Razvoj operaterske stanice/konzole za povezivanje sa raznorodnim SCADA sistemima

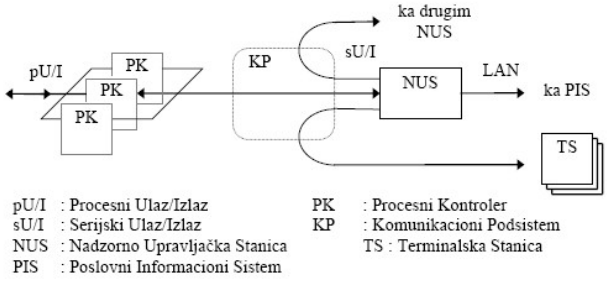
Laslo Brusnjai, Miroslav Popović, Ilija Bašičević, Velibor Mihić, Fakultet tehničkih nauka, Novi Sad

***Sadržaj* — U ovom radu je analizirana struktura i organizacija baze procesnih podataka različitih SCADA sistema, kao i raspoloživih komunikacionih tehnologija za daljinski pristup tim podacima. Na toj osnovi je razvijena programska podrška udaljene operaterske stanice koja, u okviru jedne aplikacije, omogućava istovremeno povezivanje i rad sa više različitih SCADA sistema. Korisnička sprega je realizovana korišćenjem WPF grafičkog okruženja, a sama aplikacija je napisana u programskom jeziku C#.**

***Ključne reči* — AUS, DAL, GAUS, HMI, OASyS DNA, SCADA, WPF, XAML, XML.**

I. UVOD

Akviziciono upravljački sistem (*Supervisory Control And Data Acquisition*) SCADA je skup namenskih, prostorno raspodeljenih, međusobno povezanih računarskih modula, čiji je zajednički cilj ostvarenje funkcija nadzora i/ili upravljanja fizičkim procesom u realnom vremenu. Osnovna funkcija SCADA sistema je ciklična akvizicija digitalizovanih odmeraka različitih fizičkih veličina koji određuju stanje proizvoljnog tehnološkog (fizičkog) procesa. Konfiguracija tipičnog SCADA sistema je prikazana na Sl. 1.



Sl. 1. Arhitektura SCADA sistema

Važnu komponentu svakog SCADA sistema predstavlja operaterska stanica/konzola (*HMI – Human Machine Interface*) koja omogućava ovlašćenom operateru nadzor i upravljanje tehnološkim (fizičkim) procesom čiji se rad odvija pod kontrolom SCADA sistema. Osnovni zadatak operaterske stanice je efikasan prikaz trenutnog stanja i događaja u nadziranom procesnom sistemu, kao i prihvat operaterskih upravljačkih direktiva (komandi). Posmatrano kroz programsku realizaciju, operaterska stanica obuhvata vizuelni pregled i ručno ažuriranje baze procesnih podataka SCADA sistema. Funkcije operaterske stanice uključuju prikaz i izmenu sadržaja baze podataka, a ostvaruju se putem ekranskih i štampanih izveštaja ili prenosom podataka iz SCADA-e u klasičan poslovni IS. Komunikacija između operatera i programskog okruženja se obavlja posredstvom standardnih računarskih U/I uređaja (monitor, štampač, miš i tastatura). Izveštaji i operaterske maske se najčešće realizuju u alfanumeričkom, obično tabelarnom obliku. Pored tabelarnog prikaza, koji podrazumeva alfanumerički prikaz trenutnog stanja procesnih promenljivih u unapred definisanom zadatom formatu, postoji i grafički prikaz koji predstavlja primarni i najpopularniji vid vizualizacije trenutnog stanja procesnog sistema kojim se upravlja. Tabelarni prikaz se organizuje po tehnološkoj pripadnosti ili tipu procesnih veličina. Kontrolni paneli procesnih veličina obezbeđuju detaljan prikaz stanja i svih relevantnih parametara jedne procesne veličine. Istovremeno, preko njih se u radu ("on-line") menjaju parametri koji utiču na tok sekundarne obrade merenih podataka, kao i izdavanje izvršnih komandi.

II. POSTAVLJENI ZAHTEVI

Efekat globalizacije i tržišne konkurencije doveo je do toga da se različiti SCADA sistemi, razvijeni od strane raznih (konkurentskih) IT kompanija, koriste na relativno bliskim geografskim lokacijama. Svaka od tih kompanija poseduje posebno razvijenu i prilagođenu operatersku stanicu za potrebe nadzora i upravljanja svog SCADA sistema. Kako se u drugim granama (sektorima) tržišta, tako i u ovoj, akteri relativno brzo menjaju ili čak udružuju, što dovodi do potrebe da se postojeći SCADA sistemi zamene novim ili ako je to moguće integrišu u jedan zajednički sistem. Osnovni cilj koji je postavljen je da se razvije aplikacija koja će omogućiti integraciju različitih SCADA sistema što podrazumeva istovremeni nadzor i upravljanje istih. Razvijena operaterska stanica treba da podrži tabelarni prikaz procesnih veličina kao i događaja i alarma koji opisuju trenutno stanje nadziranog tehnološkog procesa. Pod upravljanjem se podrazumeva izdavanje sledećih komandi od strane operatera:

- *Promena granica alarma.* U skladu sa promenom režima u tehnološkom sistemu, potrebno je usklađivati granice alarma i srodne

Ovaj rad je delimično finansiran od Ministarstva za nauku Republike Srbije, projekat 12004, od 2008. god.

Laslo Brusnjai, Autor, Fakultet tehničkih nauka u Novom Sadu, Srbija (e-mail: laslo.brusnjai@telventdms.com).

Dr Miroslav Popović, Koautor, Fakultet tehničkih nauka u Novom Sadu, Srbija (e-mail: miroslav.popovic@rt-rk.com).

Dr Ilija Bašičević, Koautor, Fakultet tehničkih nauka u Novom Sadu, Srbija (e-mail: ilija.basicevic@rt-rk.com).

Mr Velibor Mihić, Koautor, Fakultet tehničkih nauka u Novom Sadu, Srbija (e-mail: velibor.mihic@rt-rk.com).

parametre vezane za sekundarnu obradu izmerenih podataka.

- *Dozvola / zabrana generisanja alarma.* Uzastopne oscilacije određene procesne veličine oko granice alarma izaziva neprestano generisanje alarma, koji opterećuju rad operatera i stoga utiču na otežano praćenje stanja procesnog sistema. Stoga je uobičajena mogućnost da se operaterskom direktivom zabrani generisanje alarma, dok se ne otkloni uzrok.
- *Izbor izmerene ili ručno zadate vrednosti.* U toku eksploatacije sistema dolazi do otkaza fizičkih senzora ili merne procesne opreme. Od interesa je omogućiti ručni unos procenjene (približne) vrednosti, jer se na taj način omogućuje dalja obrada procesnih promenljivih (npr. računatih veličina).
- *Potvrdu alarma na nedvosmislen način.*

Kontrolni panel, koji zavisi od odabranog tipa procesne veličine, treba da obezbedi detaljan prikaz stanja i svih relevantnih parametara jedne procesne veličine. Novu konzolu treba modelovati tako da integracija sa novim SCADA sistemom ne zahteva izmenu osnovnog koda, već samo razvoj novog prilagodnog (*plug-in*) modula koji služi za komunikaciju sa odredišnim sistemom. Prilikom dinamičkog dodavanja novog prilagodnog modula treba da se omogući prijava dežurnog operatera koji predstavlja važan elemenat u postupku upravljanja tehnološkim procesom, kojim se određuje odgovorno lice, njegove obaveze (dužnosti) i skup dozvoljenih akcija, koje su definisane u okviru samog SCADA sistema. Razvijeni koncept je potrebno proveriti na primeru integracije sa dva različita SCADA sistema – OASyS DNA i GAUS.

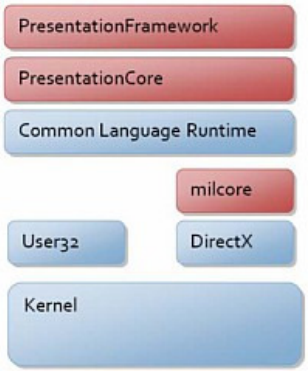
III. Rezultati analize različitih SCADA sistema

Pre samog razvoja udaljene operaterske stanice, analizirane su strukture i organizacije baze procesnih podataka i raspoloživih komunikacionih tehnologija za daljinski pristup podacima GAUS SCADA sistema [1] koja je razvijena u okviru Fakulteta tehničkih nauka u Novom Sadu, i OASyS DNA SCADA sistema [2] kompanije Telvent osnovane u Sevilji - Španija. Detaljnom analizom, utvrđeno je da su korišćeni tipovi osnovnih procesnih veličina prisutni kod oba sistema a to su analogni, digitalni i brojački. Njihovi nazivi kao i sama struktura tabela u kojima se skladište podaci o procesnim veličinama se razlikuju, počevši od imena i tipova polja unutar samih tabela pa sve do njihovih međusobnih veza.

U vezi raspoloživih komunikacionih tehnologija, primećuju se takođe velike razlike između posmatrana dva SCADA sistema. OASyS DNA poseduje sopstveni razvijeni mehanizam – *Data Access Layer (DAL)* - za udaljeni pristup i upravljanje koji se koristi od strane njihove udaljene operaterske stanice nazvane *Remote ezXOS*. U slučaju GAUS-a, udaljene terminalske stanice (TS) koriste komunikacioni podsistem koji se oslanja na korišćenje brzih LAN kanala (*Ethernet IEEE 802.3*) i TCP/IP protokola za komunikaciju sa nadzorno upravljačkom stanicom (NUS).

IV. Razvoj grafičke sprege pomoću WPF-a

Grafička sprega razvijene udaljene operaterske stanice je realizovana korišćenjem *Windows Presentation Foundation (WPF)* grafičkog podsistema u okviru .NET 3.5 okruženja [3]. Primarni programski model WPF-a je izložen kroz *managed* kod. *Managed* kod je programski kod koji se izvršava pod kontrolom virtuelne mašine - CLR, za razliku od *unmanaged* koda, čije izvršavanje obavlja direktno sam procesor računara [4]. Na Sl. 2. je crvenom (tamnom) bojom označena struktura WPF-a.



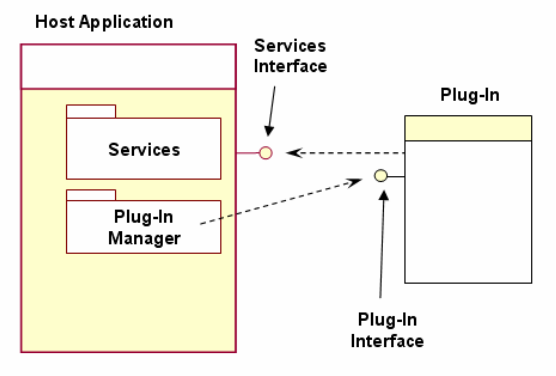
Sl. 2. Struktura WPF-a

Jedini deo WPF-a koji je *unmanaged* je *milcore*. On je zadužen za usku integraciju sa DirectX-om, koji u okviru WPF radi sva iscrtavanja. XAML (*Extensible Application Markup Language*) je deklarativni jezik zasnovan na XML-u koji se koristi u okviru WPF-a za definisanje korisničke sprege. XAML elementi se direktno preslikavaju u instance CLR objekata, dok se atributi preslikavaju u CLR osobine i događaje. Primenom XAML jezika postignute su dve važne stvari: smanjenje kompleksnosti projekta i lakši razvoj korisničke sprege. Pošto u okviru standardnog WPF grafičkog okruženja nije razvijena kontrola za tabelarni prikaz podataka, tzv. *DataGrid*, iskorišćena je *WPF Toolkit* biblioteka otvorenog izvornog koda, koja u sebi sadrži navedenu kontrolu.

V. Razvoj prilagodnog modula – plug-in

Razvijena udaljena operaterska stanica omogućava dinamičko uključivanje različitih SCADA sistema bez potrebe za izmenom osnovne aplikacije. Ovo je postignuto zahvaljujući implementaciji *plug-in* arhitekture u okviru same aplikacije. *Plug-in* arhitektura predstavlja okruženje (*framework*) koje omogućava programu da "potraži" dodatnu funkcionalnost pri pokretanju i da zatim dozvoli međusobnu saradnju sa tim modulom. U tipičnom slučaju osnovna aplikacija pruža usluge koje prilagodni modul može da koristi, uključujući način na koji se ti moduli registruju u okviru aplikacije i definiciju protokola razmene podataka između navedenih entiteta. Ovi moduli zavise od usluga koje pruža osnovna (*host*) aplikacija i obično se ne mogu izvršavati samostalno. Međutim, sama

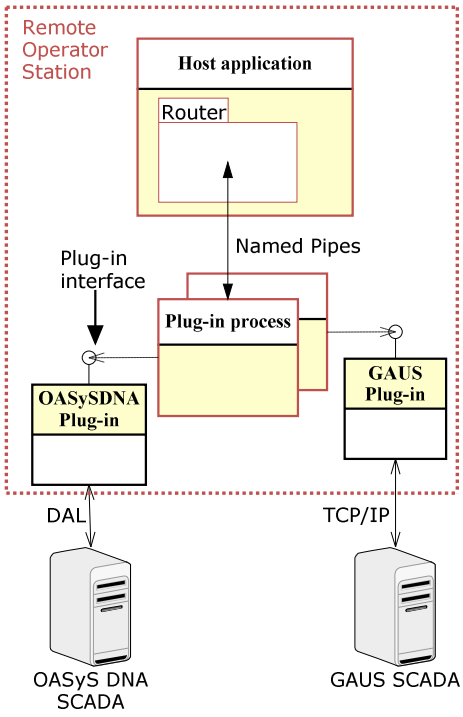
aplikacija radi nezavisno od prilagodnih modula, omogućavajući krajnjim korisnicima da obavljaju njihovo dinamičko dodavanje i ažuriranje, bez potrebe da se ona menja. Na Sl 3. je prikazan tipičan primer korišćenja *plug-in* okruženja.



Sl. 3. Primer tipičnog *plug-in* okruženja

Realizovani prilagodni modul za potrebe rešenja se razlikuje od gore prikazanog primera po tome što se usluge (*services*) koje koristi ne nalaze na strani osnovne aplikacije već se one obezbeđuju od strane SCADA sistema. Pomenuti moduli su realizovani kao DLL-ovi (*Dynamic Link Library*) koji implementiraju posebno definisanu *IPlugin* spregu (*interface*). Navedena sprega sadrži deklaracije obaveznih metoda (API) koje poziva aplikacija.

Na Sl. 4. je prikazana arhitektura realizovanog sistema sa jasno definisanim sastavnim elementima i korišćenim komunikacionim tehnologijama.



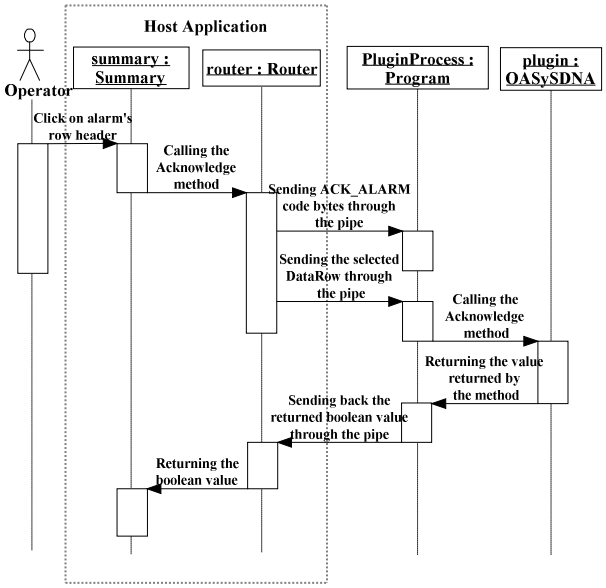
Sl. 4. Arhitektura realizovanog sistema

Istaknuta *router* komponenta u okviru osnovne aplikacije predstavlja DLL biblioteku koja obavlja apstrakciju složene međuprocesne komunikacije. Ona je zadužena i za usmeravanje podataka od osnovne aplikacije ka tačno određenom *plug-in* procesu i obrnuto.

Windows operativni sistem pruža mehanizme za komunikaciju i deljenje podataka između aplikacija. Zajednički naziv ovih funkcionalnosti je međuprocesna komunikacija (*interprocess communication*) u daljem tekstu IPC. Neki tipovi IPC-ova omogućavaju podelu posla između izdvojenih procesa unutar jednog računara, dok drugi omogućavaju podelu posla između računara u mreži. Tipično, aplikacije koje koriste IPC se svrstavaju ili u klijentske ili u uslužne. Klijent predstavlja aplikaciju ili proces koji zahteva uslugu od neke druge aplikacije ili procesa, dok je uslužna aplikacija ili proces koja odgovara na taj zahtev. Sledeći IPC mehanizmi su podržani od strane Windows operativnog sistema: *Clipboard, COM, Data Copy, DDE, File Mapping, Mailslots, Pipes, RPC* i *Windows Sockets* [4].

U ovom rešenju je odabran *Pipes* mehanizam i to *Named pipe* koji predstavlja jedan od dva tipa dvosmerne komunikacije u okviru *Pipes* grupe. *Named pipe* mehanizam se koristi za prenos podataka između različitih procesa smeštenih na jednom ili više računara u mreži. U tipičnom slučaju, *Named pipe* server stvara *Named pipe* sa dobro poznatim imenom ili sa imenom koji se nakon stvaranja šalje klijentima. Nakon što klijent sazna za ime *Named pipe*-a pristupa otvaranju drugog kraja stvorenog kanala. Kada su entiteti na oba kraja kanala povezani, razmena podataka između njih se ostvaruje pozivanjem operacije čitanja i pisanja nad stvorenim kanalom.

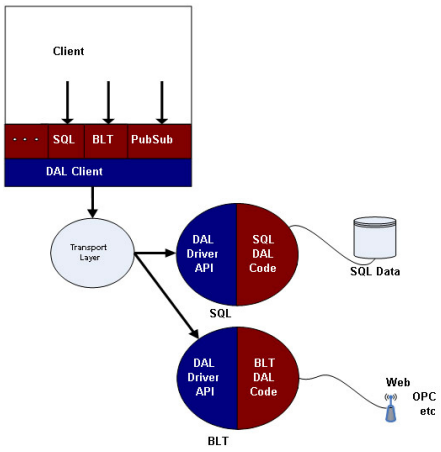
Izdvajanjem svakog prilagodnog modula u poseban proces povećala se stabilnost osnovne aplikacije. U slučaju greške u toku izvršavanja prilagodnog modula, gasi se samo proces koji se naknadno pokreće od strane osnovne aplikacije i u okviru koje se izvršavaju funkcije učitanog modula, dok sama aplikacija ostaje "netaknuta". Osnovna aplikacija je realizovana kao uslužna a sam *plug-in* proces kao klijentska strana u IPC komunikaciji.



Sl. 5. UML dijagram redosleda

Na Sl. 5. je prikazan UML dijagram redosleda (*sequence diagram*) koji opisuje redosled dešavanja

prilikom potvrđivanja alarma od strane operatera. *IPlugin* sprega je definisana u posebnoj DLL datoteci koja sadrži i definiciju (listu) svih tipova metoda čije se pozivanje zahteva od strane osnovne aplikacije. Svaki prilagodni modul mora da definiše sve metode navedene u datoj sprezi. Modul koji je specijalno razvijen za komunikaciju sa OASyS DNA SCADA-om je implementiran kao izdvojena DLL datoteka. Za komunikaciju sa OASyS DNA SCADA-om iskorišćene su već postojeće biblioteke, takođe realizovane u .NET okruženju. Kao što je na Sl. 4. prikazano, komunikacija je postignuta korišćenjem sloja za pristup podacima - *Data Access Layer*, u daljem tekstu DAL. DAL predstavlja skup programskih alata projektovan tako da omogući pristup različitim izvorima podataka, uzimajući u obzir specifične potrebe klijenata. Apstrakcija pristupa različitim izvorima podataka postignuta je programskom spregom (API) koju obezbeđuje DAL. Na Sl. 6. je prikazana arhitektura DAL-a [2].



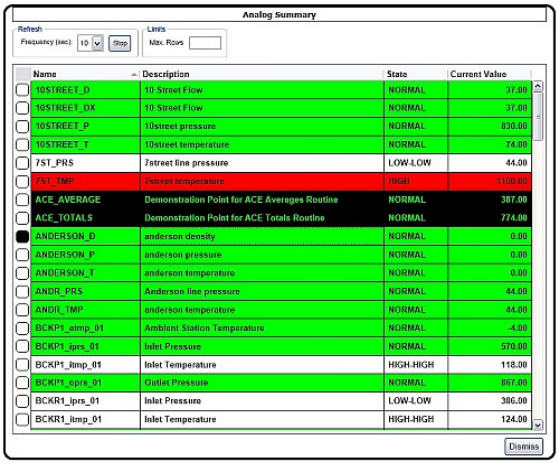
Sl. 6. Arhitektura DAL-a

Kao što se na Sl. 6. može videti, klijent zahteva preuzimanje podataka ili komandovanje pozivanjem odgovarajuće DAL API funkcije. Nakon toga se komanda prenosi pomoću transportnog sloja (*Transport Layer)* odgovarajućem DAL upravljačkom programu (*driver*). Dok se komanda obrađuje, klijent može nastaviti izvršavanje drugih operacija potpuno asinhrono. U zavisnosti od implementacije transportnog sloja, klijent i upravljački program se mogu izvršavati u okviru istog procesa, na istom računaru ali u različitim procesima ili na sasvim različitim računarima.

A. *Definisanje funkcije preslikavanja*

Analizom baze podataka različitih SCADA sistema utvrđeno je da postoji značajna razlika u njihovoj strukturi i organizaciji. Pored toga, način prikazivanja podataka i njihova obrada zavisi od samog izvora podataka, tj. SCADA sistema. Kako bi se u okviru jedne zajedničke operaterske stanice različito struktuirani podaci mogli različito prikazivati, bilo je potrebno razviti metodu koja omogućava definisanje posebne funkcije preslikavanja unutar svakog prilagodnog modula. Ovo je postignuto definisanjem šeme baze podataka u okviru prilagodnog modula koja predstavlja instancu složene strukture klasa realizovane u okviru jedne DLL datoteke. Pomoću tako definisane šeme, osnovna aplikacija tačno zna kako da interpretira podatke dobijene od strane SCADA-e.

Na Sl. 7. je prikazan realizovani generički oblik za tabelarni prikaz procesnih veličina koji prikazuje analogne tačke OASyS DNA SCADA sistema.



Sl. 7. Tabelarni prikaz procesnih veličina

VI. ZAKLJUČAK

U radu su opisane tehnologije korišćene pri razvoju udaljene operaterske stanice i sam način njene realizacije. Realizovana operaterska stanica omogućava nadzor i upravljanje različitim udaljenim SCADA sistemima i predstavlja dobar osnov za dalja poboljšanja. Postignut je visok stepen fleksibilnosti u smislu integracije različitih SCADA sistema u što je moguće kraćem vremenskom periodu. Stabilnost razvijene aplikacije je podignut na jedan viši nivo zahvaljujući višeprocesnoj arhitekturi i međuprocesnoj komunikaciji.

Razvijeni koncept je proveren na primeru integracije sa dva različita SCADA sistema - OASyS DNA i GAUS.

LITERATURA

[1] B. Atlagić, *"GAUS – Generalizovani Akviziciono Upravljački Sistem"*, Fakultet tehničkih nauka, Novi Sad, 2005.

[2] *"OASyS DNA 7.5 Internals"*, Telvent, http://www.telvent.com.

[3] M. MacDonald, *"Pro WPF in C# 2008"*, Second Edition, Apress, 2008.

[4] A. Troelsen, *"Pro C# 2008 and the .NET Platform"*, Fourth Edition, Apress, 2007.

ABSTRACT

The analysis of structure and organization of the processing database of different SCADA systems, as well as the available communication technologies for remote access to this data is carried out in this paper. On the basis of that analysis, the remote operator station has been developed, which enables simultaneous connectivity and usability of different SCADA systems, within the scope of one application. The user interface has been developed by leveraging the Microsoft's WPF grafical environment and the source code has been wirtten in C#.

DEVELOPMENT OF HMI STATION FOR CONNECTION TO DIFFERENT SCADA SYSTEMS

Laslo Brusnjai, Miroslav Popović, Ilija Bašičević, Velibor Mihić, Faculty of Technical Sciences, Novi Sad