OPC server pro úlohy automatické regulace.

Bc. Jaroslav Gajzler

Diplomová práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky
akademický rok: 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jaroslav GAJZLER**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Automatické řízení a informatika**
Téma práce:	**OCP server pro úlohy automatické regulace.**

Zásady pro vypracování:

1. **proveďte analýzu zadání,**
2. **zpracujte literární rešerši o stavu řešení,**
3. **popište strukturu výměny dat s OPC serverem obecně,**
4. **navrhněte propojení IPC a PLC pomocí OPC řešení,**
5. **proveďte otestování a ověření v simulačním režimu,**
7. **zpracujte vyhodnocení výsledků měření.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

HRUŠKA, F.: Technické prostředky automatizace IV. Snímače, převodníky, regulátory, průmyslová výpočetní technika, ovládací jednotky. UTB ve Zlíně, FT, září 2001, s. 107. ISBN 80-7318-026-X.
HRUŠKA, F.: Projektování systémů integrované automatizace. Učební texty. 2. vyd. Zlín: UTB ve Zlíně, 2002, s. 133. ISBN 80-7318-100-2.
Firemní literatura firmy SIEMENS, A.G..
Firemní literatura firmy Moravské přístroje, a.s. Zlín, Malenovice.
www.matrikon.com/drivers/opc

Vedoucí diplomové práce: **doc. Ing. František Hruška, Ph.D.**
Ústav elektrotechniky a měření

Datum zadání diplomové práce: **14. února 2006**

Termín odevzdání diplomové práce: **26. května 2006**

Ve Zlíně dne 14. února 2006

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá technologií OPC, existujícími specifikacemi, konfigurací a použitím této technologie v praktickém nasazení. Jsou zde popsány základy nastavení komunikace mezi PLC a IPC pomocí OPC. Získané znalosti jsou použity na praktickém příkladě. Je vytvořeno jednoduché SCADA/HMI rozhraní pro regulaci teploty na laboratorní úloze.

Klíčová slova: OPC, server, klient, SCADA/HMI, COM, OPC specifikace, Control WEB

ABSTRACT

In my Final Thesis I deal with the issue of OPC technology, its existing specifications and configuration and further I examine the problem of its practical usage. I have described the fundamental principles for the setup of OPC communication between PLC and IPC .The acquired knowledge I demonstrate by a practical example. I have created a simple SCADA/HMI interface for temperature regulation through laboratory work.

Keywords: OPC, server, client, SCADA/HMI, COM, OPC specifications, Control WEB

Na tomto místě bych rád poděkoval vedoucímu mé diplomové práce Doc. Ing. Františkovi Hruškovi, Ph.D. za jeho odborné rady a připomínky. Můj dík patří taky Ing. Seemanovi za odbornou konzultaci při vývoji aplikace v prostředí Control WEB.

OBSAH

ÚVOD **9**
I TEORETICKÁ ČÁST **10**
1 ANALÝZA ZADÁNÍ **11**
2 LITERÁRNÍ REŠERŠE **12**
3 ZÁKLADNÍ POPIS OPC **13**
3.1 PŘEDSTAVENÍ OPC 13
3.2 HISTORIE VZNIKU OPC 13
3.3 VÝHODY OPC 14
3.4 STANDARD OPC, SPECIFIKACE OPC, OPC FOUNDATION 15
3.5 EXISTUJÍCÍ SPECIFIKACE OPC 16
3.6 ÚSPĚŠNÁ HISTORIE OPC 17
3.7 OPC XML DA 17
3.8 ARCHITEKTURA OPC 18
3.9 PLATFORMY POUŽITELNÉ PRO OPC 19
4 MODEL COM **20**
4.1 HISTORIE COM 20
4.2 COM – ZÁKLADNÍ POPIS 22
4.3 ZÁKLADNÍ ARCHITEKTURA KOMPONENT COM 24
4.4 ZÁKLADNÍ PRAVIDLA PRO IMPLEMENTACI QUERYINTERFACE() 27
4.5 ŽIVOTNÍ CYKLUS KOMPONENTY 27
4.6 DYNAMICKÉ LINKOVÁNÍ 28
4.7 HRESULT 29
4.8 GUID, IID, CLSID, PROGID APOD. 29
4.9 REGISTR SYSTÉMU WINDOWS 30
4.10 BUDOUCNOST TECHNOLOGIE COM 30
5 OPC SPECIFIKACE **32**
5.1 OVERVIEW 33
5.2 OPC COMMON DEFINITIONS AND INTERFACES 35
5.2.1 Vyžadované definice rozhraní 36
5.2.2 Chyby a vrácené kódy. 37
5.2.3 Vypínání OPC serveru. 37
5.2.4 IOPCCommon 37
5.2.5 OPC server prohlížeč 37

5.3 OPC DATA ACCESS (DA) 37
5.3.1 Automatizační rozhraní 43
5.4 OPC ALARMS AND EVENTS (EA) 43
5.4.1 Automatizační rozhraní 46
5.5 OPC HISTORICAL DATA ACCESS (HDA) 47
5.5.1 Čtení hodnot 49
5.5.2 Aktualizování hodnot 50
5.5.3 Záznam hodnot 51
5.5.4 Automatizační rozhraní 52
5.6 OPC BATCH 53
5.7 OPC SECURITY 54
5.8 OPC DATA EXCHANGE (DX) 54
5.9 OPC XML DA 54
II PRAKTICKÁ ČÁST 56
6 PROPOJENÍ IPC S PLC POMOCÍ OPC 57
6.1 FYZICKÉ PROPOJENÍ 57
6.2 NASTAVENÍ KOMUNIKACE MEZI PLC A OPC SERVEREM OPC SIMATIC.NET 58
6.2.1 Instalace Simatic.NET 58
6.2.2 Konfigurace OPC serveru Simatic.NET 60
6.3 PŘENOS DAT MEZI OPC SERVEREM A KLIENTEM CONTROL WEB 64
6.3.1 Konfigurační soubory OPC ovladače 65
6.3.1.1 Mapovací soubor 66
6.3.1.2 Parametrický soubor 67
6.3.2 Konfigurační nástroj OPC ovladače 69
6.3.2.1 Výběr OPC serveru 70
6.3.2.2 Výběr kanálů 72
6.3.2.3 Přiřazení čísel kanálů 72
6.3.2.4 Přiřazení čísel polím kanálů 73
6.3.2.5 Parametry ovladače 73
6.3.2.6 Generování konfiguračních souborů 74
6.3.2.7 Načtení parametrického souboru 74
7 OVĚŘENÍ A OTESTOVÁNÍ V SIMULAČNÍM REŽIMU 75
7.1 APLIKACE CONTROL WEB 75
7.2 POPIS ÚLOHY DE1 – REGULACE TEPLOTY 78
7.3 ÚLOHA DE1- REGULACE TEPLOTY V PROSTŘEDÍ CONTROL WEB 80
8 VYHODNOCENÍ VÝSLEDKŮ 83
ZÁVĚR 84
SEZNAM POUŽITÉ LITERATURY 85
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 87
SEZNAM OBRÁZKŮ 88

SEZNAM TABULEK 90
SEZNAM PŘÍLOH 91

ÚVOD

Změny v průmyslovém řízení a automatizaci se zrychlují. Zvyšuje se požadavek na přesnost strojů, na jejich flexibilitu při změně technologie, na rychlost výroby a na kontrolu chyb, stejně tak jako se zvyšuje tlak na snižování nákladů. Řídící software se stává stále více důležitým faktorem ve výrobcích procesech, systémech a celkově ve firemní infrastruktuře.

Zároveň jsou velmi patrné změny v oblasti automatizace vzniklé používáním výpočetní techniky jako součást automatizační a řídící techniky, používání síťového prostředí, internetu a také vzrůstající tendence k používání více otevřených standardům. Tyto změny jsou přínosem jak pro uživatele tak pro výrobce.

Počítače jsou ve velké míře používány k vizualizaci, k převádění dat, ke kontrole procesů a k řešení dalších úkolů v automatizaci. Počítače také doplňují nebo odstraňují tradiční PLC řízení a provozní terminály. Důvodem pro tyto změny je trvající pokles ceny počítačů, zvyšující se výkon a používání více efektivního a uživatelsky přívětivého softwaru.

OPC je v současné době obecně akceptovaný jako jeden z továrních standardů mezi uživateli a také mezi firemními vývojáři. Většina ze SCADA/HMI výrobců v oboru počítačové automatizace, a také z oblasti výrobců zařízení pro automatizaci nabízejí se svými produkty OPC klienty nebo OPC serverové rozhraní. Dnešní době představuje OPC standardní rozhraní mezi automatizační technologií a aplikacemi vytvořenými v prostředí Windows. Standard OPC je založen na technologii COM, což je komponentově orientovaná technologie od firmy Microsoft a na technologii DCOM, což je obdoba technologie COM se síťovým rozšířením.

Zavádění OPC standardů přináší výrazné výhody jako jsou snížení nákladů na školení, snížení nákladů na uživatelské nastavení a snížení nákladů na údržbu. OPC je otevřeným standardem na který vytváří společnost OPC Foundation a která vydává specifikace týkající se jednotlivých částí komunikace prostřednictvím OPC. Tyto specifikace jsou volně dostupné pro všechny vývojáře a stanovují co všechno je nutné při komunikaci dodržovat, aby nevznikali nekompatibilní aplikace. To přináší do světa automatizace volnost a komponentovou nezávislost na jednotlivých výrobcích automatizační techniky a také v neposlední řadě výrazný ekonomický přínos pro koncové uživatele.

I. TEORETICKÁ ČÁST

1 ANALÝZA ZADÁNÍ

Tato práce si klade za cíl rozebrat tématiku OPC řešení komunikace, popsat teoretickou část této problematiky a prakticky ověřit zkušenosti získané v teoretické části.

První část je teoretická a popisuje současný stav a dostupnost technické literatury vztahující se k problematice OPC řešení. Následuje úvod do problematiky OPC, základní popis struktury výměny dat, historický vývoj, možnosti použití a reálného nasazení této technologie. V této kapitole lze získat základní informace co to OPC vlastně je a k čemu slouží. V následující části je popsána technologie COM na které je OPC řešení postaveno. Dále jsou popsány vlastnosti, principy, historický vývoj a vyhlídky do budoucnosti této technologie. Jako poslední v teoretické části jsou rozebrány jednotlivé specifikace standardu OPC tak, jak je vydává organizace OPC Foundation. Tyto specifikace popisují vnitřní fungování OPC technologie a slouží výrobcům jako referenční standard této technologie, který je nutno dodržet pro získání certifikace OPC Foundation.

Praktická část se zaměřuje na vyzkoušení a ověření teoreticky nabytých znalostí. V první části (tj. bod 6.) je detailně popsaný návrh propojení IPC nebo PC s PLC prostřednictvím OPC. V několika bodech je zde uveden návod na vytvoření komunikace mezi PLC a OPC serverem a taky na komunikaci mezi OPC serverem a OPC klientem. Jako OPC klient je použita aplikace Control WEB. Praktická část dále obsahuje kapitolu ověření a otestování v simulačním režimu, kde je popsané prostředí Control WEB a otestovaná aplikace vytvořená v tomto prostředí, která je fyzicky napojená na úlohu DE1 – regulace teploty. Tato úloha je vytvořená v rámci projektu Laboratoře Integrované automatizace. V poslední části je zpracované vyhodnocení výsledků.

2 LITERÁRNÍ REŠERŠE

Vypracováním standardu OPC, jeho udržováním, šířením a prezentací se zabývá mezinárodní dobrovolná organizace OPC Foundation (http://www.opcfoundation.org), se sídlem v Scottsdale, Arizona, USA. Tato organizace vznikla v roce 1996 a v současné době sdružuje více než 300 členů z řad nejvýznamnějších světových firem věnujících se monitorování, vizualizaci a dalším aplikacím z oblasti řízení a sledování technologických procesů.

Tyto volně dostupné specifikace jsou vydávané tak, jak se vyvíjí standard OPC a jsou zdarma dostupné na internetu. Představují vlastně normu standardu OPC, která definuje chování, vlastnosti, strukturu a funkce OPC. V současné době jsou dostupné následující specifikace: OPC Overview v.1.0, OPC Common Definitions and Interfaces v.1.10, OPC Data Access (OPC DA) v. 3.0, OPC Alarms and Events v. 1.1, OPC Historical Data Access v. 1.2, OPC Batch v. 2.0, OPC Security v. 1.0, OPC XML DA v. 1.0, OPC Data eXchange v. 1.0, OPC Complex Data v. 1.0.

Další dostupnou literaturu představují jednotlivé firemní materiály firem jako je například : Siemens, Matricon, Reliance, Moravské přístroje atd.. Tyto firmy jsou také většinou členy organizace OPC Foudation. Tato literatura představuje ve většině případů popis konkrétních produktů a jejich nastavení od jednotlivých výrobců s úvodní teoretickou částí, která je převzatá ze specifikací organizace OPC Foundation, která celou technologii OPC zastřešuje.

3 ZÁKLADNÍ POPIS OPC

V současné době se při přenosech dat v průmyslových řídicích systémech stále častěji používá standard s označením OPC. Následující řádky si kladou za cíl čtenářům blíže seznámit s tuto technologii a obeznámit je se základními principy OPC komunikace.

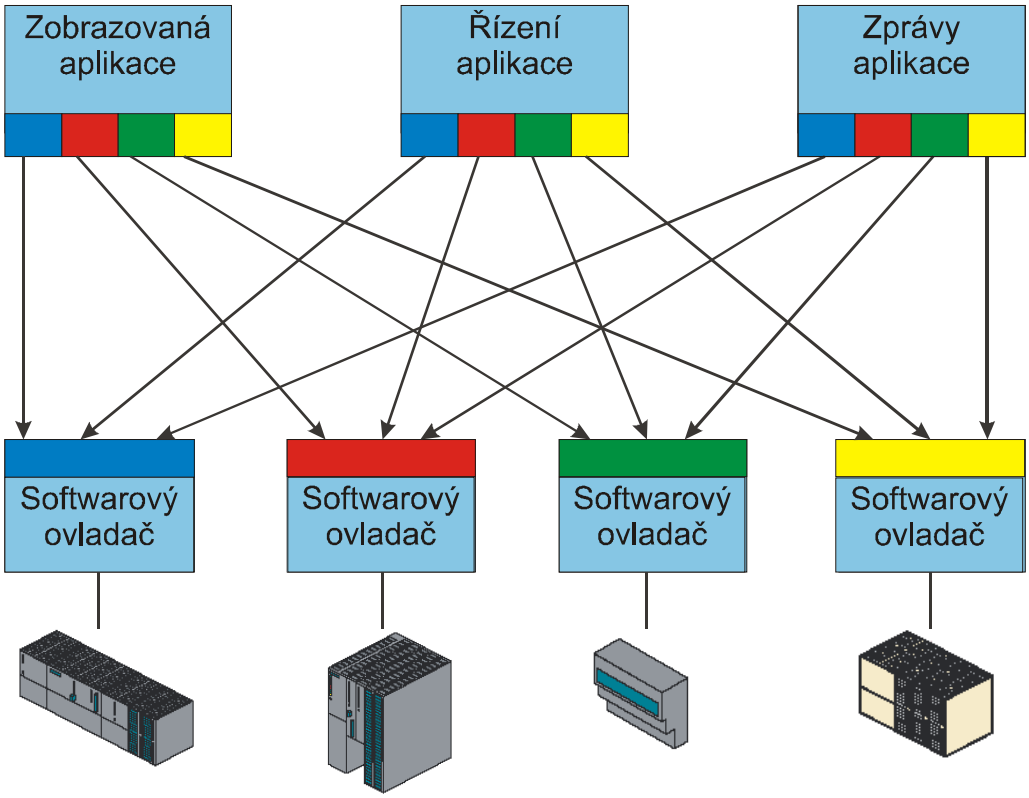
3.1 Představení OPC

OPC (OLE for Process Control) je standard průmyslové komunikace, vytvořený ve spolupráci mnoha světových dodavatelů automatizačních prostředků, hardwaru a softwaru, v oblasti automatizace a společnosti Microsoft. Je společným rozhraním pro vzájemnou komunikaci mezi různými zařízeními určenými pro monitorování a řízení technologických procesů. Jeho úkolem je zabránit závislosti daného monitorovacího nebo řídicího softwaru na výrobci hardwaru. Standard OPC je založen na metodách OLE/COM/DCOM (Object Linking and Embedding/Common Object Model/Distributed COM) společnosti Microsoft.

3.2 Historie vzniku OPC

Existuje mnoho klientských aplikačních programů, které byly vyvinuty s cílem získávat data z různých zdrojů. Přístup daného softwaru k datům je tradičně zajišťován prostřednictvím nezávisle vyvinutých ovladačů (driver). Tato metoda vede k následujícím problémům (tzv. I/O driver problem):

- každá aplikace musí obsahovat ovladač pro konkrétní hardware, který využívá (obr. 1),
- dochází k neshodám mezi ovladači od různých dodavatelů, když ne všichni dodavatelé (všechny ovladače) podporují všechny vlastnosti daného hardwaru,
- změna vlastností hardwaru je příčinou nefunkčnosti některého ovladače,
- vznikají konflikty v přístupu k hardwaru: dva různé programové balíky nemohou sdílet zařízení, pokud každý z nich neobsahuje nezávislý ovladač.

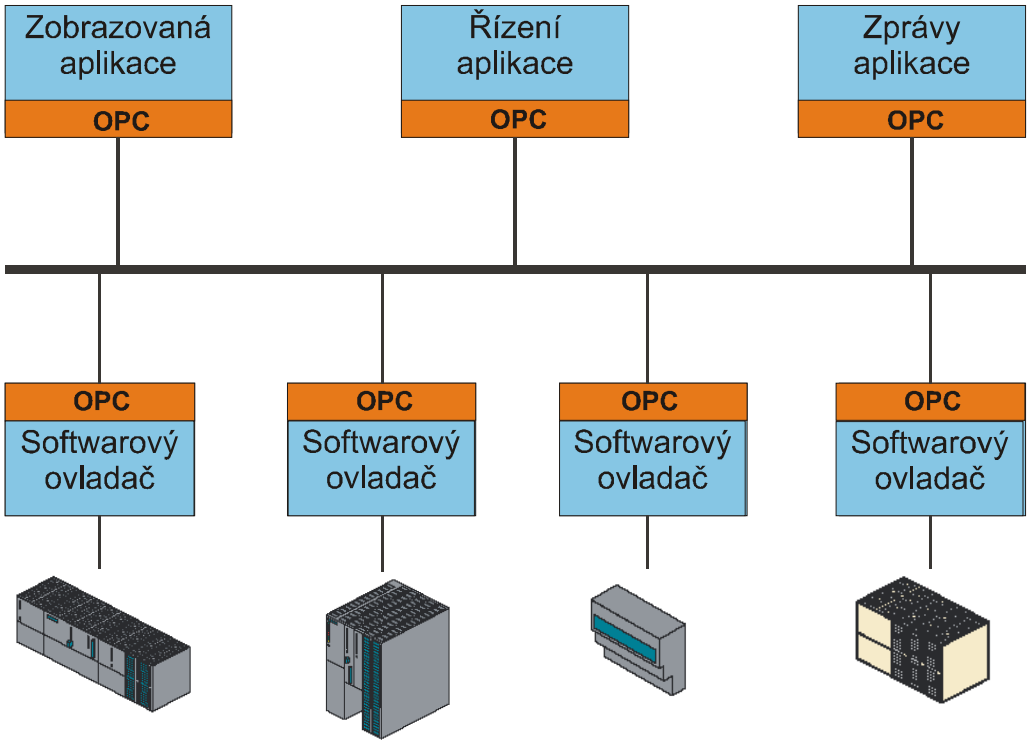


Obr. 1 Komunikace bez OPC

Mnozí výrobci hardwaru se snaží vyřešit tyto problémy vývojem dalších ovladačů, brání jim však odlišnosti v klientských protokolech. Rozdíly v protokolech jsou často důsledkem konkurenčního boje. Standard OPC vytváří mezi výrobci hardwaru a dodavateli softwaru dělicí čáru a poskytuje mechanismus umožňující získávat data z různých zdrojů a přenášet je do libovolného klientského programu nezávisle na dodavateli hardwaru. Všichni dodavatelé pak mohou vyvinout výkonově optimalizovaný server pro komunikaci se zdrojem dat.

3.3 Výhody OPC

Hlavním cílem standardizační iniciativy bylo umožnit klientským aplikačním programům konzistentní přístup k datům v technologických provozech. Současné přínosy postupného zavádění standardu OPC jsou:



Obr. 2 Komunikace přes OPC

výrobci hardwaru vystačí s jedním souborem softwarových komponent pro všechny zákazníky a jejich aplikace,

vývojáři softwaru nepotřebují psát stále nové ovladače kvůli změnám a novým vlastnostem hardwaru v jeho nových verzích,

zákazníci mají svobodu volby mezi dodavateli různých součástí a zařízení nejen pro vývoj špičkových integrovaných technologických celků, ale i pro integraci sledování a řízení technologického zařízení na celozávodní a celopodnikové úrovni,

rozhraní OPC se ve stále větší míře stává standardním rozhraním moderních programových produktů pro sledování a řízení technologických procesů, strojů a zařízení (Supervisory Control and Data Acquisition/Human-Machine Interface – SCADA/HMI), modulů programovatelných automatů a ostatních systémů (většinou již zahrnutých v ceně systému).

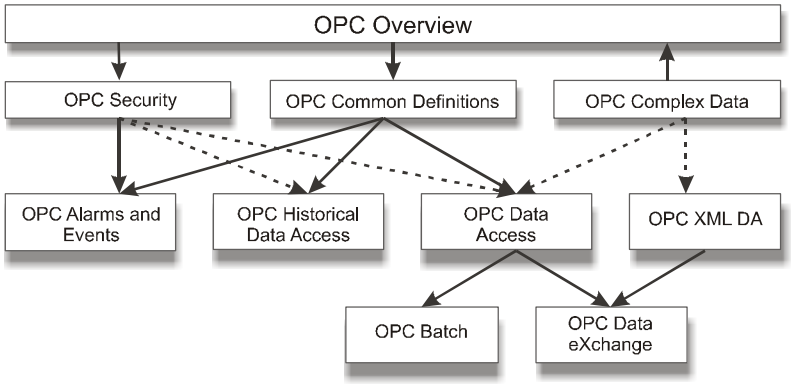
3.4 Standard OPC, specifikace OPC, OPC Foundation

Vypracováním standardu OPC, jeho udržováním, šířením a prezentací se zabývá mezinárodní dobrovolná organizace OPC Foundation (http://www.opcfoundation.org), se sídlem v Scottsdale, Arizona, USA. Tato organizace vznikla v roce 1996 a v současné době sdružuje

více než 300 členů z řad nejvýznamnějších světových firem věnujících se monitorování, vizualizaci a dalším aplikacím z oblasti řízení a sledování technologických procesů, jako např. Honeywell, Rockwell Software, Siemens, Intellution aj. Členem OPC Foundation je i společnost Microsoft, která se aktivně podílí na tvorbě nových specifikací. Z českých firem jsou členy OPC Foundation společnosti Merz (Liberec), OPC Labs (Plzeň) a Geovap (Pardubice).

3.5 Existující specifikace OPC

Standard OPC je vytvářen a udržován prostřednictvím tzv. specifikací OPC. Specifikace OPC je volně přístupná technická dokumentace stanovující pravidla, kterými se řídí chování a konfigurace standardního rozhraní OPC.



Obr. 3 Specifikace OPC a jejich vazby

Současné specifikace OPC, jejich stručné charakteristiky a vzájemné vazby jsou znázorněné na (Obr. 2) Z nich OPC Overview obsahuje základní informace o účelu, výhodách a vlastnostech OPC. Následující tři specifikace – OPC Security, OPC Common Definitions a OPC Complex Data – popisují funkční schopnosti, které mohou anebo musejí být implementovány spolu s ostatními specifikacemi.

Plná čára na (Obr. 3) spojuje nejpoužívanější typy specifikací – OPC Alarms and Events, OPC Historical Data Access, OPC Data Access a OPC XML DA. Tyto čtyři specifikace určují rozhraní komunikace pro kompletní serverový nebo klientský aplikační program. Mohou být dále rozšířeny o OPC Security (s výjimkou OPC XML DA) nebo o funkce podle OPC Complex Data. Specifikace OPC Batch doplňuje specifikaci OPC Data Access o funkce používané v recepturových (šaržových, tzv. batch) výrobách. Specifikace OPC Data

eXchange obsahuje určité nové funkce společně s funkcemi již definovanými v OPC Data Access a OPC XML DA.

Specifikace OPC Common Definitions, OPC Security, OPC Alarms and Events, OPC Historical Data, OPC Data Access a OPC Batch jsou založeny na metodě DCOM společnosti Microsoft. Specifikace OPC XML DA není na rozdíl od nich založena na metodě DCOM, ale jde o tzv. webovou službu. Poslední výjimkou je specifikace OPC Data eXchange, která využívá obě metody (DCOM i webovou službu).

3.6 Úspěšná historie OPC

Všechny specifikace OPC jsou určeny pro specifické použití v různých oblastech průmyslové automatizace, a jsou proto různě implementovány v příslušných aplikačních programech a systémech.

V současné době existuje více než 3 000 produktů využívajících OPC, které pracují po celém světě již v téměř všech odvětvích průmyslu. Daří se tak naplňovat základní vizi metody OPC – zajišťovat globální vzájemnou součinnost mezi softwarem a hardwarem různých výrobců. Komunikační rozhraní OPC rovněž následuje rozvoj internetu. Jeho orientace na standardy XML (eXtended Markup Language), SOAP (Simple Object Access Protocol) a webové služby otevírá další rozsáhlé možnosti použití přenosů dat prostřednictvím OPC v průmyslových informačních systémech.

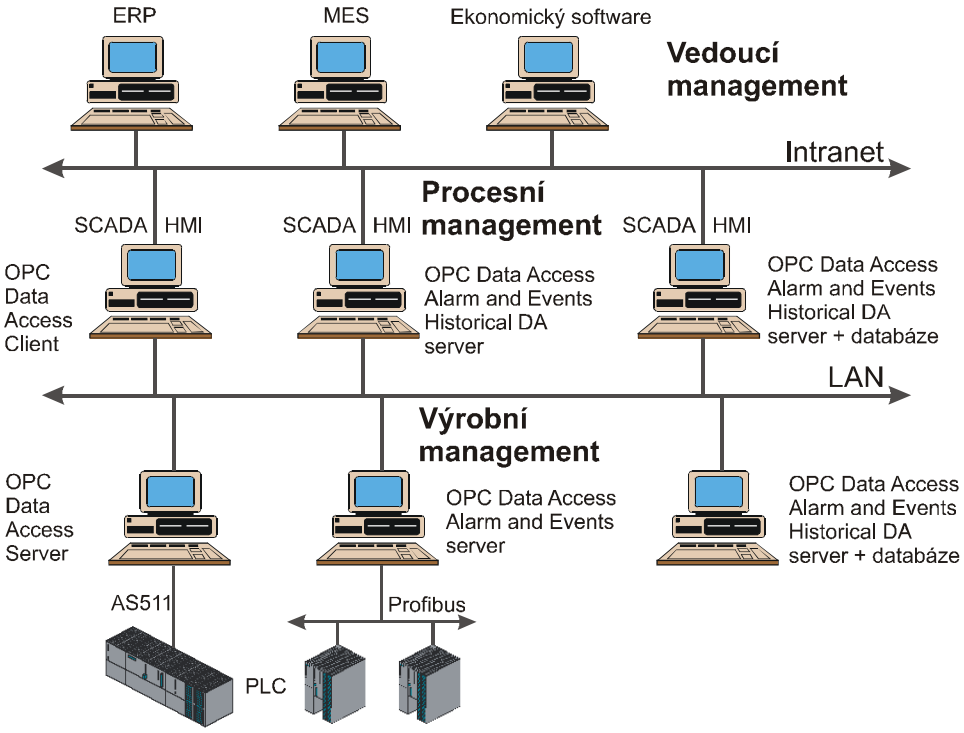
3.7 OPC XML DA

Úkolem specifikace OPC XML DA je umožnit využívání předností metody OPC bez ohledu na operační systém a fyzické umístění počítače. Produkty podporující specifikaci OPX XML DA tak mohou být použity v síti intranetu i v internetu. V obou případech je lze také aplikovat na různých provozních platformách. Jedinou podmínkou je, aby daný systém podporoval HTTP (HyperText Transfer Protocol) a XML. Přenášet data ve formátu XML je méně efektivní než v binární formě. Proto musí být klientský počítač schopen volby mezi množstvím přenášených dat a efektivitou přenosu. Rozhraní XML není možné definovat podle specifikace OPC Security, neboť tato specifikace je založena na technologii DCOM a bezpečnostním modelu systému Windows NT. Při snaze zahrnout do specifikace OPC XML DA bezpečnostní modely z různých operačních systémů se tato v konečném důsledku stane

nepoužitelnou. Proto se data při přenosu metodou OPC zabezpečují jinými způsoby, které nejsou součástí specifikace OPC (firewall apod.).

3.8 Architektura OPC

Při výměně dat podle standardu OPC je využíváno všeobecně přijaté a osvědčené schéma klient/server. K jednomu serveru se může připojit několik klientů různých výrobců, stejně tak k jednomu klientu je možné připojit OPC servery různých výrobců. Komunikace mezi programy podle standardu OPC se používá k výměně dat zejména v průmyslových informačních systémech.



Obr. 4 Architektura OPC komunikace

Typická architektura průmyslového informačního systému sestaveného s využitím metody OPC má tři úrovně řízení (obr. 4). Nejnižší úroveň, označená řízení výrobních operací (field management), obsahuje komunikační, popř. řídicí počítače připojené na jedné straně k řídicím jednotkám (programovatelným automatům; Programmable Logic Controller – PLC) a na druhé straně do místní podnikové sítě (LAN). Vedle těchto typicky serverových stanic je na dané úrovni znázorněn ještě komunikační a datový server. Jedná se o počítač, který může uchovávat technologická data v databázi (relační nebo real-time).

Na střední úrovni – řízení procesní úrovně (process management) – se nacházejí klientské počítače s vizualizačními a monitorovacími programy, které prezentují výrobní (technologické) procesy operátorům v grafické a alfanumerické podobě (operátorské rozhraní; Human Machine Interface – HMI).

Jednodušší vizualizační programy bez vazby např. na celopodnikový informační systém nebo jiné aplikační programové vybavení obsahují pouze uvedené dvě úrovně řízení – technologických operací a výrobních procesů.

Na nejvyšší úrovni, označené řízení podniku (business management), jsou nadřazené podnikové informační systémy s programy typu MES (Manufacturing Execution Systems), ERP (Enterprise Resources Planning) a různým ekonomickým softwarem. Zde se OPC využívá k vytvoření společného komunikačního rozhraní pro přenos technologických dat mezi řídicími systémy, programy typu SCADA a databázemi. Ty v reálném světě převážně pocházejí od různých výrobců a jejich propojování by bez společného komunikačního rozhraní bylo velmi složité a nesnadné.

3.9 Platformy použitelné pro OPC

Nezbytnou složkou pro komunikaci podle OPC je vrstva COM a její síťová verze DCOM (s výjimkou specifikace OPC XML DA). Funkce DCOM je standardní součástí operačních systémů Windows XP, Windows NT 4.0 a Windows 2000. Do Windows 95/98 je nutné DCOM doinstalovat. V těchto systémech je tak možné přenášet data prostřednictvím OPC v rámci jednoho PC i v LAN. Ve Windows CE lze síťově komunikovat od verze 3.0.

Na platformě Linux je rozšíření standardu OPC závislé na tom, jakým způsobem se nahrazuje vrstva DCOM. Existuje několik komerčních produktů zahraničních firem.

4 MODEL COM

Zkratka COM je zažitým označením komponentní technologie vyvinuté firmou Microsoft: Component Object Model. Zkratka označuje komponentový model, který je v současnosti jedním ze tří hlavních přístupů ke komponentnímu vývoji softwaru.

COM je metoda vývoje softwarových komponent – „malých“ spustitelných fragmentů zdrojového kódu, které poskytují své služby ostatním aplikacím, operačnímu systému a dalším komponentám – svým klientům.

Na COM jsou v současnosti postaveny další technologie společnosti Microsoft, např. ActiveX, DirectX, OLE a další. [3]

Společnost Microsoft si zakládá na tom, aby COM nebyl chápán pouze jako technologický rámec pro vývoj komponentního softwaru, ale jako způsob psaní programů (the way of writing programs). Microsoft tím chce ve vývojářích posilovat přesvědčení, že komponentám patří budoucnost a že se k tomuto způsobu vývoje bude postupně přesouvat více a více aplikací. [3]

4.1 Historie COM

Technologie COM má za sebou dlouhou cestu vývoje. Vznikala postupně, jak se předchozí technologie vyvíjely a přizpůsobovaly rostoucím požadavkům.

Na počátku byly první verze Windows, které umožňovaly spuštění více aplikací zároveň. Ruku v ruce s tímto požadavkem šla potřeba komunikace mezi těmito spuštěnými aplikacemi. Ta byla na počátku realizována velmi jednoduchým způsobem, který však přetrval až do dneška: schránkou systému Windows (clipboard).

Další technologie, která umožňovala výměnu informací mezi běžícími aplikacemi, se nazývala *Dynamic Data Exchange* (DDE): ta byla interně využívána výše zmíněnou schránkou. Principem DDE bylo zasílání zpráv mezi DDE klientem (jednou spuštěnou aplikací) a DDE serverem (druhou spuštěnou aplikací). DDE komunikace se členila do témat (topics) a položek (items); celý mechanismus je založen na zasílání zpráv systému Windows. Mechanismus DDE již není v moderních aplikacích využíván.

V roce 1992 přišla společnost Microsoft s technologií Object Linking and Embedding (OLE), která znamenala další průlom v komunikaci mezi aplikacemi. První verze OLE byla

založena na DDE, což se ukázalo jako těžkopádné a nevyhovující, v roce 1993 tedy spatřila světlo světa druhá verze – OLE2 založená na myšlence komponentní technologie. OLE je obecně založeno na propojování a vkládání dokumentů a ve výsledku umožňovala pracovat s dokumenty jedné aplikace v jiných aplikacích, které o těchto dokumentech typicky nemusely nic vědět. Uvnitř jedné aplikace (OLE klienta) se otevřelo okno druhé aplikace (OLE serveru) – tzv. OLE kontejner, který tak poskytoval funkčnost OLE serveru uvnitř OLE klienta. Důležitou vlastností bylo propojení dokumentů – kromě dokumentu si klient uchovával i informace o jeho uložení, takže mohlo více aplikací pracovat s jednou fyzickou verzí dokumentu. [3]

OLE2 byla sice postaveno na komponentní technologii, avšak sloužilo pro jeden konkrétní účel a technologii nebylo možné použít pro stavbu obecných aplikací. Až v roce 1995 se objevila zcela obecná technologie, která navazovala na OLE, avšak byla určena pro vývoj obecných aplikací na komponentním základě. Technologie se nazývá Component Object Model (COM). [3]

Ani poté však vývoj neustal. Brzy se objevila potřeba komunikovat mezi komponentami uloženými na rozdílných strojích. Microsoft se rozhodl implementovat tento požadavek za použití existující technologie Remote Procedure Calling (RPC). (RPC je jen jednou z mnoha konkrétních technologií, které definuje specifikace Distributed Computing Environment – DCE - vytvořená konsorciem Open Software Foundation - OSF). Výsledkem byla v roce 1996 distribuovaná verze technologie COM: Distributed COM (DCOM).

V současnosti vývoj pokračuje snad stále rychleji: Microsoft přináší stále nové služby, které jeho komponentové technologie poskytují (podpora bezpečnosti, škálovatelnosti, distribuovaných transakcí, vysoce bezpečných síťových aplikací apod.). Vzniká tak nová specifikace COM+ umožňující využívat všech nových služeb. Lze říci, že technologie COM+ je integrací dalších služeb:

- Microsoft Transaction Server (MTS): server pro správu komponent a transakcí, často se využívá ve spolupráci se SQL serverem Oracle nebo MS SQL a ve spolupráci s webovým serverem Microsoft Internet Information Server (IIS)
- Microsoft Message Queue Server (MSMQ): server pro práci offline a synchronizaci

Další strategickou architekturu Microsoftu, která má sloužit pro vývoj internetových aplikací na platformě Microsoft Windows je Windows Distributed Internet Application Architecture (DNA). Architektura obsahuje tři základní vrstvy:

Vrstva prezentační*:* zprostředkovává klientský styk s aplikací, obsahuje však jen uživatelské rozhraní, nikoliv aplikační logiku. Je obvykle realizována tlustými nebo tenkými klienty. Využívá technologií HTML, DHTML, skriptování, Win32 API apod.

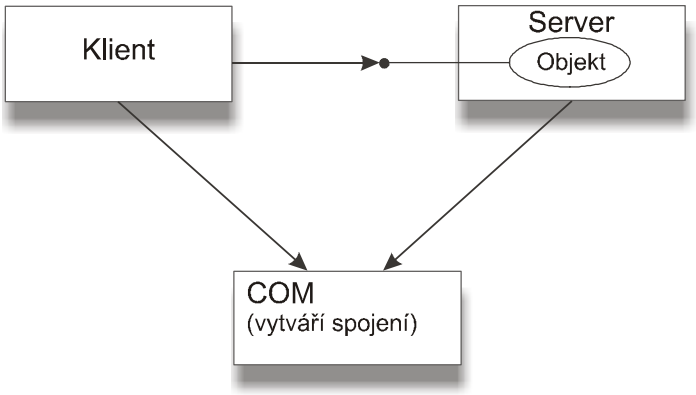
Vrstva aplikační logiky*:* implementuje funkčnost aplikace a aplikačně specifické algoritmy. Tato vrstva je nejčastěji implementována komponentními technologiemi a využívá komponent. Používají se technologie COM, COM+, MTS, MSMQ apod.

Vrstva datová: zajišťuje uložení dat, přístup k datům, bezpečnost, transakce apod. Využívají se technologie ODBC, OLE DB, SQL Server (nebo jiný SQL server připojitelný přes ODBC či OLE DB, např. Sybase, Oracle, DB2), ADO apod. [3]

4.2 COM – základní popis

Co to vlastně znamená, když se rozhodneme použít při vývoji své aplikace model COM? Technologie COM pracuje na principu klient–server. Základní model fungování je ukázán na následujícím obrázku. COM poskytuje podporu při vytvoření spojení klienta s objektem. Oba zúčastnění pak už dál komunikují přímo, bez dalších zásahů. Komunikace probíhá prostřednictvím mechanismu rozhraní. [3]

Mechanismus COM napomáhá vytvořit spojení mezi klientem a serverem (spodní větev na obrázku), následná komunikace klienta se samotným objektem však už probíhá přímo prostřednictvím rozhraní (horní větev).



Obr. 5 Mechanismus COM

Co je to COM klient? COM klientem muže být libovolná aplikace, která se připojuje ke COM serveru. Muže jím být také jiná komponenta. Klient žádá o vytvoření objektu a následně s ním pracuje způsobem, který je velmi podobný práci s klasickými objekty v rámci objektově orientovaného programování.

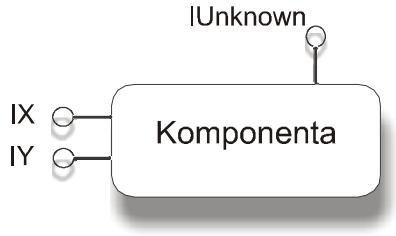
COM server pak poskytuje služby klientské aplikaci. Server je vlastně onou komponentou, a nebo systémem složeným z více komponent. Budeme-li tedy nadále hovořit o komponentách, budeme vlastně mít na mysli COM servery. Existují přitom dva základní druhy COM serverů:

- In-process servery, tedy servery, které jsou spuštěny a běží ve stejném procesu jako klientská aplikace. Dá se tady říct, že klient si u sebe „spustí“ svůj vlastní server. Takové komponenty jsou zkompilovány jako DLL knihovny, spouštějí se ve stejném procesu jako klientská aplikace a na požádání vytvářejí objekt či objekty přímo v adresovém prostoru klientské aplikace. Aplikace pak přímo volá metody rozhraní objektu, protože ví, na jaké adrese je rozhraní uloženo.
- Out-of-process servery, tedy servery, které jsou spouštěny vně adresového prostoru klientské aplikace. Out-of-process komponenty jsou spustitelné EXE soubory, které dokáží vytvářet COM objekty. Běží ve vlastním adresovém prostoru a s klientskou aplikací jsou spojeny přes zástupné objekty (proxy, stub). V adresovém prostoru klientské aplikace je vytvořen proxy objekt, který nahrazuje chybějící COM objekt. Jinak řečeno, proxy objekt vypadá stejně jako originální objekt, má stejnou sadu rozhraní, ale neumí provádět jeho metody. Jeho úkolem je tvářit se jako požadovaný COM objekt a v případě volání některé z metod tuto metodu zavolat ve skutečné komponentě a předat jí parametry volání. V adresovém prostoru komponenty se nachází jiný proxy objekt, v technologii COM nazývaný stub, který předané parametry převezme. Stub jakýmsi zrcadlovým dvojníkem proxy: sedí v adresovém prostoru komponenty a hraje vlastně COM klienta. Stub zavolá rozhraní COM objektu stejně jako by to udělal klient a obdržené návratové hodnoty odešle zpět proxy objektu v adresovém prostoru klientské aplikace. Proxy objekt je pak předá aplikaci. Podle umístění klienta a serveru lze dále rozlišovat dva druhy Out-of-process serverů – local server (komponenta i klient jsou spuštěny na stejném počítači), a remote server (komponenta a klient sedí na různých počítačích).

Ke komunikace mezi komponentou a klientem se vždy používá mechanismu RPC (Remote Procedure Call). V případě, že komponenta i klient jsou na stejném počítači, je využita jakási odnož RPC nazývaná LRPC (Local Remote Procedure Call). [3]

4.3 Základní architektura komponent COM

Typickou architekturu komponenty COM znázorňuje následující obrázek:



Obr. 6 Architektura komponenty COM

Obr. znázorňuje symbol používaný pro zobrazení komponenty se svým rozhrazením. Komponent má dvě uživatelská rozhraní IX a IY a jedno povinné rozhraní IUnknown, které každá komponenta musí mít, jelikož je to pravidlo. Komponenta je složena ze své implementace (kterou uživatel typicky nezná a nikdy nepozná) a z jednoho nebo několika rozhraní, které poskytuje svému okolí. Uživatel komponenty dostane k dispozici pouze popis rozhraní a podle tohoto popisu implementuje klienty, kteří budou s komponentou pracovat. Rozhraní se tak stává úplným základem technologie COM.

Jak se fyzicky komponenty implementují? Možností je víc, ale COM komponenty jsou velmi často implementovány uvnitř knihovny DLL. Chcete-li někomu dát nebo prodat komponentu, přinesete mu prostě soubor DLL.

Dynamicky linkované knihovny (DLL) vždy poskytují (a zveřejňují) množinu exportovaných funkcí, tedy funkcí, které mohou uživatelé knihovny používat. I v případě, že tuto množinu rozhraní neznáme, ji můžeme zjistit např. použitím příslušného nástroje (např. aplikace dumpbin.exe). Vztah DLL a jejích exportovaných funkcí je podobný vztahu komponenty a jejích rozhraní.

Komponenta poskytuje několik rozhraní, každé rozhraní se skládá z několika funkcí, které může klient volat.

V technologii COM je rozhraní definováno jako specifická paměťová struktura obsahující pole ukazatelů na funkce. Každý ukazatel obsahuje adresu funkce implementované kompo-

nentou. Jinak řečeno: implementace všech funkcí jsou ukryty v komponentě, rozhraní pouze obsahují ukazatele na tyto funkce. Zavolá-li klient funkci daného rozhraní, předá se toto volání do komponenty a provede se příslušná funkce. Dalo by říct, že rozhraní je vlastně jakási tabulka, která obsahuje názvy funkcí a potom pouze ukazuje na samotná těla funkcí, která jsou implementována kdesi v komponentě. O tom, kde jsou tato těla implementována, rozhraní ví, proto na ně může ukazovat. Klient se to však nikdy nemusí dozvědět, ten si povídá pouze s rozhraním.

COM aplikace můžeme implementovat takřka v libovolném programovacím jazyce, nejčastěji používaným je však zřejmě C++. Zmíním zde tedy jen něco málo o C++ a jeho souvislosti s modelem COM, neboť se jedná o základní spojení používané v technologii OPC.

V programovacím jazyce C++ se rozhraní implementují jako abstraktní třídy. Abstraktní třída je obecně třída obsahující alespoň jednu virtuální metodu, přičemž může obsahovat datové atributy. V kontextu COM je však za rozhraní považována jen taková abstraktní třída, která obsahuje pouze virtuální metody a neobsahuje žádné atributy (čistá abstraktní třída). [3]

Programový systém napsaný nad technologií COM tedy lze popsat takto: systém je množina komponent, komponenta je množina rozhraní, rozhraní je množina funkcí.

Každé implementované rozhraní vzniká v C++ zděděním ze třídy IUnknown, což je základní třída pro definici rozhraní (deklarovaná v souboru Unknwn.h, obsahuje základní abstraktní metody AddRef(), Release(), QueryInterface()).

Každé rozhraní musí implementovat minimálně tři metody:

- metoda AddRef() – slouží k počítání referencí na komponentu. V těle AddRef() musíme zvýšit čítač odkazů na komponentu.
- metoda Release() – slouží k počítání referencí na komponentu. V těle Release() musíme snížit čítač odkazů na komponentu.
- metoda QueryInterface() – slouží k zjištění, zda komponenta poskytuje požadované rozhraní. Pokud ano, vrátí ukazatel na toto rozhraní.

Při vytvoření nového odkazu na komponentu je nutné zavolat AddRef(), za to bývá zodpovědná sama komponenta; při zrušení odkazu na komponentu je nutné zavolat Release(), to provádí klient.

Klient obecně o komponentě nic neví. Potřebuje-li zjistit, zda je k dispozici nějaké rozhraní, zavolá metodu QueryInterface() a předá ji IID tohoto rozhraní. K provedení tohoto dotazu se v podstatě používá také rozhraní IUnknown.

Jinak řečeno klient má pouze dvě informace:

- komponenta určitě poskytuje rozhraní IUnknown (neboť toto rozhraní poskytuje každá komponenta)
- rozhraní IUnknown určitě poskytuje metodu QueryInterface() pro ověření existence zadaného rozhraní (neboť tuto metodu poskytuje každé rozhraní)

Při vytvoření komponenty dostane klient ukazatel na toto rozhraní. Z uvedených informací tedy lze dovodit způsob, jakým klient komunikuje s komponentou:

Klient vytvoří komponentu a dostane ukazatel na IUnknow, označme jej pIUnknown.

Klient by rád použil rozhraní například IIX. Zavolá proto metodu pIUnknown->QueryInterface(IID_IIX).

Pokud komponenta poskytuje rozhraní IIX (označované identifikátorem IID_IIX), vrátí klientovi ukazatel na toto rozhraní, označme jej např. pIIX.

Klient následně volá metody rozhraní pIIX, např pIIX->metoda()

Jakmile klient již rozhraní IIX nechce dále využívat, zavolá pIIX->Release().

Klient přistupuje ke komponentě výhradně přes ukazatele na jednotlivá rozhraní. Protože všechna rozhraní vznikla děděním z IUnknown, mohou být všechna rozhraní manipulována stejně přes ukazatel typovaný na IUnknown.

Komponenta je složena ze své implementace (kterou uživatel typicky nezná a nikdy nepozná) a z jednoho nebo několika rozhraní, které poskytuje svému okolí. Uživatel komponenty dostane k dispozici pouze popis rozhraní a podle tohoto popisu implementuje klienty, kteří budou s komponentou pracovat. Každé implementované rozhraní vzniká v C++ oddděním ze třídy IUnknown, což je základní třída pro definici rozhraní (deklarovaná v souboru Unknwn.h, obsahuje základní abstraktní metody AddRef(), Release(), QueryInterface()). Každé rozhraní musí povinně implementovat alespoň tři metody: AddRef(), Release() a QueryInterface(). [3]

4.4 Základní pravidla pro implementaci QueryInterface()

Abychom dodrželi zásady modelu COM, musíme při implementaci metody QueryInterface() dodržovat několik základních pravidel:

- vždy dostaneme stejný ukazatel na IUnknown. Ať použijeme jakékoliv rozhraní dané komponenty, ukazatel na rozhraní IUnknown je jen jeden a musí být odkudkoliv vrácen ve stejné podobě. Důsledkem je možnost testovat, zda dva ukazatele ukazují na tutéž komponentu.
- dostali-li jsme ukazatel na rozhraní jednou, musíme jej dostat kdykoliv znovu. Nesmí se stát, že QueryInterface() jednou řekne „ano, rozhraní IAbc existuje“ a za chvíli řekne „ne, rozhraní IAbc neexistuje“
- vždycky dostaneme ukazatel na rozhraní, které již známe. Nesmí se stát, že by nám QueryInterface() najednou začalo tvrdil, že rozhraní, které s úspěchem používáme, neexistuje
- ptáme-li se z rozhraní IIX na rozhraní IIY a dostaneme-li jej, musíme dostat i rozhraní IIX při dotazu z IIY. Jinak řečeno, pokud metoda IIX.QueryInterface() řekne, že rozhraní IIY existuje, pak musí metoda IIY.QueryInterface() naopak říct, že existuje rozhraní IIX
- existující rozhraní lze dostat z kteréhokoliv dalšího rozhraní. Nesmí se stát, že by QueryInterface() kteréhokoliv rozhraní „nevěděla“ o existenci nějakého dalšího rozhraní

Základní pravidlo pro implementaci rozhraní však zní: rozhraní označené daným identifikátorem (IID) se nikdy nezmění. Klient se tedy nemusí bát, že používá špatnou verzi rozhraní. Najde-li klient rozhraní, je to zaručeně správné rozhraní. [3]

4.5 Životní cyklus komponenty

Nyní se pojďme stručně podívat na životní cyklus komponenty, tedy na otázku, jak dlouho má být komponenta umístěna v operační paměti a kdo je zodpovědný za její uvolnění.

Zopakujme: klient na začátku své práce vytvoří instanci komponenty. Pak ale pouze ví, že má ukazatel např. na rozhraní IIX a že by rád toto rozhraní IIX používal. Obecně ale neví (například ve vícevláknovém prostředí), používá-li ještě jiné rozhraní. Pokud by klient měl

být zodpovědný za uvolnění komponenty z paměti, musel by ve všech svých modulech periodicky testovat ukazatele na IUnknown a složitě porovnávat jejich rovnost.

Přímočařejší řešení spočívá v přenesení této odpovědnosti (tedy odpovědnosti za uvolnění komponenty z operační paměti) z klienta na komponentu samotnou. Jak už jsme si řekli, komponenta si sama udržuje čítač referencí prostřednictvím funkcí AddRef() a Release().

Pravidla pro používání AddRef a Release():

- ve funkcích vracejících rozhraní (např. QueryInterface(), CreateInstance()) apod.) voláme AddRef(). Toto volání si zajišťuje komponenta sama, proto volající (tj. klient) nevolají AddRef().
- pokud již rozhraní není potřeba, zavolá se funkce Release(). Toto volání provádí kdokoliv, kdo zjistí, že rozhraní, které sám vytvořil, již nepotřebuje.

AddRef() je kromě toho nutné volat vždy, když vytvoříme nový odkaz na rozhraní (přiřadíme ukazatel, zavoláme funkci vracející ukazatel na rozhraní apod.). Především tento bod bych rád zdůraznil, protože se v něm velmi často dělají chyby. Prostě máme ukazatel pA ukazující na rozhraní IA; stačí vytvořit nový ukazatel pB a řekneme mu „ukazuj tam, kam ukazuje pA“. Tím jsme ve své podstatě zvýšili počet odkazů na komponentu, takže je nutné zavolat AddRef(). Existují samozřejmě metody, jak toto ruční volání AddRef() zautomatizovat a získat tak možnost pustit podobné starosti z hlavy.

Z hlediska implementace jsou možné dva přístupy. Udržovat jeden čítač pro celou komponentu nebo počítat odkazy na každé rozhraní zvlášť. Obecně lze za mírně lepší řešení prohlásit druhou variantu (snazší hledání chyb a lepší práce s pamětí – nepotřebná rozhraní jsou uvolněna). [3]

4.6 Dynamické linkování

Jak už jsme si řekli, obecně existují dva druhy umístění komponent:

- v dynamicky linkovaných knihovnách DLL: tzv. in-process komponenty (jsou umístěny ve stejném paměťovém prostoru jako jejich klienti)
- ve spustitelných souborech EXE: tzv. out-of-process komponenty (jsou umístěny ve vlastním paměťovém prostoru)

Jednodušší variantou jsou komponenty v DLL. Exportování funkcí z DLL se provádí pomocí klíčového slova extern „C“, navíc je nutné vytvořit soubor DEF s popisem exportovaných funkcí. [3]

4.7 HRESULT

Předávání výsledků v modelu COM je často realizováno prostřednictvím údajů datového typu HRESULT. Nejedná se o handle, ale o 32bitové celé číslo, které ve svém 31. bitu nese informaci o vážnosti (severity), dalších 15 bitů označuje službu, které se hodnota týká a bity 0 – 15 jsou vyhrazeny pro samotnou návratovou hodnotu.

Je k dispozici řada standardních předdefinovaných konstant, např. S_OK, E_FAIL, E_UNEXPECTED, E_NOINTERFACE apod.

Hodnoty bezchybného a chybného běhu bychom neměli porovnávat přímo (hr == S_OK), ale přes speciální definovaná makra (SUCCEEDED(hr)). [3]

4.8 GUID, IID, CLSID, ProgID apod.

Nelze samozřejmě vyloučit, že dvě komponenty vyvinuté dvěma nezávislými vývojáři ve dvou rozdílných zemích budou mít totožný název. Pokud by klienti přistupovali ke komponentám prostřednictvím jejich jmen, mohlo by tedy obecně docházet k chybám způsobeným záměnou komponent.

Proto model COM definuje mechanismus pro jednoznačnou identifikaci komponent (ale také dalších důležitých prvků, například rozhraní apod.). K této jednoznačné identifikaci všech klíčových prvků v modelu COM se používá identifikátorů typu GUID – Globally Unique Number. Tento identifikátor si může kdokoliv vygenerovat, existuje pro to několik způsobů, např. použít utility UUIDGEN.EXE či GUIDGEN.EXE. Existuje také makro DEFINE_GUID. Identifikátor je generován na základě mnoha údajů (čas, fyzická adresa síťové karty a další), které by dle slov tvůrců modelu COM zajistit jednoznačnost (každý vygenerovaný identifikátor by měl být unikátní).

V modelu COM se pak tyto identifikátory používají pro identifikaci rozhraní (tzv. IID – Interface ID), pro identifikaci samotných komponent (CLSID – Class ID), případně i pro identifikaci aplikací (ProgID – Program ID) apod. [3]

4.9 Registr systému Windows

Základní systémová databáze Windows, tzv. registr, uchovává informace o všem podstatném, co se ve Windows odehrává, a tedy i o komponentách, které jsou v daném počítači k dispozici.

Z hlediska modelu COM je nejpodstatnější klíč HKEY_CLASSES_ROOT – CLSID – {konkrétní_CLSID} – InProcServer32 – C:\Knihovna.dll. Tento klíč uchovává umístění komponenty s daným CLSID na disku počítače.

V okamžiku, kdy klient požaduje vytvoření instance dané komponenty, zjistí systém ve svém registru, zda je tato komponenta k dispozici a kde.

Registrace komponenty do registru se provádí pomocí funkcí DllRegisterServer() a DllUnregisterServer(), které musíme implementovat (a exportovat z DLL knihovny komponenty). Tyto funkce následně využije např. aplikace REGSVR32.EXE, která provede zaregistrování komponenty do registru. [3]

4.10 Budoucnost technologie COM

I když je technologie COM výtvorem firmy Microsoft, s její podporou se sedkáme i u jiných operačních systémů. Například, Sun Solaris, Digital UNIX, IBM OS/400, Linux, atd.

Novou technologií, která by mohla nahradit COM je technologie .NET. Je založená na Common Language Runtime (CLR), která pracuje na vrstvě operačního systému a je přenositelná na jiný operační systém, než je MS Windows. CLR má několik důležitých vlastností na kterých jsou další části skupiny technologií založeny a které umožňují každému .NET jazyku přístup i k komponentám zapsaným v jiném jazyce. Jádrem technologie .NET je Common Language Infrastructure (CLI). CLI specifikace obsahuje běžné standardy pro programovací jazyky – Common Language Specification (CLS) a Common Type System (CTS). Principem se podobá jazyku JAVA.

.NET podporuje dva různé typy aplikací – WEB aplikace a WIN aplikace. .NET je následovníkem technologie DCOM a je přímo určená pro použití v internetu. Hlavní důvody pro přechod na technologii .NET:

- Zjednodušuje vývoj komponent. Obtížné, ale nedůležité části nejsou již aplikovány.
- Je nezávislá na OS.

- Používá protokol SOAP pro snadnou aplikaci v internetovém prostředí.
- DCOM je uzavřenou Microsoft technologií, .NET specifikace jsou volně přístupné. [3]

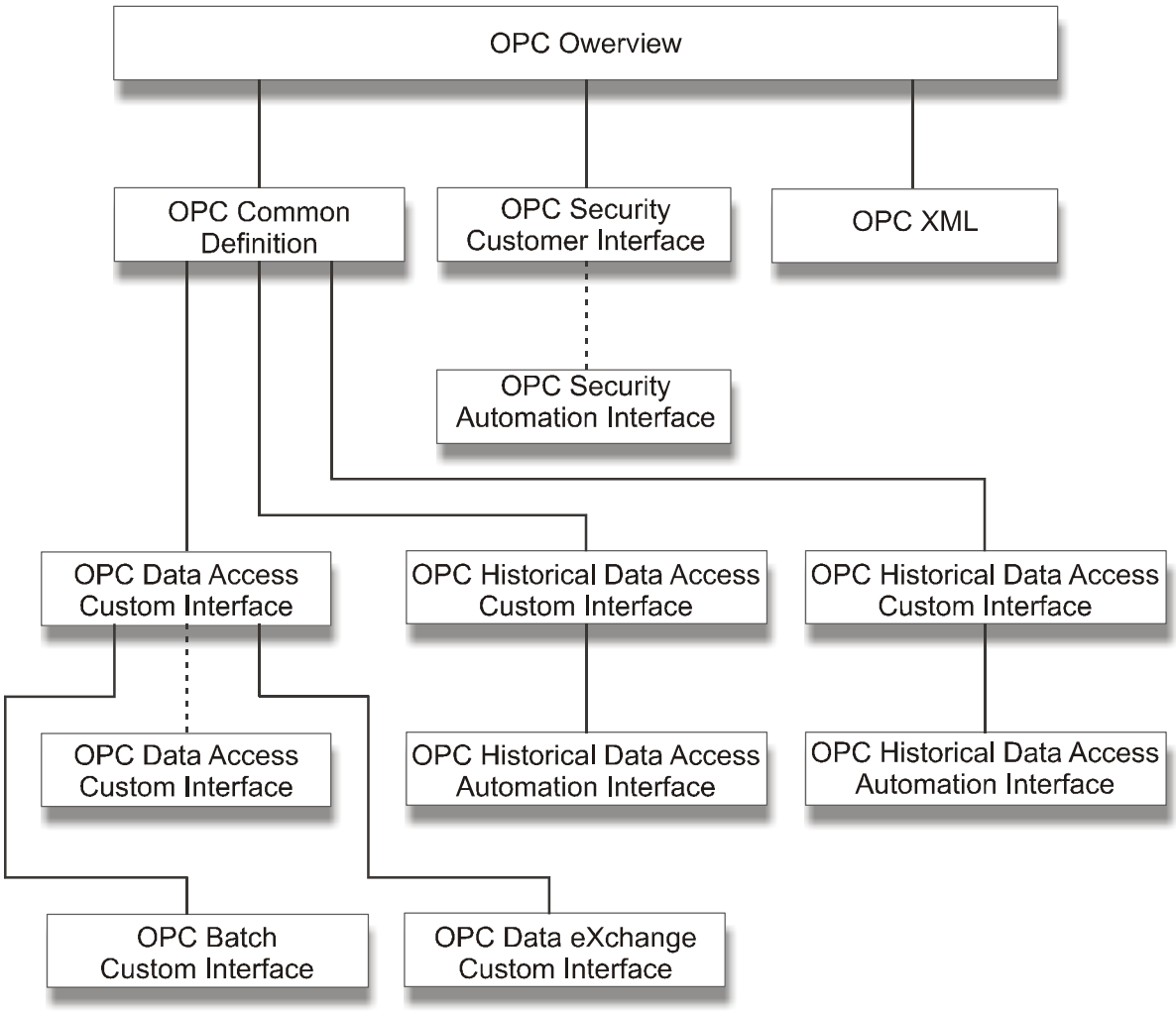
5 OPC SPECIFIKACE

Standard OPC je vytvářen a udržován prostřednictvím tzv. specifikací OPC. Specifikace OPC je volně přístupná technická dokumentace stanovující pravidla, kterými se řídí chování a konfigurace standardního rozhraní OPC.

Tab. 1 Specifikace OPC – stav ke konci 1. čtvrtletí 2006

Specifikace	Obsah	Aktuální verze
OPC Overview	obecný popis metody OPC, jejích výhod a způsobu použití	verze 1.00
OPC Common Definitions and Interfaces	specifikace popisuje definice a základní přístupová pravidla, která jsou společná pro všechny specifikace.	verze 1.10
OPC Data Access (OPC DA)	určuje přístup k datům v reálném čase; je nejčastěji používanou specifikací	verze 3.0
OPC Alarms and Events	stanovuje poskytování informací o výskytech specifikovaných událostí a výstrah klientům OPC	verze 1.1
OPC Historical Data Access	popisuje přístup k historickým datům uloženým v databázi	verze 1.2
OPC Batch	podobně jako OPC DA, ale místo spojitých provozů je určena pro technologie s šaržovou výrobou (potravinářství, farmacie apod.)	verze 2.0
OPC Security	určena k důkladnějšímu zabezpečení přístupu obsluhy při ovládání technologického zařízení z klientů OPC prostřednictvím serverů OPC s využitím zabezpečení systému Windows	verze 1.0
OPC XML DA	vymezuje integrování OPC a XML do internetových aplikací	verze 1.0
OPC Data eXchange	určena pro tzv. horizontální komunikaci mezi řídicími jednotkami s odlišnými komunikačními protokoly (např. EtherNet/IP, ProfiNet, High Speed Ethernet a Interbus) prostřednictvím sítě Ethernet	verze 1.0
OPC Complex Data	určuje způsob popisu struktury komplexních dat a metody přístupu k těmto datům	verze 1.0

Současné specifikace OPC a jejich stručné charakteristiky jsou uvedeny v (Tab. 1). Vzájemné vazby mezi jednotlivými specifikacemi jsou uvedeny a na (Obr. 7). Plná čára na spojuje nejpoužívanější typy specifikací – OPC Alarms and Events, OPC Historical Data Access, OPC Data Access a OPC XML DA. Tyto čtyři specifikace určují rozhraní komunikace pro kompletní serverový nebo klientský aplikační program. Mohou být dále rozšířeny o OPC Security (s výjimkou OPC XML DA) nebo o funkce podle OPC Complex Data.



Obr. 7 Vzájemné vazby mezi jednotlivými specifikacemi

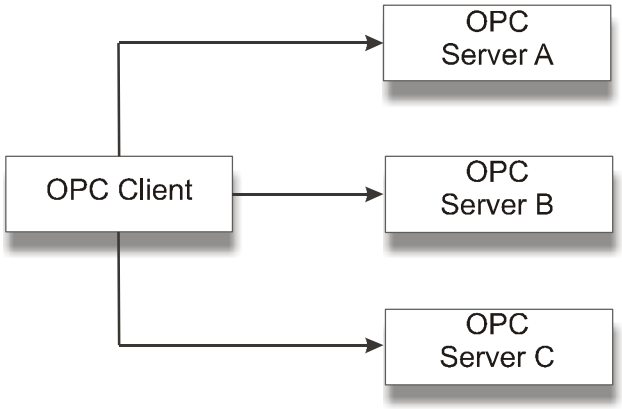
5.1 Overview

OLE for Process Control (OPC) představuje první úspěšnou iniciativu standardizující komunikační rozhraní mezi prvky průmyslové automatizace — průmyslovými automaty, čidly a akčními členy na jedné straně a řídicími či operátorskými počítači a průmyslovými informačními systémy na straně druhé. Ze standardu OPC profitují především uživatelé, kteří

díky němu přestávají být vázáni na programové a technické vybavení podporující pouze vlastnické protokoly zavedených firem (zejména pokud tyto firmy brání své postavení právními překážkami zabraňujícími implementaci daného protokolu třetím stranám). Standard OPC spravuje nezisková organizace OPC Foundation (http://www.opcfoundation.org/) a implementace tohoto standardu je dostupná všem bez jakýchkoliv licenčních poplatků.

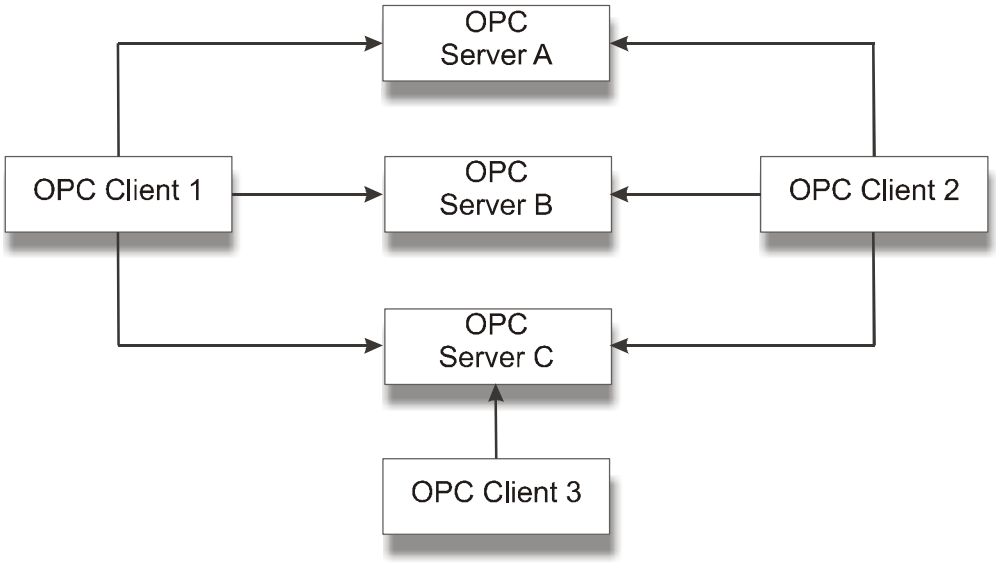
OPC tvoří programovou vrstvu mezi technickým vybavením a programy s tímto hardware komunikujícími. Je také možné poskytovat pomocí OPC dada pro kancelářské aplikace a ERP systémy. Procesní data mhou být také přímo prezentovaná v Exelu a mohou být archivována a zpracovávaná v databázích. Další možností OPC je zasílání a vzdálená správa dat a datových zdrojů přímo prostřednictvím sítě.

OPC specifikace popisují OPC COM objekty a jejich rozhraní implementované prostřednictvím OPC serveru. OPC klient může být napojený na jeden či více OPC server.



Obr. 8 OPC klient

Stejně tak jako OPC klient může být napojen na několik serverů, tak i jeden server může poskytovat data více klientům.



Obr. 9 Vztah mezi OPC serverem a klienty

Přestože OPC je primárně navržen pro přístup dat ze síťového serveru, OPC rozhraní může být použito na mnoha místech v samotné aplikaci. Na nejnižší úrovni jsou získávána procesní data z fyzických zařízení nebo SCADA a DCS systémů do aplikací. Architektura OPC umožňuje získávat klientovi data prostřednictvím OPC serverů z různých zdrojů.

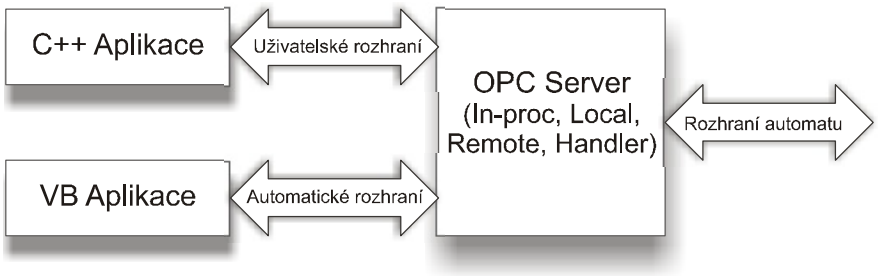
Technicky je standard OPC založen na komponentové technologii COM firmy Microsoft. Název sice napovídá spíše na použití nadstavby COM pro tvorbu složených dokumentů — OLE (Object Linking and Embedding), ale dnes lze jen těžko stanovit hranici kde končí COM a kde začíná OLE, zvláště když se hry zapojí další marketingové názvy, jako např. „Active X“. Protože sama firma Microsoft technologii COM používá k implementaci řady komponent v operačních systémech Windows, je COM široce podporován a obsahuje řadu vlastností velmi užitečných i pro OPC, jako například globální registraci komponent (každý OPC klient přesně ví jak se dostat k OPC serveru), kategorie komponent (OPC klient může snadno zjistit které OPC servery a jakých verzí jsou na daném počítači dostupné) apod. [4]

5.2 OPC Common definitions and Interfaces

Tato specifikace popisuje definice a základní přístupová pravidla, která jsou společná pro všechny specifikace.

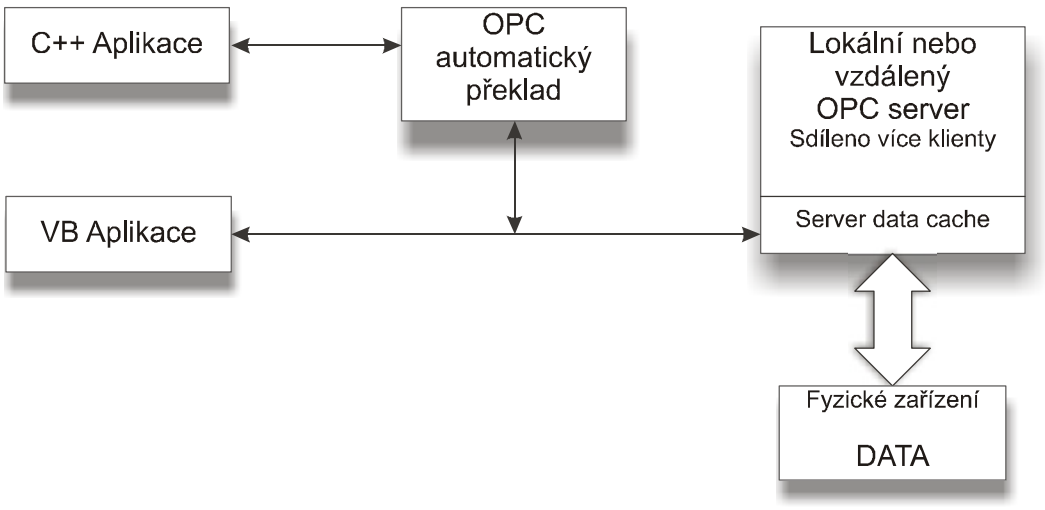
5.2.1 Vyžadované definice rozhraní

OPC specifikace definuje dva druhy rozhraní. Uživatelské a automatické rozhraní. Tvůrce OPC serveru musí implementovat všechny funkce z vyžadovaných rozhraní. OPC klient musí komunikovat prostřednictvím OPC vyžadovaných rozhraní.



Obr. 10 Uživatelské a automatické rozhraní

OPC specifikace obsahuje dvě základní rozhraní – uživatelské a automatické. V hlavním klientském programu, který je vytvořen skriptovacím jazykem jako VISUAL BASIC, bude použito automatického rozhraní. Pro klientský program napsaný v objektově orientovaném jazyku bude výhodnější a jednodušší pro vyšší výkon aplikace použít uživatelské rozhraní. OPC server musí mít povinně implementované automatické rozhraní a uživatelské je volitelné. OPC Foundation poskytuje pro automatické rozhraní standardní překladač. Tato DLL knihovna může být použita pro libovolný OPC server.



Obr. 11 OPC rozhraní

OPC server musí mít implementovány všechny vyžadované rozhraní. OPC klient komunikuje voláním funkcí s povinně implementovaných rozhraní. Pokud OPC server poskytuje volitelné rozhraní musí implementovat všechny jeho funkce. [5]

5.2.2 Chyby a vrácené kódy.

OPC specifikace popisuje rozhraní a odpovídající chování které OPC server a OPC klientské aplikace používají. Pro každou specifikaci existuje soupis chyb a kódů, které jsou při vzniku chyby zasílány zpět aplikaci. [5]

5.2.3 Vypínání OPC serveru.

Při vypínání umožňuje OPC server rozeslat požadavek k odpojení všem připojeným klientům. Funkce je dostupná prostřednictvím přípojného bodu v serverovém objektu a koresponduje na straně klienta s rozhraním IOPCShutdown. [5]

5.2.4 IOPCCommon

Rozhraní IOPCcommon rozhraní, je používána všemi typy OPC serverů (Data Access, Alarm and Events, Historical Data). Umožňuje nastavení a dotazování LocaleID (LCID), protože OPC server může komunikovat více než jedním jazykem. V tomto rozhraní jsou také funkce pro překlad HRESULT výsledných hodnot do textových zpráv a funkce, která umožňuje nastavení klientova jména. [5]

5.2.5 OPC server prohlížeč

OPC foundation dodává Server prohlížeč OPCENUM.EXE. Ten může být nainstalován na kterémkoliv počítači a bude přistupovat k lokálnímu Component Categories Manager a bude poskytovat nové rozhraní IOPCServerList, které může být používáno vzdálenými klienty. Tento server může poskytovat třídy a může být instalován na počítači hostujícím OPC server. [5]

5.3 OPC Data access (DA)

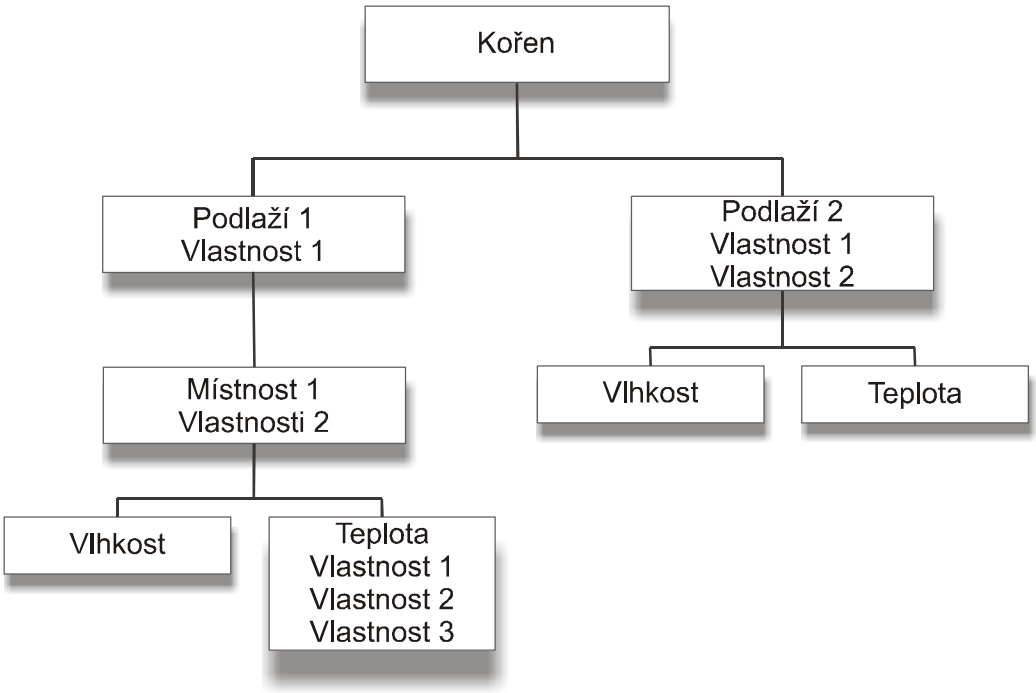
Tato specifikace určuje přístup k datům v reálném čase. Je nejčastěji používanou specifikací a také je nejstarší specifikací ze všech OPC specifikací. Definuje rozhraní mezi klientem a servery, kteří poskytují procesní data. DA servery poskytují jednomu, nebo více DA klien-

tům transparentní přístup k rozdílným datovým zdrojům (např. senzory) a kanálům (ovladače). To umožňuje DA klientovi mít v jednom časovém okamžiku přístup k více DA serverům a DA server může poskytovat data několika DA klientům.

DA specifikace definuje dva rozdílné způsoby, které DA server může implementovat a DA klient použít – jmenný prostor a OPC objektovou hierarchii.

Jmenný prostor obsahuje všechny datové zdroje a datové kanály dostupné serveru. Jmenný prostor může být stromové struktury s jakoukoliv hloubkou (hierarchický jmenný prostor). Může být taky plochý, což znamená, že všechny položky stromové struktury jsou na stejné úrovni a nejsou zde žádné uzly. V hierarchickém jmenném prostoru mohou být uzly použity jako strukturální prostředek. Jednotlivé položky jsou dostupné v uzlech struktury a reprezentují datové zdroje a datové kanály, které představují nastavovací body a měřené hodnoty zařízení. [6]

Položky a uzly mají definované atributy, neboli vlastnosti (např. měřící rozsah). Kromě těchto atributů mají položky definovanou přístupovou cestu AccessPath, jejíž použití je volitelné a význam si určuje server (např. komunikační cesta mezi serverem a zařízením).



Obr. 12 Hierarchická struktura jmenného prostoru

DA klient může vytvořit několik OPC objektů v OPC serveru k definování to je zobrazení procesu. OPCServer objekt je v hierarchii na nejvyšší úrovni. OPCGroup objekty tvoří další

úroveň a obsahují pointers k požadovaným položkám. OPCGroup slouží k strukturování jednotlivých položek. To se provádí na základě logických aspektů, funkčních aspektů, nebo na základě požadavků uživatele. Všechny OPCGroup jsou řízeny OPC serverem. Každý klient vlastní svůj OPCServer, takže existuje specifická hierarchie, což přesně odráží její požadavky.

Jak už bylo zmíněno, OPCGroup řídí potřebné položky; ty jsou zpřístupněny, přečteny a zaznamenány prostřednictvím svého rozhraní. Mezi DA serverem a DA klientem jsou vymezeny tři typy datové výměny, znázorněné v následující tabulce. [6]

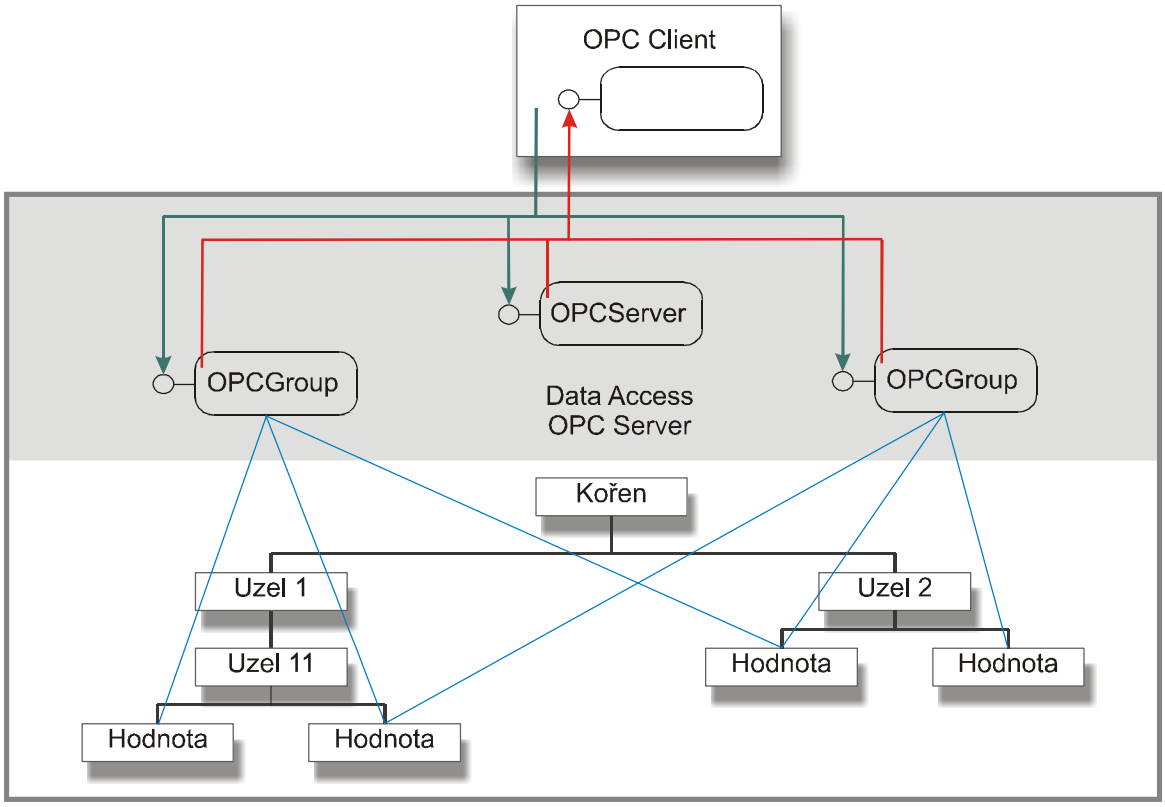
Tab. 2 Typy datové výměny

	Cache	**Device**
Synchronní čtení	√	√
Synchronní zápis		√
Asynchronní čtení		√
Asynchronní zápis		√
Obnovení dat	√	√

Hodnoty jsou získány ze serverové interní cache paměti pro zpracování dat, a nebo přímo z datového zdroje, ale zapsány jsou přímo do datového zdroje.

Při synchronním a asynchronním přenosu se klient přizpůsobuje chování zdroje dat. Při synchronním čtení, klient volá metodu a čeká na vrácenou hodnotu. Synchronní volání je použitelné pouze když máme rychlý přístup k datům jinak bude klient dlouhou dobu blokován. Při asynchronním čtení volá klient metodu ze serveru a okamžitě dostává odpověď (nedostává přímo hodnotu, ale dostane informaci jestli může či nemůže číst požadovanou hodnotu). Hodnota je poslána serverem po určitém čase, který závisí na druhu a nastavení datového přenosu. Asynchronní čtení je tudíž používané, když jsou čtená data dostupná v delších časových intervalech. [6]

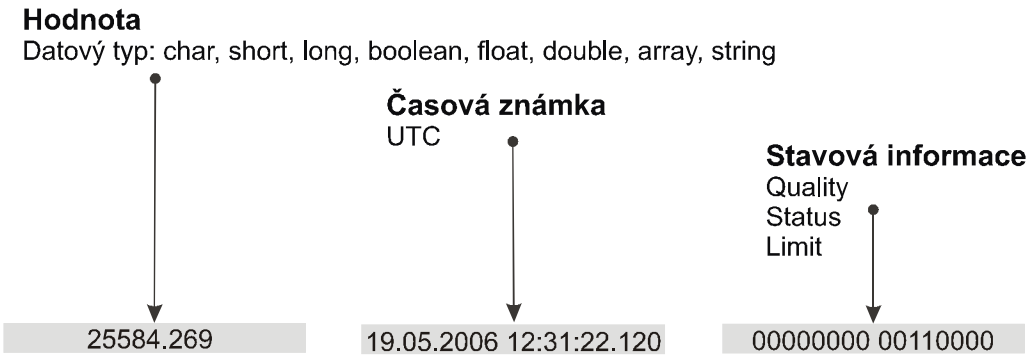
Při obnovení server obnoví hodnoty, o jejich čtení v cyklech rozhoduje nastavení obnovovací četnosti a jejich předávání do klienta, když proběhnou změny jejich hodnot, nebo stavu.



Obr. 13 OPC Data Access server – jmenná a objektová hierarchie

Data zasílaná klientovi mají specifický formát. Formát se skládá ze tří částí:

- Aktuální data (The actual data). Konkrétní procesní hodnota. Hodnota je datového typu *VARIANT*.
- Časová známka (The time stamp). Používá se standardní Windows formát *FILETIME* a čas je univerzální časová souřadnice (UTC)
- Stavová informace (The status information). Má velikost 2B a obsahuje stavové informace o přenášených datech. První dva bity reprezentují kvalitu hodnoty (Good, Bad, Uncertain). Další čtyři stavové bity nám udávají detailní informace o aktuálním stavu. Poslední dva bity obsahují informace o překročení limitních hodnot senzorů.



Obr. 14 Data Access specifikace datového formátu

Když chce OPC klient používat služeb OPC serveru, je vyžadováno vytvoření *OPC-Group* s následujícími parametry:

- Jméno skupiny (Group Name)

  Symbolické jméno, může obsahovat libovolný text (např. skupina_1, teploty, …). Pokud nezadáme žádné jméno, tak server automaticky toto jméno vygeneruje.
- Požadovaná obnovovací četnost (update rate)

  Obnovovací četnost je časový cyklus, kdy server periodicky prověřuje měřené hodnoty a načítá je když je to nutné k informování klienta o změně měřených hodnot. Obnovovací četnost se může průběžně měnit a informace o ní jsou zasílány klientovi. Rozsah obnovovací četnosti je u různých serverů specifická.
- Procento neurčitosti (Percent Deadband)

  Je to procentuální hodnota z měřícího rozsahu, která udává jak maximálně se může hodnota změnit v průběhu časového cyklu bez odezvy klientovi.
- Aktivní stav (Active State)

  Hodnoty ve skupina jsou aktualizovány a načítány pouze pokud je skupina aktivní.

Po vytvoření *OPCGroup* se přidávají další položky. Prostřednictvím jednoho volání může být přidána více než jedna položka. Každá položka může mít specifikovány následující parametry:

- Úplně kvalifikované ItemID (Fully qualified ItemID)

  To je jedinečné jméno položky, vkládané prohlížečem za jméno položky.
- Aktivní stav (Aktive State)

  Má stejný význam jako v *OPCGroup*. Hodnoty se aktualizují a načítají pokud je aktivní.

- Požadovaný datový typ (Requested Datatype)

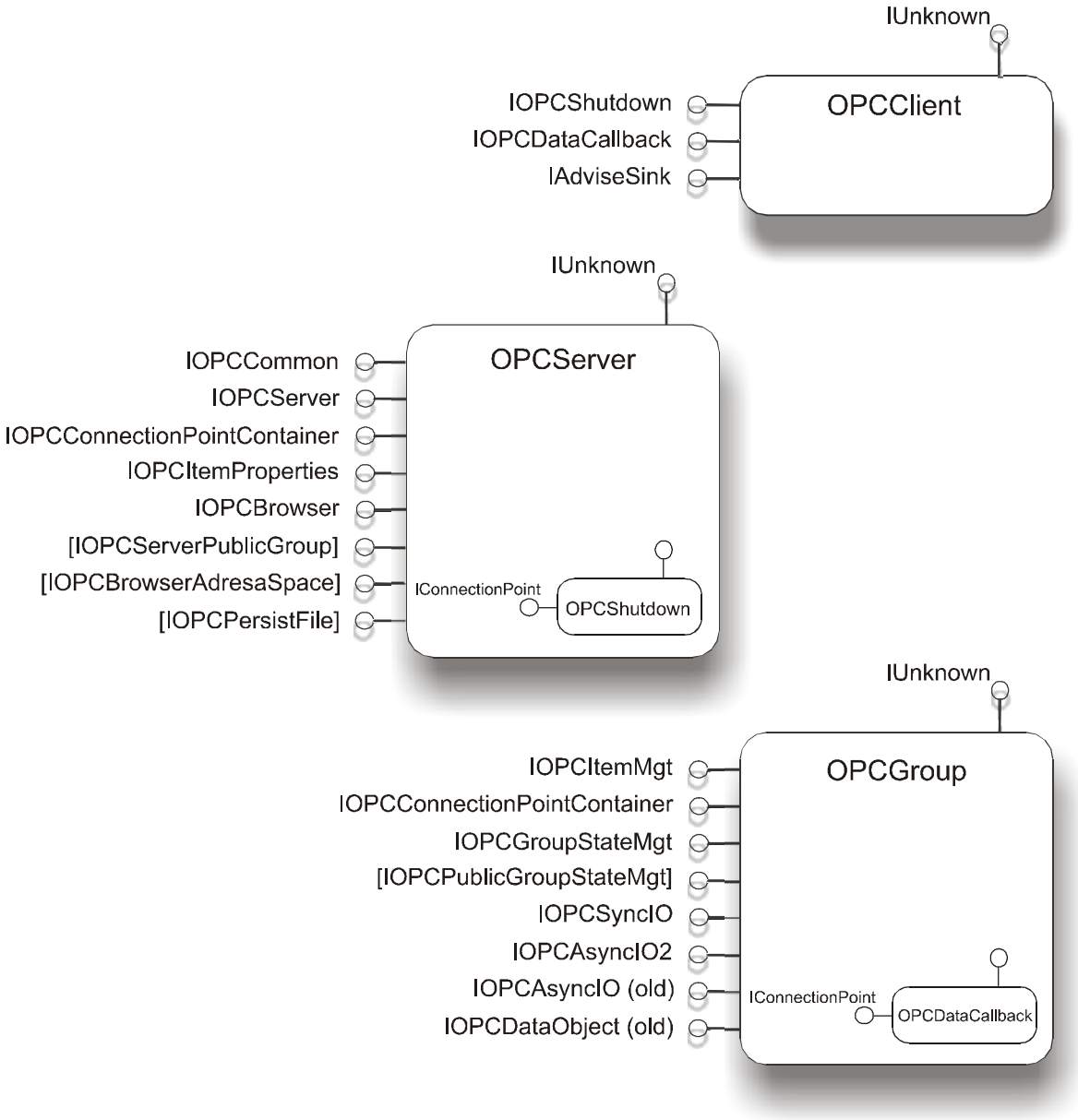
  Klient může požadovat určitý datový typ a server by měl být schopný změnit datový typ na požadovaný (např. float na integer)

Server vrací:

- Kanonický datový typ (Canonical Datatype)

  Interní datový typ položky.
- Serverový popisovač (Server Handle)

  Jedinečné číslo používané k identifikaci položek serverem.



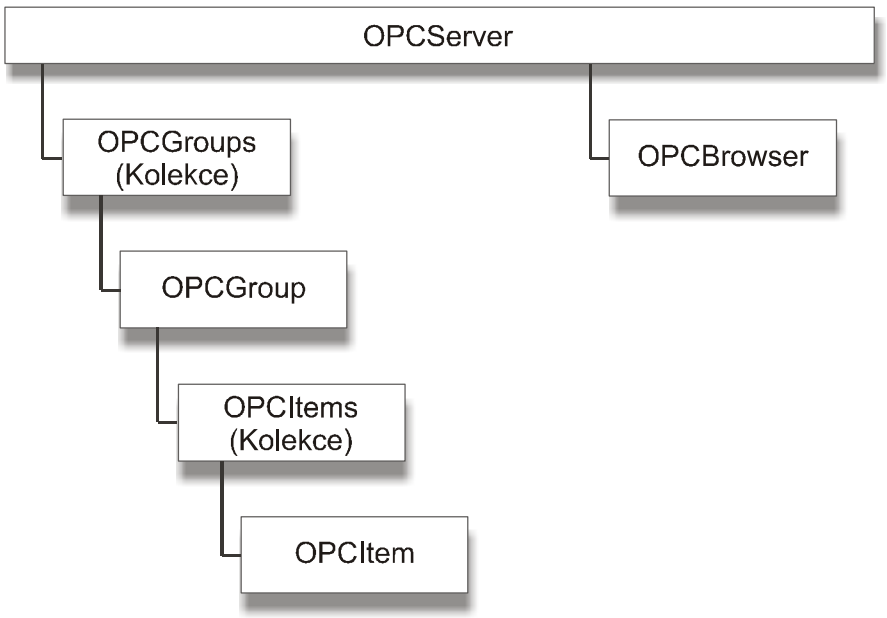
Obr. 15 OPC objekty v Data Access serveru a klientu

Na obrázku (Obr. 15) jsou zobrazené OPC objekty a jejich rozhraní, které jsou definované v DA specifikaci. Protože v současné době je aktuální specifikace 3.0 v níž je spousta odlišností od starších verzí, jsou staré rozhraní, které nejsou použité v nové verzi označené (old). Rozhraní umístněné uvnitř čtverce jsou nepovinné. [6]

5.3.1 Automatizační rozhraní

Mezi automatizačními a uživatelskými objekty jsou tři základní rozdíly:

- V automatizačním rozhraní je jednoduchý automatický objekt, který umožňuje prohledávat jmenný prostor (u uživatelské specifikaci je možnost prohledávat jmenný prostor v části OPCServer). Když server nepodporuje prohledávání, objekt neexistuje.
- Jsou zde kolekce objektů které existují mezi objekty vložených částí.
- Je zde automatický objekt *OPCItem*, klient má přímý přístup do metody tohoto objektu (v uživatelské specifikaci jsou položky skryté uvnitř *OPCGroup*) [6]



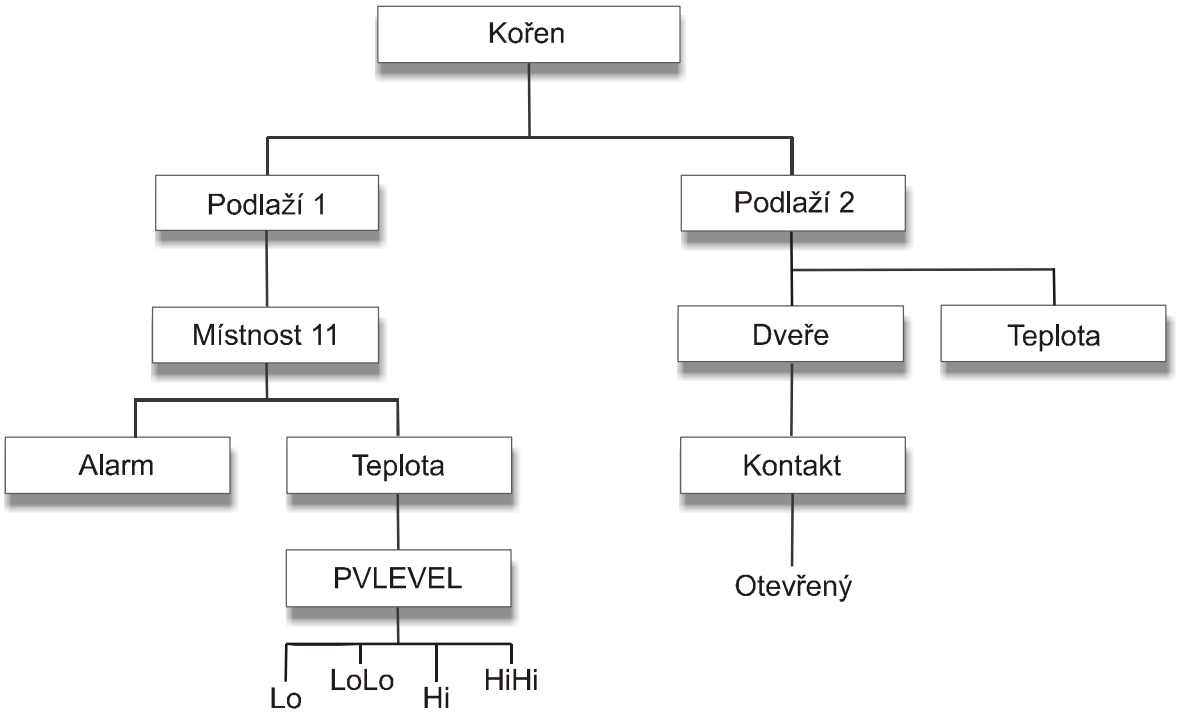
Obr. 16 Automatizační objekty v Data Access

5.4 OPC Alarms and Events (EA)

Specifikace Alarms and Events (alarmy a události) definuje rozhraní mezi klientskými a serverovými programy pro monitorování a potvrzování událostí a alarmů. EA server může

zaznamenávat a hodnotit hodnoty z různých datových zdrojů a rozhodnout zdali bylo dosaženo dané události.

AE server and DA server (viz. 2.2.2 OPC Data Access) mohou mít různé zdroje dat. Rozdíl mezi nimi je, že DA server zajišťuje relativně kontinuální datový tok a AE server neposílá žádné hodnoty klientovi, ale posílá informace, že proběhla nějaká událost (např. teplota překročila limitní hodnotu). Kritéria, která AE server používá k rozhodnutí, že událost nastala, musí být předem nastavena. Aby byly hodnoty a zařízení monitorovány, musí být tato definice provedena během adaptace serveru na konkrétní aplikaci. [7]



Obr. 17 Příklad událostí

AE server může být implementován jako samostatná jednotka, která může získat informace přímo ze zařízení nebo DA serveru. V neposlední řadě je nutná implementace v části DA klientu. AE server může s DA serverem tvořit jeden celek.

AE specifikace zohledňuje fakt, že klient může mít přístup k několika serverům a jeden server může být používán několika klienty.

Specifikace definuje událost jako rozpoznatelný děj (např. porucha zařízení). Některé události mohou být vymezeny specifickými hodnotami. Tato specifikace určí koncept podmínek a vedlejších podmínek, které tato specifikace vymezuje.

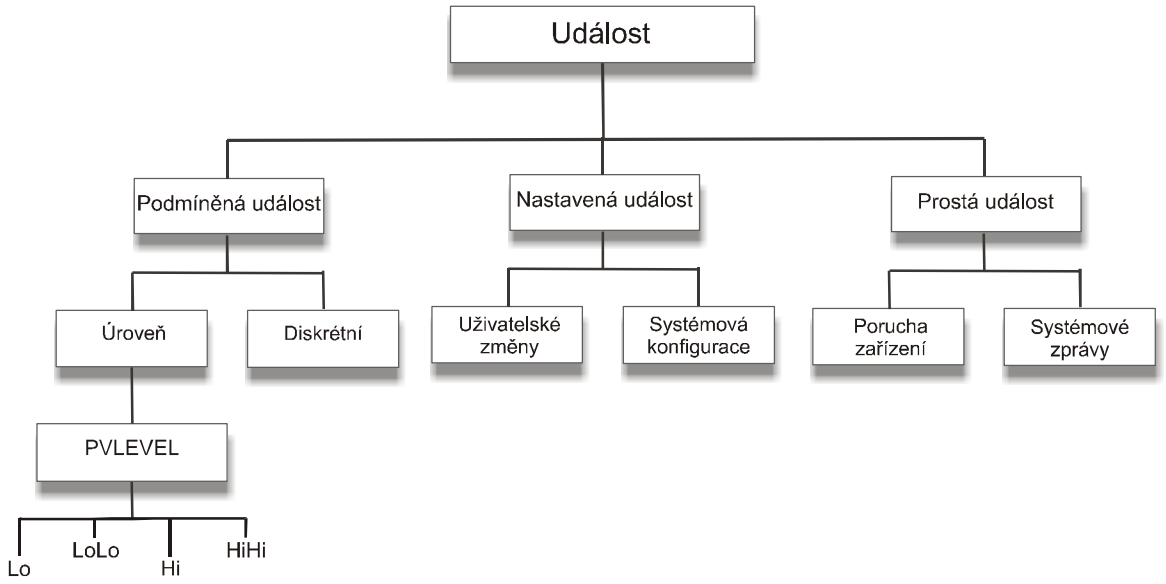
Jsou definovány následující typy událostí:

- Události závislé na podmínkách (Condition-related events)

  Zaslání události klientovi ukazuje, že se vyskytl alarm. Podmínky jsou aktivovány, nebo deaktivovány. Je zde také mechanismus, který rozpozná nastolení událostí.
- Ručně nastavitelné události

  Ty poskytují informace o výskytu událost, jejíž příčina není v samotném procesu (např. zásah operátora) a nebo která musí být nastavena ručně.
- Prosté události

  Upozorní klienta na výskyt jednoduché události (např. selhání zařízení) [7]

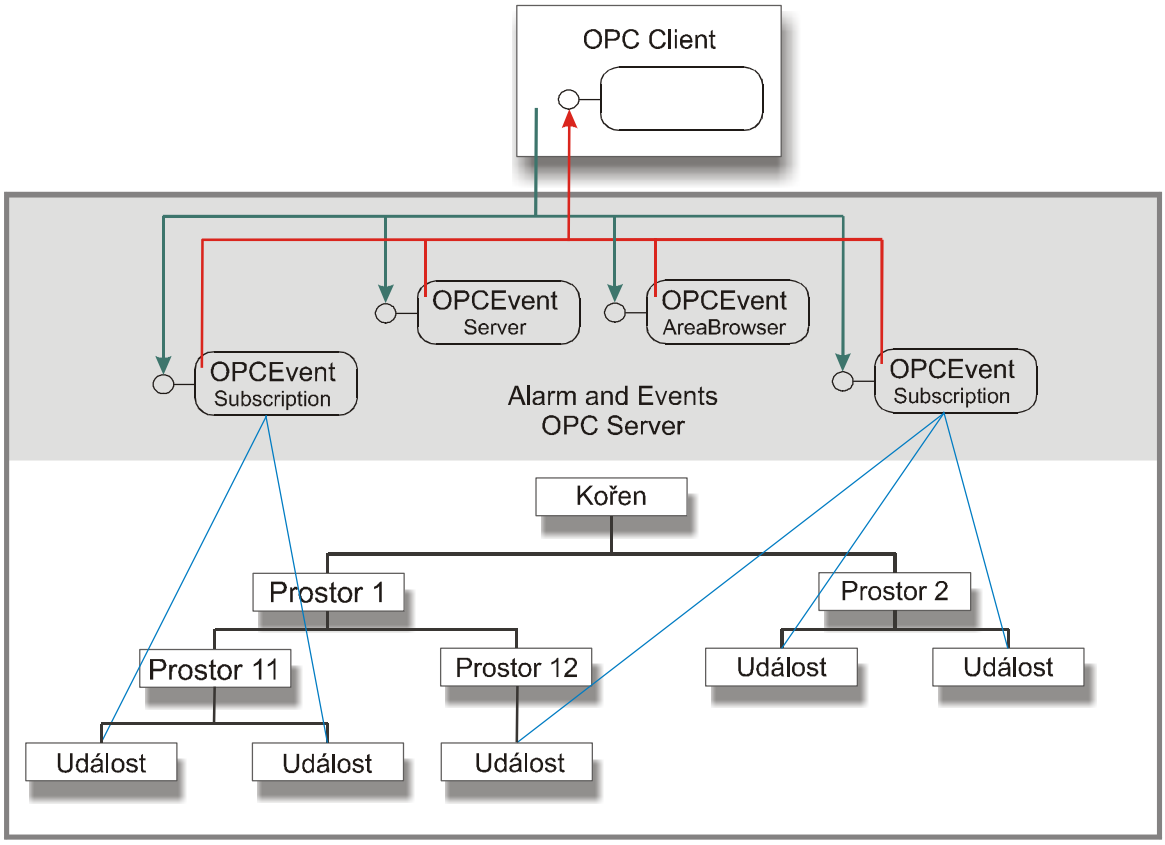


Obr. 18 Rozdělení událostí

Specifikace definuje dva způsoby pro strukturování události:

- Oblast události je stanovena topografickým rozložením události. Uzly v události popisují oblasti, ve kterých se události mohou vyskytnout. Jednotlivé položky popisují zdroje události. V jednom zdroji události může být dostupných několik událostí. (např. monitorování teploty a teplotní senzor). (viz. (obr. 17)).

Pro rozdělení událostí do logických struktur se použito třídění podle umístění (filter space). Tato specifikace rozlišuje tři typy událostí, které jsou rozděleny do kategorií. Některé kategorie jsou definovány již v specifikaci. (viz. (obr. 18))



Obr. 19 OPC Alarm and Events server – jmenná a objektová hierarchie

AE klient může vytvářet různé OPC objekty v AE Severu. Obrázek (Obr. 19) znázorňuje hierarchii příkladu. OPCEventServer objekt je nejvýše postavený v této hierarchii OPC objektů. Volitelný objekt OPCEventBrowser se používá k prohlížení oblasti události. A objekt OPCEventSubscription tvoří nejnižší část, kde jsou události vyhodnoceny.

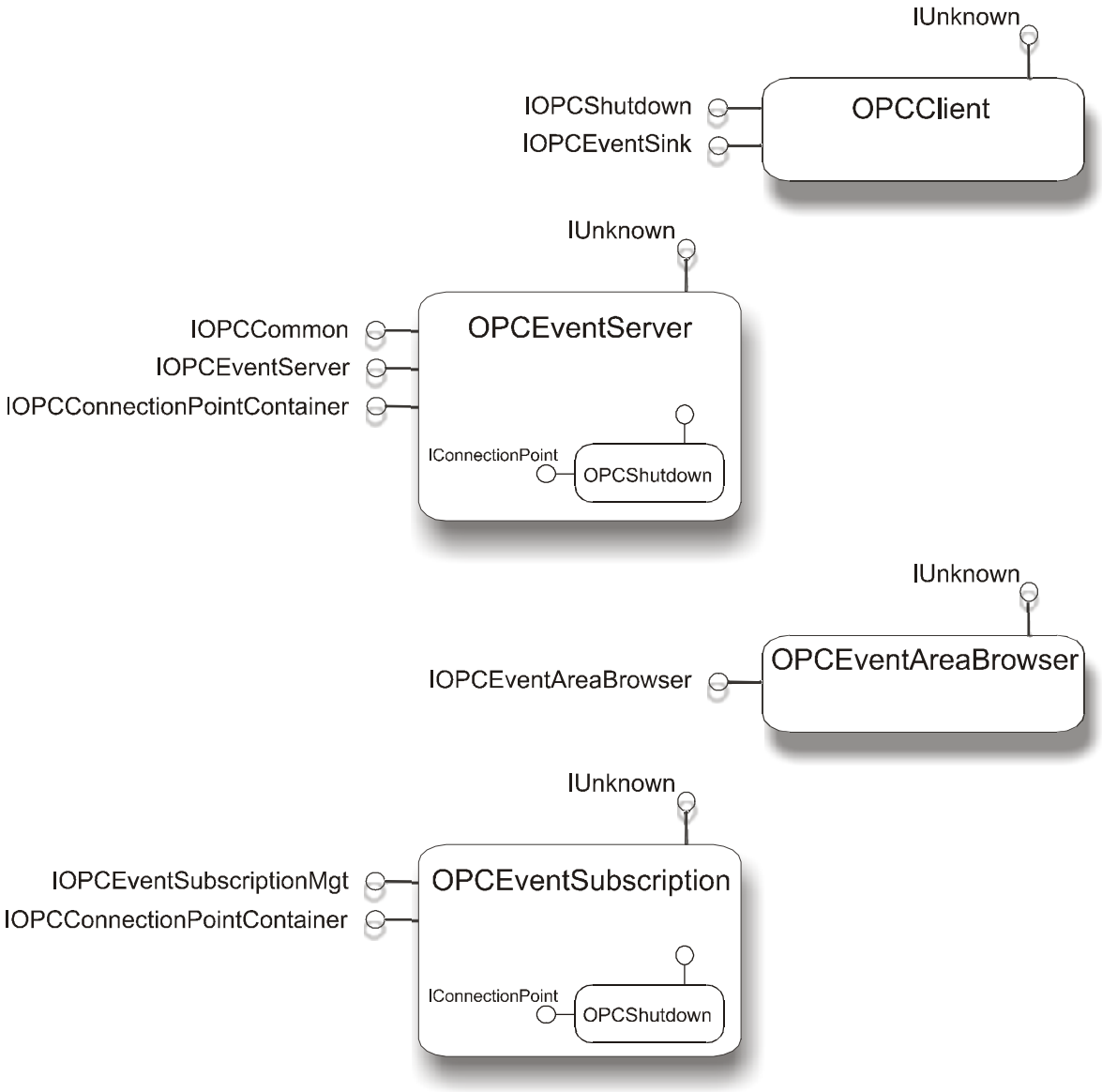
Existuje zde zpětné spojení objektu OPCEventServer s klientem, které server používá k upozornění klienta o případném ukončení. Další zpětná spojení jsou dostupná pro objekty OPSEventSubscription. Ty jsou využity pro záznam události. Obrázek (Obr. 20) znázorňuje všechny objekty a jejich rozhraní. [7]

5.4.1 Automatizační rozhraní

Mezi automatizačními a uživatelskými objekty jsou tři základní rozdíly:

- Jméno OPC AE DLL musí být změněno na jedinečné specifické jméno zdroje dat.
- Jméno OPC AE IDL souboru musí být změněno na jedinečné specifické jméno zdroje dat.

- Pomocný řetězec v souboru IDL musí být změněn tak, aby reflektovat specifikaci rozhraní OPC AE automation.



Obr. 20 Objekty v Alarms and Ebeny serveru a klientu

5.5 OPC Historical Data access (HDA)

HDA server slouží k přístupu k historickým datům uloženým v databázi. Máme základní dva typy historických dat. Zaznamenaná data, jsou to historická data uložená v databázi a agregovaná data, což jsou data načtená ze zaznamenaných dat. Agregovaná data se vytvářejí na základě požadavku klienta. Data mohou být zapisována, čtena a měněna.

Model HDA serveru dovoluje implementovat dva druhy serverů:

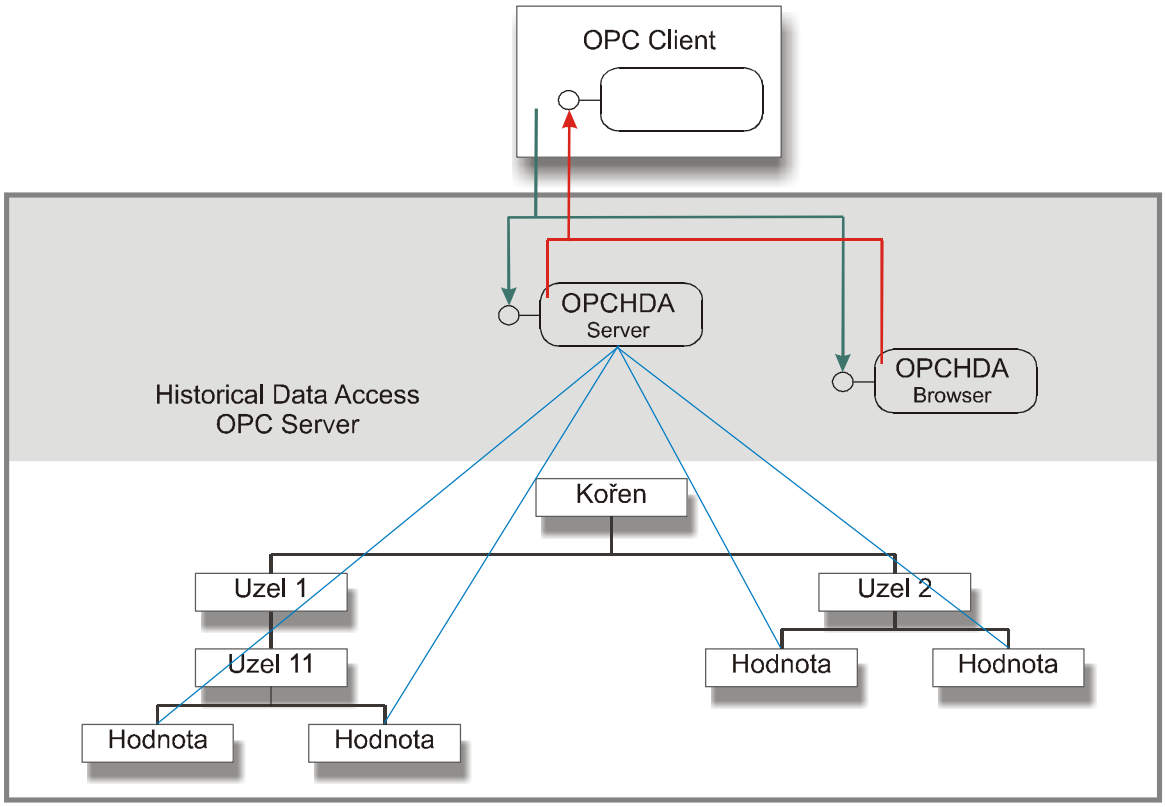
- Prostě směřované datové servery, které používají pouze některé vybrané rozhraní a přístupy k zaznamenaným datům datům.
- Komplexní servery, které umožňují datové procesy jako analýza dat, obnovení dat a čtení historie změněných dat.

Specifikace nestanovuje, které datové zdroje jsou dostupné z HDA serveru, protože datovým zdrojem může být například databáze.

HDA specifikace opět definuje dva druhy přístupu k datům které mohou být implementovány HDA serverem a používány HDA klientem. Je to jmenný prostor (namespace) a objektová hierarchie (object hierarchy).

Jmenný prostor obsahuje všechna historická data poskytnutá serverem. Tato data jsou pouze přebíraná a porovnávaná z jmenného prostoru DA serveru.

HDA klient může vyvolat různé OPC objekty v HDA serveru. *OPCHDAServer* je v hierarchii nejvyšším objektem. Poskytuje kompletní funkce pro čtení, zápis a změnu dat nebo jejich atributů. *OPCHDABrowser* poskytuje funkce pro procházení jmenného prostoru. [8]



Obr. 21 Distorical Data Access server – jmenná a objektová hierarchie

HDA specifikace definuje následující typy datových výměn mezi serverem a klientem:

- Čtení hodnot
- Aktualizování hodnot
- Komentáře k hodnotám
- Záznam hodnot

5.5.1 Čtení hodnot

OPCHDAServer používá synchronní a asynchronní čtení. Asynchronní čtení je obdobné jako u DA serveru.

Možné druhy čtených dat:

- Čtení zaznamenaných dat v periodách s, nebo bez zadaných mezí. V tomto případě nečteme pouze hodnoty, ale také kvalitu a časovou známku.
- Zpracování zaznamenaných data v periodách. HDA server poskytuje různé agregace zpracovaných procesních dat. Klient se dotazuje na existence jednotlivých agregací. Čísla jednotlivých agregací jsou definovaná ve specifikaci, ale je možné použití dalších typů agregací.
- Hodnoty v jednotném čase. Tato metoda se používá ke čtení souvztažných hodnot, které byly zaznamenány ve stejném čase (např. všechny teploty v 17:00).
- Čtení atributů položek pro specifické časy. Server podporuje čísla atributů jednotlivých položek. Některé z nich jsou definovány ve specifikaci (např. datový typ, popis, odkaz na zdroj dat).

Tabulka (Tab. 3) ukazuje relace mezi zpracovanými daty, agregacemi a atributy. V tabulce jsou zaznamenané čtyři měřené teploty v deseti hodnotách. Jsou zde taky tabulky s atributy a agregacemi (průměrná hodnota, delta – rozdíl mezi první a poslední hodnotou v časovém intervalu, minimum, maximum, rozsah – rozdíl mezi minimem a maximem v časovém intervalu). [8]

Tab. 3 Vztahy mezi zaznamenanými daty, atributy a agragacemi

Zaznamenaná Data										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Teplota1	18	18	19	20	21	20	19	18	17	18
Teplota2	17	17	18	19	20	17	17	17	18	18
Teplota3	16	17	18	18	18	17	18	19	19	18
Teplota4	17	18	18	19	20	20	20	20	19	19

Atributy				
	Teplota1	**Teplota2**	**Teplota3**	**Teplota4**
Data Type	VT_I2	VT_I2	VT_I2	VT_I2
Description	Teplota v místnosti11	Teplota v místnosti12	Teplota v místnosti13	Teplota v místnosti14
Node Name	128.7.16.123	128.7.16.123	128.7.16.124	128.7.16.124

Agregace				
	Teplota1	**Teplota2**	**Teplota3**	**Teplota4**
Průměr	18,8	1738	17,8	19
Delta	0	1	2	2
Minimum	17	17	16	17
Maximum	21	20	19	20
Rozsah	4	3	3	3

5.5.2 Aktualizování hodnot

OPCHDAServer objekt používá synchronní a asynchronní metody aktualizování hodnot, tudíž máme na výběr ze synchronního a asynchronního rozhraní.

Jednotlivé možnosti aktualizování dat:

- Vkládání hodnot. Hodnoty je možné vkládat do všech položek: proměnná, časová známka a kvalita. Není možné vkládat hodnoty do již uložených položek.
- Výměna hodnoty. Je možná jen u existujících dodnot.

- Vložení/výměna hodnoty. Kombinace předchozích možností. Když položka neexistuje je hodnota vložena a když existuje je zaměněna.
- Smazání hodnoty. Hodnoty jedné, nebo více položek mohou být občas na základě časové známky smazány.

Komentáře k hodnotám

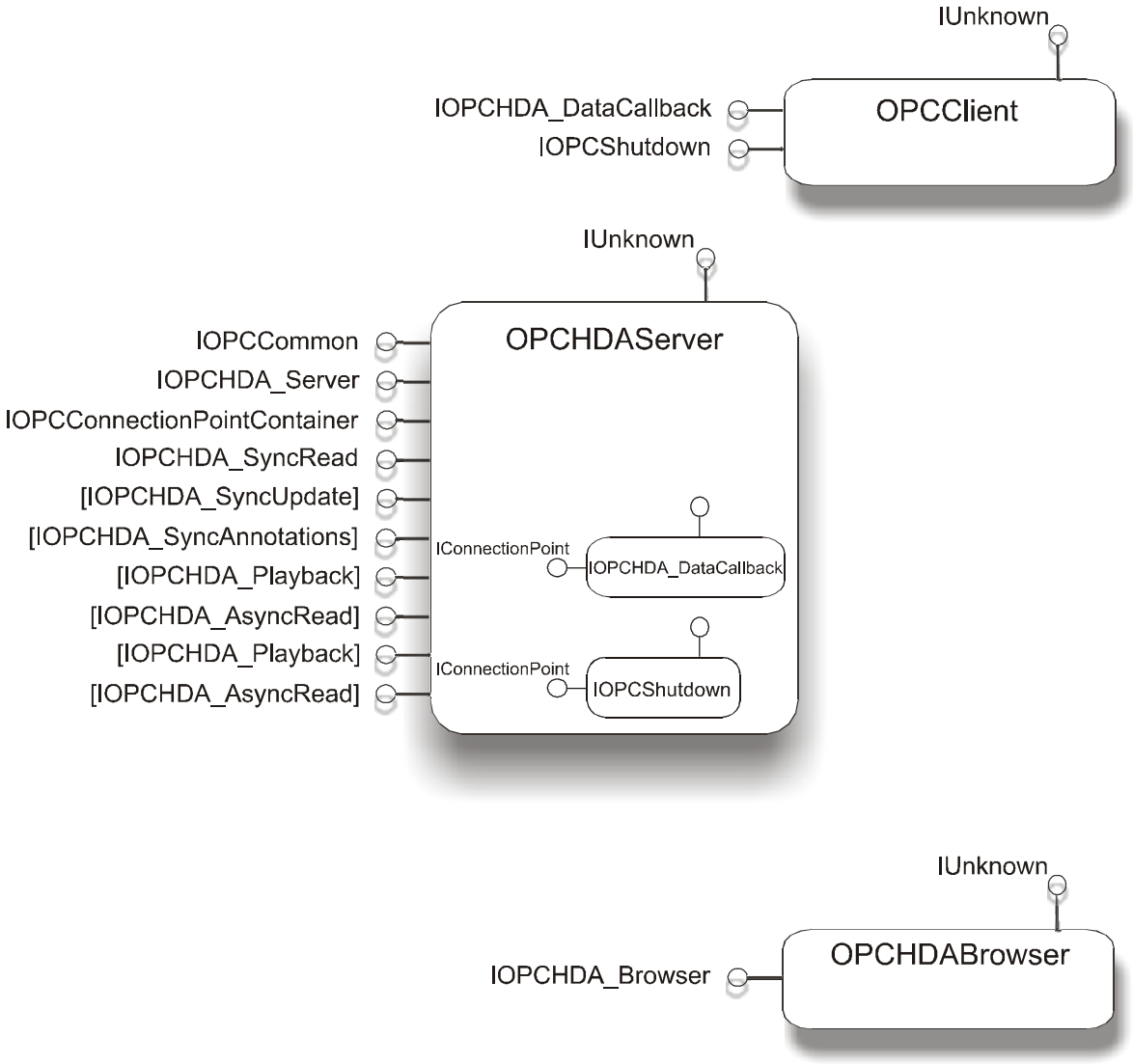
OPCServer objekt používá synchronních a asynchronních metod (nebo obojí) pro přístup k komentářům. To umožňuje čtení a vkládání komentářů. [8]

5.5.3 Záznam hodnot

Je možné používat pouze asynchronní rozhraní. Při prvním vyvolání obdrží klient inicializační data a potom jsou data zasílaná v specifických časových intervalech.

- Čtení zaznamenaných dat. Klient rozhodne, které hodnoty, časové rozsahy a obnovovací frekvence chce doručit. Klient také určuje rychlost s jakou se budou data ukládat. Na výběr má několik možností jako ukládání v reálném čase (real time), velmi rychle, velmi pomalu.
- Čtení procesních dat. Možnosti jsou shodné s čtením zaznamenaných dat.

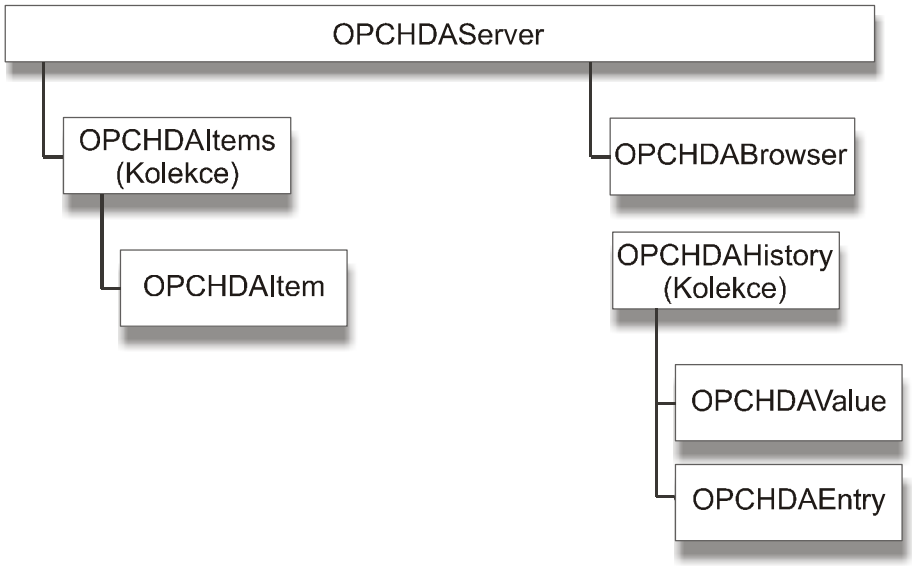
HDA server využívá pouze dvou objektů. Všechna rozhraní k historickým datům jsou soustředěna v *OPCHDAServer* ,viz. (Obr 21). [8]



Obr. 22 Historical Data Access server a klient rozhraní

5.5.4 Automatizační rozhraní

Obrázek (Obr. 23) zobrazuje objektový model definovaný specifikací. *OPCHDAItems* je kolekce obsahující *OPCHDAItems* vytvořený klientem. Při používání metod tohoto objektu mohou být ovládány a čteny jeho vlastnosti. *OPCHDAHistory* je opět kolekce, která obsahuje další *OPCHDAValue* nebo *OPCHDAEntry* objekty a reprezentující historická data. Tento objekt obsahuje výsledky volání, používání a přístupů k objektu *OPCHDAItem. OPCHDAValue* představuje konkrétní hodnoty a *OPCHDAEntry* obsahuje jejich atributy. [8]



Obr. 23 Automatizační rozhraní HDA serveru

5.6 OPC Batch

Další OPC specifikace již nejsou tak rozšířené a nemusí být nutně implementované. Na rozdíl od specifikací Data Access, Alarms and Events a Historical Data Access nepopisují žádné nové kompletní rozhraní mezi klientem a serverem. OPC Batch definuje dodatek specifikace Data Access pro dávkově (recepturové) řízené procesy. Dávky se skládají z různých receptur charakterizující výsledný produkt (toto řízení se hojně využívá v potravinářském, nebo farmaceutickém průmyslu). Po spuštění dávky dojde k výměně dat mezi serverem a zařízením. Server odešle recepturu a obdrží zprávu o průběhu procesu. OPC Batch je univerzální specifikace pro dávkové procesy. [9]

OPC Batch server je charakteristický následujícími komponenty:

- Podporu je všechny nadřízené rozhraní definované Data Access specifikaci 3.0
- Podporuje další OPC rozhraní definované OPC Batch specifikací.
- Přesně definuje jmenný prostor. Jelikož se u dávkového řízení vytváří a posílají velké objemy informací, je i jmenný prostor velmi rozsáhlý.
- Hodnoty položek v jmenném prostoru je možné z OPC Batch serveru načítat a volitelně i zapisovat.

5.7 OPC Security

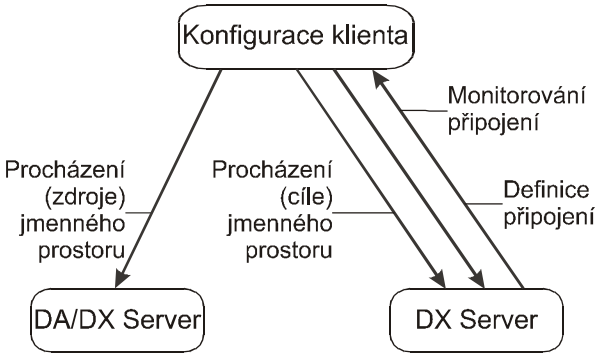
OPC klient a OPC server nemusí běžet na stejném počítači (pomocí technologie DCOM), proto musí být dodržena bezpečnostní politika komunikace k vyloučení neautorizovaného přístupu do systému. To řeší specifikace OPC Security dvěmi způsoby:

- Deklarovaná bezpečnost, používá obslužný program DCOMCFG.
- Programová bezpečnost, používá metod DCOM-Security-API

Oba způsoby mají své rozšíření a omezení. Největší překážkou v použití DCOMCFG je, že všechny definice jsou dále použity pro všechny DCOM komponenty na počítači, nebo pro jedinou komponentu. Řešením je použití programové bezpečnosti, která je více flexibilní.

Bezpečnostní specifikace definuje rozhraní s metodami, ale nedefinuje nové servery ani OPC objekty. Tato rozhraní mohou být začleněny do serverů. [10]

5.8 OPC Data eXchange (DX)



OPC Data eXchange (DX) server slouží k přímé vzájemné horizontální datové komunikaci mezi jednotlivými servery. Tato komunikace je prováděna přímo ze zdroje v serveru do cíle v jiném serveru. Datové zdroje jsou umístněné každý zvlášť v DA Serveru, nebo DX Serveru a datové cíle jsou umístněné v DX Serveru. DX server je implementován jako součást DA serveru. [13]

5.9 OPC XML DA

Při používání OPC XML DA rozhraní, mohou být data čtena, zapisována a v jednoduché formě také monitorována přes internet. K tomu slouží protokol SOAP (Simple Object Access Protocol). SOAP poskytuje jednoduché a transparentní mechanismy pro výměnu struk-

turovaných a zapsaných informací mezi počítačem a distribuovaným prostředím. Výměna OPC dat prostřednictvím internetu je platformě nezávislá a probíhá pomocí HTTP protokolu rozhraním SOAP. Rozhraním pro OPC komunikaci prostřednictvím internetu je XML.

XML (eXtensible Markup Language). Je další ze značkovacích jazyků. Tento jazyk lze využít i v jiných odvětvích nejen na webu. Ale ve spojení s internetem se o něm mluví jako o nástupci HTML. Co nám XML nabízí? XML nám umožní si navrhnout vlastní tagy (každému elementu dáte svůj vlastní styl), ale nyní můžete pomocí elementů popisovat jejich obsah.

Datový přístup používaný OPC XML má řadu funkcí, které jsou založeny na OPC DA specifikaci. Pomocí XML se posílají pouze jednoduché zprávy jako například měřené hodnoty. Tato komunikace nepředpokládá přenos řídících hodnot, které by byli na úrovni DCO OPC DA rozhraní, kvůli možné ztrátě internetového spojení. [12]

II. PRAKTICKÁ ČÁST

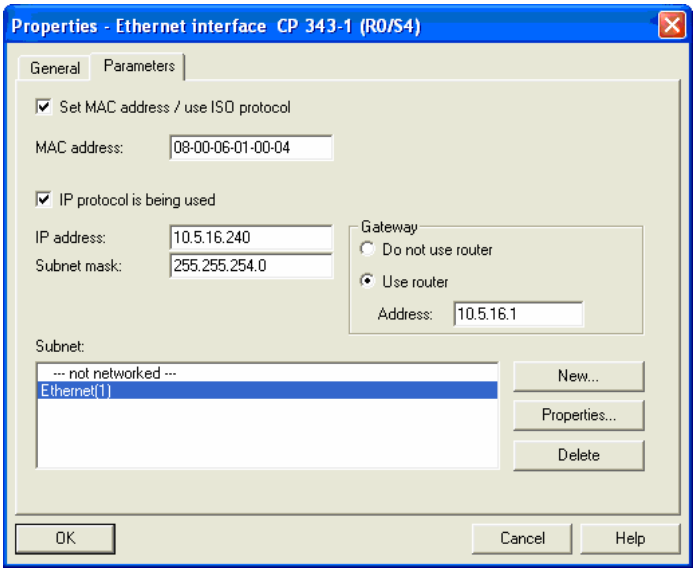
6 PROPOJENÍ IPC S PLC POMOCÍ OPC

Celá praktická část je realizovaná v rámci Laboratoře integrované automatizace (LABI). Cílem projektu „Laboratoře integrované automatizace“ (dále jen LABI) je vybudovat moderní laboratoře se vzdálenými reálnými výukovými experimenty (dále jen úloha) přístupnými lokálně i dálkově přes internet. LABI slouží pro studium v předmětech souvisejících s průmyslovou automatizací a s aplikovanou informatikou. Mohou je využívat studenti i přednášející a stanou se součástí globálního řešení laboratoří na síti internet.

LABI jsou vybudovány na fakultě automatizace a informatiky Univerzity Tomáše Bati ve Zlíně v rámci projektu „Laboratoře integrované automatizace“ FRVŠ, č. 663/2005/A/a..

6.1 Fyzické propojení

Fyzicky je propojení realizováno prostřednictvím sítě LAN standardu ethernet 100BASE-T. Jedná se o propojení ethernetové karty Siemens SIMATIC NET, CP 343-1 a běžné síťové karty počítače na kterém běží OPC server. Ke komunikaci je používaný protokol TCP/IP. Z toho také vyplívají požadavky na nastavení síťových parametrů. Nejdůležitější je nutnost přiřadit každému zařízení (každé ethernetové kartě PLC) vlastní IP adresu, která bude ve stejném segmentu sítě jako je OPC server. Dále musíme nastavit masku sítě, ta bude na všech stanicích v síti stejná (v našem případě 255.255.254.0). Jako poslední musíme nastavit adresu routeru pro přístup k úloze do a ze sítě internet. Pro příklad je zde uvedeno nastavení síťové komunikační karty pro úlohu DE4.



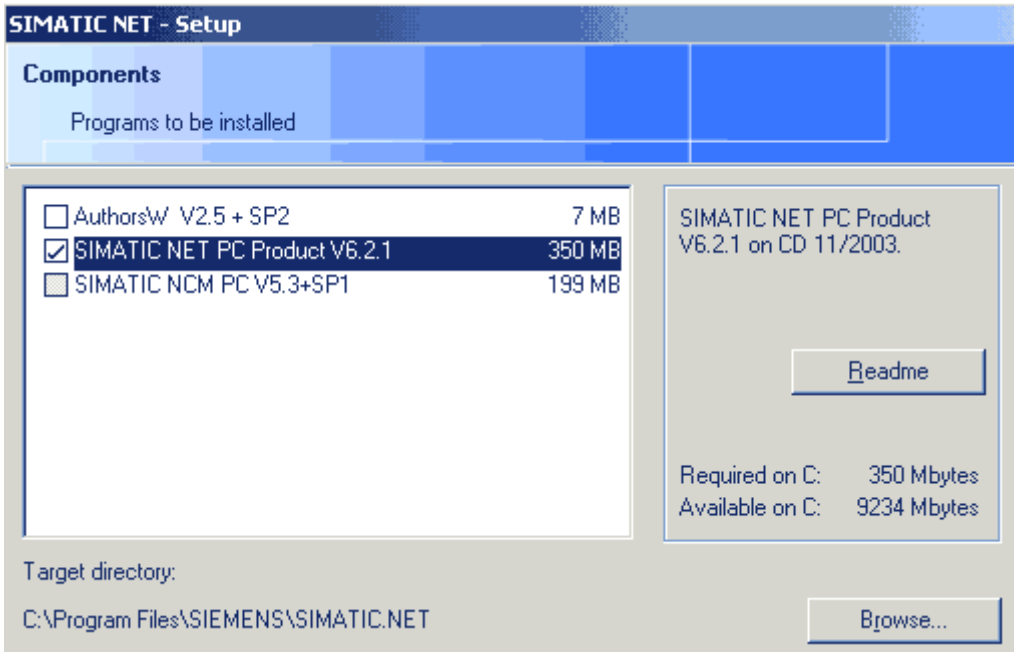
Obr. 24 Nastavení síťových parametrů

V rámci projektu LABI se vyskytuje více úloh jejichž řídící systémy jsou propojeny na OPC server je k fyzickému propojení úloh použito také dalších síťových prvků jako přepínače (switch) a samotný OPC server je napojený prostřednictvím routeru do sítě internet. Více o nastavení síťového spojení mezi PLC kartou a OPC serverem je popsáno v kapitole 6.2 Nastavení komunikace mezi PLC a OPC serverem OPC.simatic.NET.

6.2 Nastavení komunikace mezi PLC a OPC serverem OPC simatic.NET

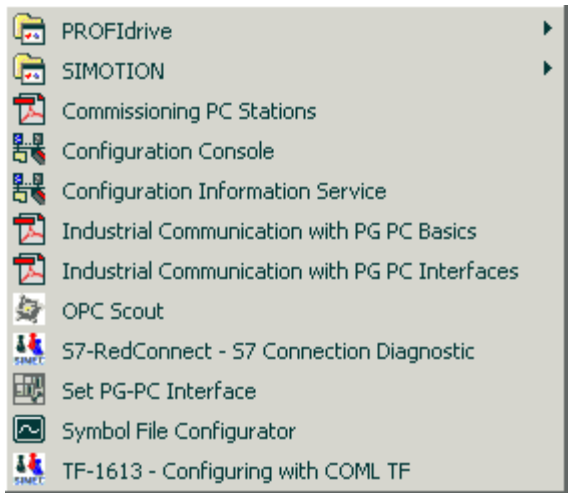
6.2.1 Instalace Simatic.NET

Po spuštění instalace se nám zobrazí úvodní okno, ve kterém si můžeme zvolit součásti instalace. Ta se skládá ze tří součástí. První je AuthorsW, která slouží k autorizaci produktu. Bez nainstalování a registrace této součásti nám bude prostředí Simatic.NET běžet pouze ve zkušebním režimu. Druhou částí je samostatný Simatic.NET. Třetí část Simatic NCM je určena pro systémy na kterých je používaná verze STEP7 nižší než 5.3, nebo na kterých není STEP7 vůbec nainstalovaný. [14]



Obr. 25 Instalace Simatic.NET

Po ukončení instalace je vyžadovaný restart počítače. Po restartování se nám na ploše zobrazí ikona konfigurace stanice, která slouží jako OPC server. Další instalované součásti prostředí Simatic.NET si můžeme prohlédnout v nabídce START.

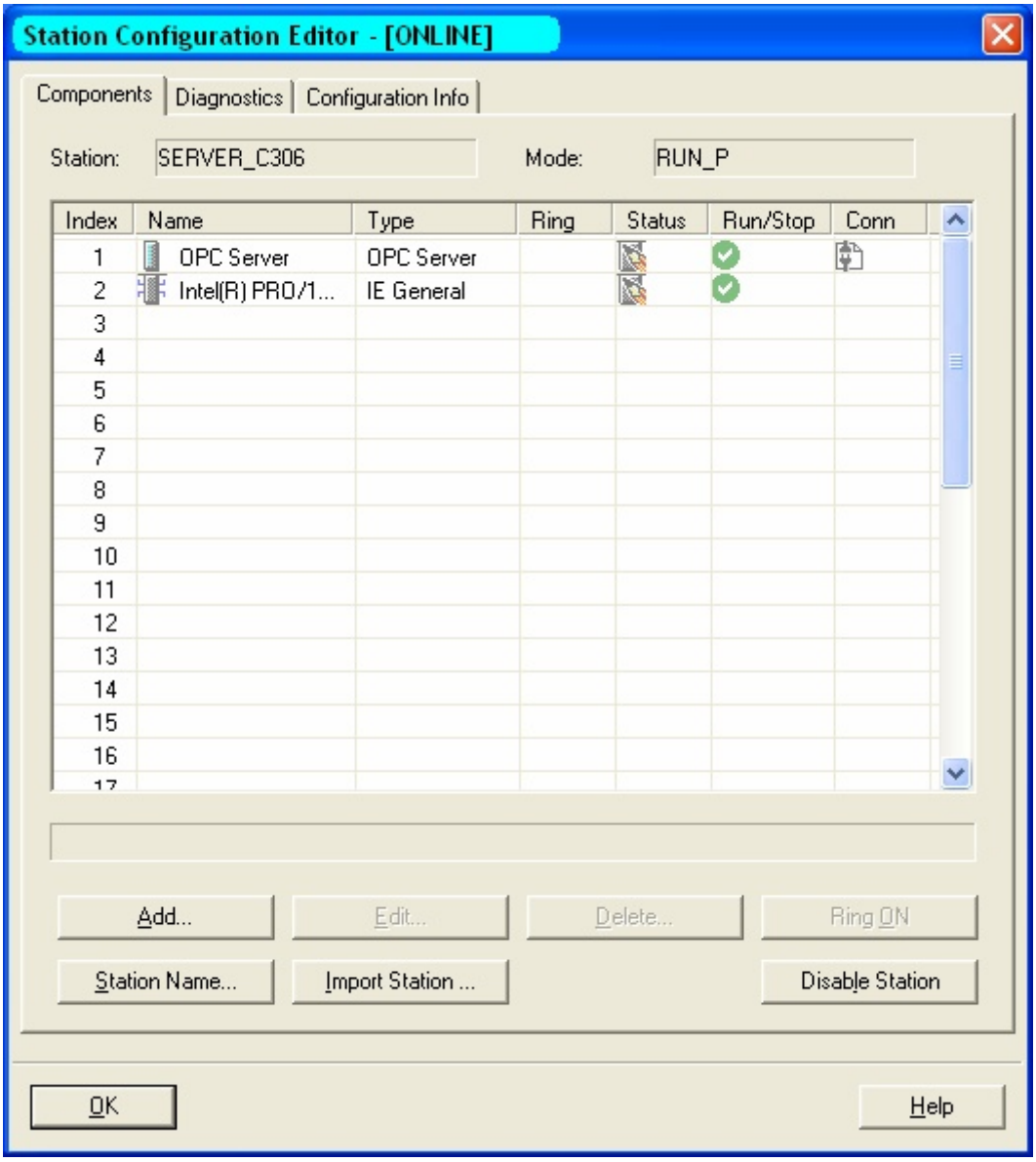


Obr. 26 Nainstalované součásti

Od této chvíle je počítač na kterém jsme provedli instalaci OPC serverem.

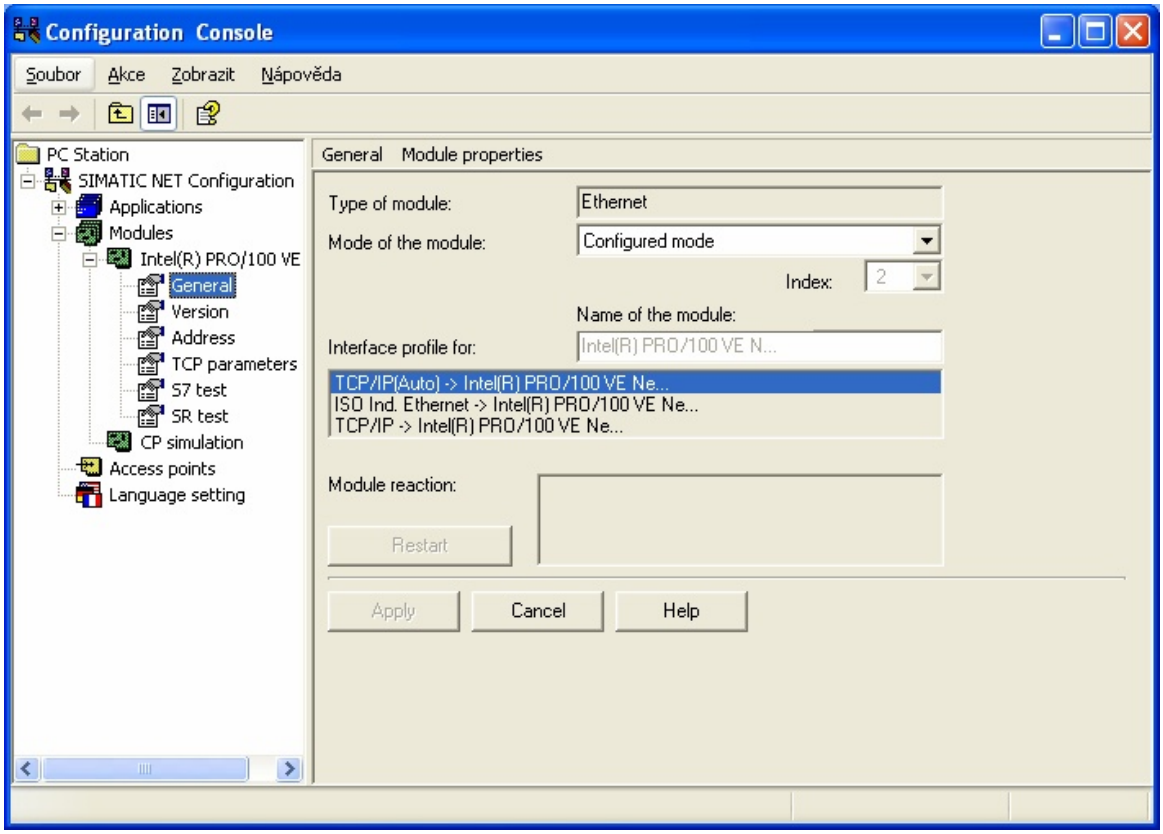
6.2.2 Konfigurace OPC serveru Simatic.NET

Nejprve spustíme Station configurator, kde nastavíme komponenty stanice - OPC server a komunikační kartu, v našem případě sítovou.

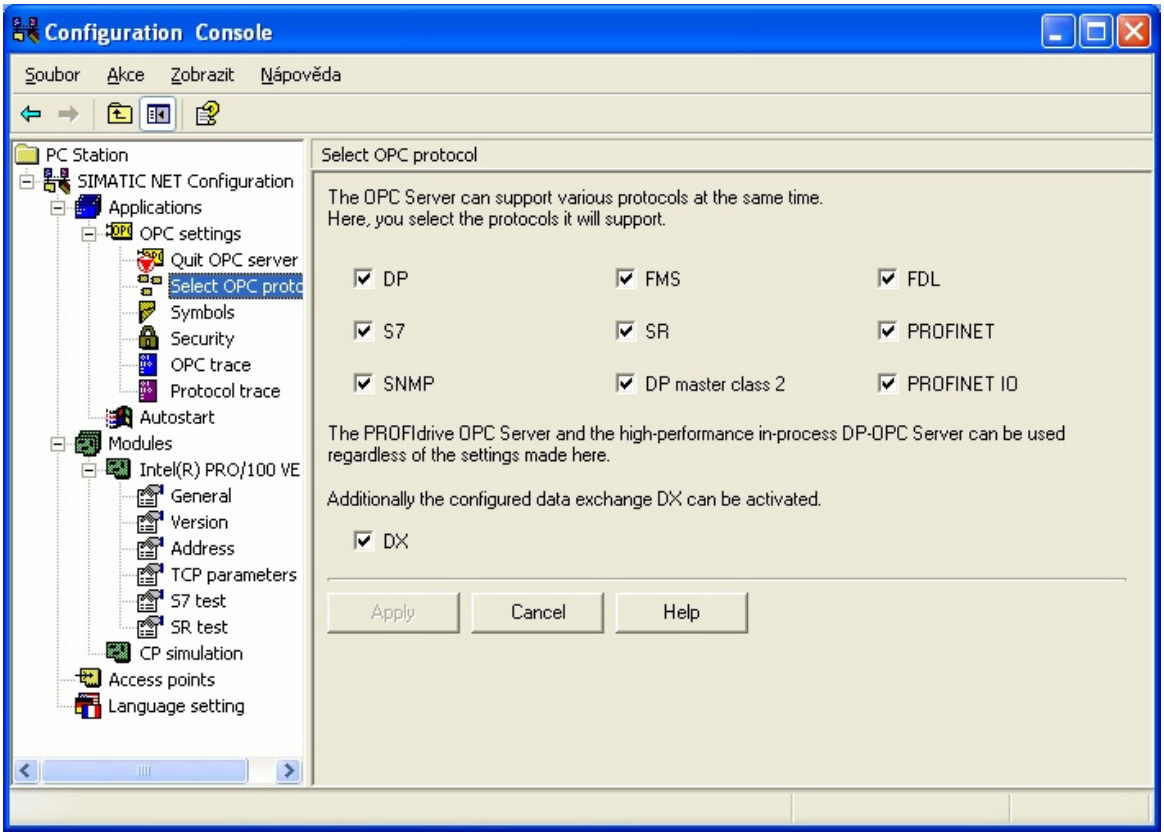


Obr. 27 Station configuration editor

V dalším kroku konfigurace je třeba spustit přes nabídku START Configuration Konsole a nastavíme parametry OPC. Důležité je nastavit „Mode of the module“ na Configured mode a potom vybrat jednotlivé komunikační protokoly, které budeme používat. Taky zde můžeme zkontrolovat nastavení sítě, jako je IP adresa, maska sítě, brána sítě, DHCP server a taky otestovat spojení mezi serverem a PLC. [14]



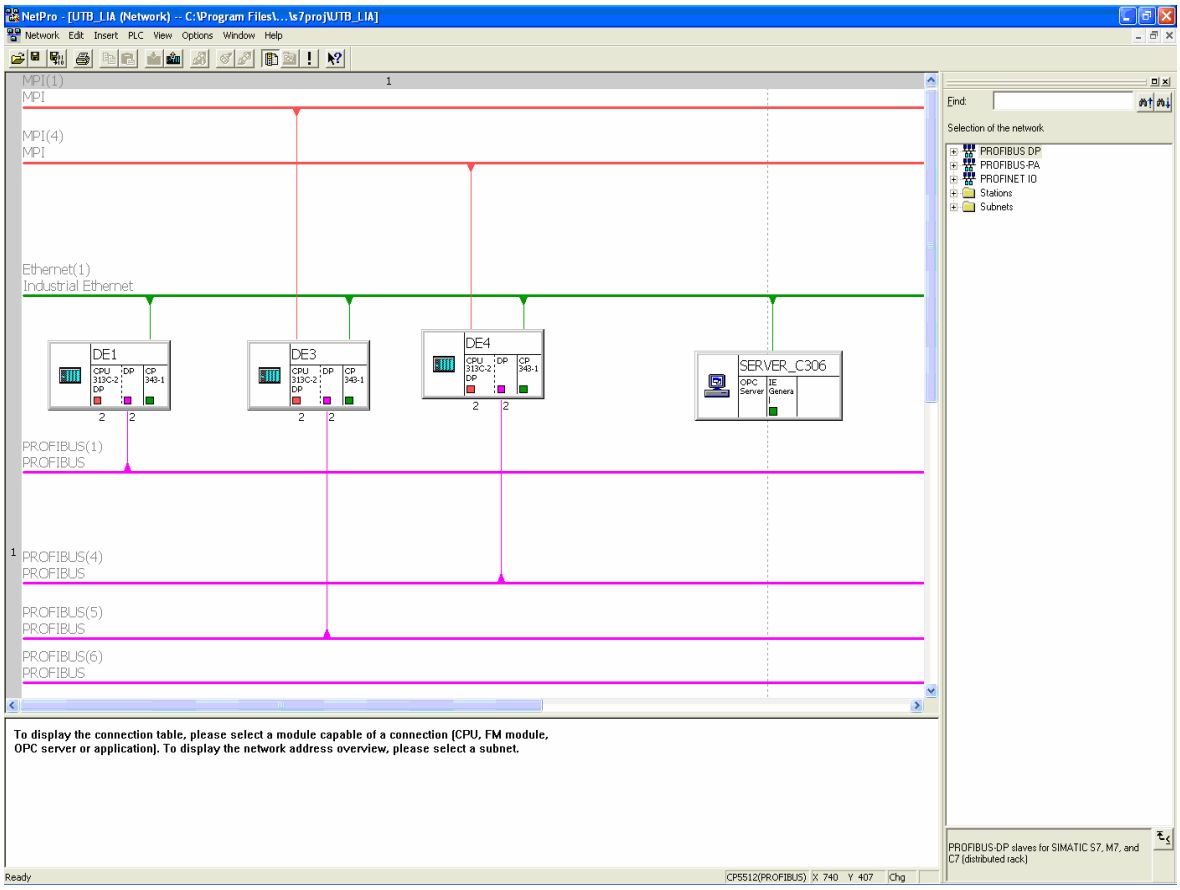
Obr. 28 Konfigurace parametrů propojení



Obr. 29 Konfigurace parametrů propojení – výběr komunikačních protokolů

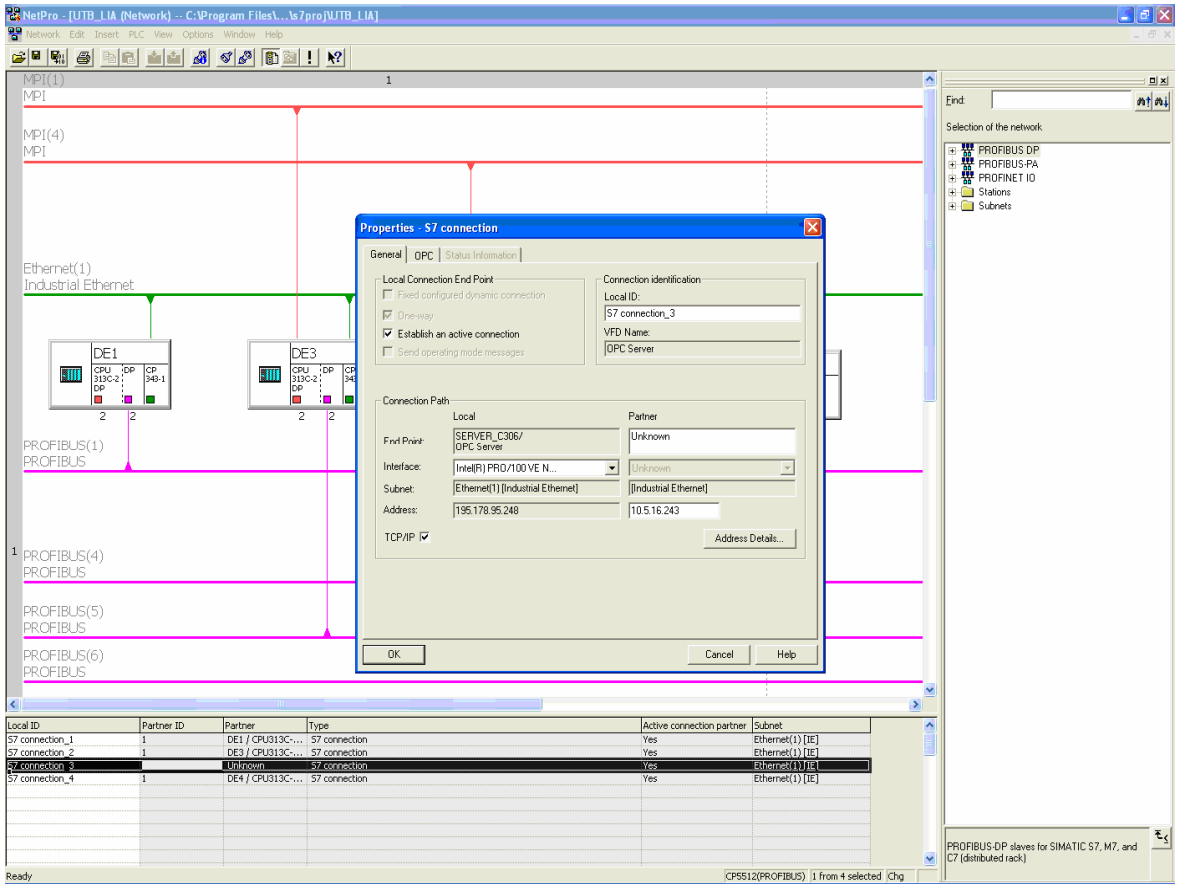
V dalším kroku spustíme Step7 Simatic Manager, nemáme-li jej, nainstalujeme si Simatic NCM PC V5.3+SP1 a spustíme ten. Založíme nový projekt (např OPC_server) a vložíme SERVER_C306 to bude naše stanice. Tato stanice se musí jmenovat naprosto stejně jako počítač, to je důležité (název zjistíte např. ve Station Configuratoru, jinak se Vám nepodaří nahrát do ní konfiguraci.

V HW konfiguraci vložíme OPC Server a komunikační kartu. Přiřadíme jí připojení na ETHERNET(1). Z HW konfigurace spustíme "Configure Network". Otevře se program NetPro a v naší stanici vidíme přiřazené spojení na ETHERNET linku. [14]



Obr. 30 Struktura sítě v programu NET PRO

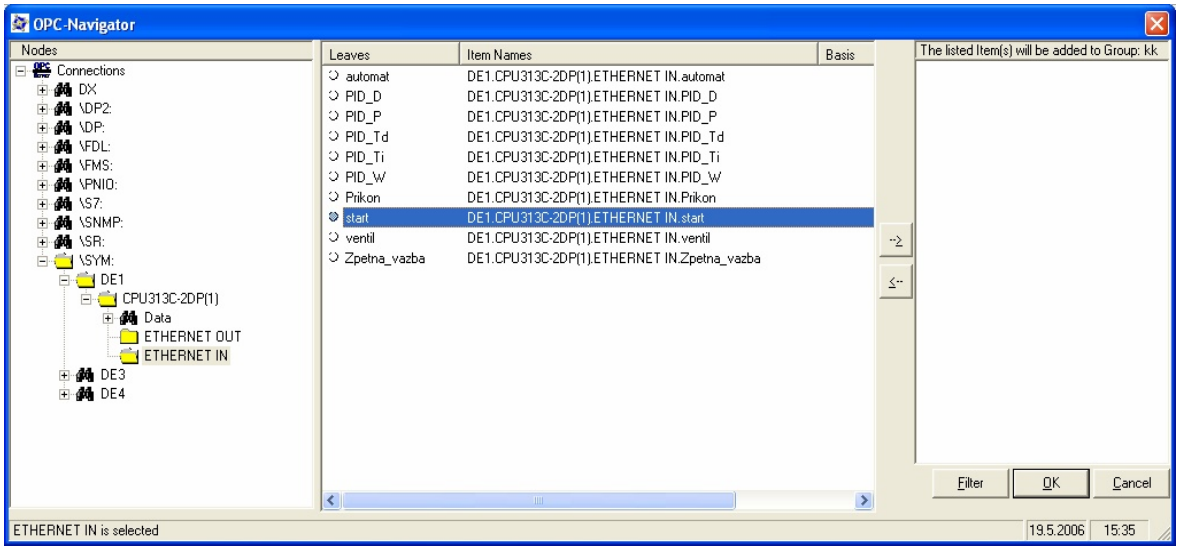
Potom zvýrazníme OPC Server a do řádku dole přiřadíme spojení z lokální (naší) stanice na "Unknown" partnera. Nastavíme ve vlastnostech připojení (S7 connection_3) adresu PLC**,** které bude připojeno k OPC. Nyní musíme nastavit IP adresu PLC stanice (v našem případě 10.5.16.243).



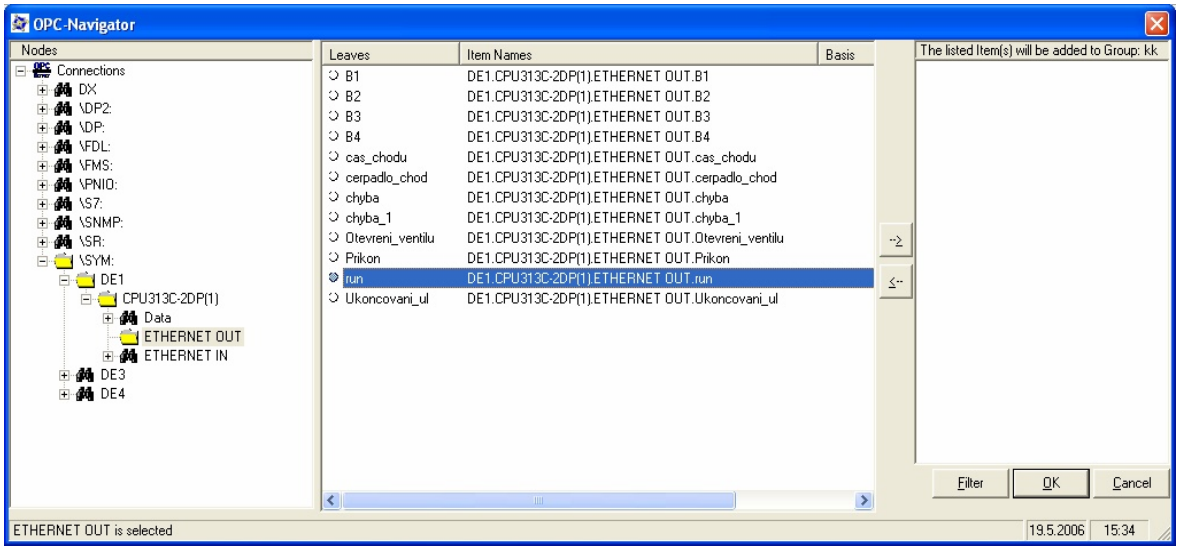
Obr. 31 Nastavení síťové komunikace

Vše uložíme a zkompilujeme. Tím se nám vytvoří hardwarová konfigurace, která je uložena v binárním souboru s příponou *.xdb. Tento soubor nahrajeme jako konfiguraci do OPC serveru a také ji importujeme jako hardwarovou konfiguraci do PLC stanice S7. Tím je konfigurace ukončena. Nyní ověříme námi nastavenou konfiguraci pomocí jednoduchého klienta obsaženého v Simatic.NETu.

Vrátíme se do hlavní nabídky a otevřeme OPC Scout. Je to jednoduchý OPC klient, který slouží ke zkontrolování funkčnosti OPC serveru. Tady už můžeme vidět dostupné servery, nás zajímá OPC.SimaticNET na který se po kliknutí připojí naše PLC. Dalším klikem otevřeme "OPC-Navigator" který načte strukturu připojeného PLC. Následující obrázky ukazují vstupní a výstupní proměnné úlohy DE1, tak jak jsou definované v OPC serveru a jak je vidí OPC klient. Pokud některou proměnnou vybereme a šipkou ji přetáhneme do pravého sloupce, tak můžeme přímo vidět aktuální hodnotu v proměnné.



Obr. 32 OPC Scout – vstupní hodnoty



Obr. 33 OPC Scout – výstupní hodnoty

6.3 Přenos dat mezi OPC serverem a klientem Control Web

OPC standard definuje řadou způsobů komunikace, aby vyhověl různým požadavkům klientů:

- Synchronní komunikace vždy čekající na přenos dat z/do zařízení.
- Synchronní komunikace pracující s vyrovnávací pamětí serveru (cache).
- Asynchronní komunikace (vždy komunikuje se zařízením).
- Periodická komunikace serveru se zařízením a zpětné volání klienta při změně dat.

Druhý a čtvrtý způsob komunikace předpokládá, že server sám dokáže vyvolávat komunikaci s periferií a buď jen ukládat přečtená data do vyrovnávací paměti (odkud mohou být synchronně čtena) nebo předávat data klientovi zpětným voláním (jeden ze způsobů asynchronní komunikace).

Ačkoliv OPC ovladač pro Control Web pracuje se servery verze 1 i verze 2, u serverů verze 1 používá výhradně synchronního přenosu dat. Asynchronní přenos dat v OPC verze 1 trpěl řadou nedostatků a mnohé servery jej nepodporovaly. To je také důvod, proč byl asynchronní přenos dat v OPC verze 2 zcela přepracován a používá se jiných rozhraní umožňujících výrazně efektivnější komunikaci.

Použití synchronního přenosu se neprojevuje negativně na průchodnosti aplikace systému Control Web - OPC ovladač z hlediska systému Control Web pracuje vždy asynchronně a neblokuje aplikaci čekáním na ukončení synchronního čtení a zápisu dat.

U OPC serverů verze 2 je implicitně používán asynchronní přenos dat a synchronní přenos je použit jen pokud je to explicitně vyžádáno Problematika přenosu dat v OPC je podstatně složitější a je kompletně popsána v kapitole „OPC Data Access“. Nicméně k použití OPC ovladače pro Control Web není znalost detailů způsobu komunikace v OPC naprosto zapotřebí, OPC ovladač používá vlastností OPC přesně odpovídající požadavkům na chování ovladače systému Control Web a uživatel má zaručeno, že všechny mechanismy komunikace systému Control Web (kvalita přenosu, časové známky, prodlevy komunikace apod.) budou převedeny do OPC a zpět zcela transparentně. [15]

6.3.1 Konfigurační soubory OPC ovladače

Jako každý jiný ovladač v systému Control Web i OPC klient potřebuje dva konfigurační soubory — parametrický soubor (.PAR) a mapovací soubor (.DMF — Driver Map File). Protože mapovací i parametrický soubor velmi závisí na použitém OPC serveru, je k dispozici nástroj s grafickým rozhraním umožňující pohodlnou tvorbu obou souborů. Uživatel tak vůbec nepotřebuje znát strukturu těchto souborů, stačí vybrat OPC server, označit položky, které mají být komunikovány jako kanály, a o generování konfiguračních souborů se postará program automaticky. Použití konfiguračního nástroje je popsáno v kapitole 6.3.2.

6.3.1.1 Mapovací soubor

Mapovací soubor (driver map file) je textový soubor obsahující popis dostupných kanálů ovladače, jejich typů a směrů. Soubor *.dmf je součástí definice ovladače a Control Web jej používá při překladu kanálů pro kontrolu směru a typů. *.dmf soubor je obyčejně součástí dodávky ovladače a zpravidla vymezuje výrobcem doporučený rozsah kanálů. Některé ovladače navíc samy dokáží *.dmf soubor sestavit podle zadané konfigurace. Soubor pro úlohu DE1 – regulace teploty vypadá následovně:

```
OPCDRV DMF soubor vytvořen konfigurátorem OPC ovladače
Soubor: C:\Program Files\Moravian Instruments\Control Web 5 Runtime CZE\Prog\HTTP\s200.dmf
Vytvořen: 01/30/2006 16:09:27
begin

1 boolean bidirectional

101 integer bidirectional
102 integer bidirectional
103 integer bidirectional
104 integer bidirectional
105 boolean bidirectional
106 boolean bidirectional
107 boolean bidirectional
108 integer bidirectional
109 integer bidirectional
110 boolean bidirectional
111 boolean bidirectional
112 integer bidirectional
113 integer bidirectional
114 integer bidirectional
115 integer bidirectional
116 integer bidirectional
117 integer bidirectional
118 integer bidirectional
119 integer bidirectional
120 boolean bidirectional
121 boolean bidirectional
122 integer bidirectional
```

Každý řádek definuje nejprve číslo kanálu ovladače (případně interval čísel), dále pak datový typ kanálu a nakonec směr kanálu. Pokud kanál definovaný v aplikaci neodpovídá údajům v *.dmf hlásí Control Web chybu překladu.

6.3.1.2 Parametrický soubor

Parametrický soubor ovladače OPC klient má strukturu odpovídající konvencím .INI souborů, což jsou prosté textové soubory se jmény sekcí uzavřenými v hranatých závorkách a s položkami v sekcích v podobě řádků `klíč=hodnota`. OPC klient vyžaduje v parametrickém souboru dvě sekce: [server] a [channels].

Sekce [server]:

Tato sekce musí obsahovat identifikátor třídy serveru, tzv. CLSID (Class Identifier). CLSID je globálně unikátní identifikátor (GUID — Globally Unique Identifier), 128-bitové číslo používané v COM k jednoznačné identifikaci tříd, rozhraní, typových knihoven apod. Formát textového zápisu tohoto čísla je definován standardem COM a odpovídá hodnotě klíče v registrační databázi (Registry) systému Windows v klíči HKEY_CLASSES_ROOT\CLSID, kde jsou registrovány všechny v systému instalované COM komponenty. V našem případě je identifikátor třídy serveru následující:

```
CLSID={B6EACB30-42D5-11D0-9517-0020AFAA4B3C}
```

Pokud má být server vytvořen na vzdáleném počítači, je nutno definovat parametr host. Tento parametr obsahuje UNC jméno serveru, na němž je požadovaný OPC server instalován.

```
host=\\remotehost
```

Jak již bylo řečeno, OPC ovladač implicitně komunikuje synchronně s OPC servery verze 1 a asynchronně s OPC servery verze 2 (pokud server verze 2 asynchronní komunikaci nepodporuje, je použita synchronní komunikace i v tomto případě). Zatímco se servery verze 1 ovladač nedokáže komunikovat asynchronně, u OPC serverů verze 2 je možné ovladači předepsat použití synchronního rozhraní řádkem:

```
sync=true
```

Vynechání tohoto parametru nebo uvedení `sync=false` znamená, že OPC ovladač upřednostní asynchronní komunikaci, pokud je to možné.

Jestliže použitý OPC server potřebuje určitou dobu po startu k dokončení konfigurace a přitom tento stav nesignalizuje pomocí stavů "nekonfigurován" a "server běží", je možné prostě vložit do startu prodlevu. Parametr wait_time udává počet sekund startovací prodlevy:

```
wait_time=2.5
```

Sekce [channels]:

Tato sekce obsahuje definice jednotlivých kanálů vždy v podobě řádku:

```
číslo_kanálu=identifikátor položky OPC serveru
```

Pokud identifikátor položky OPC serveru označuje pole, na místě čísla kanálu je interval čísel zahrnující celkový počet prvků pole. První a poslední číslo kanálu jsou odděleny znakem '-' (číslo kanálů nemůže být záporné a tak znak '-' není interpretován jako znaménko).

```
první_číslo_kanálu-poslední_číslo_kanálu=identifikátor položky pole
OPC serveru
```

Protože na identifikátor položky nejsou ve standardu OPC kladena žádná omezení, jsou za identifikátor považovány všechny znaky za prvním rovnítkem až po konec řádku. Definice kanálu nevyžaduje žádné další údaje (např. typ dat nebo směr přenosu), protože OPC rozhraní umožňují všechny informace získat za běhu serveru.

Parametrický soubor pro komunikaci s úlohou DE1:

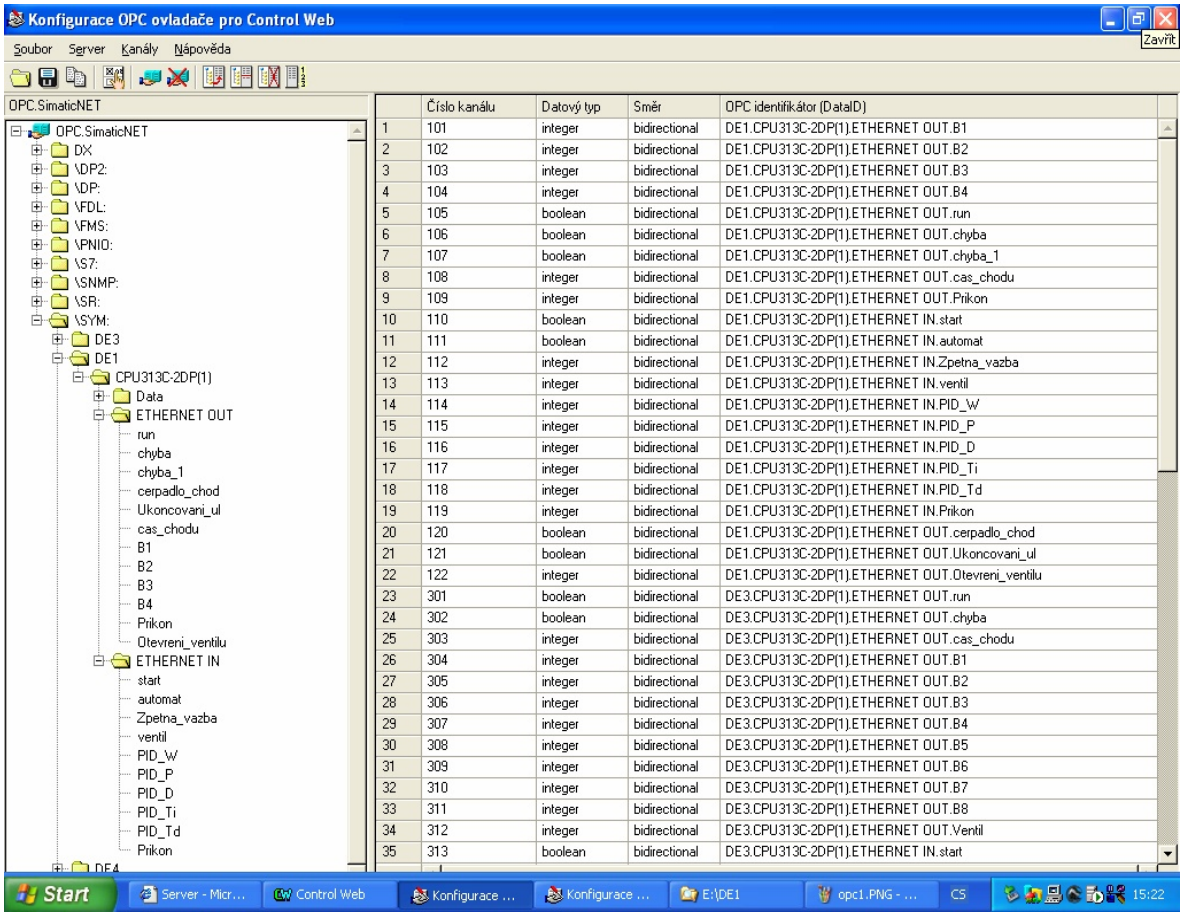
```
[comment]
OPCDRV PAR soubor vytvořen konfigurátorem OPC ovladače
Soubor: C:\Program Files\Moravian Instruments\Control Web 5 Runtime
CZE\Prog\HTTP\s200.par
Vytvořen: 01/30/2006 16:09:23

[server]
CLSID={B6EACB30-42D5-11D0-9517-0020AFAA4B3C}
sync=true
update_rate=1

[channels]
101=DE1.CPU313C-2DP(1).ETHERNET OUT.B1
102=DE1.CPU313C-2DP(1).ETHERNET OUT.B2
103=DE1.CPU313C-2DP(1).ETHERNET OUT.B3
104=DE1.CPU313C-2DP(1).ETHERNET OUT.B4
105=DE1.CPU313C-2DP(1).ETHERNET OUT.run
106=DE1.CPU313C-2DP(1).ETHERNET OUT.chyba
```

```
107=DE1.CPU313C-2DP(1).ETHERNET OUT.chyba_1
108=DE1.CPU313C-2DP(1).ETHERNET OUT.cas_chodu
109=DE1.CPU313C-2DP(1).ETHERNET OUT.Prikon
110=DE1.CPU313C-2DP(1).ETHERNET IN.start
111=DE1.CPU313C-2DP(1).ETHERNET IN.automat
112=DE1.CPU313C-2DP(1).ETHERNET IN.Zpetna_vazba
113=DE1.CPU313C-2DP(1).ETHERNET IN.ventil
114=DE1.CPU313C-2DP(1).ETHERNET IN.PID_W
115=DE1.CPU313C-2DP(1).ETHERNET IN.PID_P
116=DE1.CPU313C-2DP(1).ETHERNET IN.PID_D
117=DE1.CPU313C-2DP(1).ETHERNET IN.PID_Ti
118=DE1.CPU313C-2DP(1).ETHERNET IN.PID_Td
119=DE1.CPU313C-2DP(1).ETHERNET IN.Prikon
120=DE1.CPU313C-2DP(1).ETHERNET OUT.cerpadlo_chod
121=DE1.CPU313C-2DP(1).ETHERNET OUT.Ukoncovani_ul
122=DE1.CPU313C-2DP(1).ETHERNET OUT.Otevreni_ventilu
```

6.3.2 Konfigurační nástroj OPC ovladače

V praxi je poměrně obtížné zjistit CLSID instalovaných serverů a ještě obtížnější je zjišťovat jména jednotlivých položek v serveru definovaných a zapisovat je do parametrického souboru. Proto je spolu s OPC ovladačem dodáván konfigurační nástroj OpcDrvCf, který jednoduchým způsobem umožňuje zvolit OPC server, vybrat které položky mají být viditelné v aplikaci systému Control Web jako kanály, přiřadit k těmto položkám čísla kanálů a vygenerovat PAR a DMF soubory automaticky. Navíc je možné do schránky (clipboard) uložit textovou podobu zápisů kanálů nadefinovaných v parametrickém souboru a poté je vložit do editoru ve vývojovém prostředí systému Control Web. [15]

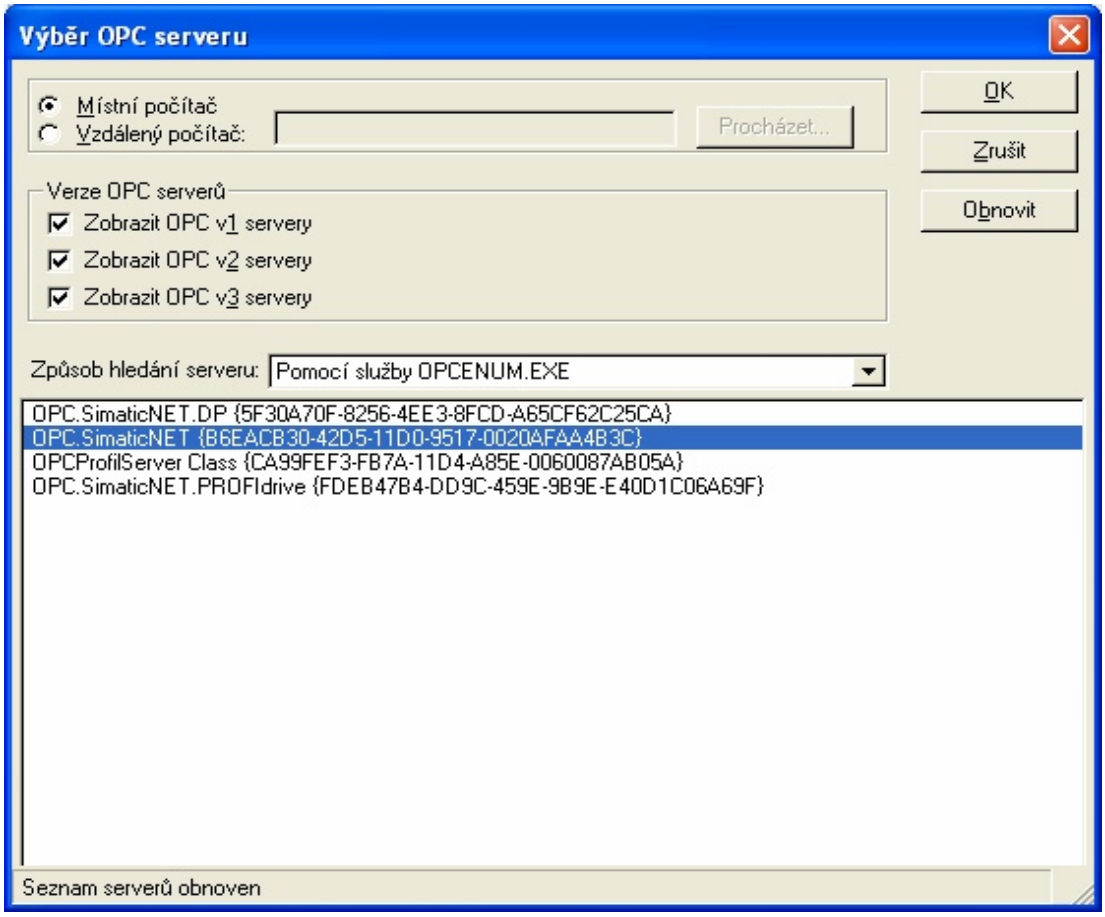
Okno konfiguračního nástroje obsahuje dva panely — v levém panelu je v podobě stromu zobrazen jmenný prostor zvoleného OPC serveru, v pravém panelu jsou zobrazovány vybrané položky.



Obr. 34 Okno konfiguračního nástroje s hodnotami pro úlohu DE1

6.3.2.1 Výběr OPC serveru

Nejprve je nutno vybrat OPC server, se kterým má v aplikaci OPC ovladač komunikovat. Menu *Server/Připojit k OPC serveru* otevře okno výběru OPC serveru.



Obr. 35 Okno výběru OPC serveru

Toto okno zobrazí jména i CLSID všech OPC serverů instalovaných na lokálním nebo na vzdáleném počítači. Je možné omezit seznam zobrazených serverů jen na verzi 1,2 nebo 3. Řada serverů poskytuje rozhraní obou verzí OPC.

Konfigurační nástroj používá k vyhledání OPC serverů mechanismus kategorií COM komponent, který je standardní součástí implementace COM v systémech Windows. Organicace OPC Foundatin definovala dva unikátní identifikátory (GUID) kategorií pro OPC servery verze 1, 2 a 3. V rámci instalace každého serveru je nezbytné danou kategorii zaregistrovat a taktéž instalovaný server k této kategorii přihlásit.

Naneštěstí stále existují servery, jejichž instalace toto pravidlo ignorují. Z těchto důvodů existuje v okně pro výběr serveru možnost volby "Nepoužívat COM kategorie k vyhledání serverů". Pokud je tato volba zvolena, systém nabídne servery nikoliv podle zaručeně unikátních identifikátorů kategorií, ale podle výskytu klíčového slova "OPC" v registrační databázi. Tak se mohou vyhledat i komponenty jež ve skutečnosti nejsou OPC servery a při

pokusu o připojení je hlášena chyba. Nicméně tato volba umožní pracovat i se serverem, který není správně instalován. [15]

6.3.2.2 Výběr kanálů

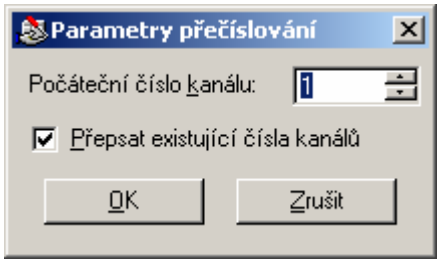
Je-li server úspěšně zaveden, objeví se v levém panelu struktura jeho jmenného prostoru. Z této struktury je možné přidávat jednotlivé položky do tabulky v pravé části aplikace pomocí dvoj-kliku nebo lze označit jednu či více položek najednou a přenést je do tabulky kanálů pomocí menu Kanály/Přidat vybrané položky. Smazání řádku z tabulky je možné po označení celého řádku kliknutím na políčko vlevo od řádku stiskem klávesy <Del>, případně pomocí nástroje Kanály/Smazat vybraný kanál. Také je možné smazat všechny řádky najednou pomocí menu Kanály/Smazat všechny kanály.

Řada serverů po svém startu zpracovává např. vlastní konfigurační soubory či provádí jinou inicializaci. Naneštěstí často servery žádným způsobem nesignalizují, že ještě nejsou připraveny (jejich status je od počátku OPC_STATUS_RUNNING). Pak je docela dobře možné, že zobrazený jmenný prostor bude neúplný. Zpravidla pomůže prostě server vybrat ještě jednou, když už je inicializován.

Konfigurační nástroj se snaží pro každý přidaný řádek (kanál) získat ze serveru také údaje o typu dat a směru komunikace. Naneštěstí existují servery, které tuto informaci neposkytují vůbec nebo ji neposkytují jen pro některé položky (například položky vytvořené jako alias jiných položek). V takovém případě je nutné ručně upravit vygenerovaný DMF soubor.

6.3.2.3 Přiřazení čísel kanálů

Nově přidané řádky mají číslo kanálu prázdné, dosud nedefinované. Čísla kanálů je možné přidat individuálně pro každý řádek nebo pomocí menu Kanály/Přečíslovat kanály.... Přečíslování kanálů může pracovat jen s kanály, jejichž číslo dosud nebylo definováno, případně může přepsat čísla všech kanálů.



Obr. 36 Přiřazení čísla kanálu

6.3.2.4 Přiřazení čísel polím kanálů

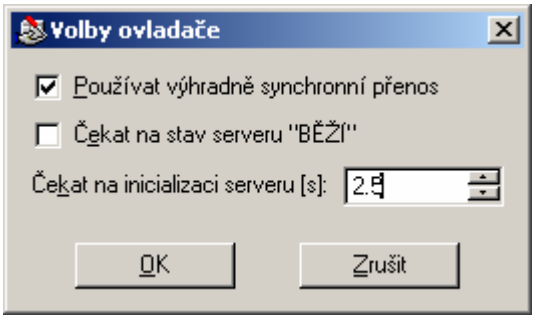
Protože OPC položka typu pole je mapována do řady kanálů a tato řada musí být souvislá, zadává se vždy pouze číslo prvního kanálu v poli. Ostatní kanály jsou číslovány vzestupně od tohoto čísla.

Čísla kanálů musí být unikátní, což platí i v případě polí. Jsou-li např. definovány kanály 1, 4 a 7, pak prvkům pole obsahujícího 3 prvky mohou být přiřazena nejdříve čísla 8 až 10. Nenechte se tedy zmýlit chybovou zprávou „Číslo kanálu redefinováno“, může být způsobena i nedostatkem volných čísel kanálů pro celé pole, i když zadané (první) číslo kanálu není použito.

Systém automatického přidělování čísel kanálů vybere nepřekrývající se čísla i v případě, že se jednotlivé kanály střídají s poli. Pokud ale použijete automatické očíslování a přitom některé kanály již mají číslo přiřazeny, nemusí být čísla kanálů nutně ve spojité řadě (čísla přiřazená polím samozřejmě spojitá budou).

6.3.2.5 Parametry ovladače

Pomocí menu Soubor/Parametry ovladače... lze otevřít dialogové okno umožňující zadání dalších parametrů ovladače — vyžádání synchronní komunikace, čekání na stav „server běží“ či zadání počáteční prodlevy po zpoždění serveru.



Obr. 37 Volba ovladače

6.3.2.6 Generování konfiguračních souborů

Mají-li všechny řádky přidělena čísla, je možné pomocí nástroje Soubor/Zapsat PAR a DMF soubory... uložit parametrický (.PAR) a mapovací (.DMF) soubor.

Pomocí nástroje Soubor/Kopírovat kanály do schránky je možné do schránky (clipboard) uložit textovou podobu zápisu kanálů v aplikaci systému Control Web. Ve vývojovém prostředí Control Web je možné tento text vložit do editoru a případně modifikovat názvy kanálů.

6.3.2.7 Načtení parametrického souboru

Pomocí nástroje Soubor/Otevřít existující PAR soubor lze do konfiguračního nástroje zavést již existující parametrický soubor. Tato akce odpojí aktivní OPC server (pokud je konfigurační nástroj na nějaký OPC server připojen) a vymaže všechny položky v seznamu kanálů. Poté se pokusí spojit s OPC serverem uvedeným v parametrickém souboru a do seznamu kanálů naplní položky v tomto souboru uvedené.

Protože v takovém případě nelze zaručit, že všechny položky uvedené v souboru OPC server skutečně poskytuje, konfigurační nástroj prověří jejich dostupnost. Pokud OPC server zadanou položku nezná, bude v seznamu kanálů uvedena červeně a nebude zařazena do generovaných konfiguračních souborů.

7 OVĚŘENÍ A OTESTOVÁNÍ V SIMULAČNÍM REŽIMU

Ověření a otestování v simulačním režimu probíhalo v rámci projektu Laboratoře Integrované automatizace. Konkrétně na úloze DE1 – regulace teploty.

7.1 Aplikace Control WEB

Control Web je objektově orientovaný systém pro vývoj a provozování vizualizace, měření, řízení, regulaci a komunikační systém pro programy pro sběr, archivaci a zpracování dat. Systém má široký rozsah aplikací které umí řídit stroje nebo technologické linky v reálném čase a souběžně umí být začleněny do rozsáhlého firemního informačního systému. Tento systém může být použit v laboratořích, ve vývojových dílnách a ve školách pro vyučovací účely.

Prostředí systému Control Web umožňuje vytvářet programy, které používají symbolické popisy tvořeného systému. Není potřeba prostudovat všechny pokročilé vlastnosti systému, protože aplikace může být jednoduše rozvíjená vytvářením zobrazovačů a kontrolních panelů a dalších komponent na distribuční desce nebo v řídícím prostoru. Jsou zde používány virtuální modely místo reálných nástrojů.

Pomocí grafického prostředí může být aplikace velmi jednoduše vytvořena uspořádáním objektů pomocí myši na pracovní ploše. Jednoduše se uchopí vhodná ikona, která je grafickým znázorněním požadovaných nástrojů a vloží se do struktury rozvíjené aplikace. Data týkající se vloženého nástroje jsou automaticky převedeny do zdrojového textu aplikace.

Systém umožňuje používat několik způsobů vývoje (je možné dostat stejný výsledek použitím různých přístupů):

- vizuálním programováním – je založeno na vytváření aplikací s pomocí myše, grafických symbolů, algoritmů a vizuálních struktur
- textovým programováním – je založeno na zapisování klíčových slov v textovém editoru
- kombinovaným programováním – je založeno částečné na vizuálním a částečné na textovém programování

Při tvorbě aplikačního programu, se používají základní prvky měření a regulace systému. Jsou to:

- konstanty a proměnné – které převádějí data v aplikaci
- vstupní/výstupní ovladače zařízení – které spojují aplikaci s reálným prostředím
- vstupní, výstupní nebo dvojsměrné kanály – které převádějí data mezi aplikací a reálným prostředím prostřednictvím ovladače
- virtuální nástroje – které zajišťují aktuální fungování aplikace a slouží k výkonu různých činností : zobrazení a nastavení hodnot, regulaci, archivaci a ostatních procesů s daty
- panely – speciální typ virtuálního nástroje, slouží pro estetickou a logickou kombinaci více virtuálních nástrojů do jedné jednotky
- časovače – speciální typ virtuálního nástroje, jsou použity pro tvorbu časových algoritmů, které používají sekvence, třídění a opakování

Při práci se systémem je důležité definovat hlavní proměnné (nebo konstanty) a použít ovladače a kanály k určení spojení programu s aktuálním vstup/výstup zařízením (pro znázornění je možné použít také virtuální a modelový ovladač). Potom se zde mohou uspořádat a propojit panely, nástroje, kontroly a ukazatele, regulátory, data managery apod.

Vývojové prostředí s kompilátorem a soubor virtuálních nástrojů je integrován do systému a současně může běžet program a jiné úkoly. Aplikační program může být spuštěn stejně jako ukončen (např. odstraněn z paměti) použitím Control Web IDE (Intergated Development Environment).

Control Web se liší od ostatních systémů hlavně pro následující vlastnosti:

- většina systémů jsou omezeny určitými způsoby (např. je limitován počtem vstup/výstupních kanálů, souběžně zobrazovanými daty atd.). Control Web nemá žádná omezení. Jediný limit je velikost paměti, disku a rychlost počítače.
- většina systémů obsahuje obyčejné konfigurovatelné a uzavřené aplikace. Control web je reálně volně programovaný a otevřený systém bez žádných limitů.

- u většiny systémů je povinné vyhodnocovat určité konfigurační data. Aplikace v prostředí Control Web má tato data v paměti počítače, podobně jako kdyby to bylo naprogramováno v objektově orientovaném programovacím jazyce, např. C++. Control Web proto kombinuje výhody ukládání a rychlost vizuálního programování se znázorněním kódu stroje.

Další vlastnosti:

- Control Web obsahuje vlastní http server, který umožňuje přístup k aplikacím přes standardní www klient z jakéhokoli místa přes intranet nebo internet.
- Control Web přináší výhody vizuálního programování využitím technologií objektového modelování s možností přepínání z vizuálního módu do textového módu a zpět, kdykoli je to nutné během vývoje aplikace.
- velká výhoda je modularita aplikace – schopnost, aby mohlo běžet současně více aplikací v jednom projektu a také na více počítačích na intranetu nebo přímo na internetu. Individuální aplikace jsou schopny sdílení běžných dat a spolupráce.
- Control Web využívá standardního rozhraní, protokolů a vlastností prostředí Microsoft Windows, např. multiřetězení, TCP/IP (Transmission Control Protocol/Internet Protocol) propojení, ODBC (Open Database Connectivity), přenos dat při použití aplikací DDE (Dynamic Data Exchange) a NetDDE protokoly. [15]

Přehled ovladačů

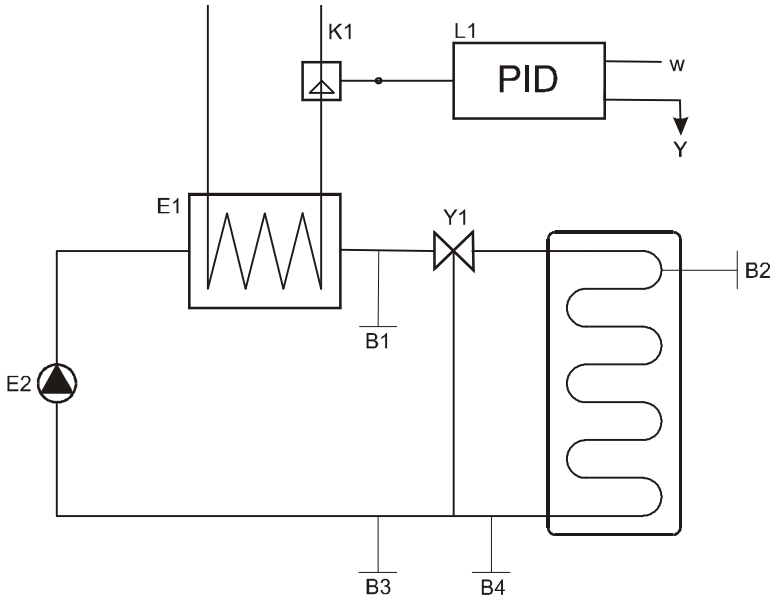
Control Web systém zahrnuje široký rozsah ovladačů pro tovární automatizovanou kontrolu, vkládací obvodovou desku pro PC, nezávislou jednotku a ostatní zařízení.

Standardní částí dodaného vývojového prostředí jsou ovladače pro účely znázornění při odstraňování chyb:

- virtuální driver – poskytuje řadu signálů pro znázornění využití
- model driver – server pro testování regulačního obvodu
- DDE driver – server pro připojovací aplikace přes DDE rozhraní
- ASCII driver – vhodný pro použití jednoduchého komunikačního protokolu charakteru ASCII přes sériové linky.

7.2 Popis úlohy DE1 – regulace teploty

Úloha DE1 s názvem „Regulace teploty“ plní funkci demonstrace reálného systému s uzavřeným okruhem regulace teploty. Zařízení obsahuje regulovanou soustavu s dopravním zpožděním a rozloženými parametry. Soustava je vytvořena zdrojem tepla (E1), oběhovým čerpadlem (E2), třícestným regulačním ventilem (Y1), spotřebičem tepla (E3). Regulátor (U1) je kompaktní programovatelný číslicový regulátor s možností dálkového parametrizování. Akčním členem je jednotka (K1) pro ovládání efektivní hodnoty napětí 230V/50 Hz toku elektrické energie.Velikost dopravního zpoždění lze nastavit změnou rychlosti proudění na čerpadle (E2) a polohou regulačního ventilu (Y1). Schéma experimentu je na obrázku 1.



Obr. 38 Schéma úlohy DE1 – regulace teploty

Měřené veličiny na soustavě (teploty B1, B2, B3, B4, poloha ventilu Y1) jsou napojeny na vstupní stranu centrální jednotky jako analogové signály. Centrální jednotka je typu PLC (Programmable Logic Controller), typ SIMATIC S7-300. Její výstupy ovládají akční jednotky systému analogovým výstupním signálem (ventil Y1). Propojení typu PROFIBUS je mezi PLC a regulátorem. Souhrn technických prostředků použitých na úloze DE1 je v tabulce.

Při spuštění úlohy pro danou měřenou teplotu (B1, B2, B3, B4) a pro danou žádanou hodnotu regulace se po startu provozu regulace provede regulační pochod. Pro zadané parame-

try regulátoru bude regulační pochod stabilní nebo nestabilní. Uživatel má možnost podle průběhu regulace si vyhodnotit příčinu nestability nebo horší kvality regulace.

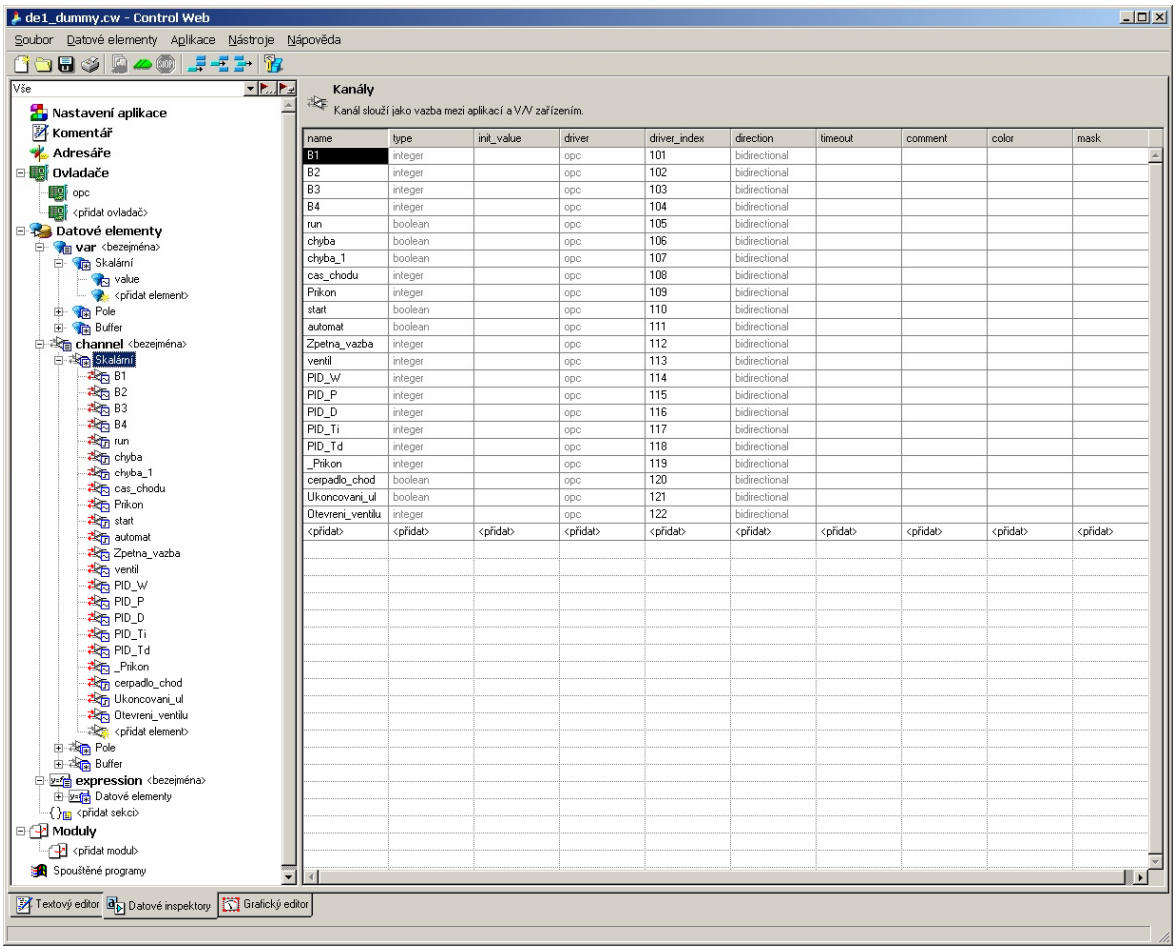
Tab. 4 Použité komponenty

Centrální jednotka	Parametr	Input	Output	Dodavatel
SIMATIC S7-300, CPU 313C-2DP	CPU SIMATIC	16DI	16 DO	Siemens
SIMATIC NET, CP 343-1	protokol TCP/IP			Siemens
SIMATIC S7-300, ANALOG INPUT SM 331		4-20mA		Siemens
SIMATIC S7-300, ANALOG OUTPUT SM 332			0-10V	Siemens
SIMATIC S7-300, POWER SUPPLY PS 307		230VAC	24VDC	Siemens
Měření, regulace, ovládání				
T1-teplota výstup ze zdroje	Pt100, 0-100°C	0-10V		Regmet
T2-teplota spotřeba	Pt100, 0-100°C	0-10V		Regmet
T3-teplota spotřeba	Pt100, 0-100°C	0-10V		Regmet
T4-teplota vratná	Pt100, 0-100°C	0-10V		Regmet
E1-elektrický ohřívač	230VAC/2,2 kW			
E2-čerpadlo	230VAC, JS 1/2"		0-10V	WILO
K1-ovládání příkonu E1	230VAC, 2,2 kW		4-20mA	Shimaden
Y1-regulační ventil	JS15, třícestný	0-10V	0-10V	Belimo
U1-číslicový regulá-tor PID	PID, RS485	4-20mA	0-10V	Siemens

7.3 Úloha DE1- regulace teploty v prostředí Control WEB

Pro ověření a otestování funkce OPC serveru OPC.simatic.NET byla v prostředí Control WEB vytvořena aplikace pro řízení a vizualizaci laboratorní úlohy DE1 – regulace teploty. Tato aplikace představuje SCADA/HMI řešení řízení úlohy DE1 prostřednictvím PC.

V první řadě bylo nutné nainstalovat ovladač, pomocí něhož komunikuje Control WEB s OPC serverem tzv. OPC Client driver. Ovladač je uložený v souboru s200.drv a pomocí nástroje Konfigurace OPC ovladače je nutné k němu vytvořit mapovací s200.dmf a parametrický s200.par soubor. Pro ilustraci zde uvádím obrázek prostředí Control WEB s rozbalenými komunikačními kanály.

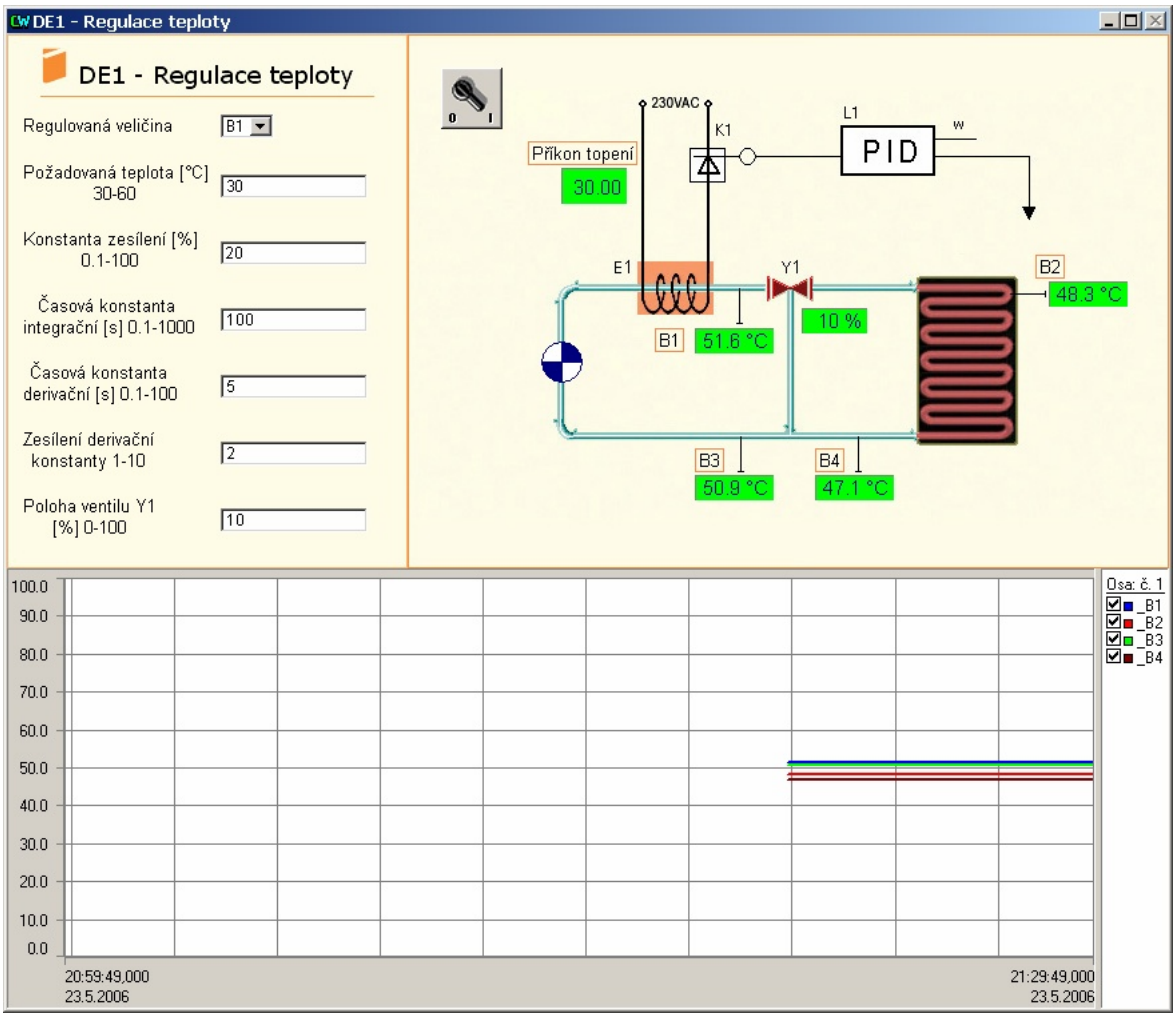


name	type	init_value	driver	driver_index	direction	timeout	comment	color	mask
B1	integer		opc	101	bidirectional				
B2	integer		opc	102	bidirectional				
B3	integer		opc	103	bidirectional				
B4	integer		opc	104	bidirectional				
run	boolean		opc	105	bidirectional				
chyba	boolean		opc	106	bidirectional				
chyba_1	boolean		opc	107	bidirectional				
cas_chodu	integer		opc	108	bidirectional				
Prikon	integer		opc	109	bidirectional				
start	boolean		opc	110	bidirectional				
automat	boolean		opc	111	bidirectional				
Zpetna_vazba	integer		opc	112	bidirectional				
ventil	integer		opc	113	bidirectional				
PID_W	integer		opc	114	bidirectional				
PID_P	integer		opc	115	bidirectional				
PID_D	integer		opc	116	bidirectional				
PID_Ti	integer		opc	117	bidirectional				
PID_Td	integer		opc	118	bidirectional				
_Prikon	integer		opc	119	bidirectional				
cerpadlo_chod	boolean		opc	120	bidirectional				
Ukoncovani_ul	boolean		opc	121	bidirectional				
Otevreni_ventilu	integer		opc	122	bidirectional				
<přidat>	<přidat>	<přidat>	<přidat>	<přidat>	<přidat>	<přidat>	<přidat>	<přidat>	<přidat>

Obr. 39 Prostředí Control WEB

Samotná tvorba aplikace DE1 - regulace teploty se provádí v grafickém prostředí aplikace Control WEB. Jednotlivé prvky se vkládají metodou drag and drop přímo do námi definovaného panelu z palety přístrojů, ale je zde také možnost vytvořit si přístroje vlastní. Po vložení přístroje se jeho vzhled a funkce upraví pomocí nástroje datové inspektory. Pomocí

tohoto nástroje se také jednotlivým prvkům přiřazují komunikační kanály, nebo se definují prvky do kterých se bude ukládat jimi zadaná hodnota a také zde můžeme nastavit různé události a jejich aktivaci.



Obr. 40 Aplikace DE1 – regulace teploty

Po spuštění aplikace z prostředí Control WEB se pro danou měřenou teplotu (B1, B2, B3, B4) a pro danou žádanou hodnotu regulace provede regulační pochod. Pro zadané parametry regulátoru bude regulační pochod stabilní nebo nestabilní. Uživatel bude mít možnost podle průběhu regulace si vyhodnotit příčinu nestability nebo horší kvality regulace.

Pro regulaci se zadávají následující parametry:

místo regulované veličiny, teplota B1 nebo B2 nebo B3 nebo B4

- Regulovaná veličina [B1, B2, B3, B4]
- Požadovaná teplota [30-60°C]

- parametr zesílení regulátoru (0,1-100%)
- parametr integrační časové konstanty regulátoru (0,1-100s)
- parametr derivační časové konstanty regulátoru (1-10s)
- zesílení derivační konstanty [1-10]
- poloha nastavení ventilu v rozsahu [0-100%]

Po zadání parametrů a spuštění regulačního procesu otočným ovladačem vlevo nahoře se nám spustí regulace. Průběhy jednotlivých teplot B1,B2, B3 a B4, můžeme je sledovat ve spodní části okna na běžícím grafu.

8 VYHODNOCENÍ VÝSLEDKŮ

Vyhodnocení výsledků komunikace PLC s IPC prostřednictvím OPC serveru se nedá jednoznačně napsat. Mnou vytvořená aplikace i nastavení serveru fungovalo bezchybně. Je otázkou jestli stejně bezchybně by komunikace fungovala při větším zatížení serveru velkým množstvím klientů, nebo jak by byla rychlá a kvalitní komunikace při používání několikanásobně větším počtu komunikačních kanálů. Zde je také nutné zohlednit řízenou aplikaci z hlediska náročnosti řízení, a také kvalitu a vytížení sítě LAN. Jiné nároky bude mít ryze REAL TIMEová aplikace a aplikace určená pouze pro zobrazování jednotlivých veličin.

Hodnoty zobrazované v aplikaci DE1- regulace teploty jsem pro zhodnocení komunikace porovnal s hodnotami které jsem si zobrazil pomocí klientu OPC Scout dodávaného přímo v balíku aplikací Simatic.NET firmy Siemens. Po zobrazení jednotlivých kanálů v tomto klientovi mohu konstatovat, že hodnoty jsou shodné s hodnotami v aplikaci DE1 – regulace teploty.

ZÁVĚR

Tato diplomová práce je zaměřená na OPC technologii. Je zde taky stručně popsaná technologie COM od firmy Microsoft, na které je OPC postavená.

V teoretické části této práce jsou popsány základní funkce, principy a možnosti použití OPC technologie při praktickém nasazení v automatizačním odvětví. Dále se práce zabývá základními principy a funkcemi objektově orientované technologie COM. Jako poslední jsou v teoretické části detailně popsány jednotlivé OPC specifikace, které tvoří jakousi normu pro vývoj, použití a správné fungování aplikací založených na výměně dat pomocí technologie OPC. Velký důraz je kladen na specifikaci OPC Data Access, která je nejvíce rozšířenou specifikací a je používána ve při všech implementacích jak na straně klienta tak i na straně serveru.

V praktické části je nejprve popsáno jak vytvořit a správně nakonfigurovat propojení mezi PLC automatem Siemens a OPC serverem OPC.Simatic.NET prostřednictvím rozhraní ethernet. V další části je detailně popsané propojení a komunikace mezi OPC serverem a klientem Control WEB. Podrobně se tato část věnuje nastavení konfiguračních souborů OPC ovladače konfiguračním nástrojem pro OPC ovladač Control WEB a také vytvořením mapovacího a parametrického souboru. V poslední části je popsaná praktická aplikace. Je to SCADA/HMI aplikace pro laboratorní úlohu DE1 – regulace teploty. Tato aplikace nám slouží pro praktické ověření funkce OPC serveru a kontrolu správného nastavení komunikačních kanálů mezi OPC serverem a klientem Control WEB. Aplikace pro vizualizaci a řízení úlohy DE1 – regulace teploty je vytvořena v prostředí Control WEB 5.

SEZNAM POUŽITÉ LITERATURY

[1] **HRUŠKA, F: Technické prostředky automatizace IV. Snímače, převodníky, regulátory, průmyslová výpočetní technika, ovládací jednotky.** UTB ve Zlíně, FT, **září 2001**, s.107 **ISBN 80-7318-026-X**

[2] **HRUŠKA, F: Projektování systémů integrované automatizace. Učební texty. 2. vyd**. Zlín: UTB ve Zlíně, **2002**, s.133 **ISBN 80-7318-100-2**

[3] **KAČMÁŘ, D: Programujeme v COM a COM+.[Programming in COM and COM+]. 1st ed**. Pratur: Computer Press, **2000**. 309 p. **ISBN 80-7226-381-1**

[4] **OPC FOUNDATION: OPC Owerview. Version 1.0. 1998. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[5] **OPC FOUNDATION: OPC Comon Definitions and Interfaces. Version 1.1. 2000. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[6] **OPC FOUNDATION: OPC Dada Access Cystom interface standard. Version 3.0. 2003. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[7] **OPC FOUNDATION: OPC Alarm and Events specification. Version 1.1. 2000. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[8] **OPC FOUNDATION: OPC Historical Data Access specification. Version 1.2. 2003. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[9] **OPC FOUNDATION: OPC Batch specification. Version 2.0. 2001. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[10] **OPC FOUNDATION: OPC Security specification. Version 1.0. 2001. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[11] **OPC FOUNDATION: OPC XML DA specification. Version 1.0. 2003. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[12] **OPC FOUNDATION: OPC Data eXchange specification. Version 1.0. 2003. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[13] **OPC FOUNDATION: OPC Data Komplex Data specification. Version 1.0. 2003. [pdf elektronická specifikace]. Dostupný z WWW: <http://www.opcfoundation.com>.**

[14] **SIEMENS A.G.: SIMATIC.NET, Ethernet comunication between OPC Server and S-7 200 incl. CP243-1 1.0. release 2/2003. [pdf elektronický manuál]. Dostupn ý z WWW: <http://www.siemens.com>.**

[15] **MORAVSKÉ PŘÍSTROJE: Control WER 5, dokumentace. [html elektronický manuál]. Dostupný z WWW: <http://www.mii.cz>.**

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

OPC	Ole for proces control
OLE	Object Linking and Embedding
SCADA	Supervisory Control and Data Acquisition
HMI	Human – Machine interface
COM	Component Object Model
DCOM	Distributed Component Object Model
PID	Proportional-Integrating-Derivative control algorithm
PLC	Programmable Logic Controller
TCP/IP	Transmission Control Protocol/Internet Protocol
IP	Internet Protocol
DDE	Dynamic Data Exchange
IDE	Intergated Development Environment
ODBC	Open Database Connectivity
XML	eXtensible Markup Language

SEZNAM OBRÁZKŮ

Obr. 1 Komunikace bez OPC....14
Obr. 2 Komunikace přes OPC....15
Obr. 3 Specifikace OPC a jejich vazby....16
Obr. 4 Architektura OPC komunikace....18
Obr. 5 Mechanismus COM....22
Obr. 6 Architektura komponenty COM....24
Obr. 7 Vzájemné vazby mezi jednotlivými specifikacemi....33
Obr. 8 OPC klient....34
Obr. 9 Vztah mezi OPC serverem a klienty....35
Obr. 10 Uživatelské a automatické rozhraní....36
Obr. 11 OPC rozhraní....36
Obr. 12 Hierarchická struktura jmenného prostoru....38
Obr. 13 OPC Data Access server – jmenná a objektová hierarchie....40
Obr. 14 Data Access specifikace datového formátu....41
Obr. 15 OPC objekty v Data Access serveru a klientu....42
Obr. 16 Automatizační objekty v Data Access....43
Obr. 17 Příklad událostí....44
Obr. 18 Rozdělení událostí....45
Obr. 19 OPC Alarm and Events server – jmenná a objektová hierarchie....46
Obr. 20 Objekty v Alarms and Ebeny serveru a klientu....47
Obr. 21 Distorical Data Access server – jmenná a objektová hierarchie....48
Obr. 22 Historical Data Access server a klient rozhraní....52
Obr. 23 Automatizační rozhraní HDA serveru....53
Obr. 24 Nastavení síťových parametrů....58
Obr. 25 Instalace Simatic.NET....59
Obr. 26 Nainstalované součásti....59
Obr. 27 Station configuration editor....60
Obr. 28 Konfigurace parametrů propojení....61
Obr. 29 Konfigurace parametrů propojení – výběr komunikačních protokolů....61
Obr. 30 Struktura sítě v programu NET PRO....62
Obr. 31 Nastavení síťové komunikace....63

Obr. 32 OPC Scout – vstupní hodnoty....64
Obr. 33 OPC Scout – výstupní hodnoty....64
Obr. 34 Okno konfiguračního nástroje s hodnotami pro úlohu DE1 70
Obr. 35 Okno výběru OPC serveru....71
Obr. 36 Přiřazení čísla kanálu 73
Obr. 37 Volba ovladače....74
Obr. 38 Schéma úlohy DE1 – regulace teploty....78
Obr. 39 Prostředí Control WEB....80
Obr. 40 Aplikace DE1 – regulace teploty....81

SEZNAM TABULEK

Tab. 1 Specifikace OPC – stav ke konci 1. čtvrtletí 2006 32
Tab. 2 Typy datové výměny 39
Tab. 3 Vztahy mezi zaznamenanými daty, atributy a agragacemi 50
Tab. 4 Použité komponenty 79

SEZNAM PŘÍLOH

Příloha P I: Výpis programu úlohy DE1 – regulace teploty

PŘÍLOHA P I: VÝPIS PROGRAMU ÚLOHY DE1 – REGULACE TEPLOTY

```
comment
end_comment;


directories
  '*.dmf' = 'D:\Program Files\Moravian Instruments\Control Web 5
CZE\Dmf';
  '*.par' = 'D:\Program Files\Moravian Instruments\Control Web 5
CZE\Par';
  '*.ico' = 'D:\Program Files\Moravian Instruments\Control Web 5
CZE\Ico\Indicator';
end_directories;


settings
  operation_mode = real_time;
  startup_options
    call_procedures = false;
    activate_receivers = false;
    output_action = set_local;
  end_startup_options;
end_settings;

driver
  opc : 'dummy.dll', 's200.dmf', 's200.par';
end_driver;

data


  var
    value : real;
  end_var;


  channel {driver = opc};
    B1 : integer {driver_index = 101; direction = bidirectional};
    B2 : integer {driver_index = 102; direction = bidirectional};
    B3 : integer {driver_index = 103; direction = bidirectional};
    B4 : integer {driver_index = 104; direction = bidirectional};
    run : boolean {driver_index = 105; direction = bidirectional};
    chyba : boolean {driver_index = 106; direction = bidirectional};
```

```
    chyba_1 : boolean {driver_index = 107; direction = bidirectio-
nal};
    cas_chodu : integer {driver_index = 108; direction = bidirectio-
nal};
    Prikon : integer {driver_index = 109; direction = bidirectional};
    start : boolean {driver_index = 110; direction = bidirectional};
    automat : boolean {driver_index = 111; direction = bidirectio-
nal};
    Zpetna_vazba : integer {driver_index = 112; direction = bidi-
rectional};
    ventil : integer {driver_index = 113; direction = bidirectional};
    PID_W : integer {driver_index = 114; direction = bidirectional};
    PID_P : integer {driver_index = 115; direction = bidirectional};
    PID_D : integer {driver_index = 116; direction = bidirectional};
    PID_Ti : integer {driver_index = 117; direction = bidirectional};
    PID_Td : integer {driver_index = 118; direction = bidirectional};
    _Prikon : integer {driver_index = 119; direction = bidirectio-
nal};
    cerpadlo_chod : boolean {driver_index = 120; direction = bidi-
rectional};
    Ukoncovani_ul : boolean {driver_index = 121; direction = bidi-
rectional};
    Otevreni_ventilu : integer {driver_index = 122; direction = bidi-
rectional};
  end_channel;


  expression
    _B1 = B1 / 10;
    _B2 = B2 / 10;
    _B3 = B3 / 10;
    _B4 = B4 / 10;
    _Y1 = ventil;
  end_expression;


end_data;


instrument


  panel panel_1;
    owner = background;
    position = 10, 25, 870, 730;
    window = normal;
    win_title = 'DE1 - Regulace teploty';
    win_disable = zoom;
```

```
end_panel;

data_viewer data_viewer_1;
  timer = 1;
  owner = panel_1;
  position = 0, 400, 870, 330;
  viewer_mode = chart;
  content = title, legend, x_axis, y_axis, grid;
  tool_content = change_group, change_mode, browse, axes_arrange;
  chart_description
    time_axis
      time_interval = '30m';
    end_time_axis;
    y_axes
      axis_element
        name = 'teplota';
        units = '°C';
        dec_places = 1;
      end_axis_element;
    end_y_axes;
  end_chart_description;
  data_group
    group_name = 'gr';
    item
      data_element = _B1;
      axis_name = 'teplota';
      color = lblue;
    end_item;
    item
      data_element = _B2;
      axis_name = 'teplota';
      color = lred;
    end_item;
    item
      data_element = _B3;
      axis_name = 'teplota';
      color = lgreen;
    end_item;
    item
      data_element = _B4;
      axis_name = 'teplota';
```

```
      color = red;
    end_item;
  end_data_group;
  active_group = 'gr';
end_data_viewer;

panel panel_2;
  owner = panel_1;
  position = 0, 0, 300, 400;
  win_title = 'Hodnoty';
  mode = window_less;
  dv_file = 'LEFT.JPG';
  colors
    color = 254, 242, 179;
  end_colors;
end_panel;

multi_switch multi_switch_2;
  owner = panel_2;
  position = 160, 60, 40, 18;
  mode = combo_box;
  item
    text = 'B1';
  end_item;
  item
    text = 'B2';
  end_item;
  item
    text = 'B3';
  end_item;
  item
    text = 'B4';
  end_item;

  procedure OnIndex( Index : longint );
  begin
    Zpetna_vazba := Index;
  end_procedure;

end_multi_switch;
```

```
label label_3;
  owner = panel_2;
  position = 10, 245;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Časová konstanta ';
    font = 'Arial (Central European)', 10, normal;
    text = 'derivační [s] 0.1-100';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label label_3;
  owner = panel_2;
  position = 10, 60;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Regulovaná veličina';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label label_3;
  owner = panel_2;
  position = 10, 345;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
```

```
    font = 'Arial (Central European)', 10, normal;
    text = 'Poloha ventilu Y1';
    font = 'Arial (Central European)', 10, normal;
    text = '[%] 0-100';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label label_3;
  owner = panel_2;
  position = 10, 295;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Zesílení derivační';
    font = 'Arial (Central European)', 10, normal;
    text = 'konstanty 1-10';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label label_3;
  owner = panel_2;
  position = 10, 195;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Časová konstanta ';
    font = 'Arial (Central European)', 10, normal;
    text = 'integrační [s] 0.1-1000';
  end_text_list;
```

```
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label label_3;
  owner = panel_2;
  position = 10, 145;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Konstanta zesílení [%]';
    font = 'Arial (Central European)', 10, normal;
    text = '0.1-100';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label label_3;
  owner = panel_2;
  position = 10, 95;
  win_disable = zoom, maximize;
  transparent;
  justify = center;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Požadovaná teplota [°C]';
    font = 'Arial (Central European)', 10, normal;
    text = '30-60';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;
```

```
string_control Y1;
  owner = panel_2;
  position = 160, 355, 110, 18;
end_string_control;

string_control PID_D;
  owner = panel_2;
  position = 160, 305, 110, 18;
end_string_control;

string_control PID_Td;
  owner = panel_2;
  position = 160, 255, 110, 18;
end_string_control;

string_control PID_Ti;
  owner = panel_2;
  position = 160, 205, 110, 18;
end_string_control;

string_control PID_P;
  owner = panel_2;
  position = 160, 155, 110, 18;
end_string_control;

string_control PID_W;
  owner = panel_2;
  position = 160, 105, 110, 18;
  init_value = '30';
end_string_control;

panel panel_3;
  owner = panel_1;
  position = 300, 0, 570, 400;
  mode = window_less;
  dv_file = 'DE1P.JPG';
  colors
    color = 222, 245, 255;
  end_colors;
end_panel;
```

```
engine engine_1;
  owner = panel_3;
  position = 100, 230, 31, 31;
end_engine;

switch RUN;
  owner = panel_3;
  position = 25, 25, 46, 46;
  win_disable = zoom, maximize;
  output = start;
end_switch;

label Příkon;
  owner = panel_3;
  position = 90, 80;
  win_disable = zoom, maximize;
  frame = 1;
  transparent;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'Příkon topení';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

meter Příkon;
  timer = 1;
  owner = panel_3;
  position = 115, 100, 50, 28;
  expression = Prikon;
  mode = text_display;
  content = med;
  font = 'Arial (Central European)', 11, normal;
  transparent;
  colors
    border_paper = lgray;
    ink = blue;
```

```
    paper = white;
    value = blue;
    low_limit = lgreen;
  end_colors;
end_meter;

meter Y1;
  timer = 1;
  owner = panel_3;
  position = 295, 205, 50, 20;
  expression = _Y1;
  mode = text_display;
  content = med;
  dec_places = 0;
  font = 'Arial (Central European)', 11, normal;
  mask = '### %';
  transparent;
  colors
    border_paper = lgray;
    ink = blue;
    paper = white;
    value = blue;
    low_limit = lgreen;
  end_colors;
end_meter;

label B2;
  owner = panel_3;
  position = 470, 165;
  win_disable = zoom, maximize;
  frame = 1;
  transparent;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'B2';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;
```

```
label B1;
  owner = panel_3;
  position = 185, 220;
  win_disable = zoom, maximize;
  frame = 1;
  transparent;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'B1';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

label B3;
  owner = panel_3;
  position = 215, 310;
  win_disable = zoom, maximize;
  frame = 1;
  transparent;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'B3';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

meter B2;
  timer = 1;
  owner = panel_3;
  position = 480, 185, 60, 20;
  expression = _B2;
  mode = text_display;
  content = med;
  dec_places = 1;
```

```
  font = 'Arial (Central European)', 11, normal;
  mask = '###.# °C';
  transparent;
  colors
    border_paper = lgray;
    ink = blue;
    paper = white;
    low_limit = lgreen;
  end_colors;
end_meter;

meter B1;
  timer = 1;
  owner = panel_3;
  position = 215, 220, 60, 20;
  expression = _B1;
  mode = text_display;
  content = med;
  dec_places = 1;
  font = 'Arial (Central European)', 11, normal;
  mask = '###.# °C';
  transparent;
  colors
    border_paper = lgray;
    ink = blue;
    paper = white;
    low_limit = lgreen;
  end_colors;
end_meter;

meter B3;
  timer = 1;
  owner = panel_3;
  position = 215, 330, 60, 20;
  expression = _B3;
  mode = text_display;
  content = med;
  dec_places = 1;
  font = 'Arial (Central European)', 11, normal;
  mask = '###.# °C';
  transparent;
```

```
  colors
    border_paper = lgray;
    ink = blue;
    paper = white;
    low_limit = lgreen;
  end_colors;
end_meter;

label label_3;
  owner = panel_3;
  position = 305, 310;
  win_disable = zoom, maximize;
  frame = 1;
  transparent;
  text_list
    font = 'Arial (Central European)', 10, normal;
    text = 'B4';
  end_text_list;
  colors
    top_shadow = 247, 140, 79;
    bottom_shadow = 247, 140, 79;
  end_colors;
end_label;

meter B4;
  timer = 1;
  owner = panel_3;
  position = 305, 330, 60, 20;
  expression = _B4;
  mode = text_display;
  content = med;
  dec_places = 1;
  font = 'Arial (Central European)', 11, normal;
  mask = '###.# °C';
  transparent;
  colors
    border_paper = lgray;
    ink = blue;
    paper = white;
    low_limit = lgreen;
  end_colors;
```

```
  end_meter;
end_instrument;
```