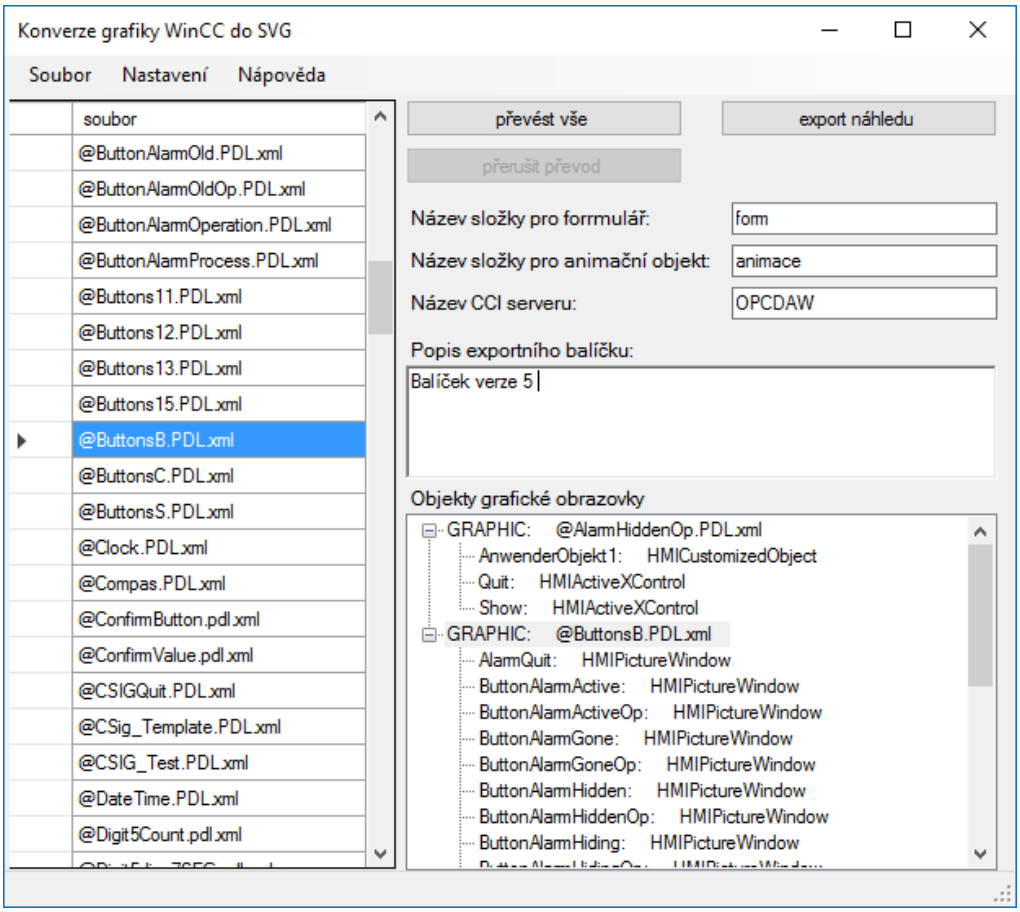
```
<FkColumns>
      <FolderId>
        <Null />
      </FolderId>
      <MenuId>
        <Null />
      </MenuId>
      <PrintFormatId Name="A4L">1</PrintFormatId>
      <SaveScriptId>
        <Null />
      </SaveScriptId>
      <SqlProcedureId>
        <Null />
      </SqlProcedureId>
      <ViewScriptId>
        <Null />
      </ViewScriptId>
</FkColumns>
```

6.2 Ovládání programu

Program je vytvořen tak, aby nemusel být instalován, protože je předpoklad, že bude často přenášen z jednoho PC na další. Instalace by zbytečně zdržovala pracovníky, vytvářející uživatelskou aplikaci v systému COMES.

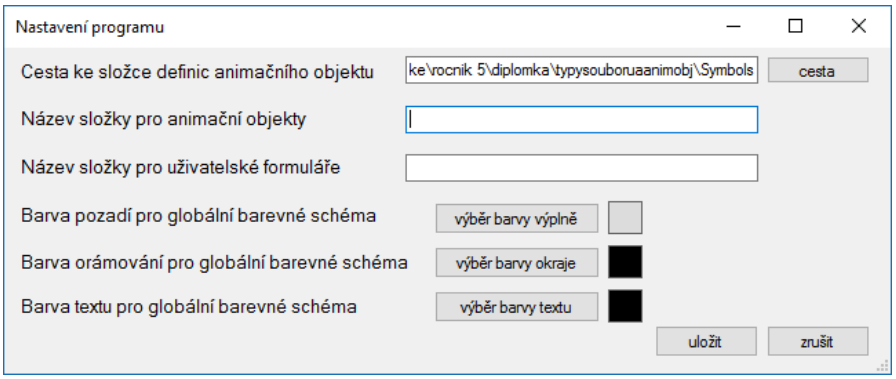
Po spuštění programu je nutné načíst složku, obsahující XML soubory vyexportované pomocí skriptu popsaného v kapitole 4. To lze učinit volbou v menu *soubor* výběrem položky *načíst složku souborů k převodu*. Pokud složka obsahuje XML soubory, zobrazí se ve výpisu v levé části obrazovky. Původně byla zvažována možnost vybírání souborů jednotlivě. Nakonec byla ale zamítnuta ze dvou následujících důvodů. Prvním důvodem byl poznatek, že systém COMES obsahuje velice propracovanou obrazovku pro nahrávání, kde je možné jednotlivé soubory vybírat, případně i měnit jejich zdrojový kód a vlastnosti. Je tedy mnohem přehlednější vybírat konkrétní soubory přímo v systému COMES. Druhým důvodem pro tento postup bylo zachování přehlednosti převedených dat. V případě, že by byly převáděny pouze jednotlivé soubory, mohlo by docházet ke tvorbě více exportních balíčků, kde by každý obsahoval část převedených grafických obrazovek, ale v každém balíčku by se nacházela data jiného stáří a tedy ne vždy aktuální. V případě mého programu tedy bude každý exportní balíček obsahovat data o všech převedených obrazovkách z doby exportu souborů ze systému WinCC. Tyto soubory musí být uloženy ve složce, ze které můj program čte převodní soubory. Je tím zajištěna možnost, že by bylo nutné vrátit se ke starší verzi grafických objektů.

Grafické rozhraní je vytvořeno tak, aby bylo co možná nejpřehlednější a aby jeho ovládání bylo rychlé. Všechna důležitá data jsou obsažena v hlavním okně programu, které je vyobrazeno na Obr. 19.



Obr. 19: Hlavní okno programu pro převod grafických objektů

Po stisknutí tlačítka pro převod je zahájeno převádění souborů. Aby bylo možné převádět i animační objekty firmy Compas, musí být v nastavení programu nastavena cesta ke složce s definičními soubory těchto objektů (Obr. 20). Zároveň je zde nastavení barev pro globální barevné schéma z programu WinCC, které bude vysvětleno v kapitole 7. Všechny položky z formuláře nastavení programu se společně s textem nápovědy ukládají do XML souboru *config.xml*, aby mohly být po startu opět načteny.



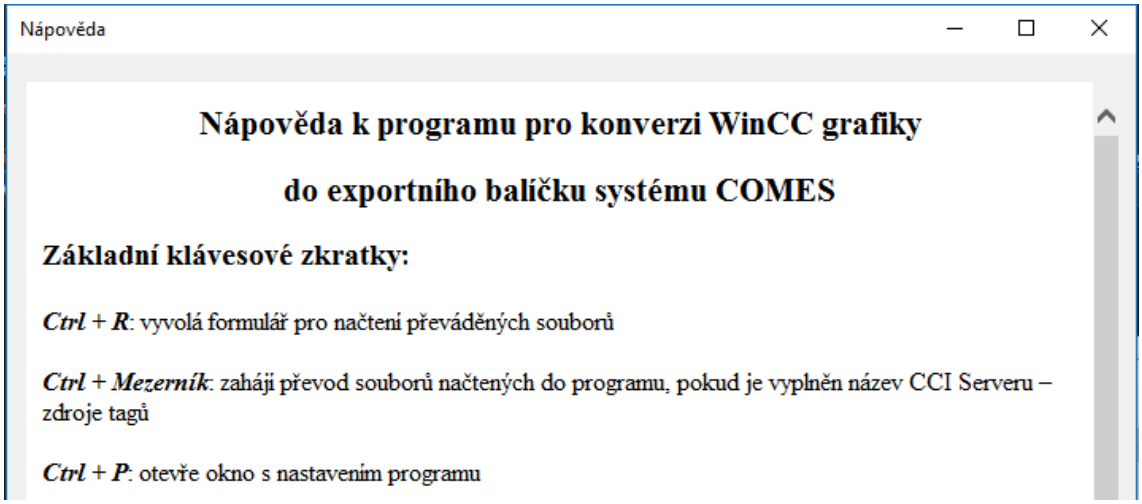
Obr. 20: Okno Nastavení programu

Pokud konfigurační soubor neexistuje, program si nastaví výchozí hodnoty, tedy prázdná textová pole, černou barvu do globálního barevného schéma a základní text nápovědy. Po stisknutí tlačítka *uložit* je konfigurační soubor vytvořen, nebo přepsán na aktuální stav.

Převod je prováděn postupně v cyklu a je pro něj vytvořena asynchronní paralelní úloha (*Task*) – okno programu je tedy funkční i při převodu. Postupně jsou převedeny jednotlivé obrazovky, přičemž převod je možné zrušit tlačítkem s názvem *Přerušit převod*.

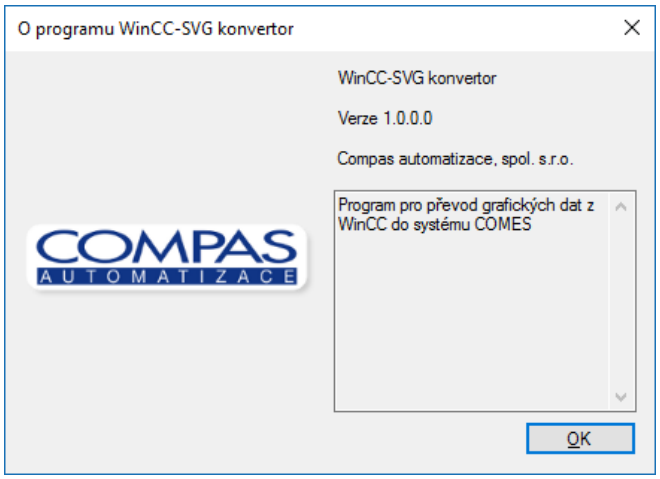
V levém dolním rohu je pomocí tzv. progress baru zobrazen stav převodu. Po dokončení je zobrazen popis „Převod dokončen“. V případě chyby je zobrazen nápis „Převod byl přerušen“. Uživatelské rozhraní umožňuje zobrazit objekty, obsažené v jednotlivých exportovaných souborech. Po dvojím kliknutí na název souboru se v pravém dolním okně zobrazí název souboru a po rozbalení nabídky u tohoto názvu také objekty a jejich datové typy z programu WinCC. Tento výpis lze exportovat tlačítkem s názvem *export náhledu* do CSV souboru a může sloužit jako kontrolní seznam pro programátora uživatelské aplikace. Odstranit položku z výpisu lze označením konkrétní položky a kliknutím pravým tlačítkem myši na tuto položku.

V programu je vytvořeno také okno s nápovědou, obsahující HTML stránku, kde je vysvětleno ovládání programu. Část tohoto okna je vyobrazena na Obr. 21.



Obr. 21: Část obrazovky s nápovědou k programu realizujícímu převod souborů

Posledním oknem je okno „O programu WinCC-SVG konvertor“ (Obr. 22), kde jsou uvedeny základní informace o programu.



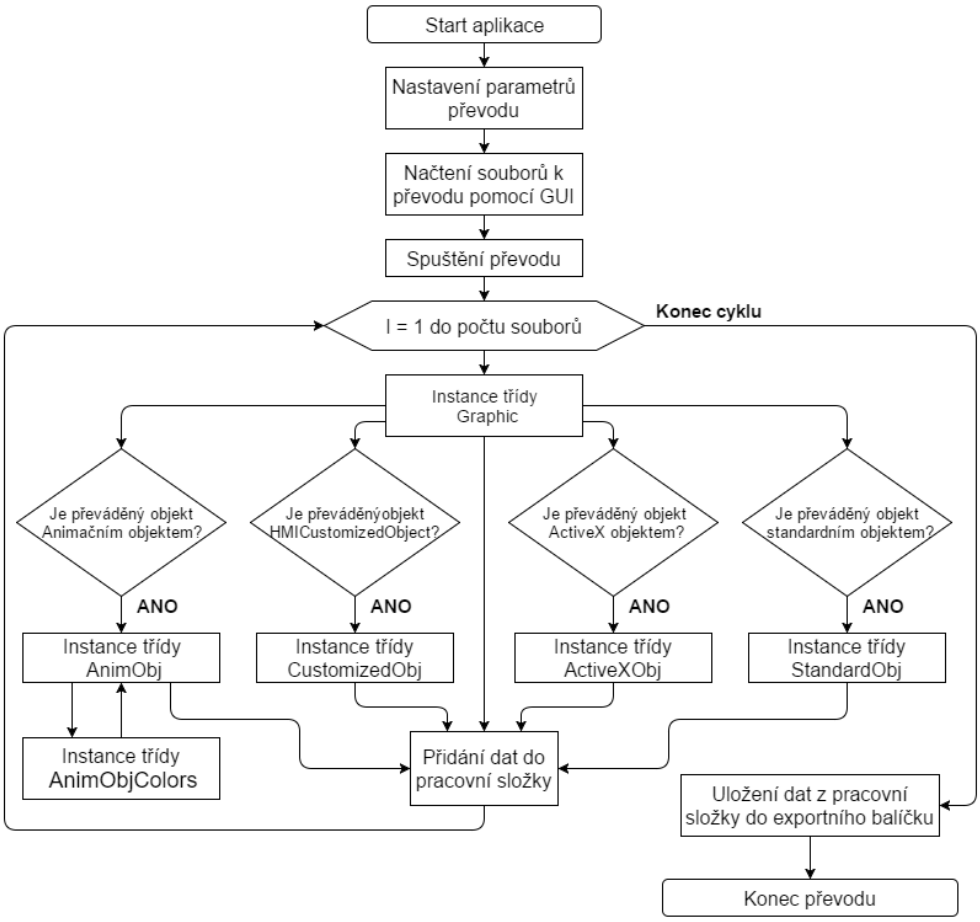
Obr. 22: Okno popisující základní informace o programu

Tato kapitola popisovala vnější rozhraní programu a tedy tu část, se kterou se uživatel při využití tohoto programu setká. V následující kapitole bude program rozebrán z hlediska vnitřní funkčnosti, aby bylo patrné, jak je program koncipován a jaké jsou důvody pro tyto kroky.

6.3 Struktura programu

V předchozí kapitole byl popsán program z hlediska uživatelského ovládání. Nyní bude rozebrána vnitřní funkčnost. Program je tvořen tak, aby byla možnost data převést s využitím napsaných tříd, obsahujících metody, zajišťující funkčnost převodu. Tyto třídy nebyly navrženy na jiné operace, než je převod a vytvoření exportního balíčku. Jejich použití tedy není zamýšleno jako obecné pro tvorbu grafických objektů a nepočítá se s jejich přenesením a obecným využitím v jiných programech. Program využívá pracovní složku nazvanou *Export*, která je uložena v dočasných souborech počítače.

V programu je kromě tříd, zajišťujících uživatelské rozhraní vytvořeno 7 dalších tříd. Tyto třídy obsahují výkonný kód pro převod dat. Protože jsou poměrně rozsáhlé a značně se liší, budou popsány jednotlivě v následujících podkapitolách. Na obrázku Obr. 23 je zobrazen vývojový diagram, který zjednodušeně vysvětluje postup převodu.



Obr. 23: Vývojový diagram vyobrazující postup převodu

6.3.1 Třída Graphic

Tato třída slouží jako základní prvek pro vytváření exportního balíčku. Obsahuje metody pro přidávání dat do XML souborů, vytváření těchto souborů a metody pro vytvoření exportního balíčku. Při převodu je první, které jsou předána data z původního XML souboru, obsahujícího popis objektů. Objekt vytvořený jako instance třídy *Graphic* zpracovává konkrétní data o celé grafické obrazovce, neřeší tedy zpracování jednotlivých typů objektů, pouze vytváří instance dalších tříd pro převod těchto objektů podle toho, jakého typu by měl daný objekt být. Výkonný kód této třídy tedy provádí třídění objektů do skupiny dle typu objektu.

Základem po vytvoření instance této třídy a naplnění základních paměťových míst pomocí konstruktoru je metoda *Convert*. Tato metoda zahájí převod jedné konkrétní obrazovky, jejíž data jsou do metody vložena pomocí vstupních parametrů. V této metodě je již zmiňované rozřazování objektů a vytváření instancí konkrétního typu objektu.

Rozřazení probíhá v cyklu. Objekty jsou zpracovávány jeden po druhém. Rozřazení je realizováno pomocí příkazu *switch*. Těmto objektům jsou předávány parametry, které jsou zároveň vstupem metody *Convert*, která se od stejnojmenných metod v dalších třídách liší. Jde o údaje zadané skrze grafické rozhraní uživatelem a slouží pro správné převedení objektů. Lze uvést cestu k definičním souborům animačních objektů, název CCI serveru pro systém COMES, názvy složek pro formuláře a animace, parametry nastavení programu a další. Dále je na tyto objekty předán XML dokument, do kterého jsou data vkládána po převodu. Tento XML dokument je v podstatě výsledným uživatelským formulářem pro systém COMES. V metodě *Convert* je do tohoto dokumentu přidán pouze základní HTML element *div* s definicí vlastností tak, aby byly všechny části formuláře zobrazeny správně a aby v případě přesahu mimo grafických dat mimo zobrazenou plochu bylo možné rolovat. Výsledný element poté vypadá následovně:

```
<div style="width:100%; height:100%;position:absolute;left: 0px;
top: 0px; overflow:scroll;">
```

Kód formuláře lze rozdělit na 2 části. První je uvozená elementem *svg* a jsou do ní vkládány grafické objekty vykreslované pomocí technologie SVG. Tento element je vložen do základního elementu *div*. Druhá část je také vložena do základního elementu a je tvořena HTML elementy, například vstupními poly, checkboxy a dalšími prvky, které nejsou převáděny do SVG.

Dalšími důležitými metodami, bez kterých by nebyl převod funkční, jsou metody pro vytváření souborů referenčních a definičních vlastností objektů (metody s názvy *CreateReferenceFile* a *CreateCongfigFile*), vytvoření souboru s vlastnostmi balíčku (metoda *GeneratePackageInfo*) a metoda pro vytvoření zip archivu exportního balíčku (metoda *CreateZipFile*). Tyto metody jsou všechny definovány jako statické, aby byla možnost jejich využití v dalších třídách. Protože jsou soubory referenčních vlastností téměř shodné, je zde pouze jedna metoda pro jejich vytvoření a typ souboru je zadán vstupním parametrem. V případě souborů definičních vlastností jsou zde větší odlišností, proto byla definice metody přetížena pro každý typ souboru.

Tento popis stručně popisuje základní metody a funkčnost vytvořenou ve třídě. Třída obsahuje další pomocné metody, které ale nedefinují funkčnost programu a proto nejsou uvedeny.

6.3.2 Třída HMIObject

Třída HMIObject je vytvořena jako abstraktní a slouží jako základ, definující funkčnost tříd, které zajišťují převod objektů. Třída obsahuje definici základních vlastností, které jsou zděděny do všech metod. Jedná se o základní vlastnosti grafických objektů, které jsou shodné pro všechny třídy realizující převod. Patří mezi ně jméno objektu šířka a výška objektu, pozice od levého horního rohu vykreslovací plochy úhel natočení, popisek, název CCI serveru a další.

Dalším prvkem, který je důležitý pro převod, jsou 3 knihovny (datový typ *Dictionary*), pojmenované *HMIType*, *objType* a *HMIProperty*. První jmenovaná definuje, na jaký prvek se převede konkrétní objekt. Jako příklad lze uvést objekt obdélníku, v systému WinCC definovaném jako *HMIRectangle*. Hodnota, kterou knihovna vrátí, bude *rect*, což je název objektu obdélníku v definici SVG jazyka. Takto jsou převedeny i další objekty.

Druhá jmenovaná knihovna, tedy *objType*, definuje, jaký typ prvku bude vytvořen. Návratová hodnota může být *svg*, nebo *html*. Tato knihovna rozřazuje prvky tak, aby byly vkládány do části definující SVG grafiku, nebo jak je patrné, jako HTML elementy.

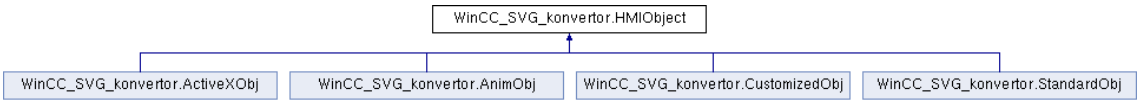
Poslední knihovnou, je knihovna popisující jednotlivé vlastnosti objektů. Určuje, jak budou vlastnosti převáděny. Pokud je výstupní hodnota nevyplněna, tato vlastnost nebude převáděna. Jako příklad lze uvést vlastnost systému WinCC s názvem *BackColor*, tedy barvu výplně, která bude pro SVG objekty převedena jako *Fill*.

Posledním prvkem této třídy jsou metody, které jsou zděděny do dalších tříd. Jedná se hlavně o předpis metody *Convert*, který je pro všechny třídy realizující převod stejný. Její implementace je však pro každou třídu jiná, protože se jedná o virtuální metodu. Jde o nejdůležitější metodu programu. Její předpis vypadá následujícím způsobem:

```
virtual public void Convert(XmlDocument doc, string path, string
animComesFolderName, string formComesFolderName, string
cciServerName,int formCount , List<string> globalColorScheme, string
animObjDefFolderPath = "Symbols")
```

Poslední důležitou metodou v této třídě je metoda *GenerateIFrame*. Tato metoda zajišťuje vygenerování struktury elementů pro vložení animačních objektů v systému COMES, včetně navázání na tagy.

Jak je patrné z předchozích řádků, tato třída slouží jako základ pro další skupinu tříd, popsaných v dalších kapitolách. Tyto třídy již realizují převod. O které třídy se jedná je patrné z následujícího diagramu (Obr. 24).



Obr. 24: Diagram dědičnosti pro třídu HMIObject. Ukazuje, které třídy dědí ze třídy HMIObject

6.3.3 Třída AnimObj

Tato třída slouží pro převod animačních objektů, vytvořených firmou Compas. Je zde využita dědičnost, tedy tato třída dědí ze třídy *HMIObject*. Třída se od ostatních tříd liší v tom, že vytváří v podstatě 3 části kódu. Prvním je kód zobrazení animačního objektu. Zde jsou definovány objekty, ze kterých se objekt skládá. Druhým je výkonný kód objektu, zajišťující animování objektu. Posledním typem kódu je *iframe*, vkládaný do HTML dokumentu uživatelského formuláře.

Aby bylo možné zpracovat objekty tohoto typu, musí být načten soubor s definicí animačního objektu – děje se tak podle údajů předaných z instance třídy *Graphic* a také z dat vyčtených z vlastností původního objektu. Definiční soubor je tvořen CSV souborem, tedy tabulkou a proto je do objektu načten jako datová tabulka, obsahující

stejné informace. Tato tabulka je blíže popsána v kapitole 3.2.1. Tabulka je po načtení postupně procházena a údaje z ní jsou zpracovány a přetvořeny do souboru animačního objektu, který je do exportního balíčku vložen pomocí statických metod, popsaných v kapitole 6.3.1.

Podobně, jako třída *HMIObject*, obsahuje třída *AnimObj* 4 knihovny – *_animObjType, _animObjProperty, _animObjTagType* a *_AnimAttrTypeDict*. První dvě knihovny jsou podobné těm definovaným ve třídě *HMIObject*. První definuje převod typu objektu z definiční tabulky na typ objektu definovaný v SVG jazyce, druhá určuje převod vlastností z definice animačního objektu na vlastnosti SVG objektů a třetí určuje, jaký datový typ původní vstupně výstupní veličiny v definičním XML souboru bude nastaven v systému COMES jako rozhraní pro konkrétní animační objekt. Čtvrtá knihovna slouží k definici toho, zda se parametr objektu uvedený v datové tabulce týká pouze aktuálního objektu nebo i k němu přidruženého orámování.

Třída obsahuje také další datová místa pro ukládání vlastností. Protože jsou zde vlastnosti pevně definovány, obsahuje třída datová místa pro všechny důležité vlastnosti, popsané v kapitole 3.2.1. Tato datová místa jsou naplněna daty pomocí metody *DecodeProperties*, která vlastnosti z původního definičního XML souboru převádí do požadované podoby a ukládá do již zmíněných datových míst.

Převod je realizován voláním metody *Convert*. V této metodě je nejprve pomocí metody *LoadAnimObjDescr* zajištěno naplnění datové tabulky. Následně je vygenerován kód vzhledu animačního objektu metodou *GenerateAnimObjFile*. Tento kód je uložen jako XML dokument. V dalším kroku je pomocí metody *GenerateCsCode* vygenerován řetězec, definující výkonný kód animačního objektu, sloužící k animování jednotlivých prvků. Nakonec je vygenerována definice komunikačního rozhraní animačního objektu a všechny tyto definiční části animačního objektu jsou pomocí statických metod pro uložení animačního objektu ze třídy *Graphic* uloženy do struktury exportního balíčku. Po provedení všech těchto kroků je vygenerována struktura elementů, obsahujících element *iframe*, který je vložen do kódu uživatelského formuláře a slouží pro zobrazení animačního objektu na definovaném místě.

Za zmínku také stojí pomocná metoda, realizující převod barevného kódu do správného tvaru. Je nazvána *DecodeColor* a zajišťuje, aby byl kód barvy převeden do tvaru vhodném pro SVG, v tomto případě zápisu RGB barev v hexadecimální soustavě. Řetezec s číslem barvy musí mít následující tvar – *#rrggbb,* kde *rr* je hodnota červené barvy v hexadecimální soustavě, *gg* hodnota zelené barvy a *bb* hodnota modré barvy. Tuto metodu bylo nutné implementovat, protože v definici animačního objektu jsou sice barvy definovány také pomocí hexadecimálního zápisu jednotlivých složek, ale ve tvaru *0xbbggrr*, tedy je zde jiné pořadí složek a barvy by tím pádem neodpovídaly vzoru.

6.3.4 Třída StandardObj

Jestliže v předchozí kapitole uvedená třída realizovala převod jednoho prvku, tato třída slouží k převodu všech standardních prvků z prostředí WinCC a je proto nejsložitější.

Třída opět dědí z třídy *HMIObject*. Je nutno uvést, že převáděné objekty se v mnohých parametrech podobají a převodní algoritmy jsou často stejné, nebo velice podobné, proto byla zvolena cesta vytvoření jedné třídy, která vnitřně typy jednoduchých objektů rozděluje a podle toho, o který objekt se jedná, je zvolen typ převodu.

Převod objektů pomocí této třídy probíhá následujícím způsobem. V prvním kroku je v metodě *Convert* opět volána metoda *DecodeProperties*, zajišťující zaznamenání hodnot základních vlastností, nebo v případě této třídy navíc také vlastností, jejichž hodnotu je třeba uchovat, ale nelze je transformovat přímo do jedné nové vlastnosti. Hodnota takovéto vlastnosti je poté využita pro různé účely a liší se i podle typu převáděného objektu. V následujícím kroku je zjištěn typ převáděného objektu. Poté je pomocí knihovny *_objType*, definované ve třídě *HMIObject* zjištěno, zda se převáděný objekt bude převádět do části uživatelského formuláře určené pro SVG objekty, nebo zda bude vložen jako HTML kód. V následujícím kroku je pomocí metody *DecodeType* určeno, jak se bude konkrétní typ souboru převádět. Převedení je prováděno u objektů různým způsobem, u většiny je ale použita jedna ze dvou metod, realizujících převod. Jedná se o metody s názvem *SetElementAttributes* pro objekty převáděné do SVG a *SetHtmlAttributes* pro objekty převáděné na prvky HTML. Tyto metody realizují výběr, převod a úpravy vlastností převáděného objektu. Pro převedení souboru je zde několik pomocných metod, z nichž jako nejdůležitější lze jmenovat metodu s názvem *PropertyDecode*. Tato metoda převádí jednotlivé vlastnosti a případně realizuje úpravu hodnot do správného tvaru, aby bylo možné vlastnost přiřadit k objektu. Nutno podotknout, že většina exportovaných vlastností v současné podobě programu není využita a transformována do nových vlastností. V mnohých případech jsou využity základní vlastnosti pro zobrazení a zbytek není využit. V posledním kroku převodu je objekt přidán do HTML dokumentu, do kterého by měl být zahrnut. Může se jednat přímo o uživatelský formulář konkrétní vizualizační obrazovky, nebo může být objekt vložen do animačního objektu, který vznikne převodem typu *HMICustomizedObject*.

Opět, jako v předchozí metodě je zde implementována pomocná metoda *DecodeColor* pro převod barevného kódu. V tomto případě se ale nejedná o převod z hexadecimálního zápisu, nýbrž z dekadického, který metoda převede do hexadecimálního tvaru a opět přeskládá barvy do správného pořadí. Kód metody je tedy velice podobný.

6.3.5 Třída ActiveXObj

Tato třída je oproti dvěma předchozím mnohem jednodušší. Protože objekty, definované jako ActiveX objekty, nelze převádět z důvodu nemožnosti exportu dat, byla tato třída definována tak, aby místo ActiveX objektu vložila na správné místo obdélník s černým orámováním o velikosti původního objektu. K tomuto obdélníku je vložen text, zjednodušující identifikaci objektu. Je zde uveden název knihovny, ze které prvek pochází (uvozen popisem *ServerName*) a pojmenování objektu (uvozeno popisem *Name*).

Po domluvě s firmou Compas nebyly další funkčnosti implementovány. Je počítáno s tím, že programátor, vytvářející uživatelskou aplikaci, vytvoří objekt manuálně, nebo objekt vloží jako obrázek, vytvořený z původního prvku.

Protože tato třída neobsahuje žádné další funkčnosti, je zde implementována pouze metoda *Convert*, která slouží k přečtení základních parametrů z vyexportovaných dat a z těchto dat vytvoří výše jmenované prvky. Tyto prvky následně vloží do kódu uživatelského formuláře, definujícího obrazovku.

6.3.6 Třída CustomizedObj

Třída *CustomizedObj* byla vytvořena hlavně z důvodu, že nebylo možné jednoduše převést objekty typu *HMICustomizedObject*. Objekt tohoto typu je tvořen seskupením základních objektů, například obdélníků, obrázků a vstupně výstupních polí. Oproti klasickému seskupení do skupin však má definováno rozhraní. Přístup k definovaným vlastnostem je tedy nepřímo přes toto rozhraní. Z tohoto důvodu je z objektů seskupených do tohoto typu objektu vytvořen pro systém COMES animační objekt, který obsahuje prvky, definované uvnitř. Bohužel se nepodařilo ve skriptu, realizujícím export dat ze systému WinCC nalézt požadované propojení mezi definovaným rozhraním (vlastnostmi zastřešujícího objektu) a uvnitř zapouzdřenými objekty. Lze vyexportovat vnější vlastnosti a určit, která vlastnost dílčího objektu je napojena na vnější vlastnosti a tedy i na rozhraní, bohužel však nelze určit, na kterou konkrétní vlastnost bude proměnná napojena. Proto je do systému COMES převeden pouze kód pro zobrazení. Výkonný kód objektu a rozhraní pro napojení nejsou implementovány a musí být přidány ručně.

Třída opět dědí z třídy *HMIObject*. Třída obsahuje kód metody *Convert*, který prochází vnitřní strukturu objektu typu *HMICustomizedObject* a podle typu vnitřního objektu vytvoří instanci některé ze tří předchozích tříd. Tato instance poté zrealizuje převod a vrátí kód, který je vložen do animačního objektu. Protože je vlastní převod realizován v instancích jiných tříd, obsahuje tato třída pouze jednu další metodu a tou je metoda *DecodeProperties*. Tato metoda, stejně jako u předchozích tříd, realizuje čtení a převod základních vlastností objektu, které jsou následně využity pro vložení do uživatelského formuláře převáděné vizualizační obrazovky.

6.3.7 Třída AnimObjColors

Třída AnimObjColors je poslední třídou mého programu. Její instance jsou pouze pomocné a slouží pro uchování a standardizaci barev pro animační objekt, tedy pro třídu *AnimObj*. Kromě datových oblastí, zajišťujících uložení hodnot jsou zde metody, které jsou vytvořeny tak, aby vracely řetězce s těmi částmi výkonného kódu animačního objektu, které se týkají barev (jedná se o kód v jazyce C#, zajišťující funkčnost animačního objektu). Jedná se například o metody pro generování inicializace proměnných, nebo části kódu definující barvy v případě blikání.

Třída byla vytvořena hlavně z důvodu zprovoznění blikání prvků animačního objektu. Tato operace podstatně zvyšuje složitost výkonného kódu animace, proto byla práce

s barvami objektů vložena do samostatné třídy, jejíž instance jsou při převodu animačního objektu využívány.

6.4 Převáděné objekty

V obrazovkách systému WinCC lze používat různé typy objektů od jednoduchých tvarů po složitější objekty z knihoven WinCC. Protože objekty formátu SVG jsou tvořeny z jednoduchých tvarů, není možné původní objekty přesně převádět odpovídajícím způsobem. Jednotlivé stránky vytvořené ve WinCC jsou převáděny do uživatelských formulářů v systému COMES a animační objekty, které jsou do WinCC externě doprogramovány firmou Compas, jsou převedeny na prvky systému COMES s názvem *Animace*. Místo ActiveX objektů systému WinCC je do obrazovek vložen pouze obdélník o původní velikosti těchto objektů a nápis, určující knihovnu, ze které objekt pochází a lokální název přidaný hlavně pro jednodušší najití těchto prvků v kódu formuláře a případně i původní vizualizaci.

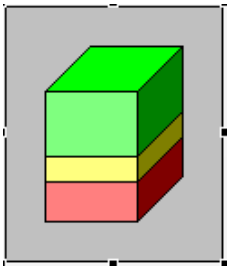
Každý z převáděných objektů obsahuje vlastnosti, které definují jeho funkci a grafické zobrazení. Některé vlastnosti je nutné převést, ale většinu vlastností není možné do systému COMES a objektů SVG převést. Všechny objekty obsahují definici základních vlastností, jako je definice polohy, výšky a šířky objektu, jména a viditelnosti objektu. Další vlastnosti mohou být specifické pro konkrétní objekt a jejich hodnota může provádět jinou operaci.

V systému WinCC se nachází i další typy objektů, které je možné vložit do obrazovek. Jedná se o tzv. *Controls*, pomocí kterých je možné vkládat kód z prostředí *.NET,* ActiveX objekty, případně aplikaci typu *WPF* (Windows presentation foundation). Tyto objekty nejsou převáděny, protože k nim neexistuje v systému COMES adekvátní náhrada. Jejich případný výskyt v obrazovkách je tedy ignorován a je na programátorovi systému, zda vytvoří objekty, splňující shodnou funkčnost manuálně, nebo zda bude využita některá z již existujících možností systému COMES.

Typy objektů, se kterými se v převodním programu pracuje a specifika jejich převodu, jsou definována v následujícím výpisu. U některých budou uvedeny i příklady převodu grafických dat, aby mohl čtenář lépe pochopit popisující text.

HMI3DBarGraph

Jedná se o objekt, představující zobrazení dat ve 3D sloupcovém grafu (Obr. 25). Do systému COMES je převeden pouze jako klasický 2D graf, vytvořený pomocí animačního objektu. Bližší popis je uveden v popisu objektu *HMIBarGraph*, se kterým je shodný. Výsledkem převodu je tedy stejný objekt, který lze vidět na Obr. 27 vpravo.

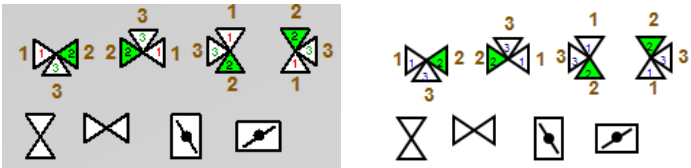


Obr. 25: Původní objekt HMI3DBarGraph v systému WinCC

HMIActiveXControl

Tento typ objektu v sobě skrývá všechny typy ActiveX objektů, nacházející se v systému WinCC. Tyto objekty se rozlišují pomocí parametru *ServerName,* který alespoň přiblíží, ze které knihovny prvek pochází. Může se jednat o okno zobrazující alarmy, prvek z knihovny *HMI Symbol library*, animační objekt firmy Compas nebo další objekty z knihoven importovaných do systému WinCC. U prvních dvou jmenovaných, nebo lépe řečeno u všech objektů tohoto typu s vyjímkou animačního objektu firmy Compas nelze grafické zobrazení získat. Tyto objekty jsou převáděny pouze jako obdélníky s vloženým textem (viz kapitola 6.3.5). Jedná se o převod do formátu SVG. Očekává se, že vložený obdélník a text budou manuálně nahrazeny objektem, který splňuje funkci, nebo budou smazány, pokud je prvek v nové vizualizaci nadbytečný.

V případě animačních objektů firmy Compas je situace odlišná. K objektům existuje definiční soubor, který je dostupný a je proto možné tyto objekty převést. Výsledkem je objekt *Animace* v systému COMES a odkaz na tento objekt ve formuláři pomocí elementu *iframe*. Tyto objekty můžou zobrazovat v podstatě jakoukoli grafiku a text, proto jsou na Obr. 26 pro ilustraci uvedeny pouze některé z nich.



Obr. 26: Vlevo - skupina animačních objektů firmy Compas v systému WinCC, vpravo - převedené objekty v systému COMES

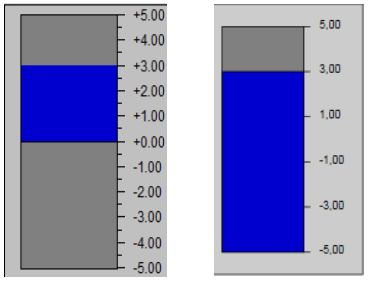
HMIApplicationWindow

HMIApplicationWindow je typ objektu, definující aplikační okna. Může sloužit pro zobrazení různých informací, například tiskových úloh. Prvek není převáděn, protože neobsahuje žádné grafické prvky vhodné k převodu.

HMIBarGraph

Jedná se o objekt 2D bargrafu, který zobrazuje hodnoty. Protože nebylo možné převést graf pomocí vyexportovaných grafických dat, byl vytvořen objekt, který by jej nahradil. Jedná se o klasické zobrazení prvku tohoto typu, které umožňuje zobrazení jak v klasické vertikální, tak i v horizontální podobě. Grafická shoda zde není přesná, byl kladen důraz

na reprezentaci zobrazované hodnoty. Společně s objektem *HMI3DBarGraph* se jedná o jediné dva objekty splňující funkčnost, které nejsou vytvořeny přímým převodem z převodního skriptu. Příklad převodu objektu lze vidět na Obr. 27.



Obr. 27: Vlevo - bargraf systému WinCC, vpravo bargraf vytvořený pro systém WinCC

HMIButton

Objekt tlačítka, převáděný do systému COMES jako klasické tlačítko ve formátu HTML. Při převodu tohoto prvku nedochází k žádným zvláštním událostem. Objekt není napojen na proměnné. Počítá se s přiřazením zvláštní události, například potvrzení hodnot v zadaném formuláři.

HMICircle

Tímto objektem je definován kruh. V převedeném prvku je definována barva orámování a výplně, šířka orámování, viditelnost. Do systému COMES je převáděn jako SVG element typu *circle*.

HMICircularArc

Jedná se o objekt definující část kruhového oblouku. Je definován poloměrem a úhly od svislé osy pro počáteční a pro koncový bod. Společně s prvky *HMIEllipseSegment, HMIEllipseArc* a *HMIPieSegment* se jedná o nejproblémovější prvek převodu, protože tvorba výsečí a oblouků se pomocí formátu SVG vytváří pomocí naprosto odlišného přístupu, než je tomu v případě systému WinCC.

V případě SVG je využito kreslení tzv. křivek, tedy elementu *path*. Tento element potřebuje pro definici jiné parametry. Vychází se zde ze souřadnic počátečního a koncového bodu, poloměrů hlavní a vedlejší poloosy, natočení elipsy o určitý úhel a informace, zda je úhel oblouku (rozdíl koncového a startovního úhlu) větší, než 180° a zda má být křivka vykreslována pomocí pravotočivé křivky (po směru hodin) nebo pomocí levotočivé křivky. Vzhledem k těmto rozdílům bylo nutné některé parametry určit – vykreslování je vždy pravotočivé a startovní a koncové body musely být vyexportovány ze systému WinCC. V tomto systému nejsou uvedeny přímo jako startovní a koncový bod, ale jsou vedeny jako body pro uchycení dalších objektů, takže bylo možné i tuto informaci získat. Pro ilustraci je uvedena vlastnost *d*, definující křivku výseče:

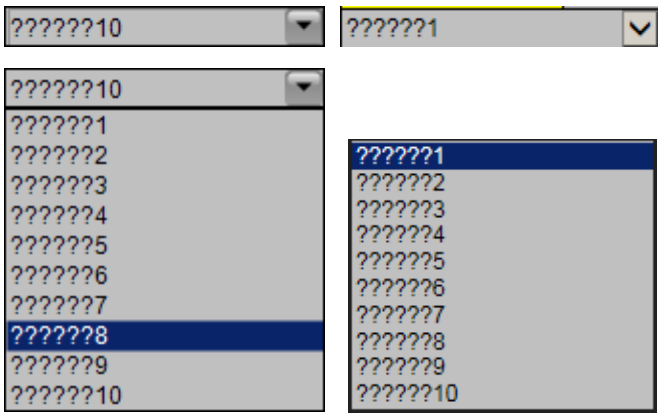
```
d="M1520,40 A70,70 0 0 1 1450,110"
```

V příkladu písmeno *M* znamená přesun na absolutní souřadnice počátečního bodu, písmeno *A* uvozuje tvorbu oblouku s poloměry os 70 pixelů, úhlem natočení 0°, informací, že oblouk má méně než 180°, rotací v pravotočivém směru a koncovým bodem na souřadnicích x = 1450 a y = 110. Nutno uvést, že pokud by písmena v definici byla malá, jednalo by se o relativní souřadnice. Pro představu, jak takovýto objekt vypadá, je na následujícím obrázku (Obr. 28) uveden příklad.

Obr. 28: Vlevo - objekt HMICircularArc v systému WinCC, vpravo - objekt po převedení do systém COMES

HMIComboBox

Objekt výběrového roletového listu (Obr. 29). Jeho napojení na proměnné není po dohodě s firmou Compas realizováno. Geometrické parametry, počty prvků k výběru i textové popisy jsou definovány přesně jako u původního prvku v systému WinCC. Prvek je převáděn na HTML element *select,* obsahující jednotlivé prvky pro výběr ve vnořených elementech typu *option*. Zde je nutno připomenout, že vizualizace v systému COMES slouží hlavně pro zobrazení stavu technologie. Pro ovládání je využito jiných systému, například v této práci využívaný systém WinCC. Je tedy pravděpodobné, že většina prvků tohoto typu bude odstraněna.



Obr. 29: Vlevo nahoře - objekt HMIComboBox systému WinCC, vlevo dole - stejný prvek v rozbaleném stavu, vpravo nahoře - alternativa v systému COMES, vpravo dole - rozbalený prvek v systému COMES

HMICustomizedObject

HMICustomizedObject je speciálním objektem, slučujícím jiné objekty. Jeho převedení je realizováno ve třídě *CustomizedObj*, uvedené v kapitole 6.3.6, kde je uveden i postup převodu a jeho problémy. Název animačního objektu se shoduje s názvem objektu v systému WinCC.

HMIEllipse

Tento objekt definuje elipsu. Specifikum elipsy v systému WinCC je to, že má hlavní osu vždy buď s úhlem 0°, nebo 90°. nelze s ní tedy libovolně rotovat. Do systému COMES je převáděna jako SVG element *ellipse* a obsahuje všechny vlastnosti pro statické zobrazení objektu.

HMIEllipseArc

Jedná se o objekt, definující eliptický oblouk, tedy část elipsy bez výplně. U tohoto objektu je postup převodu stejný, jako v případě výše jmenovaného objektu *HMICircularArc*. Je převáděn na SVG element *path* a sdílí také stejné problémy při převodu. Odlišnost je pouze v hodnotách poloměrů os, kdy u eliptického oblouku se poloměry os liší. Výsledný objekt pak může vypadat například tak, jak je uvedeno na obrázku Obr. 30.

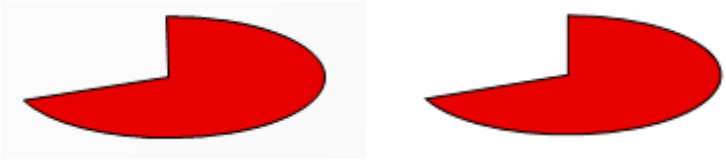
Obr. 30: Vlevo - objekt HMIEllipseArc v systému WinCC, vpravo - alternativa v systému COMES pomocí elementu path

HMIEllipseSegment

Další objekt definující část elipsy. V tomto případě se jedná o výseč a vybraná část tedy obsahuje i výplň a celý objekt je orámován. Tyto změny se projeví hlavně v definici objektu, kde musí být přidáno i kreslení čar tak, aby křivka byla uzavřena. Problémy převodu jsou opět shodné, jako u předchozích dvou jmenovaných oblouků a objekt je tedy převáděn také na SVG element *path*. Zde je ale navíc nutné z vyexportovaných údajů zpětně dopočítat souřadnice středu elipsy, ze které výseč pochází, protože údaj o středu je neznámý. Vlastnost *d*, která definuje křivku, se mírně liší od předešlých oblouků a vypadá následujícím způsobem:

```
d="M1050,50 L1140,50 A90,60 0 0 1 1050,110 z"
```

Opět zde je přesun na bod pomocí písmene *M* a souřadnic bodu. V tomto případě se však jedná o střed elipsy, ze které je výseč tvořena. Dále je zde ale kreslení čáry od středu k počátečnímu bodu, pomocí písmene *L*, následuje definice oblouku ve stejném tvaru, jak bylo popsáno u objektu *HMICircularArc*. Definice je zakončena písmenem *z*, které zajistí, že křivka je uzavřena. Příklad objektu lze vidět na následujícím obrázku (Obr. 31).



Obr. 31: Vlevo - objekt HMIElipseSegment v systému WinCC, vpravo - alternativa v systému COMES

HMIGraphicObject

Tento objekt slouží v systému WinCC pro zobrazení obrázků, které mohou být i předem definované. Jedná se například o pomocné značky, zjednodušující orientaci ve vizualizaci. Do systému COMES je převáděn jako HTML element uživatelského formuláře s názvem *img*. Obrázky zobrazené tímto elementem musí být do systému COMES manuálně přidány a cesty k obrázkům musí být nastaveny do elementu. Jedná se tedy pouze o jakýsi polotovar, který musí být programátorem ručně upraven.

HMIGroup

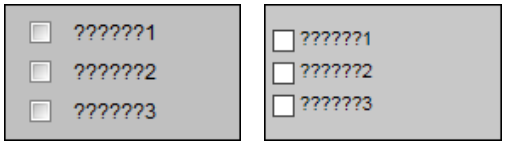
Objekt *HMIGroup* slouží ke slučování různých objektů do jednoho bloku, který usnadňuje práci se skupinou objektů při návrhu zákaznické aplikace. Při převodu do systému COMES je tento objekt zpětně rozložen na jednotlivé objekty. Skupina je tedy rozdělena a po převedení se pracuje s jednotlivými objekty. K tomuto kroku bylo přistoupeno z důvodu, že v systému COMES se již pracuje přímo s definičním kódem objektů, oproti grafickému rozhraní u systému WinCC. Formát SVG umožňuje slučování do skupin, ale vzhledem k tomu, že po převedení jsou prvky na správných absolutních souřadnicích, byla tato skupina vypuštěna s tím, že v případě potřeby může být manuálně přidána.

HMIGroupDisplay

Tento typ objektu není převeden. Jedná se o objekt, zobrazující různé údaje a funguje jako soustava zobrazovačů. Vzhledem k tomu, že jsou definovány pouze základní vlastnosti a neexistuje definice grafických údajů, není tento objekt převáděn.

HMICheckBox

U objektu *HMICheckBox* dochází k převodu na HTML element *div*. Převedeny jsou grafické proporce, textové popisy i správné počty zaškrtávacích polí. Element *div* definuje orámování a uvnitř tohoto elementu jsou zanořeny elementy typu *input*, definující vlastní zaškrtávací pole, jak je možné vidět na Obr. 32 vpravo. Podobně jako u prvku *HMIComboBox* není tento prvek dále napojen na proměnné v systému a je na programátorovi, zda bude prvek obsloužen, nebo bude smazán.



Obr. 32: Vlevo - prvek typu HMICheckBox systému WinCC, vpravo - převedený prvek do systému COMES

HMIIOField

Objekt pro zadávání, nebo zobrazování hodnot. Pokud je zadána proměnná, realizující zobrazení, je přidána obsluha pro zobrazení této hodnoty. Prvek je převáděn na HTML element *input*. Pokud je nastaven pouze na zobrazování hodnot, je převedenému elementu nastaven parametr *readonly,* aby nebylo možné zasahovat do zobrazované hodnoty. U tohoto prvku je realizován převod včetně napojení na proměnnou, která má být zobrazována.

HMILine

Objekt *HMILine* definuje úsečku. Je převáděn na SVG element *line* a obsahuje všechny grafické vlastnosti, definované v systému WinCC. Objekt je tedy převáděn jako statická čára. Případné dynamické změny vzhledu musí být přidány programátorem.

HMIListBox

Tento objekt funguje podobně, jako objekt *HMIComboBox*. Jedná se tedy o výběr hodnot. Oproti objektu *HMIComboBox* je ale zobrazeno více prvků k výběru (mohou být i všechny). V zobrazení uživatelského formuláře v systému COMES je však objekt zobrazen jako klasický roletový výběrový list, tedy převod je opět realizován do prvku typu *select* a výsledek lze vidět na Obr. 29. Stejně jako v případě objektu *HMIComboBox*, není objekt napojen na proměnné v systému COMES.

HMIMultiLineEdit

Tento objekt je podobně, jako některé další objekty převeden na HTML element *textarea.* Slouží například k sepsání textu o větším počtu řádků. Převáděny jsou parametry textu a definice geometrických vlastností. Jedná se v podstatě o orámované textové pole.

HMIObjConnection

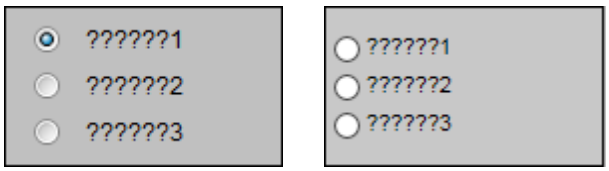
Objekt typu HMIObjConnection slouží k propojení dvou prvků pomocí lomené čáry. Protože nejsou přesně definovány body zlomu, ale pouze propojované objekty, není tento objekt do systému COMES převáděn.

HMIOLEObject

Objekt, umožňující vložení do vizualizace okno, obsahující například PDF dokument, nebo okna jiných aplikací. Tento typ objektu není možné převést do systému COMES a není proto převáděn.

HMIOptionGroup

Objekt typu *HMIOptionGroup* je do uživatelského formuláře systému COMES převáděn jako skupina HTML elementů typu *input*, podobně jako při převodu objektu *HMICheckBox* zanořených v HTML elementu *div* (Obr. 33). Také tento objekt není napojen na proměnné.



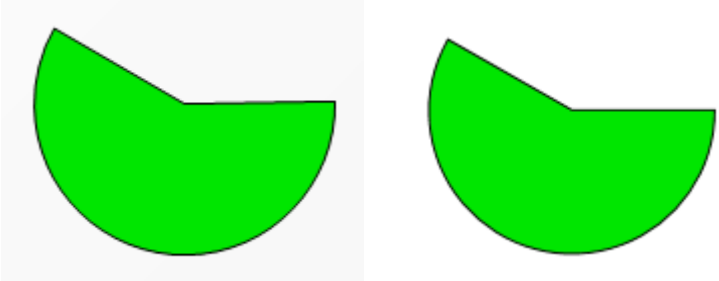
Obr. 33: Vlevo - objekt HMIOptionGroup systému WinCC, vpravo - alternativa pro systém COMES

HMIPictureWindow

HMIPictureWindow je objekt, který umožňuje ve vizualizaci zobrazit klasický obrázek. Do uživatelského formuláře systému COMES je převáděn jako HTML element *img*. Pro správnou funkci tohoto elementu musí být do systému COMES vložen také obrázek, který má být zobrazován. Podle toho, na které místo je do systému uložen, musí být také nastavena cesta ve výsledném elementu, aby byl obrázek správně zobrazen.

HMIPieSegment

HMIPieSegment je posledním ze skupiny objektů, zajišťujících zobrazení výsečí. V tomto případě se jedná o kruhovou výseč, tedy uzavřenou křivku s výplní, tvořící část kruhu. Výseč je do uživatelského formuláře systému COMES opět převáděna jako SVG element *path*. Při převodu se vyskytují stejné problémy, jako v případě objektu *HMIEllipseSegment*, pouze jsou zde nastaveny poloměry os na stejnou hodnotu. Příklad úspěšně převedeného objektu lze vidět na následujícím obrázku (Obr. 34).



Obr. 34: Vlevo - objekt HMIPieSegment v systému WinCC, vpravo - objekt po převedení do systému COMES

HMIPolygon

Objekt typu *HMIPolygon* definuje uzavřenou křivku. Tato křivka může obsahovat výplň a orámování. Je převedena do systému COMES jako SVG element s názvem *polygon*. Při převodu musela být upravována data, definující body. Ve vyexportovaných datech je každý bod uveden v elementu *Points* jako vlastnost tohoto elementu s hodnotou definující

tento bod. Objevují se zde tedy atributy s názvem *Left1*, *Top1*, *Left2* a další. Jejich počet je určen počtem zlomových bodů na křivce. Do systému COMES jsou však body převedeny pod jedinou vlastnost elementu s názvem *points*. Hodnota této vlastnosti je definována řetězcem znaků, ve kterém jsou souřadnice bodu oddělených čárkou. Mezi definicemi bodů je mezera.

HMIPolyLine

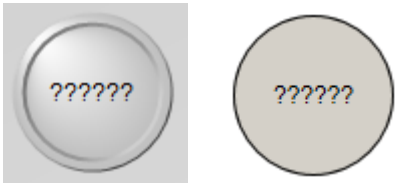
Pro objekt typu *HMIPolyline* platí stejná charakteristika převodu, jako pro objekt typu *HMIPolygon*. Výsledkem převodu je SVG element s názvem *polyline*. Tento objekt neobsahuje výplň a nejedná se o uzavřenou křivku. Body jsou definovány opět pomocí vlastnosti s názvem *points*.

HMIRectangle

HMIRectangle definuje objekt obdélníku. Výsledkem převodu je SVG Element s názvem *rect*. Objekt obsahuje definici výplně a orámování. Vzhledem k tomu, že jsou si vlastnosti ze systému WinCC podobné s vlastnostmi používanými ve formátu SVG, převod neobsahuje žádný specifický postup.

HMIRoundButton

Objekt typu *HMIRoundButton* je v podstatě stejný, jako objekt typu *HMIButton*. Jedná se tedy o tlačítko. V tomto případě je kruhového tvaru, viz Obr. 35. Do uživatelského formuláře systému COMES je po převodu přidán HTML element *button*, kterému je pomocí CSS stylů nastaven parametr *border-radius* na hodnotu poloviny šířky objektu. Je tedy vytvořeno kruhové tlačítko.



Obr. 35: Vlevo - kulaté tlačítko systému WinCC, vpravo - tlačítko systému COMES

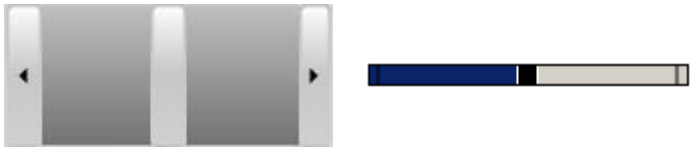
HMIRoundRectangle

Tento objekt lze popsat jako klasický obdélník se zaoblenými rohy. Je převáděn stejně, jako objekt typu *HMIRectangle*, pouze jsou přidány vlastnosti *rx* a *ry*, které definují poloměr hlavní osy a vedlejší osy eliptického zaoblení.

HMISlider

Tento objekt je převáděn na HTML element *input* a je mu nastaven typ *range*, který definuje, že se jedná o posuvník. Při převodu jsou předávány parametry rozsahu a proměnná, která určuje zobrazenou hodnotu – polohu posuvníku. Protože v prohlížeči Internet Explorer nefunguje stylování pomocí CSS stylů pro tento prvek, jeho zobrazení

se od posuvníku v systému WinCC značně liší, jak lze vidět na Obr. 36. V podobě převedených souborů, které byly vloženy do systému COMES, plní pouze funkci vizualizace. Aby bylo možné pomocí tohoto a dalších prvků upravovat hodnoty, musí být definován C# skript pro uložení dat.



Obr. 36: Vlevo - objekt posuvníku v systému WinCC, vpravo - posuvník ve formátu HTML

HMIStaticText

Jedná se o objekt textového pole, které může obsahovat orámování s podbarvením. Je převeden na element *textarea*. Může tedy obsahovat víceřádkový text a využívá se hlavně pro tvorbu popisků.

HMIStatusDisplay

Prvek typu *HMIStatusDisplay* slouží pro zobrazení stavu proměnné pomocí grafického vyjádření, tedy obrázkem. Zobrazení se mění podle toho, který bit je v proměnné aktivní. Protože formát HTML, ani formát SVG neobsahují vhodnou alternativu pro takovéto zobrazení, byl tento problém vyřešen následujícím způsobem. Převod byl realizován tak, aby výsledkem byl HTML element *img*, který umožňuje zobrazit libovolný obrázek. Tomuto obrázku byla přiřazena proměnná, jejíž hodnota je původním prvkem zobrazována. Proměnné byl přiřazen enumerativní typ tak, aby se podle hodnoty měnila cesta k obrázku. Protože se jedná o specifikum systému COMES, je tento postup je detailněji popsán v kapitole 6.5.

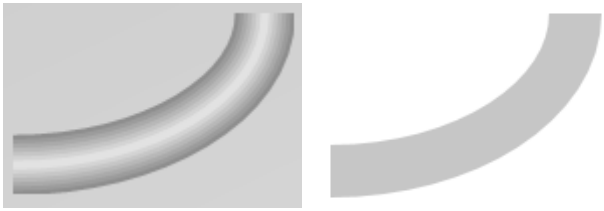
HMITextList

Objekt *HMITextList* je podobný prvku *HMIIOField*. To znamená, že může sloužit pro zobrazení, nebo zadávání hodnot a textů. Oproti jmenovanému však obsahuje více řádků. Do uživatelského formuláře systému COMES je převáděn jako HTML element *textarea*. U tohoto objektu je možné napojení na proměnné. Při převodu je tedy zkontrolováno, zda je prvek napojen na proměnnou, jejíž hodnota má být zobrazována. Pokud toto propojení existuje, je přidáno i v systému COMES.

HMITubeArcObject

Tento objekt umožňuje společně s dalšími třemi objekty jmenovanými dále v systému WinCC kreslit vizualizaci potrubí. V podstatě se jedná pouze o eliptický oblouk, takže má definován počáteční a koncový úhel a poloměry vodorovné a svislé osy a platí pro něj stejný postup převodu. Jediný rozdíl je v tom, že vizualizace v systému WinCC jej zobrazuje s barevným přechodem, aby na pohled připomínal potrubí. To však ve formátu SVG nelze realizovat pomocí jednoho objektu. Objekt je tedy převeden na eliptický

jednobarevný oblouk světle šedé barvy, jak lze vidět na Obr. 37. Výsledkem je stejně jako u objektu *HMIEllipseArc* SVG element *path*, jak je detailněji popsáno výše u popisu eliptického oblouku.



Obr. 37: Vlevo - objekt HMITubeArc systému WinCC, vpravo - objekt převedený do systému COMES

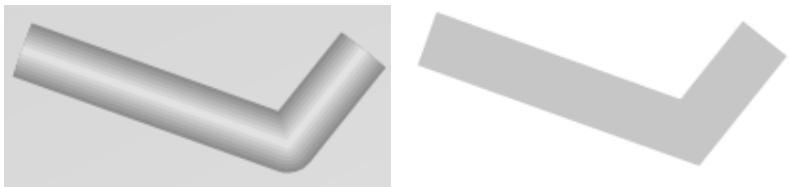
HMITubeDoubleTeeObject

Tento objekt je dalším ze čtyř objektů, sloužících pro kreslení potrubí. Jedná se o křížení potrubí. Opět se jedná o objekty s barevným přechodem, který není převeden. Výsledkem převodu je dvojice čar (elementy *line*) světle šedé barvy a požadované tloušťky, které jsou sloučeny do skupiny pomocí elementu *g*. Převod je realizován tak, že se nakreslí čáry přesně do poloviny hodnot rozměrů obdélníku, který definuje rozměr původního objektu. Na následujícím obrázku je patrný příklad takového objektu před převedením a po převedení (Obr. 38).

Obr. 38: Vlevo - objekt HMITubeDoubleTeeObject systému WinCC, vpravo - převedené objekt do systému COMES

HMITubePolyline

Třetí ze skupiny objektů pro kreslení potrubí je objekt pro kreslení lomené čáry s barevným přechodem opět zobrazujícím vzhled potrubí. Tento objekt je převeden na element *polyline* zadané tloušťky a světle šedé barvy, jak je možné vidět na Obr. 39.



Obr. 39: Vlevo - objekt HMITubePolyline systému WinCC, vpravo - náhrada objektu pro systém COMES

HMITubeTeeObject

Poslední ze skupiny objektů pro kreslení potrubí je takzvaný T-kus. Převod je podobný, jako u objektu křížení. Opět je výsledkem dvojice čar, seskupených do elementu *g* se

stejnými parametry, jako v předchozích dvou případech. V tomto případě je ale horizontální čára umístěna na kraj objektu (viz Obr. 40). Vertikální čára je umístěna opět do poloviny objektu, jako tomu je u objektu *HMITubeDoubleTeeObject*.

Obr. 40: Vlevo - objekt HMITubeTeeObject systému WinCC, vpravo - náhrada pro systém COMES

6.5 Napojení objektů systému COMES na proměnné

Převod objektů v této práci je po dohodě s firmou Compas realizován hlavně tak, aby byl převeden vizuální vzhled objektů. Protože se s mnohými objekty v systému COMES pracuje jiným způsobem, je i jejich napojení na proměnné realizováno různě. U některých typů objektů lze tohoto převodu dosáhnout i automatickým převodem. Pro tuto práci jsou důležité 2 typy napojení.

Prvním případem je vložení animačního objektu do formuláře. Tento způsob převodu je detailně popsán v kapitole 2.1.6.3.

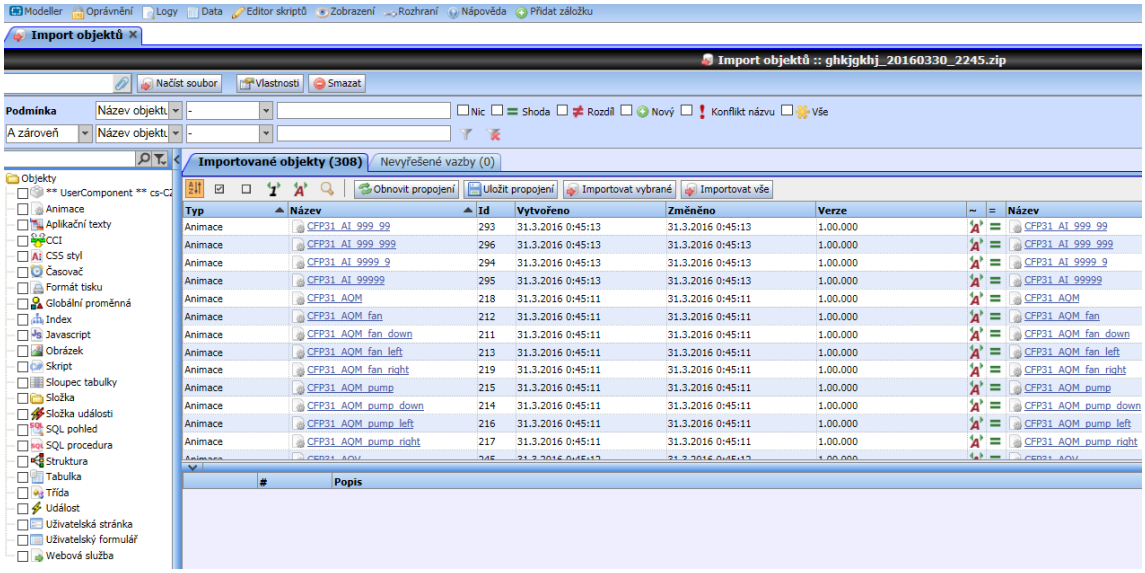
Druhým případem je předání hodnoty proměnné přímo elementu, který je uveden v uživatelském formuláři. Zde je definice podobná, jsou zde však určité odlišnosti. Elementu lze předat až 7 proměnných, přičemž každá může obsluhovat změnu jiného parametru. Opět je zde předáván název proměnné, název CCI Serveru a vzorkovací perioda. V tomto případě lze však navíc přiřadit vlastnost *property* která nám definuje, jaký parametr se v prvku bude měnit a vlastnost s názvem *enums*, která může hodnotě proměnné přiřadit jiné číslo, nebo text. Můžeme tak například měnit podle hodnoty proměnné měnit cestu, kde máme uložen obrázek. Tímto způsobem je možné vytvořit z obyčejného elementu *img* implementaci pro objekt *HMIStatusDisplay* (kapitola 0). Pro příklad lze uvést část definice objektu:

```
<img name="Status Display1" tag="@ZL_trendy_zvlaknovaci_vana1"
sample="2000" cci="OPCDAW" property="src" enums=1|@Arror_1_1.bmp;
2|@Arror_1.bmp;..
```

6.6 Vložení dat do systému COMES

V případě, že potřebujeme v systému COMES použít zdrojové kódy, nebo jiná data uložená v exportním balíčku, je nutné se do systému přihlásit účtem s dostatečnými oprávněními pro vstup do konfigurace modulu COMES Modeller. V tomto modulu se nachází obrazovka s názvem *Import objektů*, která slouží k nahrávání dat z exportních balíčků do systému COMES. Vzhled je velice podobný obrazovce pro export objektů, ale je zde navíc mnoho užitečných funkcí, které usnadňují vložení definovaných souborů do systému. Vzhled obrazovky lze vidět na Obr. 41.

V horní části obrazovky se nachází pole pro výběr exportního balíčku, zobrazení vlastností exportního balíčku a vymazání načteného balíčku. Pod těmito volbami se nachází panel pro nastavení filtrů, velice podobný tomu z průzkumníku souborů. Na levé straně obrazovky je opět panel pro výběr typů objektů. Napravo od tohoto panelu se nachází plocha, zobrazující načtené soubory a případně také jejich ekvivalenty v systému COMES. Jak je patrné z Obr. 41, je možné importovat všechny soubory, nebo pouze určitý výběr. Při stisknutí jednoho z tlačítek pro import dojde ke zkontrolování importovaných souborů – proto není možné načíst například skripty obsahující chyby. Chyby v souborech jsou vypsány do spodního panelu, kde se nachází popis, číslo řádku a název souboru, ve kterém chyba nastala.

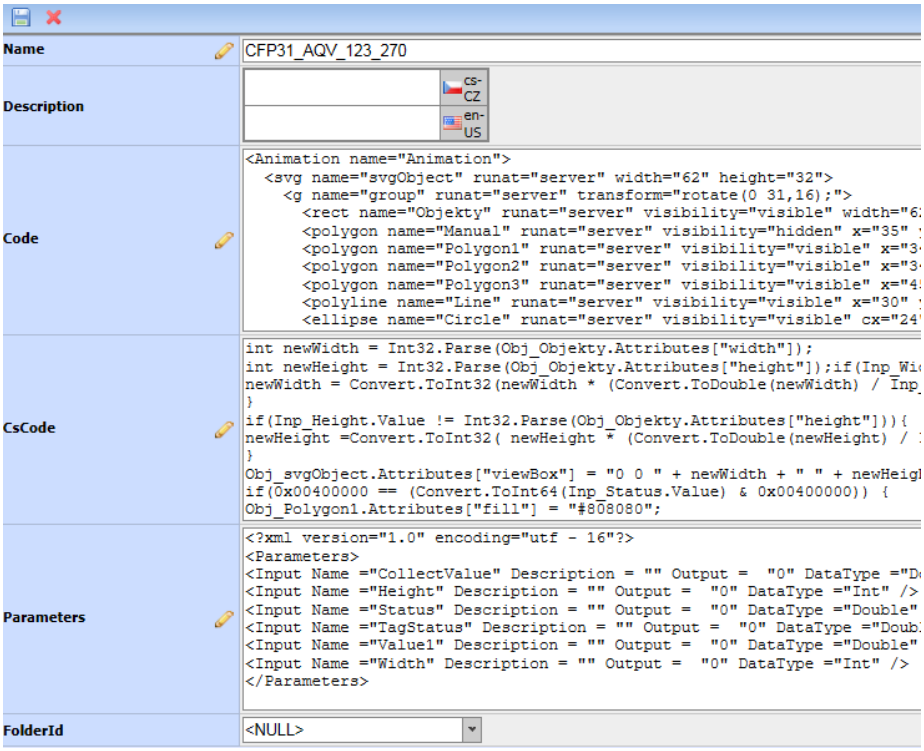


Obr. 41: Obrazovka pro import objektů z exportního balíčku

Pokud se import nezdaří, nebo je třeba udělat určité úpravy na souborech, lze po kliknutí na název konkrétního importovaného objektu otevřít obrazovku pro úpravy. Část obrazovky pro úpravy objektů lze vidět na Obr. 42. Tato obrazovka umožňuje kromě změny kódu a dalších parametrů také nastavit, do jaké složky bude tento soubor nahrán. Po uložení dojde k úpravě načteného souboru. Je tedy možné tuto verzi nahrát do systému opětovným stisknutím jednoho z tlačítek pro import.

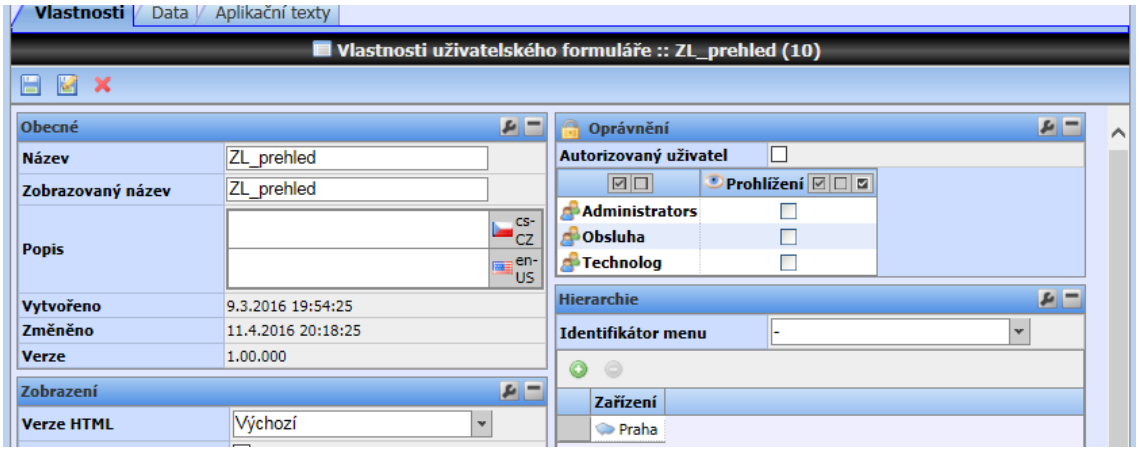
Pokud by i přesto nebylo možné soubory nahrát, je možné, že licence modulu COMES Modeller je pouze na omezený počet souborů pro zobrazení a import je tedy zablokován.

V případě mého programu jsou převáděny soubory, definující uživatelské formuláře, animační objekty a složky. Tyto objekty je po úspěšném nahrání souborů do systému COMES možné nalézt v průzkumníku souborů a opět je vyexportovat. Další možností, kde je možné tyto soubory nalézt je editor skriptů. Zde lze soubory upravovat a vytvořit tak uživatelskou aplikaci.



Obr. 42: Obrazovka systému COMES pro editaci animačního objektu určeného k importu do systému

Po provedení implementace do systému lze v editoru skriptů nastavit zobrazení v menu systému tak, aby mohla být konvertovaná data prezentována uživateli (Obr. 43). Data se poté zobrazí v menu uživatelům s dostatečnými právy pro přístup.



Obr. 43: Okno vlastností formuláře. Pro nastavení zobrazení je důležitá záložka hierarchie.

Import jednotlivých objektů a nastavení parametrů objektů v editoru skriptů je časově nejnáročnější část implementace nových grafických dat a to z toho důvodu, že je již plně na programátorovi, který daný import řeší. Samotný převod objektů je realizován plně automaticky a doba převodu se podle počtu souborů pohybuje maximálně v řádu desítek vteřin.

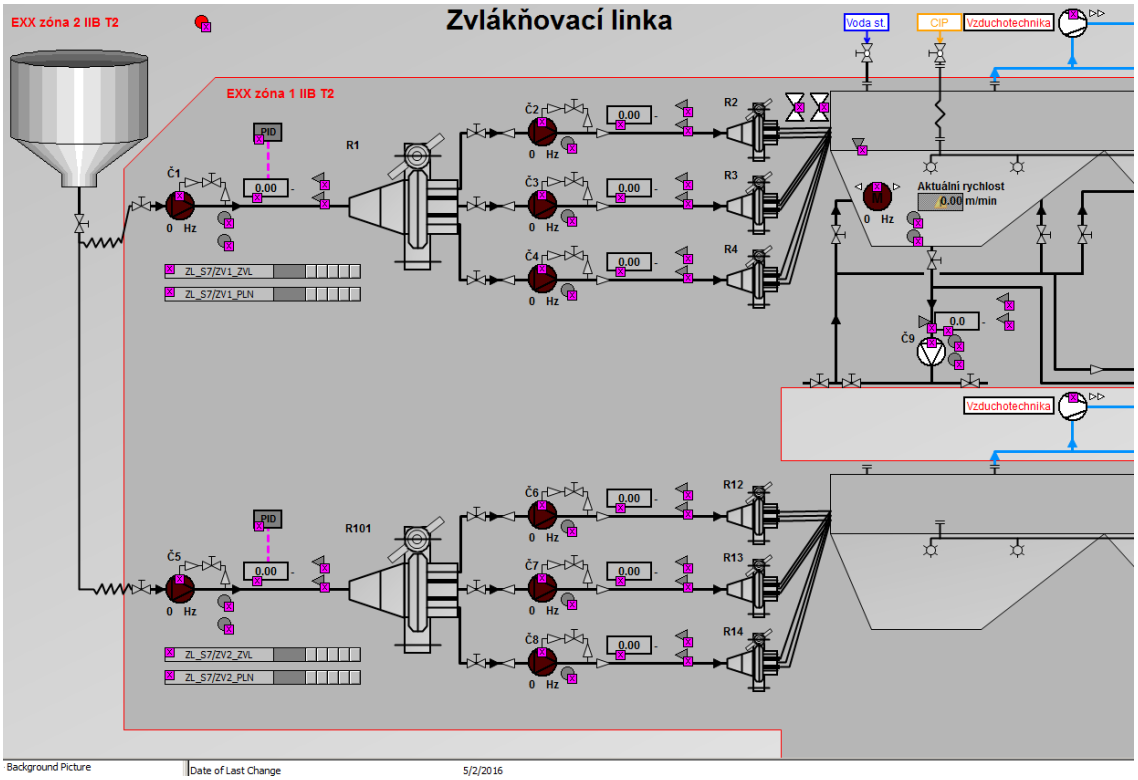
Aby byla patrná kvalita převodu, jsou v následující kapitole pro ilustraci uvedeny ukázky exportu z testovací aplikace.

7 TESTOVACÍ APLIKACE

Pro vypracování této práce byla firmou Compas dodána testovací aplikace, na které byl testován převodní skript a aplikace pro konverzi dat. Jedná se o průmyslový provoz, obsahující například vizualizaci a ovládání zvlákňovací linky. Vzhledem k tomu, že se tato práce nezaobírá konkrétním projektem, ale obecným nástrojem pro převod grafických dat, nebylo využití této testovací aplikace firmou Compas hlouběji popsáno.

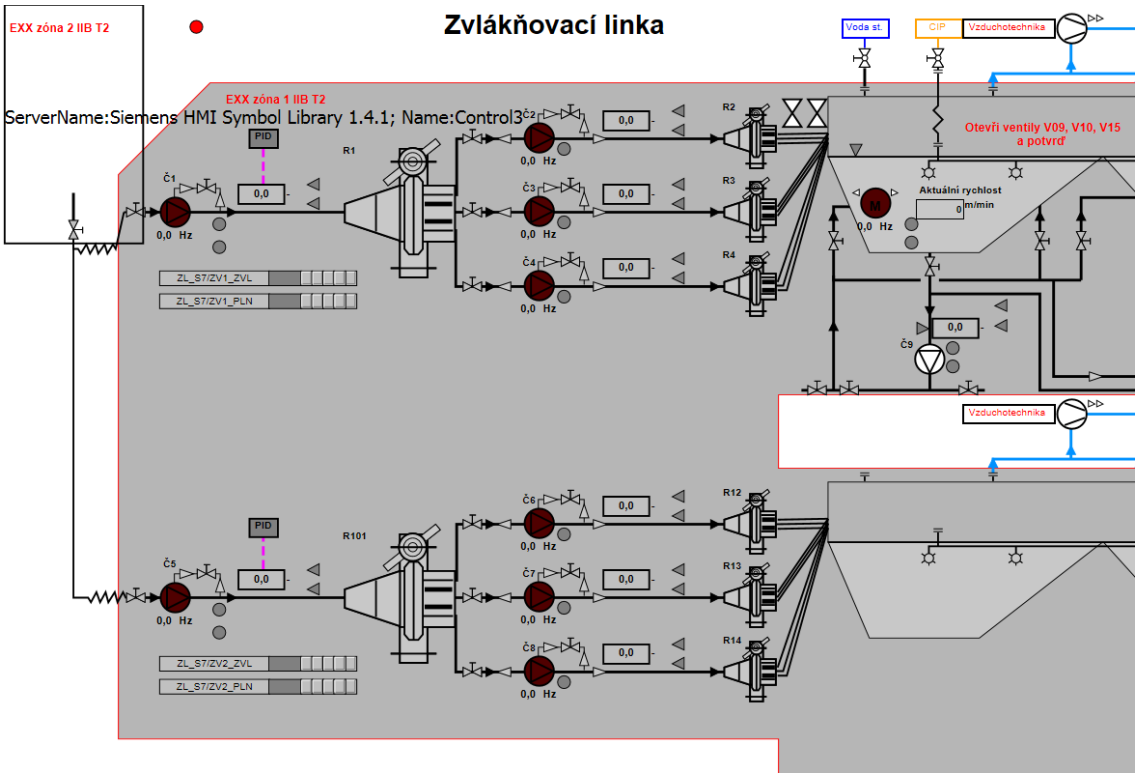
V této kapitole bude vždy uvedena dvojice obrázků vizualizace. Prvním bude původní vzor, a druhým zobrazení v systému COMES po převodu mnou vytvořeným postupem převodu. Je třeba uvést, že po domluvě s firmou Compas nebyla barva pozadí převáděna podle systému WinCC, ale ponechána bílá, jak je běžné v případě systému COMES.

Tato aplikace umožňuje otestování převodu většiny typů základních objektů, které byly podrobně popsány v předchozích kapitolách. Celý projekt obsahuje mnoho obrazovek. Mnoho z nich však nebude do systému COMES přidáno z různých důvodů, uvedených dále. Na následujících dvou obrázcích (Obr. 44 a Obr. 45) je zobrazení vizualizace zvlákňovací linky. Fialově podbarvené značky jsou zobrazeny proto, že je systém v určitém stavu. Na Obr. 45 zobrazeny nejsou, protože systém COMES není napojen na simulaci PLC. Tyto značky se zobrazují u animačních objektů firmy Compas, proto je zde dobře patrné, o které objekty se jedná a jak jsou tyto objekty převáděny.



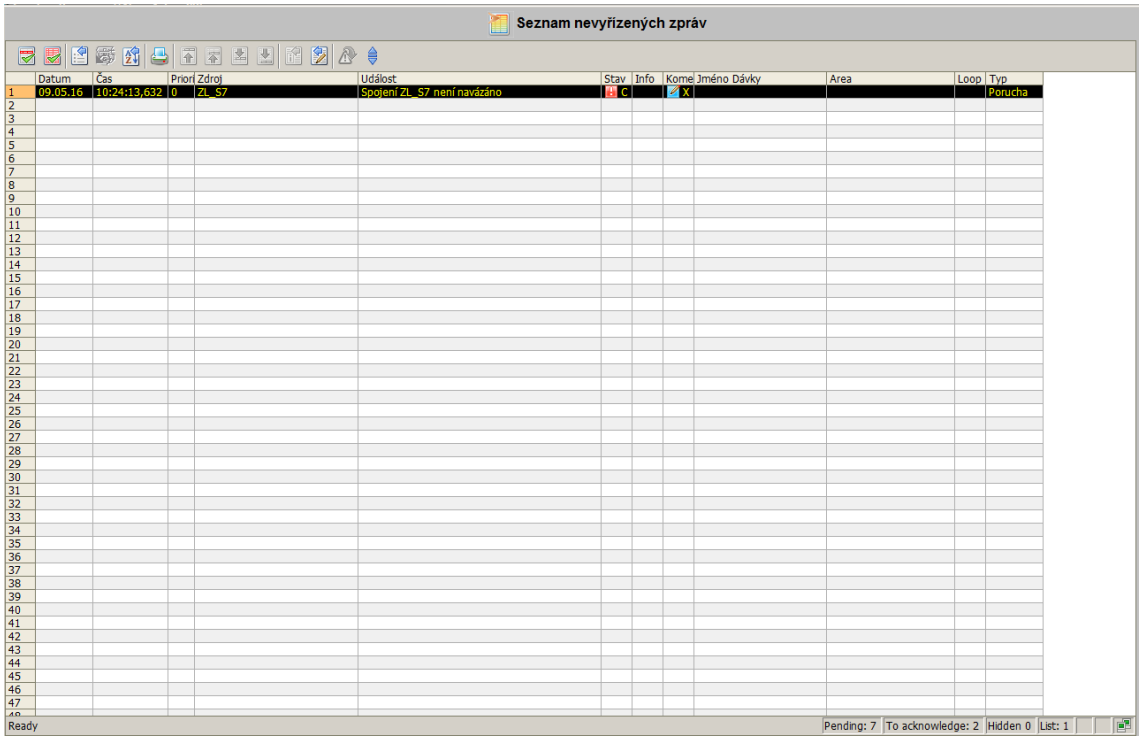
Obr. 44: Část obrazovky pro vizualizaci zvlákňovací linky

Aby se texty zobrazovaly správně, musí být jejich velikost zmenšena o 1 pixel. Bez této úpravy by se v některých případech zobrazily pouze části textů. To je způsobené tím, že velikost textu pro zobrazení v systému COMES se mírně liší od velikosti uvedené pod stejnou velikostí v systému WinCC. Dalším problémem je správa fontů. Stává se, že některé texty jsou psány fontem, který systém COMES a webový prohlížeč nepozná. Proto byl po domluvě s firmou Compas nastaven jeden pevný font a tím je *Arial*.



Obr. 45: Část převedené obrazovky pro vizualizaci zvlákňovací linky. Vlevo nahoře se nachází převedený objekt ActiveX

První typ obrazovek, které nebudou vloženy, jsou například obrazovky zobrazující alarmy a logová hlášení.

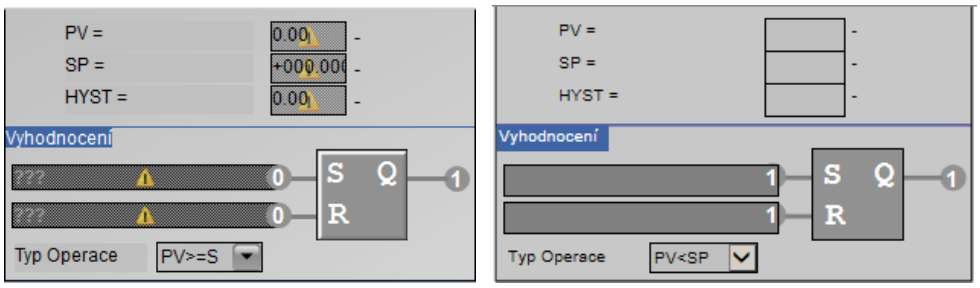


Obr. 46: Obrazovka pro správu alarmů pomocí prvku ActiveX v systému WinCC

Tyto obrazovky jsou v systému COMES řešeny jiným způsobem – systém COMES má vlastní správu alarmových hlášení a logů zabudovanou přímo v sobě a v tomto případě by ani převod nebyl přínosný, protože se jedná o speciální ActiveX objekt systému WinCC, viz Obr. 46 a Obr. 47.

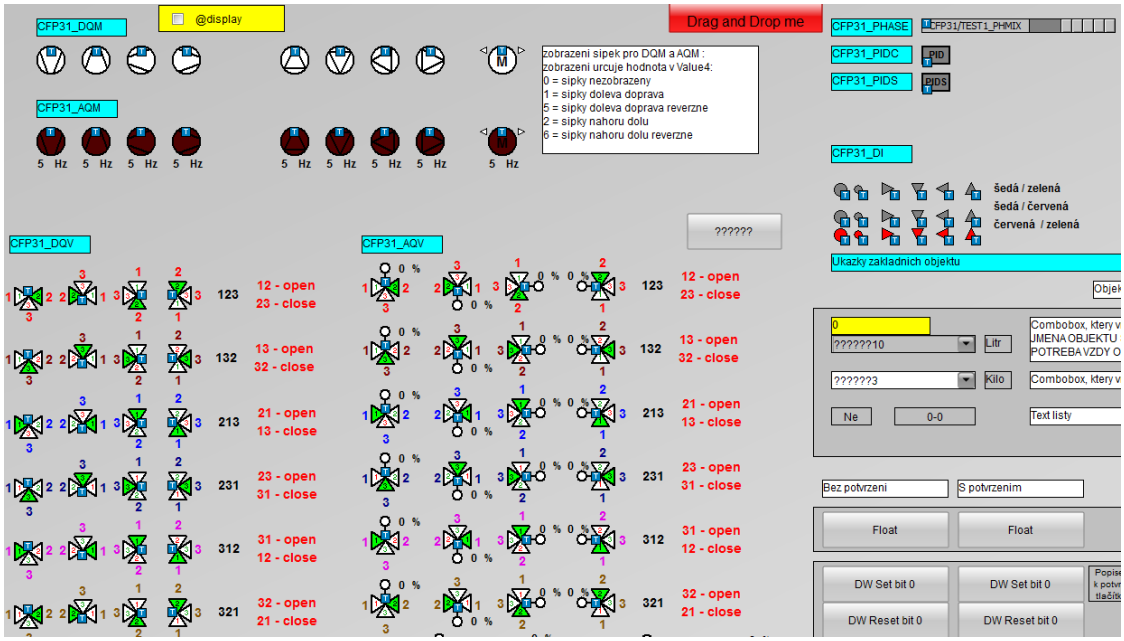
Obr. 47: Obrazovka pro správu alarmů po převedení do systému COMES

Druhým typem obrazovek, které nejspíš nebudou převedeny, jsou zadávací okna, jejichž obsah je v systému COMES častěji předáván přes tabulky vytvořené ve formulářích. Na následujících obrázcích (Obr. 48) je patrné, že výsledné obrazovky se v některých případech mohou lišit. Tyto odlišnosti jsou dané dvěma faktory. Tím prvním je fakt, že elementy ve formátu SVG jsou vkládány na pozadí uživatelských formulářů. Tento postup je zvolen proto, že do SVG se převádí hlavně statické objekty, které nerealizují žádné dynamické změny. Počítá se tedy s jejich umístěním na pozadí. Formát SVG neumožňuje rozvrstvení do vrstev a proto nejvíce na pozadí je prvek, který je vložen jako první do elementu *svg* (podobně jako v případě HTML). Elementy, které jsou poté použity pro ovládání a dynamické změny jsou animační objekty, nebo HTML elementy. Oba typy jsou vkládány v kódu pod element *svg* a jsou tedy zobrazeny navrchu. Druhým důvodem je fakt, že některé objekty v systému WinCC mají při exportu nastaveny rozdílné parametry, než jak se zobrazují při reálném běhu aplikace. Příkladem mohou být podbarvení popisků v obrazovkách na Obr. 48. Vlevo, v případě WinCC, jsou popisky podbarveny šedou barvou o jiném odstínu, než v okně na obrázku vpravo. V tomto případě je důvodem parametr *GlobalColorScheme,* který některým objektům nastaví rozdílné barvy, než jsou definovány v objektu, přesněji řečeno nastaví barevné schéma, definované pro celou aplikaci WinCC v tzv globálním barevném schématu. Toto schéma není pomocí skriptu přístupné, takže není možné požadované barvy nastavit automaticky. Proto je do nastavení mého programu vložena možnost nastavit barvu podbarvení manuálně (viz Obr. 20). V definicích vlastností v grafickém editoru systému WinCC by jim byla přiřazena například bílá barva a převod by se tak značně lišil. Touto úpravou je zajištěno, že se převedené obrazovky budou co možná nejvíce podobat původní vizualizaci.



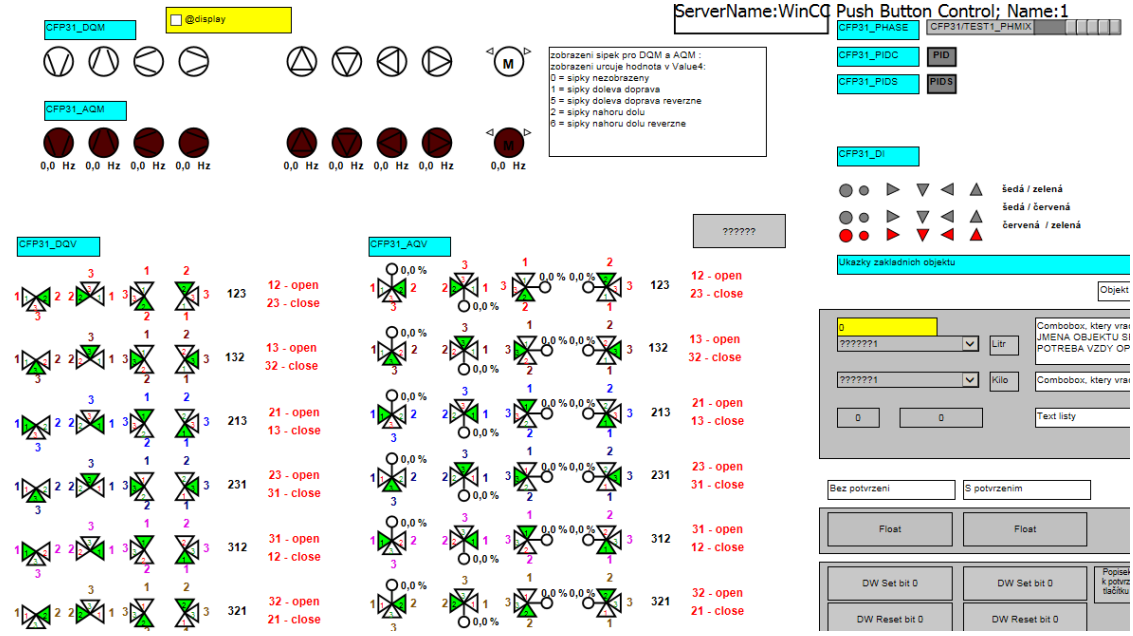
Obr. 48: Vlevo - formulářové okno pro zadání hodnot v systému WinCC, vpravo - stejný formulář, převedený do systému COMES

Posledním typem obrazovek, které nemusí být vloženy, jsou obrazovky, které zprostředkovávají data, nepotřebná pro tuto úroveň řízení. Jak bylo popsáno v předcházejících kapitolách lze si mezi vloženými objekty volit. Pro ilustraci je uvedena ještě jedna dvojice obrázků z okna vizualizace (Obr. 49 a Obr. 50), které slouží pro testování animačních objektů firmy Compas a dalších grafických prvků.



Obr. 49: Okno pro testování animačních a dalších objektů v systému WinCC značky s modrým podbarvením a nápisem T značí propojení systému WinCC se simulací PLC

Na těchto obrazovkách je možné vidět prvky různých typů, od animačních objektů přes tlačítka a vstupní pole po obyčejná popisová pole. Mírné odlišnosti jsou u HTML prvků, které mají definován svůj základní vzhled, což lze vidět například u tlačítek. Úprava na přesný vzhled by způsobila nárůst složitosti kódu při zachování stejné funkčnosti. Proto bylo zobrazení prvků ponecháno v základním stylu a prvky tedy neobsahují barevné přechody. Barvy pozadí, okrajů a textu však převedeny jsou.



Obr. 50: Převedené okno pro testování objektů - zobrazení v systému COMES

8 ZÁVĚR

Tato diplomová práce se zabývá převodem grafických dat ze SCADA systému Siemens Simatic WinCC do systému COMES, který využívá webových technologií. Práce obsahuje 4 hlavní části. První část se týká představení těchto průmyslových systémů. Zde bylo největším problémem získat dostatek informací o částech obou systémů, které jsou využívány pro vypracování této práce a musely tak být rozebrány více do hloubky. Díky tomu, že jsem měl oba systémy k dispozici, mohl jsem vypsané vlastnosti zjistit experimentálně, případně nalezené informace ověřit.

Druhá část práce se zabývá rozborem možností vizualizace průmyslové technologie pomocí webových technologií. Byly rozebrány aktuálně používané technologie a jejich vlastnosti. Při studování dané problematiky jsem došel k závěru, že pro průmyslové využití bude nejlepší formát SVG z důvodů blíže popsaných v páté kapitole. Lze zmínit hlavně dobrou přístupnost jednotlivých grafických elementů tohoto formátu při přístupu pomocí skriptů. Dalším důvodem je také široká podpora, velká pravděpodobnost rozvoje tohoto formátu v budoucnosti. Tuto část práce se dařilo zpracovávat velice dobře díky velkému množství volně dostupných informací a ukázkových aplikací ke všem uvažovaným formátům.

Další částí práce byla tvorba skriptu, sloužícího pro export vizualizačních obrazovek do formátu XML. Byl vytvořen skript, který obsahuje data a vlastnosti jednotlivých objektů v obrazovkách. Pro každou obrazovku vytvoří tento skript soubor s vlastnostmi. Výhodou mého skriptu je možnost exportu pouze aktivní obrazovky nebo celého projektu. Při exportu se nepodařilo vyexportovat data obsahující geometrické parametry a složení objektů z knihovny HMI symbol library a dalších objektů typu ActiveX s výjimkou animačních objektů firmy Compas, které mají definici v externích, snadno přístupných souborech. Tento problém byl po konzultacích se zástupci firmy Compas vyřešen tak, že data z této knihovny budou ručně přidávána do výsledného souboru v systému COMES s tím, že z exportu bude možné využít obecné parametry, jako například velikost grafického objektu, jeho umístění a navázání na konkrétní tagy. U některých objektů byl problém najít požadované parametry. Zde musím podotknout, že práce se systémem WinCC byla při tvorbě skriptu relativně zdlouhavá, protože kromě oficiální příručky pro psaní skriptů neexistuje mnoho podkladů a tato příručka neobsahuje mnoho příkladů použití. Práce s tímto systémem také není zcela intuitivní, proto byl hlavně zpočátku problém s vytvořením alespoň částečně využitelného exportu.

Poslední část práce se týkala vytvoření nástroje, který by umožňoval vyexportované XML soubory dále zpracovat a tato zpracovaná data vložit do systému COMES, kde by sloužila jako jedna z hlavních součástí pro vytvoření uživatelské aplikace. Pro tvorbu jsem zvolil programovací jazyk C#, který má pro takovéto využití široké možnosti a je dobře znám i zástupcům firmy Compas, kteří budou mnou vytvořenou aplikaci reálně využívat. Tato část byla z celé práce časově nejzdlouhavější a nejobsáhlejší. Vzhledem k velkému množství typů objektů a jejich možnému převodu vzniklo při vypracovávání

několik hlavních problémů. Prvním problémem byl již zmíněný převod objektů typu ActiveX, popsaný výše.

Dalším problémem byl převod eliptických a kruhových výsečí z důvodu velice odlišné definice v obou využívaných průmyslových systémech. Tento problém se podařilo vyřešit a prakticky otestovat funkčnost převodu.

Jedním z velkých problémů při tvorbě byla nesourodost vlastností objektů, vyexportovaných ze systému WinCC. Pro příklad lze uvést objekty bargrafů. V případě 2D objektu jsou limity nastavovány jako vlastnosti objektu, které mají stejnou funkci, jako v případě vlastností 3D grafu. V případě 3D grafu však mají odlišný název a bylo tedy nutné vytvořit zpracování pro každý objekt zvlášť. Tyto rozdíly se vyskytovaly v mnoha případech a přesto, že byly vyřešeny, způsobily velký nárůst složitosti kódu aplikace.

Posledním problémem je převod objektů typu HMICustomizedObject, které tvoří seskupení jednoduchých objektů do pokročilejší skupiny s definovanými vlastnostmi. Tyto objekty jsou převedeny na Animaci v systému COMES. Nepodařilo se však najít spojitost mezi zastřešenými objekty a vnějšími parametry, tedy jejich přiřazení k jednotlivým objektům. Jsou proto vygenerovány objekty Animace bez vnitřního výkonného kódu a rozhraní. Je proto nutné, aby programátor zákaznické aplikace funkčnosti tohoto objektu upravil podle potřeby.

Je třeba zmínit, že mnoho parametrů nemohlo být do systému COMES převedeno. Jako příklad lze jmenovat vyplňování objektů výplní, jejíž úroveň lze měnit dynamicky. Ve formátu SVG je tato možnost omezena a pro rozšíření funkcí do takové podoby, aby bylo možné objektům nastavovat požadovaný vzhled shodně, jako v systému WinCC, musel by být pro každý objekt vytvořen v systému COMES objekt Animace, obsahující několik objektů, realizujících tvorbu výplně a rozsáhlý výkonný kód, realizující tuto funkci. Protože však tato vlastnost a mnoho dalších nejsou v praxi firmou Compas často využívány, nejsou po dohodě převedeny. Většinu takových objektů zastávají animační objekty zmíněné firmy, které převáděny jsou.

Aplikace je schopna převést objekty, jmenované v této práci a její funkčnost a možnost použití byla testována převodem dat z testovací aplikace, jak je patrné z kapitoly 7 této práce. Převedené objekty mohou obsahovat drobné odlišnosti, jejich podstata je však zachována. Praktické nasazení aplikace je tedy možné a bude realizováno po odevzdání této práce přímo firmou Compas, pro kterou byl tento nástroj vyvinut.

Literatura

[1] VML Tutorial. *Gary Beene's Information Center* [online]. 2015 [cit. 2015-11-29]. Dostupné z: http://www.garybeene.com/3d/3d-vmltut.htm

[2] Vector Markup Language (VML): World Wide Web Consortium Note 13-May-1998. *World Wide Web Consortium (W3C)* [online]. 1996, 2015 [cit. 2015-11-29]. Dostupné z: http://www.w3.org/TR/1998/NOTE-VML-19980513

[3] Scalable Vector Graphics (SVG) 1.1 (Second Edition): W3C Recommendation 16 August 2011. *World Wide Web Consortium (W3C)*[online]. 1996, 2015 [cit. 2015-11-31]. Dostupné z: http://www.w3.org/TR/SVG/

[4] HEROUT, Pavel. *XSLT 2.0 a SVG: XPath a Java prakticky*. První vydání. České Budějovice: Kopp, 2010. ISBN 978-80-7232-406-4.

[5] FULTON, Steve a Jeff FULTON. *HTML 5 Canvas*. 2nd edition. Sebastopol: O'REILLY, 2013. ISBN 978-1-449-33498-7.

[6] JOHNSON, Glenn. *Programming in HTML5 with JavaScript andCSS3: Training Guide*. Second Printing. USA: Microsoft Press, 2013. ISBN 978-0-7356-7438-7.

[7] BRÁZDA, Roman. COMPAS AUTOMATIZACE SPOL. S R.O.*Výrobní informační systém COMES verze 3: Úvod do systému*. Žďár nad Sázavou, 2009.

[8] BRÁZDA, Roman. COMPAS AUTOMATIZACE SPOL. S R.O.*Výrobní informační systém COMES verze 3: Modul Logon*. Žďár nad Sázavou, 2009.

[9] BRÁZDA, Roman. COMPAS AUTOMATIZACE SPOL. S R.O.*Výrobní informační systém COMES verze 3: Modul CCI*. Žďár nad Sázavou, 2009.

[10] BRÁZDA, Roman. COMPAS AUTOMATIZACE SPOL. S R.O.*Výrobní informační systém COMES verze 3: Modul Historian*. Žďár nad Sázavou, 2009.

[11] BRÁZDA, Roman. COMPAS AUTOMATIZACE SPOL. S R.O.*Výrobní informační systém COMES verze 3: Modul Modeller*. Žďár nad Sázavou, 2009.

[12] KÁRNÍK, Jiří. *Sběr dat z PLC do systému MES*. Brno, 2013, 54 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Jan Pásek, CSc.

[13] SIMATIC WinCC Options: HMI Software - Siemens. *Siemens Global Website* [online]. 1996, 2015 [cit. 2015-10-31]. Dostupné z: http://w3.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/wincc-options/Pages/Default.aspx

[14] SIEMENS AG. *SIMATIC WinCC: Process visualization with Plant Intelligence*. Postfach, Germany, 2012. Dostupné také z: https://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-wincc_en.pdf

[15] SIEMENS AG. *Simatic HMI- WinCC V7.2 SIMATIC HMI WinCC V7.2 System description: System Manual*. 2013. Dostupné také z: https://support.industry.siemens.com/cs/attachments/99730295/WinCCSystemDescription_en-US.pdf?download=true

[16] SIEMENS AG. *Simatic HMI - WinCC V7.2 WinCC: Scripting (VBS, ANSI-C, VBA): System Manual*. 2013. Dostupné také z: https://support.industry.siemens.com/cs/attachments/99730295/WinCCSystemDescription_en-US.pdf?download=true

[17] KAMENSKÝ, Pavel. COMPAS AUTOMATIZACE, SPOL. S R.O.*ActiveX pro jednoduché animace (nejen) pro WinCC: NÁVOD K POUŽÍVÁNÍ*. Žďár nad Sázavou, 2006. TT_PCS7_10_AnimObj.doc

[18] Document Object Model (DOM). *World Wide Web Consortium (W3C)* [online]. 1997, 2015 [cit. 2015-12-01]. Dostupné z: http://www.w3.org/DOM/#what

[19] Document Object Model. *Wikipedia: the free encyclopedia*[online]. San Francisco (CA): Wikimedia Foundation, 2001-2015, 8.2.2014 [cit. 2015-12-01]. Dostupné z: https://cs.wikipedia.org/wiki/Document_Object_Model

[20] GREEFF, Gerhard a Ranjan GHOSHAL. *Practical E-manufacturing and supply chain management*. London: Newnes, 2004, xi, 461 p.

[21] JavaScript HTML DOM: The HTML DOM (Document Object Model). *W3Schools.com: THE WORLD'S LARGEST WEB DEVELOPER SITE* [online]. Copyright 1999-2015 [cit. 2015-12-27]. Dostupné z: http://www.w3schools.com/js/js_htmldom.asp

[22] COMPAS AUTOMATIZACE SPOL. S R.O.*Výrobní informační systém COMES verze 3: COMES: Úvod do systému*. Žďár nad Sázavou, 2011. 12 s.

[23] COMPAS AUTOMATIZACE SPOL. S R.O. *Výrobní informační systém COMES verze 3: COMES Modeller: Editor skriptů*. Žďár nad Sázavou, 2011. 56 s.

[24] COMPAS AUTOMATIZACE SPOL. S R.O. *Výrobní informační systém COMES verze 3: COMES Modeller: Příručka pro nastavení*. Žďár nad Sázavou, 2011. 49 s.

[25] Ganzheitliche Strategie: Systems Lifecycle Management. PLM Portal [online]. München: Impressum, 2012 [cit. 2016-05-02]. Dostupné z: http://www.plmportal.org/de/forschung-detail/ganzheitliche-strategie-systems-lifecycle-management-syslm.html

[26] SHARP, John. *Microsoft Visual C# 2010: krok za krokem*. Brno: Computer Press, 2010. Krok za krokem (Computer Press). ISBN 9788025131473.

Seznam zkratek

PCS – process control systems (systémy procesního řízení)
HMI – human machine interface (rozhraní mezi člověkem a strojem)
SCADA – supervizory control and data acquistion (supervizní řízení a sběr dat)
MES – manufacturing execution systém (výrobní informační systém)
ERP – enterprise resource planning (plánování podnikových zdrojů)
SCM – supply chain management (řízení dodavatelského řetězce)
PLM – product lifetime management (řízení životního cyklu výrobku)
XML – extensible markup language (rozšiřitelný značkovací jazyk)
VML – vector markup language (vektorový značkovací jazyk)
SVG – scalable vector graphics (škálovatelná vektorová grafika)
HTML – hypertext markup language (hypertextový značkovací jazyk)
DOM – document object model (objektový model dokumentu)
CSS – cascading style sheet (kaskádové styly)
GEP – good engineering practice (ověřené a uznávané inženýrské technické činnosti)
GAMP – good automated manufacturing practice (soubor pokynů pro výrobce ve farmaceutickém průmyslu)
OEE – overall equipment effectiveness (celková efektivnost zařízení)
MTFF – mean time to first failure (střední doba do první poruchy)
MTBF – mean time between failures (střední doba mezi poruchami)
CSV – comma-separated values (hodnoty oddělené čárkami – formát souboru)
SQL – structured query language (strukturovaný dotazovací jazyk)
HTTPS – hypertext transfer protocol secure (zabezpečený internetový protokol pro výměnu hypertextových dokumentů ve formátu HTML)
SES – sequence execution system (nástroj pro řízení sekvenčních procesů)
ODK – Open Development Kit

Seznam příloh

Příloha 1. DVD - obsahuje tyto části:

- Zdrojový kód převodního skriptu pro systém WinCC
- Příklad exportu ze systému WinCC do XML
- Příklad vytvořeného exportního balíčku
- Vzorové soubory definice animačního objektu
- Zdrojové kódy převodního programu pro převod grafických dat z XML do exportního balíčku
- Elektronickou verzi práce
- Dokumentaci k programu