UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Uroš Piletič

Aplikacija za varnostno kopiranje podatkov iz krmilnih računalnikov

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2012

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00191/2012
Datum: 03.01.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **UROŠ PILETIČ**

Naslov: **APLIKACIJA ZA VARNOSTNO KOPIRANJE PODATKOV IZ KRMILNIH RAČUNALNIKOV**

APPLICATION FOR BACKUP AND RESTORE DATA OF CONTROL COMPUTERS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Proučite in predstavite koncept krmilnega računalnika in področje varnostnega kopiranja in pri tem dajte poudarek na predstavitvi načinov varnostnega kopiranja. Na podlagi analize načinov varnostnega kopiranja utemeljite argumente za lasten razvoj aplikacije za varnostno kopiranje podatkov iz krmilnih računalnikov. Na podlagi predstavljenih argumentov nato z uporabo klasičnega cikla razvoja programske opreme aplikacijo razvijte.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Uroš Piletič,

z vpisno številko 63080291,

sem avtor diplomskega dela z naslovom:

APLIKACIJA ZA VARNOSTNO KOPIRANJE PODATKOV IZ KRMILNIH RAČUNALNIKOV

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) doc. dr. Rok Rupnik,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja: _____

ZAHVALA

Iskreno se zahvaljujem mentorju, doc. dr. Roku Rupniku s Fakultete za računalništvo in informatiko, za številne nasvete in pomoč pri izdelavi diplomskega dela.

Zahvaljujem se tudi svoji družini in punci, ki so mi vestno stali ob strani v dobrih in slabih trenutkih, med nastajanjem diplomskega dela.

Zahvaljujem pa se tudi svojim sodelavcem, v podjetju Trac d.o.o, ki so me s svojim znanjem in izkušnjami usmerjali pri izdelavi diplomskega dela.

KAZALO

POVZETEK

V diplomskem delu smo se osredotočili na reševanje problematike varnostnega kopiranja in ponastavitve podatkov krmilnih računalnikov. Z razvito aplikacijo želimo preprečiti izgubo ključnih procesnih podatkov v nekem avtomatiziranem proizvodnem procesu in omogočiti čim hitrejšo vzpostavitev delovanja procesa po izpadu. V primeru izpada krmilnega računalnika, je hitra vzpostavitev ponovnega delovanja ključnega pomena, saj prepreči oziroma minimizira izpadli čas procesa in posledično možnost nastale škode.

V uvodnem delu diplomskega dela, je predstavljen pomen varnostnega kopiranja v današnjem času ter opisani osnovni načini varnostnega kopiranja. V nadaljevanju je opisan standard OPC ter predstavljene vrste vmesnikov OPC, med njimi tudi OPC DA, ki ga uporabljamo za dostop do podatkov in strukture krmilnih računalnikov. Na koncu uvodnega dela so na splošno opisani krmilni računalniki ter med razvojem uporabljeni krmilni računalnik proizvajalca Siemens, katerim je prilagojena razvita programska rešitev. Osrednji del predstavlja uporabljena orodja in tehnologije uporabljene med razvojem ter natančneje prikazuje postopek izdelave programske rešitve skozi analizo, načrtovanje in razvoj. Glavni del predstavlja predstavitev razvite programske rešitve ter opisuje način oziroma razdelitev implementacije le-te. V zaključnem delu so predstavljene težave pri razvoju in prikazane možne nadgradnje ter izboljšave razvite rešitve.

Ključne besede

Varnostno kopiranje, OPC, OPC DA, OPC UA, PLC, Krmilni računalnik, Aplikacija za varnostno kopiranje.

ABSTRACT

In this thesis we focus on solving the problems of backup data and restore of the control computers. With the developed application we want to prevent the loss of key process data in an automated production process and enable fast establishment of the process, after a failure. In case of control computer failure, rapid completion of re-operation is crucial as it prevents or minimizes foregone process and consequently the possibility of damage.

The introductory part of the thesis presents the importance of data backup today, and the basic types of backups. The following describes the standard OPC interfaces and illustrates types of OPC interfaces, including OPC DA, which is used to access the data structures and data of control computers. At the end of the introductory part are generally described control computers and Siemens control computer used during development. The essential part lists used tools and technologies used during development and further describes development of software solution through analysis, design and development. The main part includes presentation of developed software solution, and describes structure and implementation of it. The final part of this thesis includes description of difficulties, which appeared during development and possible upgrades and improvements of developed software solution.

Key words

Backup, OPC, OPC DA, OPC UA, PLC, programmable logic computer, backup application.

1 UVOD

V svetu kot ga poznamo danes, nas praktično na vsakem koraku obdajajo podatki. Bodisi so to vsakodnevni podatki, ki definirajo različne vsakodnevne informacije, bodisi pomembni osebni podatki ali ključni podatki za izvajanje poslovnega procesa v podjetju. Ključne podatke si ponavadi prizadevamo ohraniti, saj bi njihova izguba lahko pomenila ure dodatnega dela, oziroma v hujših primerih tudi ogromne finančne izgube. Finančna škoda največkrat nastane zaradi prekinitve dela ali celotnega proizvodnega procesa v primeru, da gre za izgubo pomembnih podatkov za zagotavljanje poslovanja nekega podjetja.

Z namenom obvarovanja ključnih procesnih podatkov v nekem proizvodnem procesu smo si zadali cilj izdelati programsko rešitev, ki bo z varnostnim kopiranjem zagotavljala zaščito ključnih procesnih podatkov na krmilnih računalnikih in hitro ponastavitev teh podatkov, na izbrani krmilni računalnik. Krmilni računalniki so ponavadi ključni del avtomatiziranih sistemov, ki zagotavljajo podporo določenemu delu proizvodnega procesa in bi njihov izpad lahko pomenil zaustavitev tega procesa. Res je, da izpad nižje nivojske krmilne naprave, kot je krmilni računalnik ni pogost pojav, vendar v primeru izpada ter posledično izgube pomembnih procesnih podatkov, to pomeni daljšo prekinitev, oziroma zaustavitve proizvodnega procesa in posledično občutne finančne izgube za podjetje.

V diplomskem delu bomo predstavil standard OPC, ki nam je pri razvoju zagotavljal dostop do strukture in podatkov krmilnih računalnikov. Nadaljevali bomo s splošnim opisom krmilnih računalnikov ter predstavili pri razvoju uporabljeni krmilni računalnik podjetja Siemens na katerem je temeljil razvoj naše programske rešitve. Opisali bomo tudi uporabljena orodja ter tehnologije. Sledila bo podrobna razlaga postopka razvoja programske rešitve, vse od analize in načrtovanja, do izdelave ter opisa možnih izboljšav. Osredotočili se bomo predvsem na razlago postopka razvoja in opis posameznih delov, ki sestavljajo razvito programsko rešitev.

2 VARNOSTNO KOPIRANJE

Vsi se na dnevni ravni srečujemo s podatki različnih tipov, ki jih povezujemo v informacije in z njimi prejemamo, oziroma širimo svoje znanje. Nekateri izmed teh podatkov so nam zelo pomembni, ne glede na to ali so osebne narave ali kritični podatki za potek poslovnega procesa v podjetju. Izguba osebnih podatkov, ponavadi pomeni le nekaj slabe volje, medtem, ko ima lahko izguba podatkov kritičnih za nek poslovni proces precej večje posledice. Podjetja zato posvečajo veliko pozornosti zaščiti teh podatkov.

Znano je, da so podatki shranjeni na računalniku, eni izmed najbolj izpostavljenih, oziroma potencialno ogroženih podatkov. Glavni vzroki za izgubo podatkov so [1]:

- strojna/ sistemska odpoved (44%),
- človeški faktor / napaka (32%),
- napaka programske opreme (4%),
- virusi ali omrežni napadi (7%),
- naravne katastrofe / nesreče (3%).

Izguba kritičnih poslovnih podatkov, pa v nekem podjetju posledično povzroči tudi finančne težave, zaradi zaustavitve ali začasne prekinitve poslovnega procesa. Študije kažejo, da ima lahko izpad ali prekinitev, ki traja več kot deset dni, trajne posledice na podjetje. V polovici takšnih primerov, celo povzroči propad podjetja, v naslednjih petih letih, saj podjetje ne zmore nadomestiti izgube nastale med izpadom [2]. Tabela 1 prikazuje ocenjeni povprečni urni vpliv izgubljenih, oziroma nedostopnih podatkov na nekaterih vsakdanjih področjih.

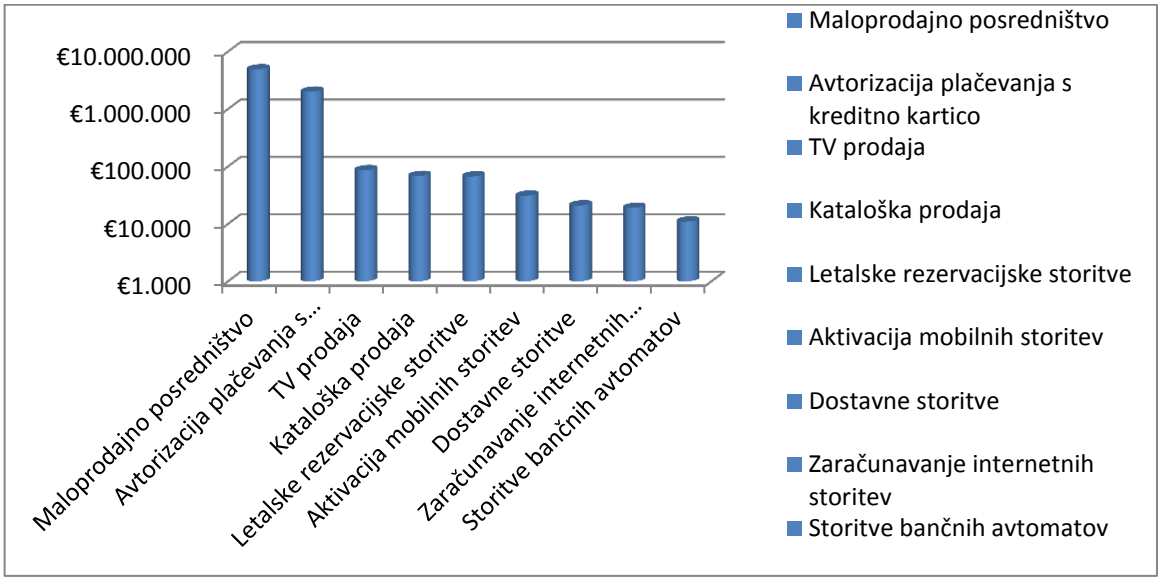


Tabela 1: Povprečni urni vpliv nedostopnih podatkov, na nekaterih vsakdanjih področjih.

V praksi se lahko pred nekaterimi vzroki za izgubo podatkov zaščitimo, oziroma jih preprečimo na več načinov:

- uporaba RAID tehnologije za ustvarjanje redundantnih podatkov,
- uporaba zrcaljenih sistemov (angl. mirrored systems),
- redna izdelava varnostnih kopij s temu namensko programsko opremo,
- odročno elektronsko trezorjenje (angl. electronic vaulting),
- zaščita podatkov z antivirusno programsko opremo,
- izdelava dobrega načrta za primer izpada podatkovnega sistema,
- izdelava dobrega načrta zagotavljanja neprekinjenega delovanja.

Vsaka izmed zgoraj navedenih tehnologij samostojno ne omogočajo popolne zaščite. Da bi svoj sistem ter podatke na njem kar najbolje zaščitili lahko izberemo več načinov/tehnologij zaščite, ki se med seboj dopolnjujejo (npr. varnostno kopiranje in protivirusna programska oprema). Eden izmed osnovnih splošno primernih načinov zaščite podatkov, katerega se poslužujemo tudi pri aplikaciji v sklopu tega diplomskega dela, pa je varnostno kopiranje in obnavljanje.

Kot že samo ime pove, je osnovna naloga varnostnega kopiranja, generiranje podvojenih instanc podatkov, ki jih je mogoče s postopkom obnavljanja v primeru napake obnoviti, oziroma ponastaviti. Varnostno kopiranje izvajamo bodisi ročno, na zahtevo (podatke kopiramo na USB ključ, zunanji trdi disk, optične medije ali kakšen drug medij za shranjevanje podatkov), bodisi periodično na neko časovno enoto (urno, dnevno, mesečno ali kakšno drugo časovno periodo). Na trgu obstaja veliko število aplikacij za izvajanje

varnostnega kopiranja ter orodij za obdelavo večjih količin podatkov, ki imajo že integrirano podporo varnostnemu kopiranju in obnovitvi podatkov.

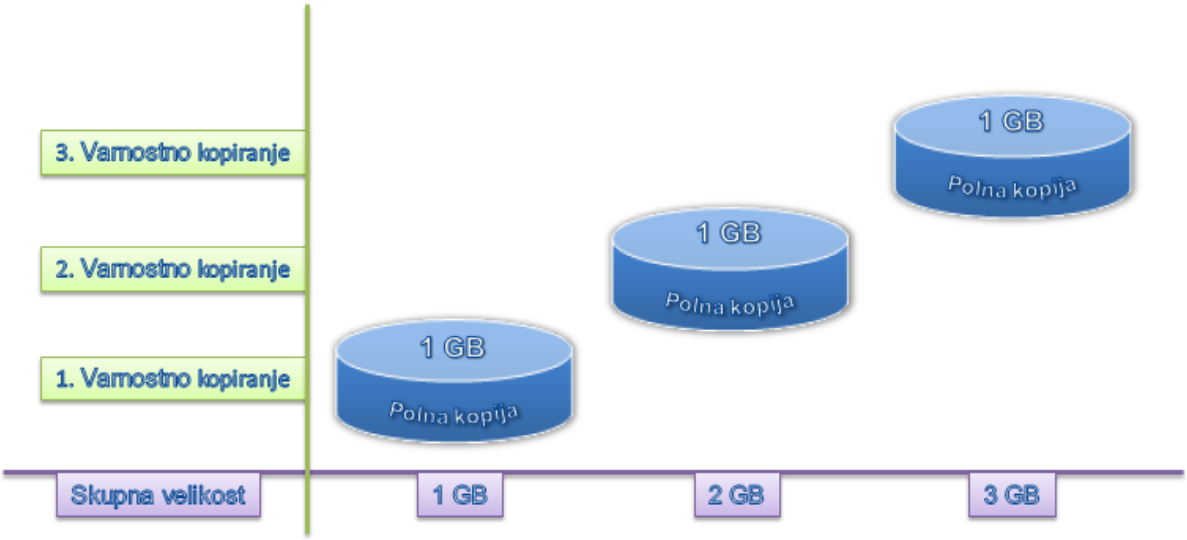
2.1 Strategije varnostnega kopiranja

Poznamo štiri načine, oziroma strategije varnostnega kopiranja [3]. Vsaka strategija natančno določa vsebino posamezne varnostne kopije in postopek obnovitve. Za pravilno izbiro strategije in posledično optimalno zaščito podatkov, moramo najprej poznati kapaciteto in tip podatkov, namenjenih varnostnemu kopiranju. Nato lahko določimo potrebno hitrost obnovitve podatkov, hitrost varnostnega kopiranja, varnost, ter frekvenco varnostnega kopiranja. Na podlagi obstoječih in želenih karakteristik, se odločimo za eno ali kombinacijo strategij varnostnega kopiranja.

2.1.1 Polno varnostno kopiranje

Polno varnostno kopiranje, je začetna, oziroma izhodiščna točka za vse ostale strategije varnostnega kopiranja. Pri tej strategiji, vsaka instanca varnostne kopije, zajema vse izbrane podatke. Na ta način varnostno kopiranje omogoča hitrejšo obnovitev podatkov v primeru napake ter upravljanje posameznih varnostnih kopij. Prav tako je zagotovljena večja varnost, saj potrebujemo za obnovitev podatkov le eno ustrezno instanco varnostne kopije. Prinaša pa tudi slabosti, kot so časovno zahtevnost ustvarjanje varnostnih kopij, ter visoka prostorska zahtevnost, saj se pri vsaki instanci varnostne kopije shranijo vsi podatki.

Strategija polnega varnostnega kopiranja je zaradi polnega nabora podatkov pri vsakem varnostnem kopiranju najboljša rešitev, v primeru manjših količin podatkov in pri daljših periodah varnostnega kopiranja. Zaradi zagotavljanja hitre obnovitve podatkov in centralizirane celotne vsebine varnostne kopije, smo v nadaljevanju pri razvoju programske rešitve uporabili to strategijo varnostnega kopiranja.

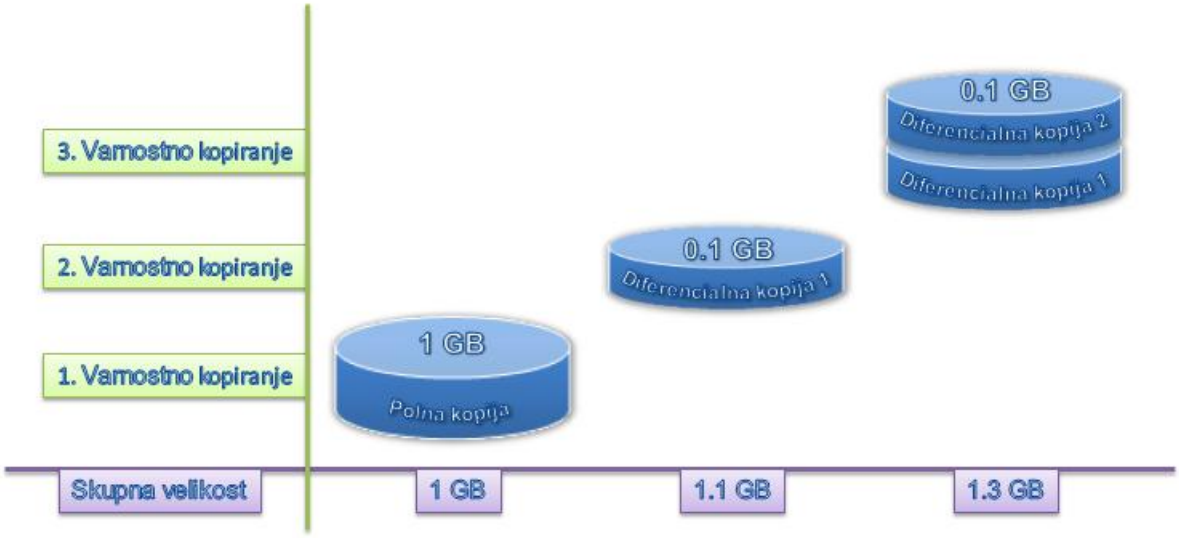


Slika 1: Primer polnega varnostnega kopiranja.

2.1.2 Diferencialno varnostno kopiranje

Ta strategija varnostnega kopiranja deluje po principu zajemanja spremenjenih podatkov, od zadnjega polnega varnostnega kopiranja, ki se izvede na začetku. Ob vsakem varnostnem kopiranju se tako zabeležijo le spremenjeni podatki, ter zadnja shranjena diferencialna kopija (če že obstaja). Obnovitev podatkov je tako manj časovno zahtevna, kot pri inkrementalni strategiji varnostnega kopiranja, saj za obnovitev potrebujemo le prvo polno instanco varnostne kopije ter zadnjo diferencialno kopijo (s tem je zagotovljena tudi večja varnost). Zaradi shranjevanja le spremenjenih podatkov, je ta strategija veliko manj časovno in prostorsko potratna, kot strategija polnega varnostnega kopiranja. Ta strategija omogoča počasnejšo obnovitev podatkov, v primerjavi s polnim varnostnim in počasnejše varnostno kopiranje, ter večjo prostorsko potratnost, v primerjavi z inkrementalno strategijo.

Strategija zagotavlja dober kompromis med hitrostjo in prostorsko potratnostjo. Je dobra izbira, v primeru potrebe po vsestransko zadovoljivih karakteristikah.

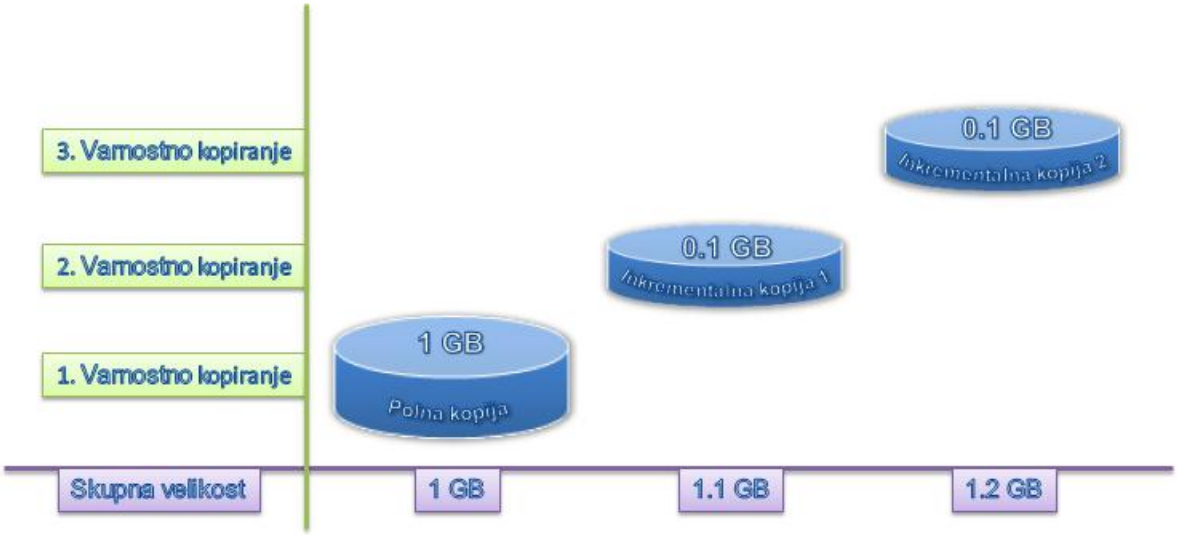


Slika 2: Primer diferencialnega varnostnega kopiranja.

2.1.3 Inkrementalno varnostno kopiranje

Strategija inkrementalnega varnostnega kopiranja, v vsako instanco varnostne kopije vključi le podatke, ki so se spremenili od zadnje polne, diferencialne ali inkrementalne varnostne kopije. Tak način delovanja zagotavlja majhno časovno zahtevnost pri varnostnem kopiranju podatkov in najmanjšo prostorsko zahtevnost, v primerjavi z ostalimi strategijami. Slaba stran te strategije, pa se kaže pri zahtevnem ponastavljanju podatkov. Za obnovitev podatkov na zadnje zabeleženo stanje, je potrebno zajeti prvo polno varnostno kopijo, ter vse nadaljnje instance, oziroma inkremente. Strategija je tako v primerjavi z ostalimi strategijami najmanj časovno učinkovita pri obnavljanju podatkov. Prav tako ta strategija ne zagotavlja pričakovane varnosti v smislu obnavljanja podatkov. V primeru napake, oziroma nepravilnosti na enem izmed inkrementov (ali polni varnostni kopije), je onemogočena pravilna ponastavitev podatkov, saj so za uspešno ponastavitev podatkov na zadnje stanje potrebne vse instance varnostnih kopij v brezhibnem stanju.

Inkrementalna strategija varnostnega kopiranja je s svojo prostorsko nezahtevnostjo in hitrim varnostnim kopiranjem najbolj primerna za velike količine podatkov, ki se pogosto spreminjajo. Zaradi varnostne pomanjkljivosti, ter zahtevnejšega ponastavljanja podatkov, je priporočljivo to strategijo dopolniti s strategijo polnega varnostnega kopiranja na daljšo časovno periodo.

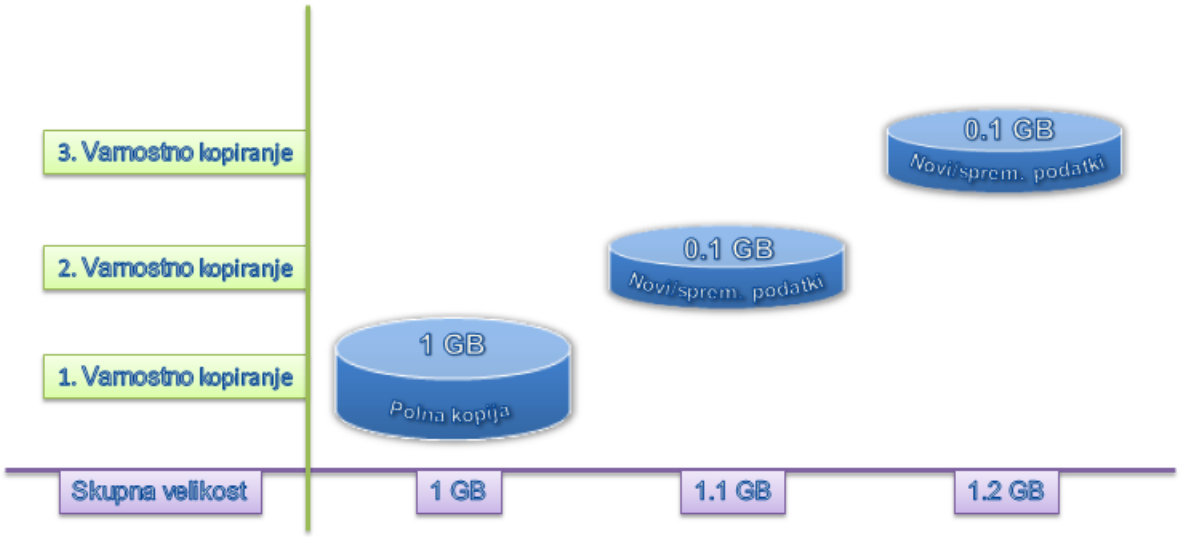


Slika 3: Primer inkrementalnega varnostnega kopiranja.

2.1.4 Zrcalno varnostno kopiranje

Zrcalna strategija deluje na osnovi polnega varnostnega kopiranja, s to izjemo, da kopirane podatke ne kompresira ampak prezrcali (ustvari natanko identično kopijo). Takšna varnostna kopija je takoj uporabna, saj je za njeno uporabnost ne potrebujemo obnavljati. Nekompresirano zrcaljenje podatkov omogoča najbolj časovno učinkovito varnostno kopiranje in zagotavlja največjo prostorsko zahtevnost, v primerjavi z ostalimi strategijami. Prinaša pa tudi večjo slabost, saj tako generirane varnostne kopije ne moremo zaščititi pred neavtoriziranim dostopom. Strategija kopiranja je močno podobna strategiji inkrementalnega varnostnemu kopiranju, saj prav tako ustvari polno varnostno kopijo (vendar zrcalno) podatkov in nato beleži le nove, oziroma spremenjene podatke.

Zrcalno varnostno kopiranje je torej najbolj uporabno v primerih manjših količin podatkov, ki jih želimo tudi v obliki varnostnih kopij ohraniti v berljivem stanju. Uporaba te strategije pa ni primerna v primeru varnostnega kopiranja tajnih podatkov, za katere je nujna zaščita pred neavtoriziranim dostopom.



Slika 4: Primer zrcalnega varnostnega kopiranja.

2.2 Mediji za shranjevanje varnostnih kopij

Izbira pravilnega medija za shranjevanje varnostnih kopij podatkov, je ena izmed ključnih odločitev za vzpostavitev učinkovitega sistema varnostnega kopiranja. Na voljo imamo vrsto medijev, kot so: magnetni trakovi, trdi diski, diskete, CD, DVD, Blue-ray optični mediji, razne pomnilniške kartice, FTP strežnike ali v našem primeru uporabljena podatkovna baza [6]. Preden se lotimo izbire ustreznega medija, moramo poznati tip podatkov, ter oceniti približno količino podatkov, ki se bodo varnostno kopirali. Na podlagi tega izberemo ustrezno strategijo in periodo varnostnega kopiranja, ki ustreza želenim časovnim, prostorskim, ter morebitnim ostalim zahtevam. Pravilno izbrani medij za shranjevanje varnostnih kopij podatkov, mora glede na potrebe, oziroma zahteve uporabnika izpolnjevati sledeče zahteve:

- **Dostopnost**
  V primeru potrebe po stalni razpoložljivosti varnostno kopiranih podatkov, oziroma hitremu dostopu do podatkov.
- **Zanesljivost**
  Za shranjevanje varnostnih kopij moramo izbrati medij z visoko stopnjo zanesljivosti, v danih okoliščinah varnostnega kopiranja.
- **Varnost podatkov**
  V primeru varnostnega kopiranja zasebnih, oziroma tajnih podatkov moramo izbrati medij, ki omogoča preprečevanje neavtoriziranega dostopa. Razni optični mediji, kot so CD, DVD ter podobni, v tem primeru niso primerni. Takšne vrste podatkov je

najprimernejše shranjevati na temu primerne strežnike, zaščitene pred neavtoriziranim dostopom.

- **Obstojnost**
  V primeru shranjevanja varnostnih kopij za daljši čas izberemo medij, ki je sposoben trajno shraniti potrebne podatke.
- **Izpolnjevanje prostorskih zahtev**
  Izbrati moramo medij na podlagi količine podatkov, ki jih želimo shraniti in upoštevati frekvenco izvajanja varnostnega kopiranja. Ključnega pomena je tudi strategija varnostnega kopiranja, ki določa kakšna količina podatkov se shrani ob vsakem varnostnem kopiranju.
- **Izpolnjevanje časovnih zahtev**
  V primeru večje količine podatkov in večje frekvence izvajanja varnostnega kopiranja moramo izbrati medij, ki je sposoben zagotoviti potreben pretok podatkov. Pri tem upoštevamo predvsem želeno hitrost izvajanja varnostnega kopiranja in hitrost obnovitve podatkov.
- **Prenosljivost**
  V primeru potrebe po prenosljivosti, oziroma mobilnosti varnostnih kopij je potrebno izbrati medij, ki poleg prenosljivosti nudi potrebni nivo zanesljivosti in varnosti. Navadno v tem primeru uporabimo medije kot so CD, DVD ali ostale prenosljive pomnilniške medije.

3 OPC

V zadnjih nekaj desetletjih opažamo pospešen razvoj industrijskih sistemov za avtomatizacijo. Od prve pojavitev avtomatiziranih sistemov so le-ti močno napredovali [8]. Vse več je uporabljenih standardov, komunikacijskih protokolov, naprav, programske opreme, vmesnikov in podobnih komponent, ki med seboj komunicirajo in tvorijo enoten avtomatiziran sistem. Zaradi časovno in posledično stroškovno potratne integracije različnih komponent različnih proizvajalcev v tovrstne sisteme, se je pojavila velika potreba po standardizaciji komunikacije med programsko opremo, ter vgrajenimi komponentami sistema.

Leta 1995 se je skupina konkurenčnih vodilnih podjetij na področju avtomatizacije, odločila združiti moči in rešiti to omejitev. Razvili so enoten standard za izmenjavo podatkov v realnem času na Microsoftovi platformi in tehnologiji COM/DCOM, ter jo poimenovali OPC (OLE for Process Control). Leto kasneje je bila ustanovljena fundacija OPC (Slika 5), ki od takrat koordinira in razvija vse OPC specifikacije in produkte, ter skrbi za razvoj in upoštevanje industrijskih potreb pri razvoju novih in obstoječih specifikacij. Posamezna specifikacija določa zahteve in potrebe, ki jim morajo ustrezati vsi produkti OPC (strežniki, odjemalci itd.) in služi kot referenca ponudnikom, ter razvijalcem OPC produktov [9].

Fundacija OPC je v primerjavi z ostalimi skupinami, ki so poskušale razviti podoben standard delovala občutno hitreje, saj je za razliko od le-teh standard OPC razvijala na že obstoječih standardih. Sodobna tehnologija OPC tako danes omogoča podjetjem povezavo ERP aplikacij za upravljanje podjetja (angl. Enterprise Resource Planning), MES aplikacij za upravljanje proizvodnje (angl. Manufacturing Execution System), aplikacij HMI za interakcijo človek-stroj (angl. Human Machine Interface) ter ostalih aplikacij, brez pisanja specifičnih gonilnikov za posamezno napravo ali ročnega prenosa podatkov za uporabo ,na višje nivojskih sistemih. Podjetja lahko tako svoj čas in denar usmerijo v povečanje produktivnosti in ne več toliko v zagotavljanje povezljivosti znotraj podjetja.

Slika 5: Logotip Fundacije OPC.

Prvi mejnik v razvoju OPC standarda je predstavljala specifikacija OPC DA (angl. OPC Data Acess Specification), ki je definirala mehanizme in funkcionalnosti branja in pisanja procesnih podatkov in je še danes najpogosteje uporabljen OPC vmesnik. Potrebam in trendom razvoja so sledile še mnoge druge specifikacije, kot so: specifikacija za nadzor in

obdelavo dogodkov in alarmov OPC AE (angl. OPC Alarms and Events Specification), specifikacija za dostop do zgodovinskih procesnih podatkov OPC HDA (angl. OPC Historical Data Access Specification), specifikacija za dostop do procesnih podatkov preko XML prehoda, imenovana OPC XML-DA (angl. OPC XML Data Access Specification), novejšo specifikacijo OPC UA (angl. OPC Unified Arhitecture) za odpravljanje pomanjkljivosti tehnologije COM/DCOM, ter vrsto ostalih opisnih ter razširitvenih specifikacij. Število produktov OPC se je tako le v nekaj letih močno povečalo in omogočilo standardu OPC prehod iz dobro zasnovanega koncepta, v univerzalno sprejet standard, ki omogoča izmenjavo podatkov med različnimi sistemi za avtomatizacijo.

Fundacija OPC šteje danes že 427 članov, med temi skoraj vse najpomembnejše dobavitelje opreme za avtomatizacijo. Po zaslugi fundacije OPC se danes ta standard uporablja praktično v vsakem avtomatiziranem sistemu in vedno več je podjetij, ki ponujajo svoje OPC produkte. Za zagotavljanje kakovosti in kompatibilnosti med lastno razvitimi produkti, je fundacija OPC uvedla dvonivojski sistem certificiranja. Z logotipi, kot jih prikazuje Slika 6, so tako označeni le tisti produkti OPC, ki so certificirani in ustrezajo strogim zahtevam kompatibilnosti in so prestali številne zmogljivostne teste.

Slika 6: Logotipa certificiranega produkta OPC.

Pri izbiri ustreznega produkta, ki nam bo omogočil povezavo in komunikacijo med različnimi komponentami našega sistema, moramo biti pazljivi in izbrati tisti produkt oziroma vmesnik, ki je ustrezno certificiran.

3.1 OPC vmesniki

Osnovna zamisel fundacije OPC pri razvoju standarda OPC je bila, da bodo proizvajalci različnih komponent za avtomatizacijo (npr. krmilnih računalnikov) vgrajevali OPC strežnike v svoje produkte in tako zagotovili možnost povezovanja preko arhitekture odjemalec-strežnik, katere se poslužujejo vsi vmesniki OPC. Aplikacija, ki predstavlja odjemalca OPC posreduje zahteve za branje in pisanje strežniku OPC. Strežnik OPC na eni strani komunicira z odjemalcem OPC (v našem primeru aplikacijo za varnostno kopiranje) in na drugi strani z nižje-nivojskimi napravami, katerih podatke v razumljivi obliki ponuja odjemalcem OPC, preko svojega vmesnika. Aplikacije, ki predstavljajo odjemalce OPC lahko tako pišejo in berejo podatke z več različnih nižje-nivojskih naprav (v našem primeru krmilnih računalnikov), preko skupnega strežnika OPC.

Glede na zahteve, oziroma potrebe po dostopu do različnih vrst podatkov (trenutni podatki, zgodovinski podatki, alarmi itd.), lahko strežnik OPC podpira klasični vmesnik kot so: OPC DA, OPC AE, OPC HDA, OPC XML-DA ali poenoten vmesnik OPC UA, s katerim lahko zajamemo vse funkcionalnosti klasičnih vmesnikov.

3.1.1 Klasični vmesniki OPC

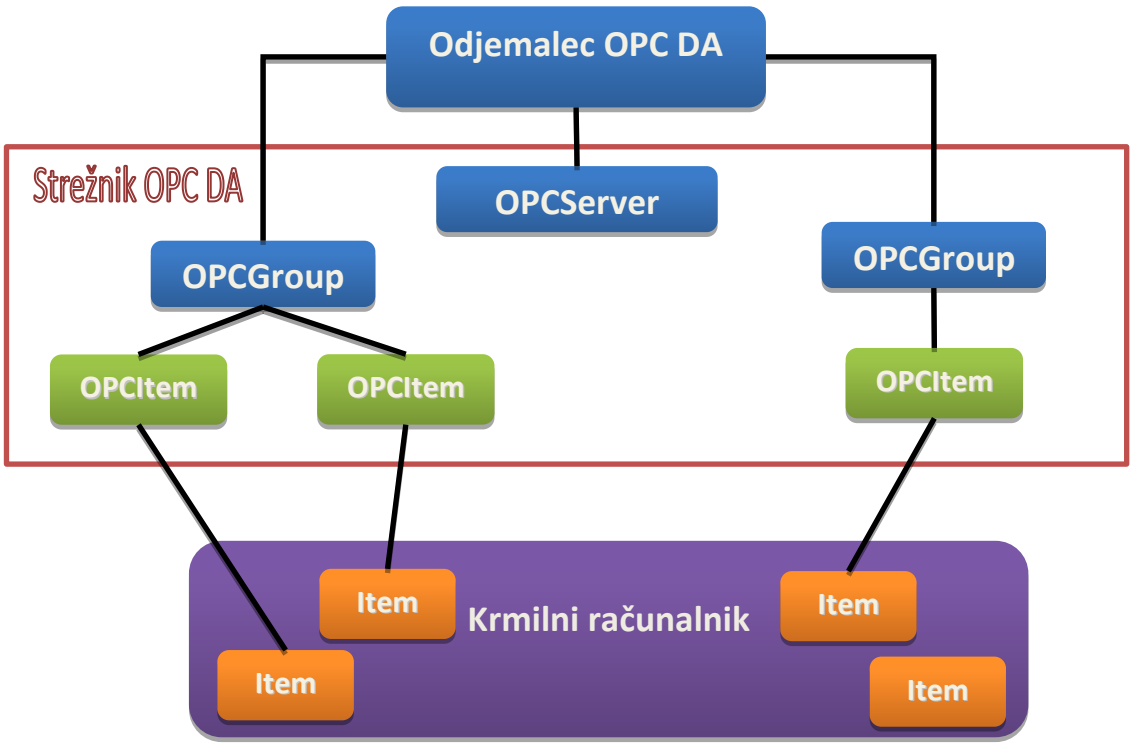
Med osnovne klasične OPC vmesnike uvrščamo vmesnike OPC DA, OPC AE ter OPC HDA. Danes še vedno najpogosteje uporabljen je vmesnik OPC DA, ki omogoča dostop do realno-časovnih, oziroma trenutnih procesnih podatkov [10]. Vmesnik OPC AE omogoča dostop do procesnih dogodkov, ter alarmov. Preko vmesnika OPC HDA, pa dostopamo do zgodovinskih procesnih podatkov. Kot razširitev klasičnim vmesnikom OPC, pa so na voljo tudi ostali vmesniki, kot je vmesnik OPC XML-DA, ki omogoča dostop do procesnih podatkov preko XML prehoda.

Pri vseh klasičnih OPC specifikacijah (z izjemo OPC XML-DA), oziroma vmesnikih, komunikacija med odjemalcem in strežnikom, temelji na Microsoftovi tehnologiji COM (angl. Component Object Model) in DCOM (angl. Distributed Component Object Model). Tehnologija COM omogoča odjemalcu uporabo, oziroma strežnikovih funkcij ali metod, ki jih le-ta ponuja preko svojega COM vmesnika. Tehnologija DCOM pa je le razširitev COM tehnologije, v smeri omogočanja enakih funkcionalnosti porazdeljeno (angl. Distributed) preko omrežja. Uporaba te tehnologije je fundaciji OPC omogočila hiter razvoj standarda OPC, saj tako ni bilo potrebe po razvoju novega komunikacijskega protokola. Slabost COM/DCOM tehnologije se kaže predvsem v zahtevnosti konfiguracije, odvisnosti od platforme ter težavam pri komunikaciji preko požarnega zidu.

3.1.1.1 Vmesniki OPC Data Access

Vmesnik OPC Data Access, poznan tudi kot OPC DA, je najstarejši ter najpogosteje uporabljen OPC vmesnik. Aplikacijam, ki implementirajo odjemalce OPC DA omogoča komunikacijo z nižje-nivojskimi napravami. Odjemalci OPC DA lahko preko vmesnika OPC DA na razne nižje-nivojske naprave zapisujejo, ter z njih berejo trenutne procesne podatke [10].

Aplikacija, ki implementira odjemalca OPC DA vzpostavi povezavo s strežnikom OPC DA tako, da ustvari objekt *OPCServer*, ki predstavlja korenski element v hierarhiji OPC objektov. Z vzpostavitvijo povezave preko objekta *OPCServer*, lahko odjemalec pregleduje strukturo strežnika, priključene podatkovne vire ,oziroma nižje-nivojske naprave, lastnosti strežnika ter priključenih podatkovnih virov. Za dostop do posameznega podatka na specifičnem podatkovnem viru, ter njegovih metapodatkov odjemalec ustvari objekt *OPCItem*, ki predstavlja končni element v hierarhiji. Ta objekt predstavlja neposredno povezavo do podatka, ki ga želimo prebrati s podatkovnega vira (npr. podatek na krmilnem računalniku). Posamezne objekte *OPCItem* odjemalec doda v objekt *OPCGroup*, ki predstavlja množico podatkov, oziroma objektov *OPCItem*. Tako združene objekte lahko nato beremo ali zapisujemo kot celoto. Omogočeni so tudi nekateri načini branja posameznega podatka preko objekta *OPCItem*, ki v ozadju avtomatično generirajo začasni objekt *OPCGroup*. Slika 7 prikazuje primer hierarhije OPC DA objektov za dostop do podatkov.



Slika 7: Primer hierarhije OPC DA objektov za dostop do podatkov na krmilnem računalniku.

Obstaja več načinov izmenjave podatkov med odjemalcem in strežnikom OPC DA: sinhrono branje (angl. Synchronous Read), asinhrono branje (angl. Asynchronous Read), osveževanje (angl. Refresh) in naročnina (angl. Subscription).

- **Sinhrono branje**
  Odjemalec OPC DA poda zahtevo za branje podatka s strežnika OPC DA in čaka na povratno informacijo. Sinhrono branje je najbolj primerno pri hitrem načinu dostopa.
- **Asinhrono branje**
  Odjemalec OPC DA takoj po zahtevi za branje dobi povratno informacijo s strežnika OPC DA (dejanska hitrost je odvisna od načina dostopa). Odjemalec v tem primeru dobi zahtevani podatek z določenim zamikom. Asinhrono branje je najbolj primerno pri daljših dostopnih časih.
- **Osveževanje**
  Odjemalec OPC DA prebere vse aktivne podatke, oziroma objekte *OPCItem*, ki pripadajo aktivnim objektom *OPCGroup*. Aktivnost posameznega objekta OPCItem in OPCGroup lahko uporabnik samodejno nastavlja, preko parametra *Active*.
- **Naročnina**
  Strežnik OPC DA periodično bere podatke s podatkovnega vira in jih v primeru spremembe podatka ali statusa le-tega posreduje odjemalcu OPC DA. Slednji lahko prilagodi periodo preverjanja podatkov.

Poleg načina izmenjave podatkov, lahko določimo tudi način dostopa do določenega podatka, preko strežnika OPC DA. Odjemalec OPC DA lahko podatek prebere z notranjega pomnilnika strežnika OPC DA (angl. Cache) ali neposredno s podatkovnega vira (angl. Device). Za razliko od branja podatka se zapisovanje podatkov vedno izvršuje direktno na podatkovni vir in nikoli preko pomnilnika strežnika OPC DA.

Vsak prebrani podatek je opremljen še z metapodatkom o časovnem žigu branja (angl. Timestamp) in metapodatkom o statusu prebranega podatka (angl. Status). Slednji statusni podatek, med drugimi manj pomembnimi podatki, vsebuje tudi podatek o kvaliteti branja. Ta podatek ima lahko vrednost *good* (kvaliteta podatka je dobra), *bad* (podatek ali podatkovni vir ni na voljo) ali *uncertain* (podatek ali podatkovni vir je na voljo vendar ni pravilno definiran).

Fundacija OPC je do sedaj izdala že vrsto verzij OPC DA specifikacije. Odjemalec OPC ni nujno kompatibilen z vsemi verzijami OPC DA strežnika in obratno. V ta namen imamo na voljo vrsto aplikacij za upravljanje izmenjave podatkov, ki omogočajo komunikacijo drugače nekompatibilnim odjemalcem in strežnikom OPC DA.

3.1.1.1.1 Vmesniki OPC XML-DA

XML (angl. Extensible Markup Language) je fleksibilen in enostaven označevalni jezik za opis strukturiranih podatkov, ki je na prvi pogled zelo podoben jeziku HTML (angl. Hyper Text Markup Language) [10]. Zaradi svoje preprostosti in pregledne zgradbe, je bil s strani Fundacije OPC prepoznan kot pomembna tehnologija za omogočanje omrežne izmenjave podatkov med OPC podprtimi napravami. Produkt zamisli je postal vmesnik OPC XML-DA, ki ponuja vse ključne funkcionalnosti vmesnika OPC DA (opisan v poglavju 3.1.1.1) in namesto tehnologije COM/DCOM uporablja protokol SOAP (angl. Simple Object Access Protocol). Protokol SOAP združuje tehnologijo XML za opis parametrov interakcije, med klientom in strežnikom ter tehnologijo HTTP (angl HyperText Transfer Protocol), kot transportni protokol. Vmesnik OPC XML-DA je tako postal prvi platformno neodvisni vmesnik OPC.

Zaradi uporabljene tehnologije za izmenjavo podatkov, vmesnik OPC XML-DA implementira le sledeče funkcionalnosti, oziroma funkcije vmesnika OPC DA:

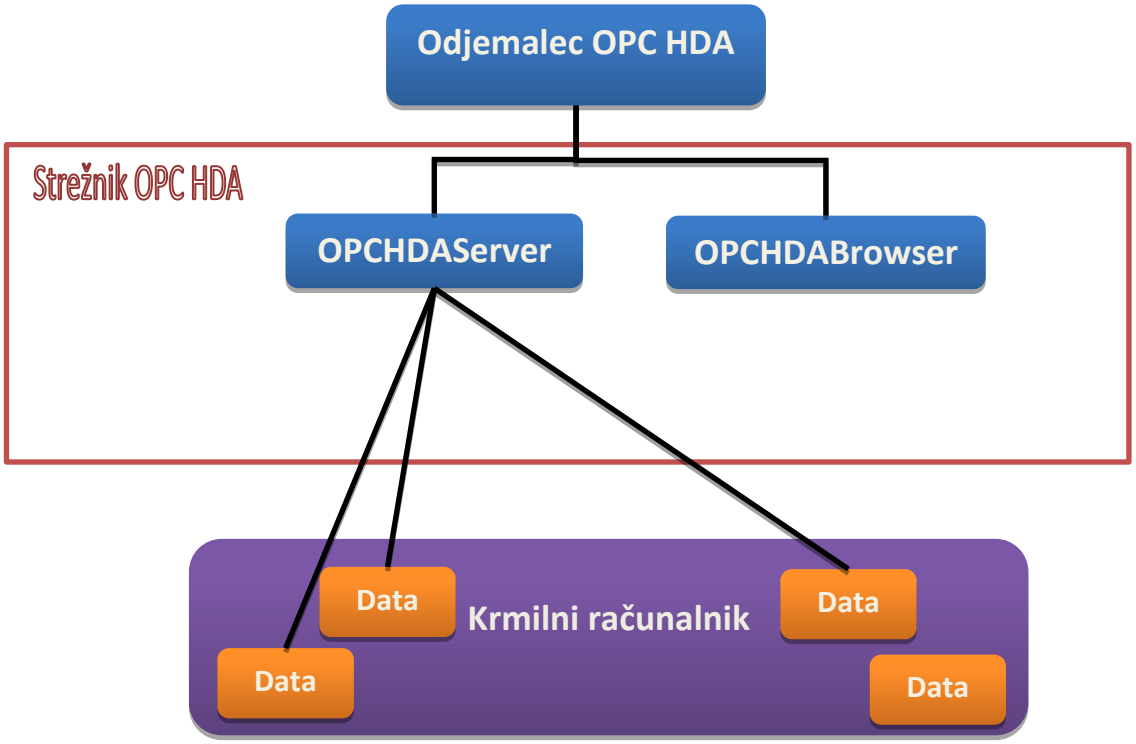
- *GetStatus*
  Odjemalec uporabi za preverjanje razpoložljivosti strežnika OPC XML-DA, ter pridobivanje statusnih informacij.
- *Browse*
  Omogoča pridobivanje informacij o imenskem prostoru.
- *GetProperties*
  Omogoča pridobivanje lastnosti dosegljivih podatkov.
- *Read*
  Odjemalec uporabi za pridobivanje vrednosti enega ali več podatkovnih elementov.
- *Write*
  Odjemalec uporabi za pisanje vrednosti enega ali več podatkovnih elementov.
- *Subscription*
  Omogoča vzpostavitev naročnine, oziroma avtomatične izmenjave podanih podatkovnih elementov med odjemalcem in strežnikom OPC XML-DA. Odjemalec dobi le spremenjene vrednosti zahtevanih elementov.
- *SubscriptionCancel*
  Odjemalec uporabi za preklic naročnine.
- *SubscriptionPolledRefresh*
  Odjemalec uporabi za pridobivanje vrednosti zahtevanih podatkovnih elementov preko naročnine.

Za povezavo strežnikov OPC XML-DA s klasičnimi odjemalci OPC DA in obratno lahko uporabimo vrsto že izdelanih prehodov s strani strežnika ali odjemalca. Vmesniki OPC XML-DA se kljub svoji platformni neodvisnosti ni uspel dobro uveljaviti, saj zaradi svoje okrnjene funkcionalnosti ni konkurenčen novejšim vmesnikom OPC.

3.1.1.2 Vmesniki OPC Historical Data Access

Vmesnik OPC HDA omogoča dostop do arhiviranih zgodovinskih procesnih podatkov. Vmesnik te vrste lahko uporabimo za spremljanje specifičnih vrednosti med proizvodnim procesom ter na podlagi pridobljenih podatkov izvajamo analizo le-teh, optimizacijo, preverjamo skladnost z zahtevami o delovanju naprave ali le vizualiziramo delovni proces [10].

Kljub temu, da vmesnik OPC HDA deli uporabljeno tehnologijo z vmesnikom OPC DA (opisan v poglavju 3.1.1.1) sta si po vrsti ponujenih podatkov ravno nasprotna, saj slednji omogoča prikaz trenutnih procesnih podatkov. Prav tako kot vsi ostali vmesniki OPC, tudi vmesnik OPC HDA deluje po principu odjemalec-strežnik. Odjemalec OPC HDA lahko podobno kot pri OPC DA, na strežniku OPC HDA ustvari hierarhijo objektov preko katerih dostopa do arhiviranih zgodovinskih podatkov. Slika 8 prikazuje primer OPC HDA hierarhije.



Slika 8: Primer hierarhije OPC HDA.

Glavni, oziroma korenski objekt predstavlja *OPCHDAServer*, ki odjemalcu ponuja sledeče funkcionalnosti:

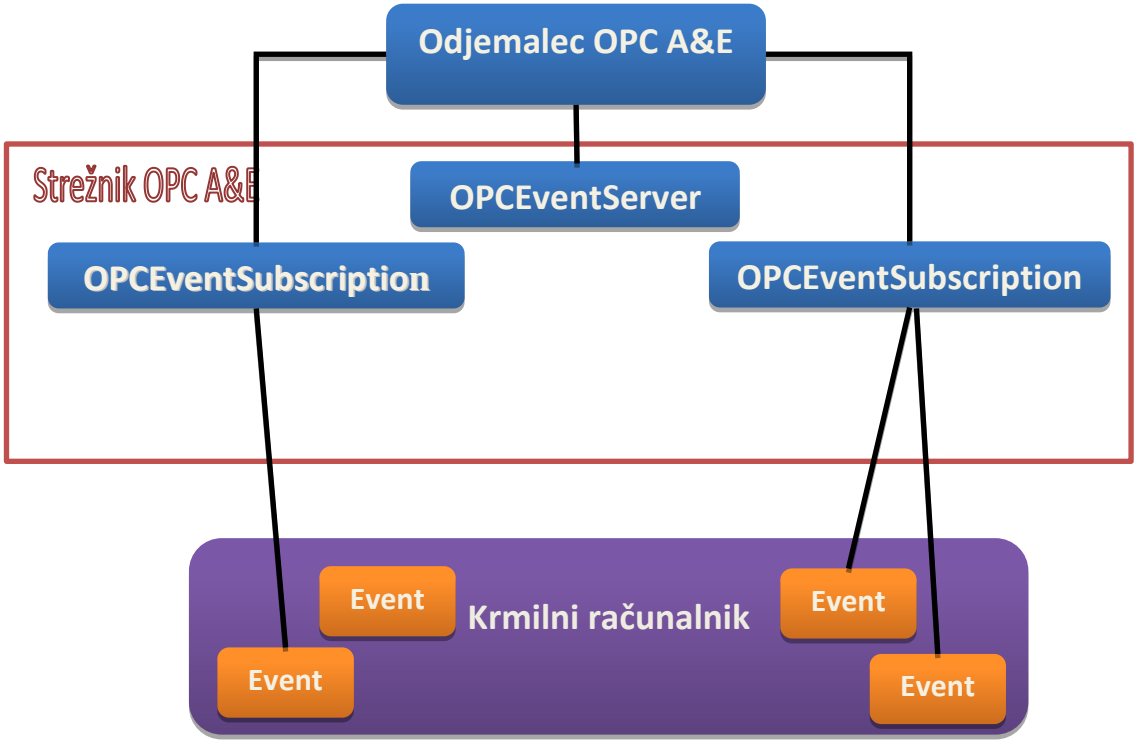
- branje podatkov,
- posodabljanje, zapisovanje in brisanje zgodovinskih podatkov,
- branje in dodajanje opomb specifičnim podatkom,
- vzpostavljanje periodične transakcije podatkov med odjemalcem in strežnikom.

Zaradi neuporabljenosti s strani aplikacije za varnostno kopiranje, razvite v sklopu tega diplomskega dela, ne bomo podajali podrobnosti o načinu delovanja vmesnika OPC HDA.

3.1.1.3 Vmesniki OPC Alarms & Events

Vmesniki OPC A&E se uporabljajo za spremljanje in potrjevanje procesnih dogodkov in alarmov, ki opozarjajo na napake ali zgolj dogodke na enem ali več podatkovnih virih [10]. Kot podatkovni vir, lahko strežnik uporablja nižje-nivojske naprave, aplikacije ali strežnik OPC DA. Strežnik OPC A&E za razliko od OPC DA ne posreduje podatkov o vrednosti elementa na zahtevo odjemalca, ampak avtomatsko dostavi obvestilo o dogodku, oziroma alarmu do odjemalca.

Podobno kot pri OPC DA, lahko odjemalec tudi na strežniku OPC A&E ustvari hierarhijo objektov za dostop in spremljanje dogodkov. Slika 9 predstavlja primer OPC A&E hierarhije.



Slika 9: Primer OPC A&E hierarhije.

Glavni, oziroma korenski objekt predstavlja *OPCEventServer* preko katerega odjemalec dostopa do OPC A&E strežnika. Z ustvarjanjem objekta *OPCEventSubscription* se odjemalec nato naroči na spremljanje dogodkov in alarmov. Preko tega objekta lahko odjemalec določi tudi kriterije za prejemanje obvestil in tako prejme le obvestila o tistih dogodkih, ki izpolnjujejo postavljene kriterije (npr. temperatura območja 1 je presegla dovoljeno mejo.). Za namen pregledovanja območja dogodkov, lahko odjemalec ustvari ločen objekt *OPCEventBrowser*.

Zaradi neuporabljenosti s strani aplikacije za varnostno kopiranje, razvite v sklopu tega diplomskega dela, ne bomo podajali podrobnosti o načinu delovanja vmesnika OPC A&E.

3.1.2 Vmesniki OPC Unified Architecture

V času razvoja specifikacije za implementacijo klasičnih OPC vmesnikov prioriteta razvoja ni bila usmerjena v zagotavljanje varnosti, neodvisnosti od platforme ali omrežne dostopnosti ampak le način za izmenjavo podatkov in zagotavljanje osnovne stopnje interoperabilnosti med napravami različnih proizvajalcev. Kompleksnost proizvodnih sistemov in sistemov za upravljanje in nadzor venomer narašča in zahteva tehnologijo, ki omogoča realizacijo tako zapletenih sistemov. V ta namen je fundacija OPC razvila novo specifikacijo imenovano OPC Unified Architecture ali OPC UA, ki združuje funkcionalnosti klasičnih OPC specifikacij in hkrati omogoča realizacijo tako enostavnih, kot tudi zelo kompleksnih sistemov [9]. Pri

klasičnih vmesnikih OPC, je moral odjemalec v primeru dostopa do različnih strežnikov OPC vsebovati instanco določenega vmesnika za vsako vrsto strežnika posebej (OPC DA, OPC HDA, OPC A&E). Pri OPC UA za zagotavljanje podobne funkcionalnosti, potrebujemo le en OPC UA vmesnik.

Bistvena sprememba, ki jo implementira OPC UA, je opustitev Microsoftove tehnologije COM/DCOM za komunikacijo med odjemalcem in strežnikom. Za to specifikacija OPC UA dopušča možnost izbire komunikacijskega protokola, kot je TCP/IP, HTTP ali SOAP. Vsi protokoli, ki so na voljo zagotavljajo komunikacijo preko interneta ter požarnih zidov neodvisno od platforme. Za komunikacijo med odjemalcem in strežnikom skrbi tako imenovani komunikacijski sklad, ki izvaja varnostne, kodirne ter komunikacijske operacije. OPC UA vpeljuje tudi nove ter izboljšane funkcionalnosti za zagotavljanje zanesljivosti z redundantnostjo (angl. Redundancy), prepoznavanje prekinitev v komunikaciji odjemalec-strežnik (angl. Heartbeat), sistem potrjevanja izmenjave podatkov (s tem preprečimo izgubo podatkov ob izpadu komunikacije, med odjemalcem in strežnikom), ter daje vsesplošen poudarek na razširljivosti, zmogljivosti in interoperabilnosti. Kot gre pričakovati je tudi sama specifikacija OPV UA precej kompleksnejša od klasičnih specifikacij. Razdeljena je na več delov, ki opisujejo posamezne mehanizme delovanja vmesnikov OPC UA.

Danes še vedno najpogosteje uporabljen vmesnik OPC, ostaja OPC DA. Kljub uporabi klasičnih vmesnikov s strani odjemalca ali strežnika OPC, lahko uporabimo vrsto programskih rešitev, ki omogočajo komunikacijo med klasičnimi vmesniki ter vmesniki OPC UA [11].

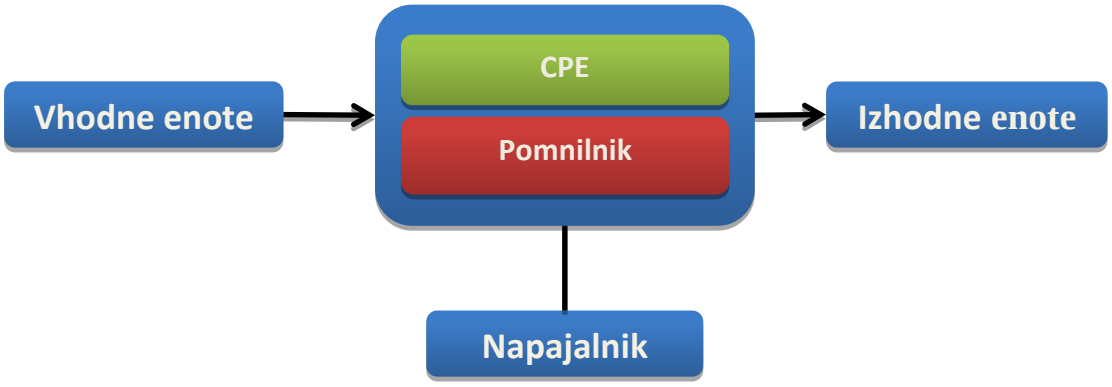
4 KRMILNI RAČUNALNIKI

Že od začetka 70. let prejšnjega stoletja, se za nadzor in krmiljenje vseh vrst mehanskih in električnih sklopov avtomatiziranih sistemov uporabljajo krmilni računalniki, oziroma programirljivi logični krmilniki (angl. Programmable Logical Computer ali PLC) [12]. Na tem področju so nadomestili trdo ožičeno logiko ali relejske sisteme, ki so bili sestavljeni iz številnih elektromehanskih relejev. Sodobni krmilni računalniki so naprave, ki pogosto predstavljajo temelj celotne avtomatizacijske strukture in za razliko od predhodnih načinov krmiljenja omogočajo hitro in enostavno odkrivanje napak ter zmožnost hitrega prilagajanja na spremembe.

Od svojih začetnih dni so krmilni računalniki močno napredovali. Danes omogočajo hitrejše obdelovanje podatkov, zanesljivejše delovanje, razne razširitvene možnosti, podpirajo številne komunikacijske protokole, vsebujejo zmogljivejše vhodno/izhodne sisteme, omogočajo programiranje v več programskih jezikih, vsebujejo napredno diagnostiko delovanja in mnoge druge izboljšave.

4.1.1 Zgradba in delovanje

Podobno kot klasični osebni računalniki, vsebujejo tudi krmilni računalniki centralno procesno enoto, oziroma procesor, pomnilnik, komunikacijski vmesnik, ter vhodno/izhodni vmesnik [12]. Nekateri modeli krmilnikov celo podpirajo modularno zgradbo, kar pomeni, da krmilni računalnik sestavimo iz posameznih modulov, ki predstavljajo posamezne komponente (napajalni modul, centralno procesni modul, komunikacijski modul itd.). To omogoča poljubno razširitev funkcionalnosti krmilnega računalnika, ki pa je vendarle omejena z zmogljivostjo procesne ter napajalne enote. Slika 10 prikazuje osnovne sestavne dele krmilnega računalnika.



Slika 10: Osnovna zgradba krmilnega računalnika.

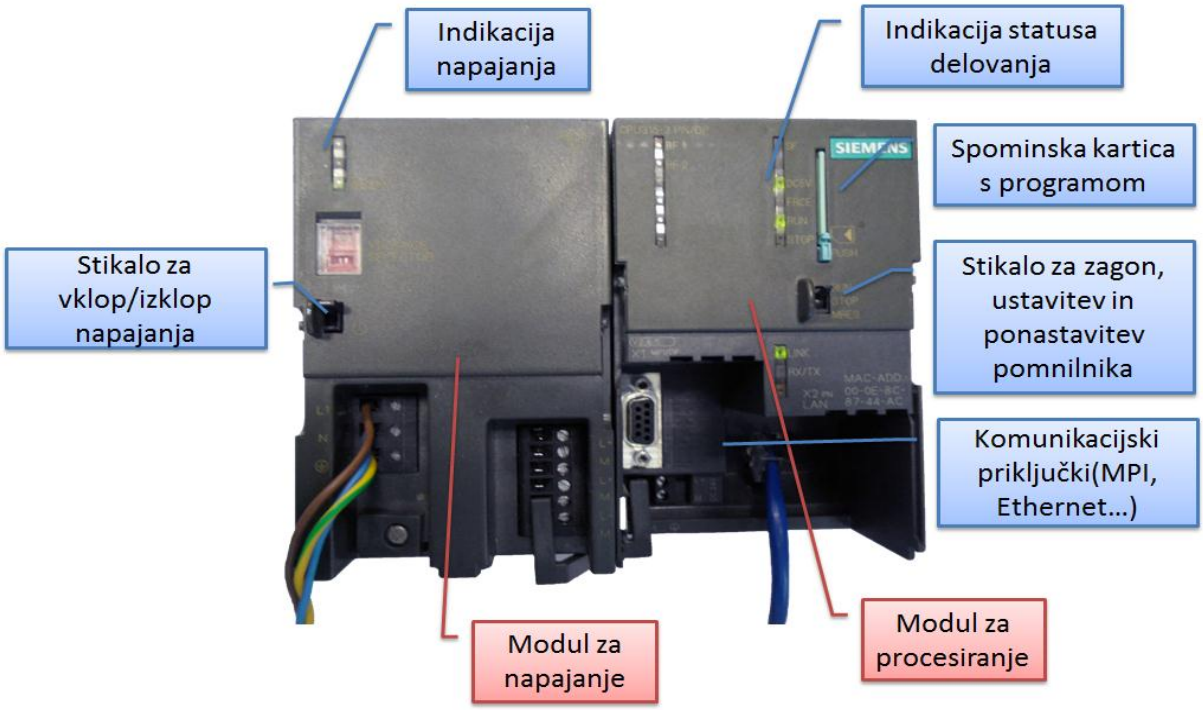
Delovanje krmilnega računalnika je določeno z uporabniškim programom, ki je shranjen v pomnilniku. Pomnilnik je razdeljen na del rezerviran za uporabniški program ter del namenjen shranjevanju podatkov. Centralno procesna enota nadzira vse dele krmilnega računalnika ter izvaja uporabniški program.

V našem primeru oziroma testnem krmilnem računalniku znamke Siemens, uporabniški program za svoje delovanje definira in uporablja logične ter podatkovne bloke. V nadaljevanju bodo nam najbolj pomembni bloki, podatkovni bloki DB (angl. Shared Data Blocks), ki se uporabljajo za shranjevanje podatkov, oziroma vsebujejo vrednosti spremenljivk, s katerimi razpolaga uporabniški program.

Vhodne ter izhodne enote skrbijo za povezavo z različnimi napravami in predstavljajo vmesnik, ki procesni enoti omogoča izvajanje operacij nad povezanimi napravami, ter branje procesnih podatkov s povezanih naprav. Na vhodne enote krmilnega računalnika so ponavadi povezane naprave za zajem podatkov o trenutnem stanju parametrov okolja, oziroma veličin delovnega procesa (napolnjenost rezervoarja, temperatura, pretok, koncentracija plina, tlak ipd.). Vhodne naprave so ponavadi razni senzorji, stikala, tipala in podobne naprave. Na izhodne enote pa so ponavadi povezane naprave, ki služijo za izvajanje neke operacije (zagon motorja, odpiranje in zapiranje ventila, vklop črpalke itd.). Izhodne naprave ponavadi predstavljajo črpalke, ventili, motorji, skratka naprave za izvajanje raznih akcij.

4.2 Krmilni računalnik SIEMENS SIMATIC S7-300 CPU 315-2 PN-DP

Siemens je danes eno izmed vodilnih podjetij, na področju razvoja krmilnih računalnikov. Na trgu ponujajo proizvode za implementacijo celovitih rešitev, na področju avtomatiziranih sistemov, pod skupnim imenom SIMATIC [13]. Pod okrilje te družine izdelkov, pa spada tudi krmilni računalnik, ki smo ga uporabili pri razvoju programske rešitve, razvite v sklopu tega diplomskega dela. Uporabljeni krmilni računalnik pripada seriji krmilnikov S7-300, ki predstavlja srednjo cenovno in zmogljivostno skupino izdelkov za avtomatizacijo. Gre za razširljiv modularen krmilni računalnik tipa S7-315-2, namenjen raznovrstnim krmiljenjem procesov. Pri razvoju programske rešitve za varnostno kopiranje smo se osredotočili na kompatibilnost predvsem s krmilnimi računalniki proizvajalca Siemens, saj zavzemajo največji tržni delež oziroma jih najpogosteje srečujemo na našem trgu. Slika 11 prikazuje glavne sestavne dele krmilnega računalnika uporabljenega pri razvoju programske rešitve.



Slika 11: Glavni deli uporabljenega krmilnega računalnika SIEMENS.

5 RAZVOJ APLIKACIJE ZA VARNOSTNO KOPIRANJE PODATKOV S KRMILNIH RAČUNALNIKOV

Pri razvoju vsake aplikacije gre le-ta skozi tako imenovani življenjski cikel programske opreme, oziroma proces razvoja programske opreme. Ta proces ponavadi zajema več orodij in tehnologij, ter je razdeljen na več aktivnosti, ki vodijo k cilju izdelave neke programske rešitve. V okviru diplomskega dela smo razvili aplikacijo, ki izvaja varnostno kopiranje izbranih parametrov s krmilnih računalnikov in ponuja možnost ponastavite podatkov le-tega na stanje izbrane varnostne kopije. Aplikacija ponuja uporabniku interakcijo preko dovršenega uporabniškega vmesnika, preko katerega lahko izvaja operacije, kot so:

- pregled in upravljanje parametrov izbranih za varnostno kopiranje,
- pregled varnostnih kopij glede na izbiro krmilnega računalnika,
- ponastavitev parametrov izbranega krmilnega računalnika,
- nastavitev urnika varnostnega kopiranja,
- nastavitev periodičnega preverjanja delovanja izbranih krmilnih računalnikov,
- pregled dogodkov nastalih med izvajanje aplikacije in varnostnim kopiranjem.

V nadaljevanju smo predstavili uporabljena orodja in tehnologije ter opisali aktivnosti, ki so bile del procesa razvoja programske opreme oziroma aplikacije za varnostno kopiranje.

5.1 Uporabljena orodja in tehnologije

5.1.1 Višji programski jezik C#

Jezik C# je objektno orientiran višji programski jezik in je namenjen razvoju aplikacij na platformi Microsoft .NET (angl. .NET Framework), ki ga lahko dodamo operacijskemu sistemu Windows [14]. Primeren je tako za programerske začetnike, kot poznavalce in strokovnjake, saj omogoča hitro učenje ter implementacijo tako enostavnih kot kompleksnih programskih rešitev. Pri razvoju aplikacije smo uporabili programski jezik C# v razvojnem okolju Visual Studio .NET, ki je namenjen razvoju aplikacij v tem in ostalih programskih jezikih podprtih na platformi .NET.

5.1.2 Razvojno okolje Visual Studio .NET

Visual Studio je integrirano razvojno okolje (angl. Integrated Development Environment ali IDE), razvito s strani podjetja Microsoft [15]. Namenjeno je razvoju vse od konzolnih aplikacij, aplikacij z grafičnim uporabniškim vmesnikom, servisnih aplikacije pa do spletnih aplikacij, strani in servisov. Podpira razvoj aplikacij v programskih jezikih, kot so C#, C++, C, Visual Basic, F# itd. Okolje vključuje vizualni urejevalnik kode, razhroščevalnik, prevajalnik, prav tako pa je z mnogimi dodatki mogoče razširiti območje podprtih programskih jezikov in funkcionalnosti. Razvoj aplikacij je mogoč za platforme Microsoft Windows, Microsoft Mobile, Microsoft CE oziroma platforme, ki podpirajo, oziroma implementirajo ogrodje .NET.

Aplikacije razvite v okviru tega diplomskega dela smo razvijali v programskem jeziku C#, ter verziji 2008 razvojnega okolja Visual Studio.

5.1.3 Orodje PowerDesigner

PowerDesigner je vodilno orodje za podatkovno modeliranje in upravljanje podatkovne arhitekture [16]. Razvilo ga je podjetje Sybase, z namenom zagotavljanja lažjega načrtovanja podatkovne strukture ter prenos le-te v dejansko stanje. Podpira razvoj poslovnih procesnih modelov, konceptualnih podatkovni model ter samodejno pretvorbo v fizični podatkovni model, generiranje poročil, XML modeliranje itd. Z uporabo tega orodja, smo v fazi načrtovanja (poglavje 5.3.1) izdelali konceptualni, ter fizični podatkovni model, kar nam je zmanjšalo možnost nastajanja napak pri načrtovanju in posledično močno olajšalo ter pospešilo razvoj programske rešitve.

5.1.4 SQL strežnik Microsoft SQL Express Server

Microsoft SQL Express Server, je brezplačna različica Microsoft SQL Server relacijskega podatkovnega strežnika SQL [17]. Razvilo ga je podjetje Microsoft, namenjen pa je hranjenju in dostopu do podatkov (oddaljenem preko omrežja ali lokalnem), s strani podatkovno nezahtevne, oziroma ožje zastavljene programske opreme. Omejen je z uporabo ene fizične centralno procesne enote, oziroma procesorja, maksimalno velikostjo posamezne podatkovne baze 10 GB (omejitve glede števila podatkovnih baz ni) in uporabo največ 1 GB spomina (RAM).

5.1.5 Orodje Microsoft SQL Server Management Studio

Microsoft SQL Server Management Studio, je tipičen sistem za upravljanje s podatkovnimi bazami (angl. Database Management System, DBMS), ki uporabniku omogoča konfiguracijo, upravljanje in administracijo strežnikov Microsoft SQL Server [18]. Pri razvoju smo orodje uporabljali za kreiranje in upravljanje tabel in relacij podatkovne baze ter pregled podatkov med razvojem in testiranjem aplikacije.

5.1.6 Razvojna komponenta OPC DA .Net

OPC DA .Net je razvojna komponenta, razvita s strani podjetja Advosol in je namenjena vključitvi v razvojno okolje Visual Studio [11]. Zagotavlja potrebne razrede, kontrole in orodja, ki omogočajo hiter in enostaven razvoj OPC DA odjemalcev. V razvitih aplikacijah nam je omogočila iskanje dosegljivih strežnikov OPC DA, pregled njihove strukture, ter strukture priključenih krmilnih računalnikov. Z uporabo implementiranih metod smo lahko tako izvajali branje in pisanje potrebnih podatkov, na izbrane krmilne računalnike.

5.2 Analiza zahtev

Ključna aktivnost pri procesu razvoja programske opreme, je analiza ter definicija uporabniških zahtev. Pravilna in jasna definicija in analiza zahtev izniči možnost kasnejšega pojavljanja dodatnih, oziroma spremenjenih zahtev in težav pri implementaciji programske rešitve ter omogoči lažjo, predvsem pa tudi pravilno implementacijo aplikacije. V sklopu te aktivnosti je potrebno analizirati zahteve in potrebe naročnika, določiti prednostne oziroma prioritetne zahteve, upoštevati omejitve, ki vplivajo na razvoj aplikacije, ter določiti funkcionalnost aplikacije. Ker pri razvoju naše aplikacije naročnika ni bilo, oziroma ni definiran, smo pri analizi zahtev predvidevali možne potrebe, zahteve ter omejitve na podlagi potencialnega uporabnika aplikacije.

Zahtevana funkcionalnost

- Aplikacija mora omogočati hitro izvajanje in pravilno generiranje varnostnih kopij izbranih parametrov, s krmilnih računalnikov.
- Varnostno kopiranje se izvaja periodično, odvisno od nastavitve uporabnika preko uporabniškega vmesnika.
- Omogočeno je pregledovanje obstoječih varnostnih kopij, glede na posamezen krmilni računalnik.
- Uporabniku mora biti omogočena hitra in enostavna ponastavitev parametrov krmilnega računalnika, na stanje izbrane varnostne kopije.
- Uporabniku je omogočeno generiranje ter upravljanje parametrov za varnostno kopiranje.
- Varnostno kopiranje se mora izvajati neodvisno od izvajanja aplikacije, oziroma v obliki windows service aplikacije.
- Uporabniku mora biti omogočeno tudi pregledovanje varnostnih kopij, nastavitev urnika varnostnega kopiranja in ročni zagon kopiranja neodvisno od urnika kopiranja.
- Aplikacija mora omogočati pregled dogodkov nastalih med izvajanjem aplikacije, ter opozarjati na nastale napake in omejitve.
- Uporabniški vmesnik aplikacije mora biti uporabniku prijazen in intuitiven.

Omejitve sistema

- Varnostne kopije se strukturirano shranjujejo v podatkovni bazi SQL.
- Aplikacija naj deluje na platformi Windows.
- Komunikacija s krmilnimi računalniki naj poteka preko standarda OPC, oziroma že nameščenih strežnikov OPC in vmesnikov OPC DA.

Omejitve pri razvoju

- Čas za razvoj aplikacije je omejen na čas namenjen izdelavi aplikacije, v okviru izdelave diplomskega dela.
- Na razpolago so le krmilni računalniki proizvajalca Siemens.

5.3 Načrtovanje

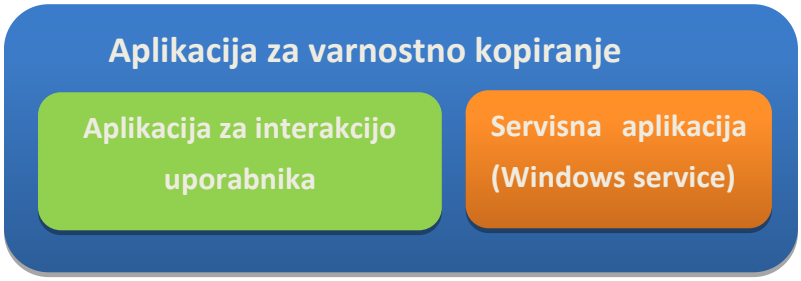
Poleg analize zahtev, oziroma potreb potencialnega uporabnika je pred začetkom izdelave aplikacije potrebno temeljito opraviti načrtovanje sistema. V tej fazi upoštevamo vse zahteve pridobljene v fazi analize in definiramo načrt, oziroma hierarhično strukturo sistem, ki ga bomo izdelali ter upoštevamo vse dejavnike okolja (proizvodnje), v katerem bo izdelani sistem deloval. Načrtovanje mora biti izvedeno temeljito in premišljeno, saj lahko napake v tej fazi pomenijo velike težave ali celo neuspešnost pri kasnejši realizaciji sistema in vodijo k redefiniciji celotnega načrtovanja ali dela le-tega in posledično do ponovitve faze izdelave. Pri načrtovanju sistema moramo tako poleg že znanih specifikacij, zagotoviti tudi zmožnost kasnejše razširitve sistema, enostavno vzdrževanje in uporabo ter zagotavljanje varnosti in zmogljivosti.

Kot rezultat uspešnega načrtovanja pridobimo učinkovit načrt arhitekture sistema, definicijo uporabniškega vmesnika, oziroma modulov namenjenih interakciji uporabnika s sistemom in arhitekturo podatkovne strukture (podatkovne baze), za shranjevanje in upravljanje s podatki sistema.

5.3.1 Načrtovanje arhitekture aplikacije

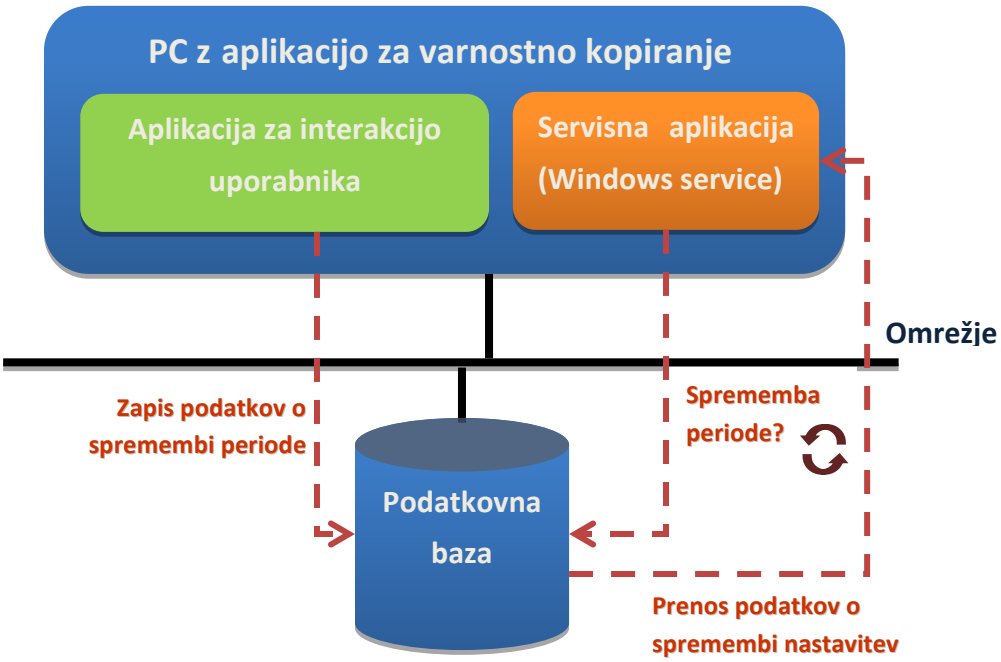
V fazi načrtovanja arhitekture sistema smo izdelali visokonivojski načrt aplikacije, določili njeno zgradbo ter način delovanja aplikacije. Pri načrtovanju arhitekture smo upoštevali vse želene funkcionalnosti aplikacije, predvidene omejitve sistema in omejitve pri razvoju, definirane v fazi analize zahtev.

Ena izmed zahtev definiranih v fazi analize določa, da mora izdelana aplikacija zagotavljati izvajanje periodičnega varnostnega kopiranja, neodvisno od uporabniškega vmesnika, oziroma mora delovati ločeno v obliki windows service aplikacije. Zato smo v fazi načrtovanja definirali aplikacijo, kot dva ločena dela, kot je prikazano na Slika 12.



Slika 12: Struktura aplikacije za varnostno kopiranje.

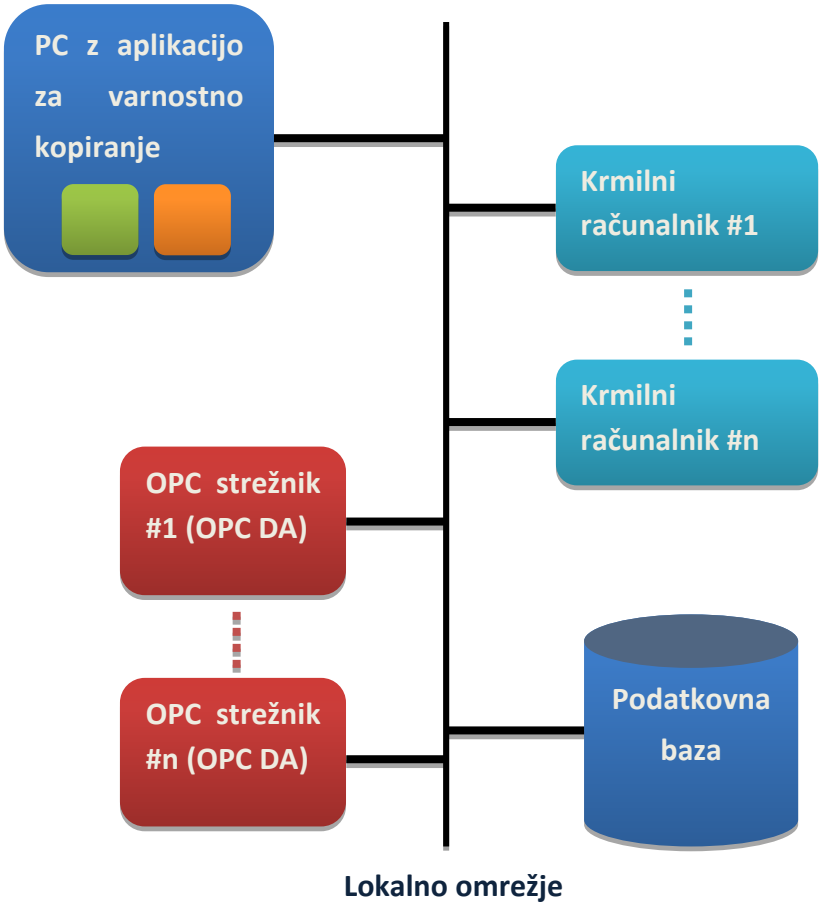
Prvi del predstavlja aplikacija za interakcijo uporabnika in je vizualni vmesnik med funkcionalnostmi aplikacije kot celote in uporabnikom. V sklopu te aplikacije so realizirani moduli, ki predstavljajo posamezno funkcionalnost aplikacije za varnostno kopiranje. Uporabnik lahko tako z izbiro ustreznega modula izvaja operacije, ki so bile v fazi analize zahtev definirane kot funkcionalne zahteve. Drugi del predstavlja servisna aplikacija, ki deluje v ozadju in skrbi za periodično izvajanje varnostnega kopiranja ter izvajanje nadzora delovanja izbranih krmilnih računalnikov (perioda se nastavi v prvem delu aplikacije). Ta aplikacija deluje neprekinjeno in ni odvisna od delovanja prvega dela aplikacije, oziroma aplikacije za interakcijo uporabnika. Prvi in drugi del aplikacije komunicirata preko podatkovne baze, preko katere aplikacija za interakcijo z uporabnikom posreduje servisni aplikaciji potrebne podatke, ki jih potrebuje za periodično izvajanje varnostnega kopiranja. Slika 13 prikazuje primer takšne komunikacije.



Slika 13: Primer komunikacije med deli aplikacije.

Ker oba dela, oziroma aplikaciji medsebojno komunicirata le preko podatkovne baze, se lahko aplikaciji izvajata na ločenih računalnikih, ki imata povezavo do skupne podatkovne baze. V tem primeru je potrebno zagotoviti način preverjanja odziva servisne aplikacije, na zapis nastavitvenih sprememb na podatkovno bazo, kar je implementirano v aplikaciji za interakcijo uporabnika.

V fazi načrtovanja smo za namen lažjega razumevanja in implementiranja zahtev izdelali primer arhitekture tipičnega delovnega okolja aplikacije za varnostno kopiranje, ki ga prikazuje Slika 14.



Slika 14: Predvideno delovno okolje aplikacije za varnostno kopiranje.

Prioritetna funkcionalnost aplikacije za varnostno kopiranje, je periodično izvajanje varnostnega kopiranja z izbranih krmilnih računalnikov. Kot smo že omenili v poglavju OPC, za komunikacijo med višje-nivojsko aplikacijo, ki je v našem primeru aplikacija za varnostno kopiranje in nižje-nivojskimi napravami, kot so krmilni računalniki, uporabljamo OPC strežnike. Aplikacija tako prepozna in omogoča dostop do krmilnih računalnikov preko strežnikov OPC z implementiranim vmesnikom OPC DA. Kot je definirano med funkcionalnimi zahtevami, aplikacija uporabniku omogoča iskanje krmilnih računalnikov v lokalnem omrežju in izbiro parametrov za varnostno kopiranje, iz naslovnega prostora izbranega krmilnega računalnika. Aplikacija podatke o izbranih parametrih shrani v podatkovno bazo, kamor se shranjujejo tudi podatki o napakah in dogodkih nastalih med delovanjem aplikacije, nastavitvah aplikacije in njenega delovanja, varnostno kopiranih podatkih ter drugi pomembni podatki, ki so nastali med izvajanjem ali so pomembni za delovanje aplikacije.

Pomembno funkcionalnost aplikacije predstavlja tudi ponastavitev parametrov krmilnega računalnika. To je nasprotna operacija varnostnemu kopiranju, s katero uporabnik sproži zapis

vrednosti varnostno kopiranih parametrov shranjenih v podatkovni bazi, nazaj na izbrani krmilni računalnik.

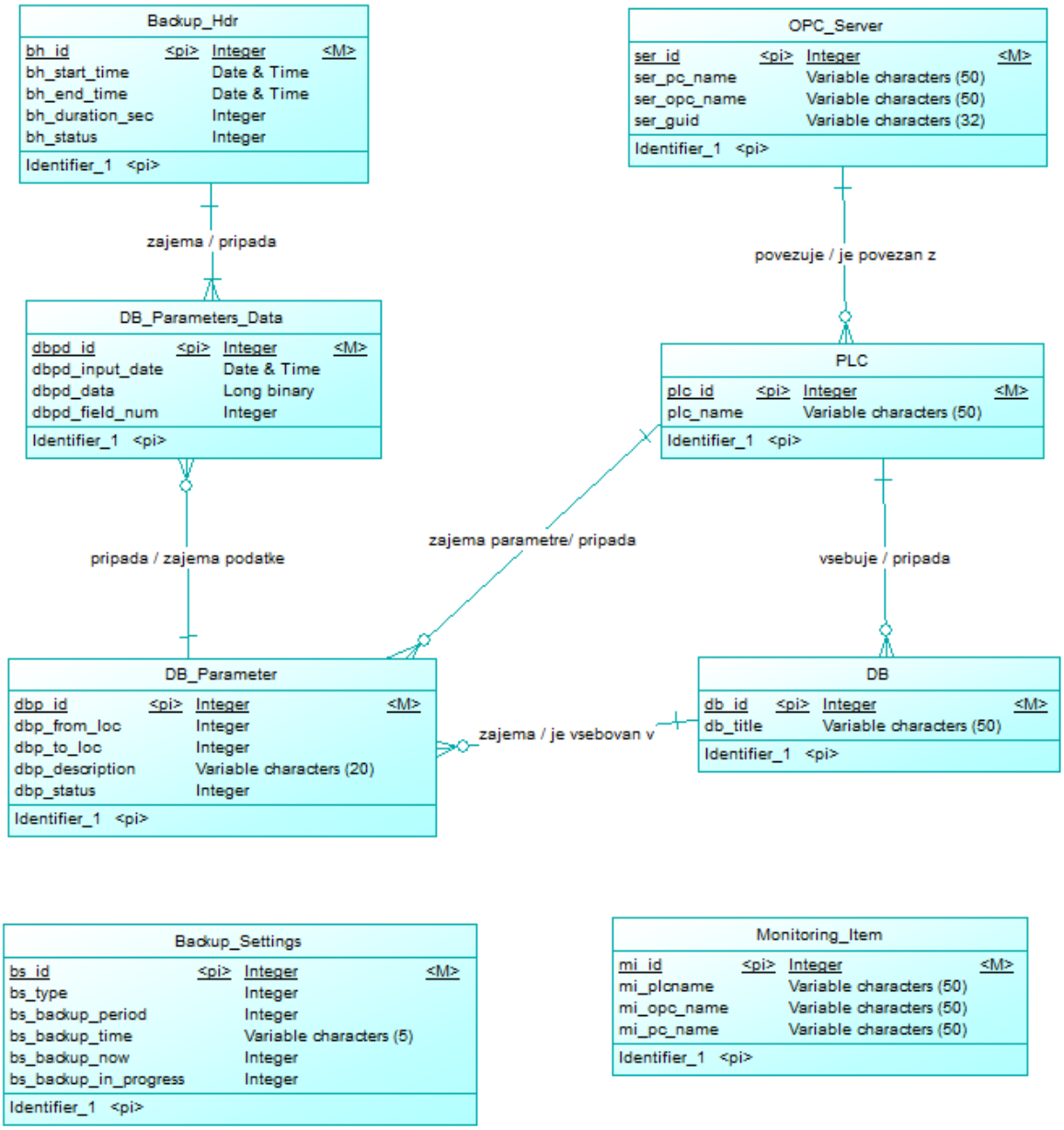
V fazi načrtovanja arhitekture aplikacije smo definirali tudi arhitekturo podatkovne strukture, oziroma podatkovne baze. Izdelali smo konceptualni ter na podlagi le-tega še fizični podatkovni model in s tem ponazorili koncept potrebne podatkovne baze ter pridobili dejansko obliko podatkovne baze, ki omogoča implementacijo vseh potrebnih funkcionalnosti aplikacije. Načrtovanje podatkovnega modela je pomemben del načrtovanja vsakega sistema, saj zmanjša možnost napačne definicije podatkovne strukture ter omogoča hitro ugotavljanje pomanjkljivosti, ter odpravo napak, ki imajo lahko v kasnejši fazi izdelave mnogo večje posledice. Za izdelavo konceptualnega in fizičnega podatkovnega modela smo uporabili orodje PowerDesigner, ki med drugim omogoča hitro in enostavno izdelavo konceptualnega modela ter avtomatsko pretvorbo v fizični podatkovni model.

5.3.1.1 Konceptualni podatkovni model

Z izdelavo konceptualnega podatkovnega modela pohitrimo in poenostavimo razvoj podatkovne baze ter prispevamo k boljšemu razumevanju celotnega sistema, ki ga želimo implementirati. Pri izdelavi konceptualnega modela poizkušamo sestaviti strukturo podatkovnih objektov in podatkov, ki z medsebojnimi povezavami predstavljajo celovit opis pomena podatkov, ki nastopajo v našem sistemu.

Konceptualni podatkovni model smo izdelali na podlagi miselno zastavljenega modela in ga opisali z uporabo entitet, atributov, relacij ter ključev. V modelu entitete predstavljajo posamezne objekte, ki ponazarjajo neko zaključeno celoto in jih lahko enolično definiramo oziroma opišemo. Posamezna entiteta je opisana z atributi, ki predstavljajo njene lastnosti. Atribut je lahko definiran tudi kot ključ, kar pomeni da enolično identificira posamezno instanco zapisa entitete. Za ponazoritev odnosa, oziroma povezave med entitetami pa uporabljamo relacije.

Slika 15 prikazuje konceptualni podatkovni model aplikacije.



Slika 15: Konceptualni podatkovni model.

Konceptualni podatkovni model aplikacije za varnostno kopiranje je sestavljen iz osmih entitet, ki s svojimi atributi ter medsebojnimi relacijami predstavljajo podatkovno strukturo aplikacije. Entiteta *Backup_Hdr* vsebuje splošne podatke o izdelanih varnostnih kopijah in sicer podatek o času začetka in konca izdelave varnostne kopije (*bh_start_time*, *bh_end_time*), podatek o trajanju varnostnega kopiranja v sekundah (*bh_duration_sec*) in podatek o statusu izdelane varnostne kopije (*bh_status*). Entiteta *DB_Parameters_Data* vsebuje podatke o vrednosti parametrov zajetih v sklopu izvajanja varnostnega kopiranja, ki pripadajo določenemu zapisu varnostne kopije. Zapis entitete *DB_Parameters_Data* poleg podatka o vrednosti parametra (*dbpd_data*) vsebuje tudi podatke o času zapisa (*dbpd_input_date*) in številu polj oziroma bajtov (*dbpd_field_num*), ki jih iz izbranega

podatkovnega bloka (DB) na krmilniku zajema podatek o vrednosti parametra. Vsak zapis entitete *DB_Parameters_Data* pa pripada natanko enemu zapisu entitete *DB_Parameter*, ki vsebuje splošne informacije o parametrih izbranih za varnostno kopiranje. Tak zapis vsebuje podatek o začetni in končni lokaciji parametra (*dbp_from_loc* in *dbp_to_loc*, ki predstavljajo območje množice bajtov v nekem podatkovnem bloku krmilnika), podatke o opisu parametra (*dbp_description*) in statusu (*dbp_status*), ki pove ali je parameter aktiven, oziroma bo dejaven pri naslednjem varnostnem kopiranju. Vsak zapis entitete *DB_Parameter* pripada določenemu zapisu entitete *PLC*, ki predstavlja krmilni računalnik in v ta namen vsebuje podatek o imenu krmilnega računalnika *plc_name*. Prav tako zapis entitete *DB_Parameter* pripada natanko enemu zapisu entitete *DB*, ki vsebuje podatek o nazivu podatkovnega bloka in tako predstavlja posamezen podatkovni blok, ki lahko zajema več parametrov in pripada natanko enemu zapisu krmilnega računalnika *PLC*. Vsak zapis o krmilnem računalniku v entiteti *PLC* pa pripada natanko enemu zapisu entitete *OPC_Server*, ki vsebuje podatke o strežniku OPC predstavljenem s podatkom o imenu računalnika na katerem je nameščen strežnik OPC (*ser_pc_name*), s podatkom o nazivu strežnika OPC (*ser_opc_name*) in podatkom o enoličnem identifikatorju vrste strežnika OPC (*ser_guid*).

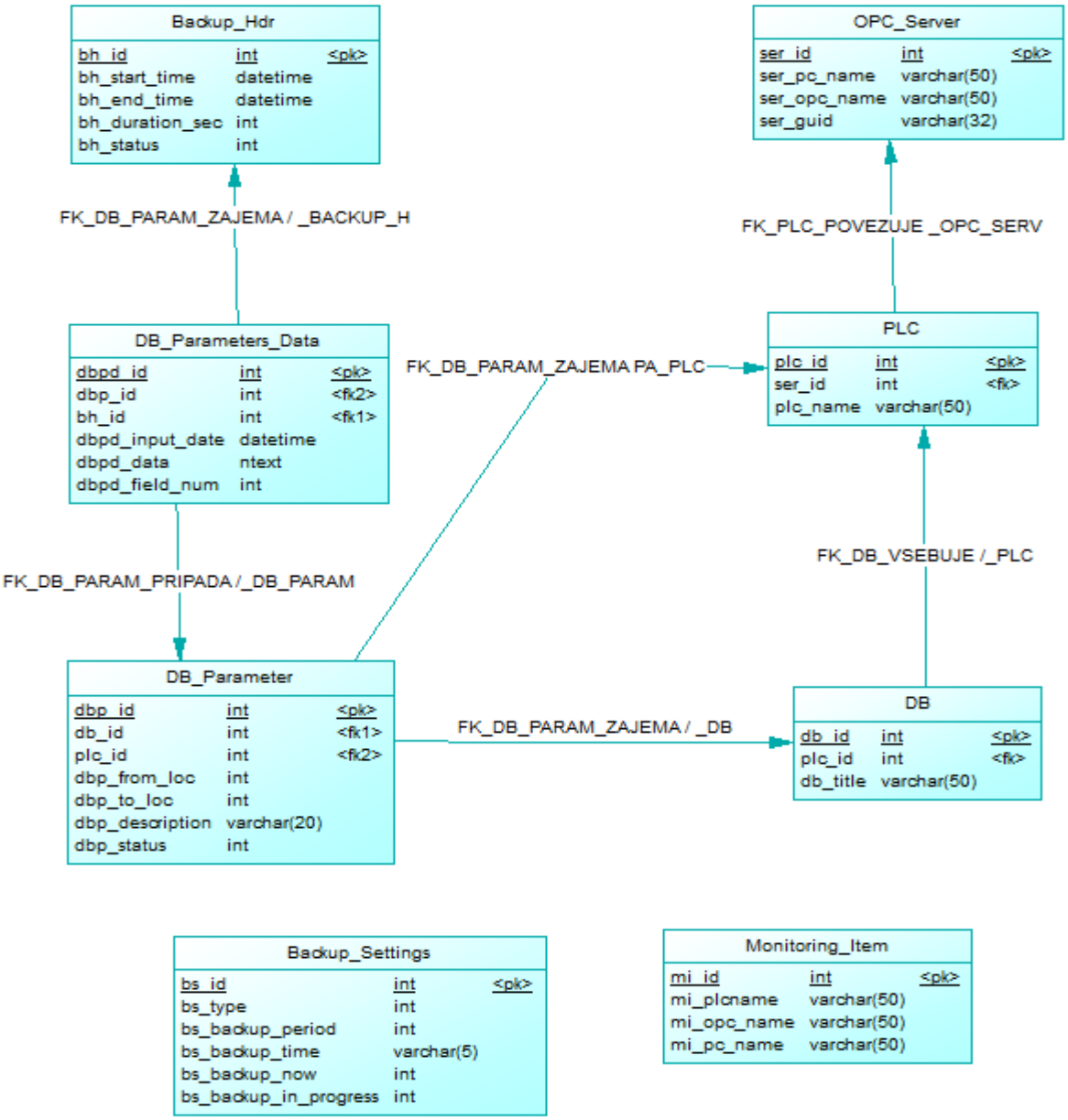
Entiteta *Monitoring_Item* služi hranjenju podatkov o krmilnih računalnikih, katerih delovanje se bo periodično preverjalo. V ta namen v tej entiteti hranimo podatke o nazivu krmilnega računalnika (*mi_plc_name*), nazivu strežnika OPC (*mi_plc_name*) in nazivu računalnika tega strežnika (*mi_pc_name*).

Za sporazumevanje, oziroma izmenjavo podatkov pa smo dodali entiteto *Backup_Settings*, katere prva vrstica bo uporabljena za izmenjavo podatkov o vrsti periode varnostnega kopiranja (*bs_type*), razponu periode izbrane izbrane vrste (*bs_backup_period*) in času varnostnega kopiranja (*bs_backup_time*). Vsebuje pa tudi podatek, ki servisni aplikaciji pove, da je izdana ročna zahteva za varnostno kopiranje (*bs_backup_now*) in podatek, ki med delovanjem služi kot indikator varnostnega kopiranja v teku (bs_backup_in_progress).

Vsaka entiteta poleg navedenih podatkov vsebuje tudi primarni ključ, ki enolično določa vsak zapis entitete.

5.3.1.2 Fizični podatkovni model

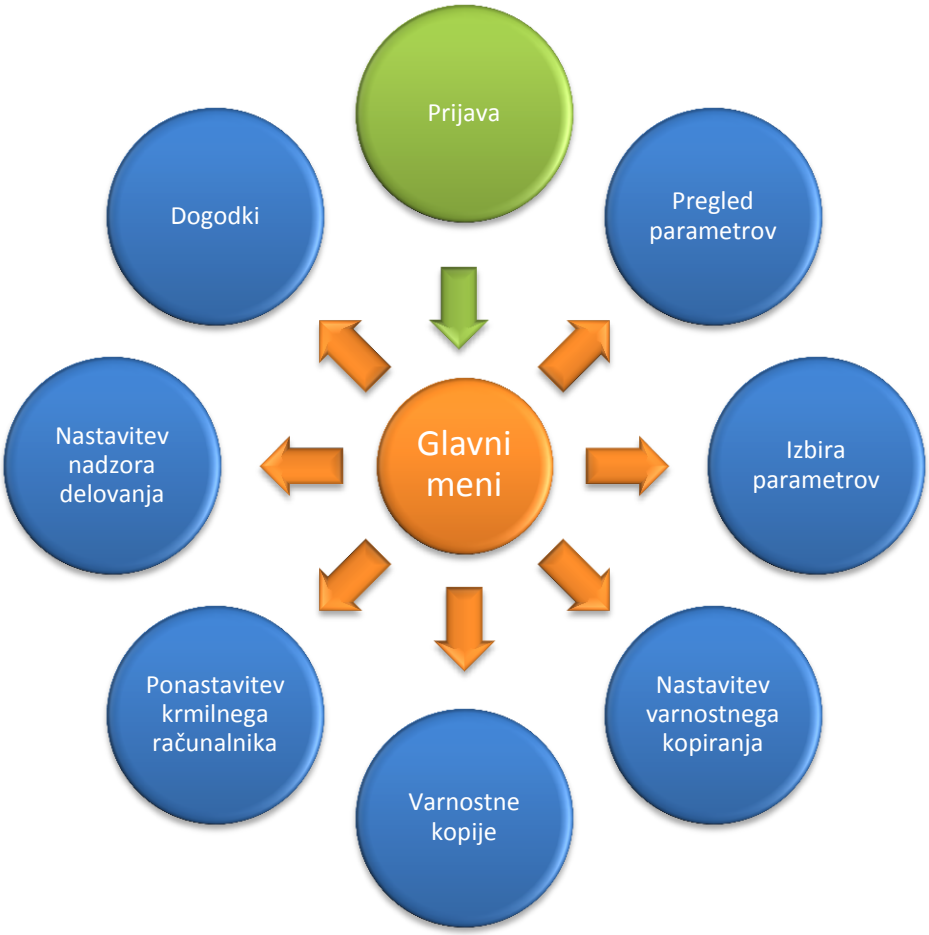
Fizični podatkovni model, v primerjavi s konceptualnim modelom predstavlja dejansko obliko podatkovne baze, v kateri bodo shranjeni podatki, s katerimi razpolaga aplikacija. Fizični model tako zajema vse tabele, stolpce, primarne in tuje ključe, indekse, podatkovne tipe, obveznost in omejitve podatkov, procedure ter ostale sestavne dele, ki so odvisni od izbranega sistema, s katerim bomo upravljali podatkovno bazo, oziroma SUPB (angl. Relational Database Management System - RDBMS). Z orodjem Power Designer, ki omogoča samodejno pretvorbo konceptualnega modela v fizični model, smo izdelali model fizične zgradbe podatkovne baze prikazan na Slika 16.



Slika 16: Fizični podatkovni model.

5.3.2 Načrtovanje modulov aplikacije

V tej fazi načrtovanja smo podrobneje definirali posamezne module, oziroma dele na katere bo razdeljena aplikacija namenjena interakciji z uporabnikom. Posamezen modul predstavlja del aplikacije, ki služi za izvajanje določene operacije, oziroma zajema določene funkcionalnosti. Slika 17 prikazuje modularno zgradbo aplikacije.



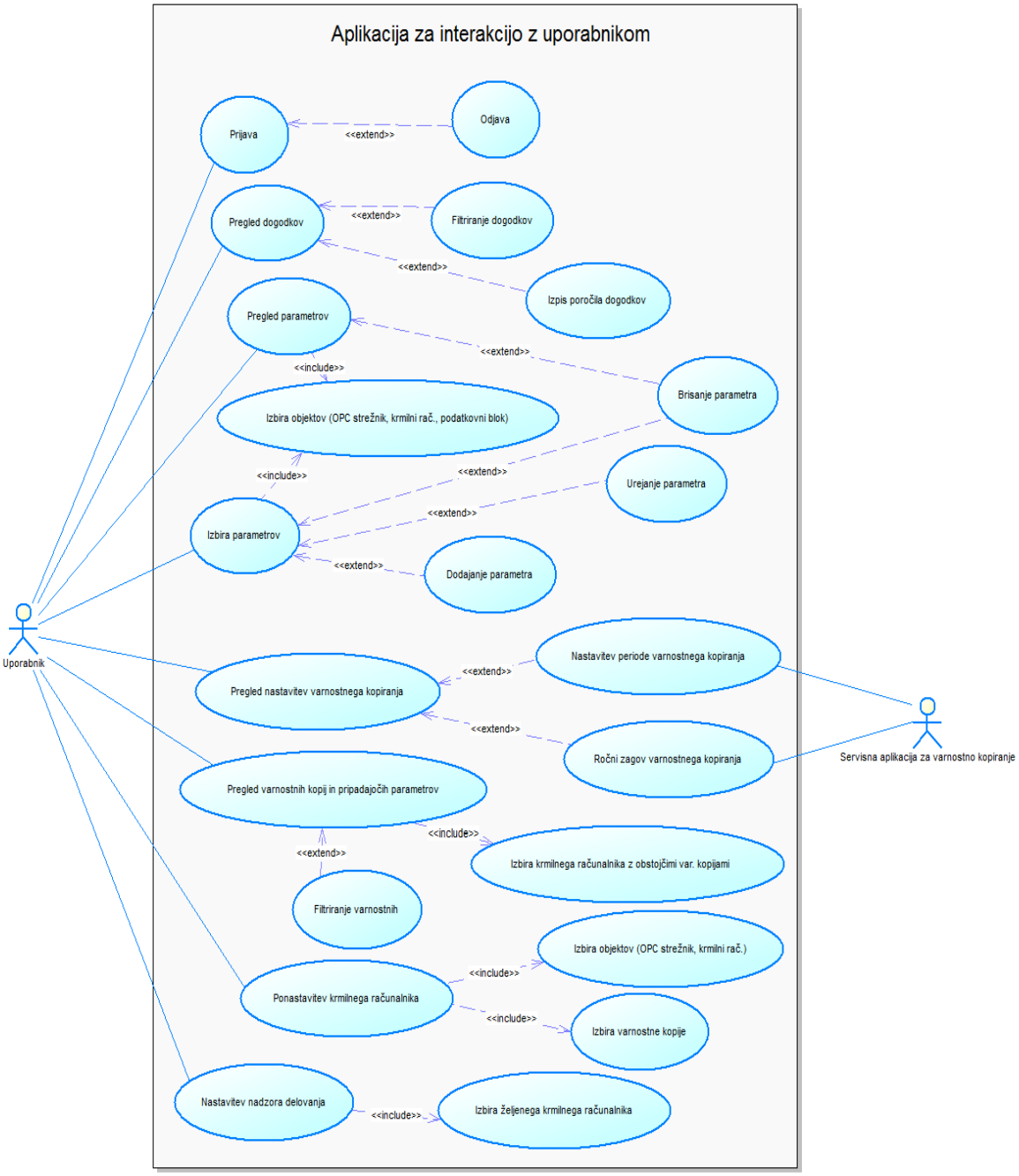
Slika 17: Modularna zgradba aplikacije.

Aplikacija za varnostno kopiranje, je v osnovi razdeljena na devet glavnih modulov, ki implementirajo eno ali več funkcionalnost določenih v fazi analize zahtev, oziroma imajo splošno-namensko funkcijo (prijava in glavni meni). Glavni moduli aplikacije so:

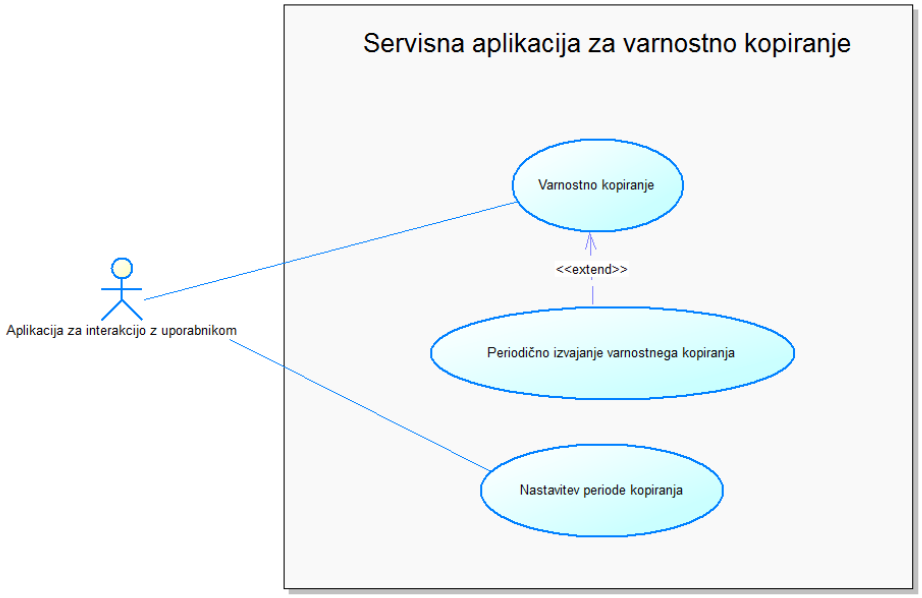
- **Prijava**
  Zahteva v naprej nastavljeno geslo uporabnika in je skupno za vse uporabnike aplikacije. Predstavlja edini način dostopa do glavnega menija aplikacije.
- **Glavni meni**
  Omogoča enostaven pregled in dostop do vseh funkcionalnosti aplikacije. Vsebuje tudi splošne nastavitve in informacije aplikacije.

- **Izbira parametrov**
  Omogoča enostavno iskanje in pregled vseh krmilnih računalnikov v lokalnem omrežju in OPC strežnikov ter računalnikov, ki omogočajo dostop do določenega krmilnega računalnika in njegove notranje strukture podatkovnih blokov. Uporabnik lahko preko tega dela aplikacije enostavno upravlja, ter izbira nove parametre za varnostno kopiranje s poljubnega krmilnega računalnika.
- **Pregled parametrov**
  Prikazuje hierarhično strukturo izbranih parametrov s krmilnih računalnikov.
- **Nastavitev varnostnega kopiranja**
  Uporabnik lahko v tem delu aplikacije nastavlja periodo samodejnega varnostnega kopiranja, ki ga izvaja servisna aplikacija. Dodana je tudi možnost ročnega zagona varnostnega kopiranja.
- **Varnostne kopije**
  V tem delu je uporabniku omogočen pregled vseh že izvedenih varnostnih kopij in kopiranih parametrov, glede na izbrani krmilni računalnik in časovno obdobje.
- **Ponastavitev krmilnega računalnika**
  Uporabnik v tem delu izbere krmilni računalnik in na podlagi ponujenih varnostnih kopij izbere eno, na katere stanje želi ponastaviti krmilni računalnik.
- **Nastavitev nadzora delovanja**
  Omogoča pregled dosegljivih krmilnih računalnikov in dodajanje le-teh na seznam preverjanja, ki določa delovanje katerih krmilnih računalnikov bo periodično preverjala servisna aplikacija.
- **Dogodki**
  Uporabnik lahko v tem delu pregleduje vse dogodke, napake in opozorila, ki so se zgodila v nekem izbranem časovnem intervalu delovanja aplikacije.

Na podlagi definiranih modulov aplikacije smo v orodju PowerDesigner izdelali diagrama primera uporabe, s katerima smo natančneje določili funkcionalnosti ponujene uporabniku in komunikacijo aplikacije za interakcijo z uporabnikom, ter servisne aplikacije za izvajanje periodičnega varnostnega kopiranja in kopiranja na zahtevo.



Slika 18: Diagram primera uporabe za del aplikacije za interakcijo z uporabnikom.



Slika 19: Diagram primera uporabe za servisni del aplikacije.

5.4 Razvoj

Po uspešno izvedeni analizi zahtev in načrtovanju sledi prenos pridobljenih rezultatov v dejansko razvito programsko rešitev. V našem primeru smo za realizacijo uporabili razvojno okolje Microsoft Visual Studio 2008 in višji programski jezik C#. Za potrebe shranjevanja podatkov in realizacije strukture podatkovne baze, kot smo jo določili v fazi načrtovanja (poglavje 5.3.1.1), pa smo uporabili Microsoft SQL Server Express Edition bazo podatkov. Za dostop do nižje-nivojskih naprav, v našem primeru krmilnih računalnikov, se uporabljajo strežniki OPC. V naši aplikacijski rešitvi smo se osredotočili na dostop, oziroma branje in pisanje podatkov na krmilni računalnik preko vmesnika OPC DA (več o vmesnikih OPC DA v poglavju 3.1.1.1), saj je le-ta danes še vedno najbolj razširjen. Za realizacijo povezave s strežniki OPC DA, smo zato uporabili temu namensko razvojno komponento OPC DA .Net podjetja Advosol, ki je namenjena razvoju aplikacij odjemalcev OPC DA tudi v programskem jeziku C#.

Ker je naša programska rešitev razdeljena na dve aplikaciji, smo v nadaljevanju za vsako od teh opisali glavne razvite funkcionalnosti.

5.4.1 Razvoj aplikacije za interakcijo z uporabnikom

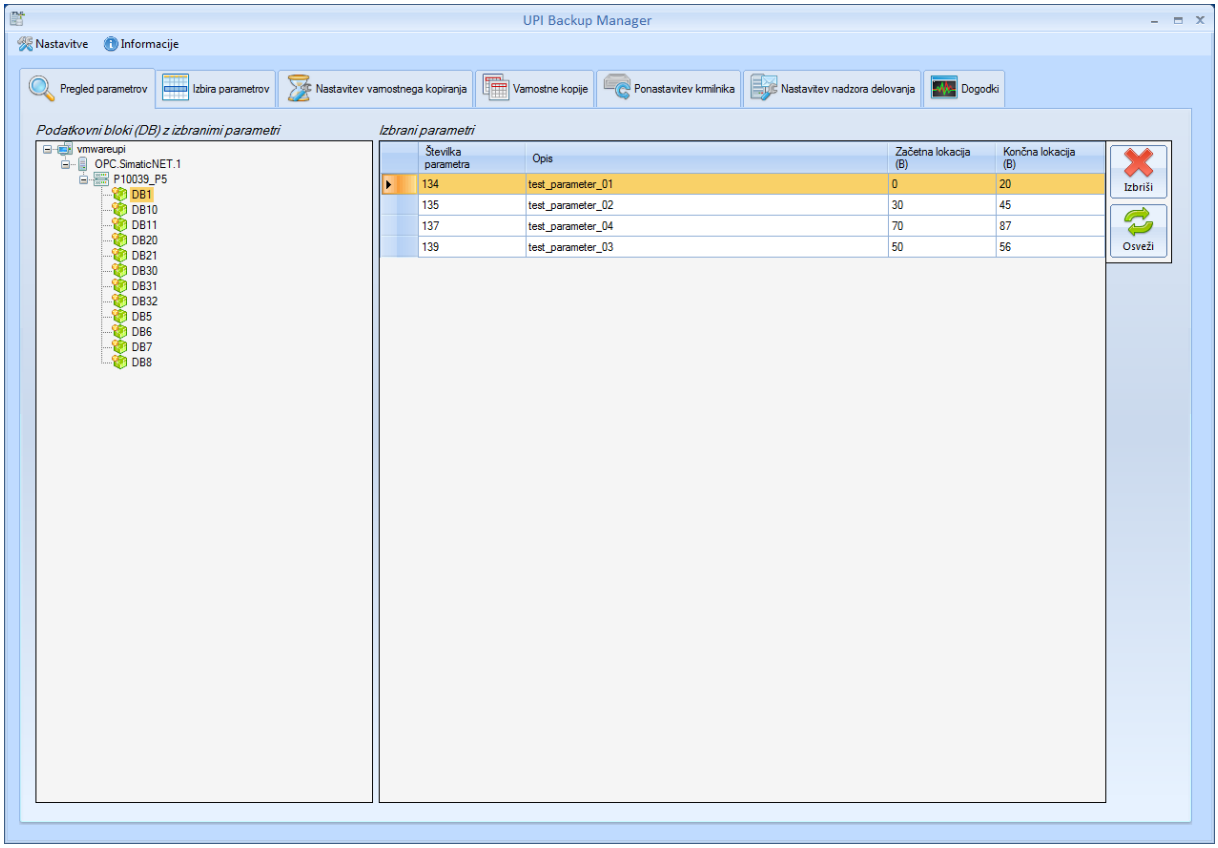
Kot smo že v fazi načrtovanja navedli, je aplikacija za interakcijo z uporabnikom namenjena uporabnikovem pregledu podatkov in upravljanju s funkcionalnostmi programske rešitve. Aplikacija je razdeljena na več delov ali modulov (glej poglavje 5.3.2), ki zajemajo oziroma implementirajo določeno funkcionalnost. Pri razvoju posameznega dela aplikacije smo stremeli k enostavnosti uporabe, zagotavljanju potrebnih funkcij, uporabniku prijaznem uporabniškem vmesniku in dobri vizualizaciji podatkov ter funkcij.

5.4.1.1 Pregled parametrov

Odločili smo se, da se bo po uspešni prijavi uporabnika v sistem le-temu najprej prikazala možnost pregleda parametrov. Ta modul omogoča uporabniku strukturiran pregled hierarhije podatkovnih blokov, ki vsebujejo parametre izbrane s strani uporabnika za namen varnostnega kopiranja. Uporabniški vmesnik tega dela aplikacije sestavlja prikaz drevesne strukture v obliki *ime računalnika* / *strežnik OPC* / *krmilni računalnik* / *podatkovni blok(DB)* in tabelarični prikaz izbranih parametrov za varnostno kopiranje, ki pripadajo določenemu podatkovnemu bloku DB. Za namen hitre odstranitve izbranega parametra s seznama za varnostno kopiranje in osvežitve prikaza, sta na voljo tudi gumba *Izbriši* in *Osveži*.

Drevesni prikaz strukture generiramo s klicem metode *showStructurePreview*, ki na podlagi podatkov zabeleženih v bazi podatkov izdela drevesno strukturo. Za prikaz drevesne strukture

smo uporabili komponento *TreeView*, kjer posamezne objekte strukture združujemo na podlagi relacije starš-otrok (angl. parent-child). Objektu razreda TreeView smo dodali tudi *AfterSelect* dogodek (angl. event), ki ob izbiri elementa drevesne strukture preveri izbrani element in v primeru, da je bil izbran podatkovni blok, iz baze podatkov prebere podatke o pripadajočih parametrih tega podatkovnega bloka in jih prikaže v tabelaričnem prikazu.



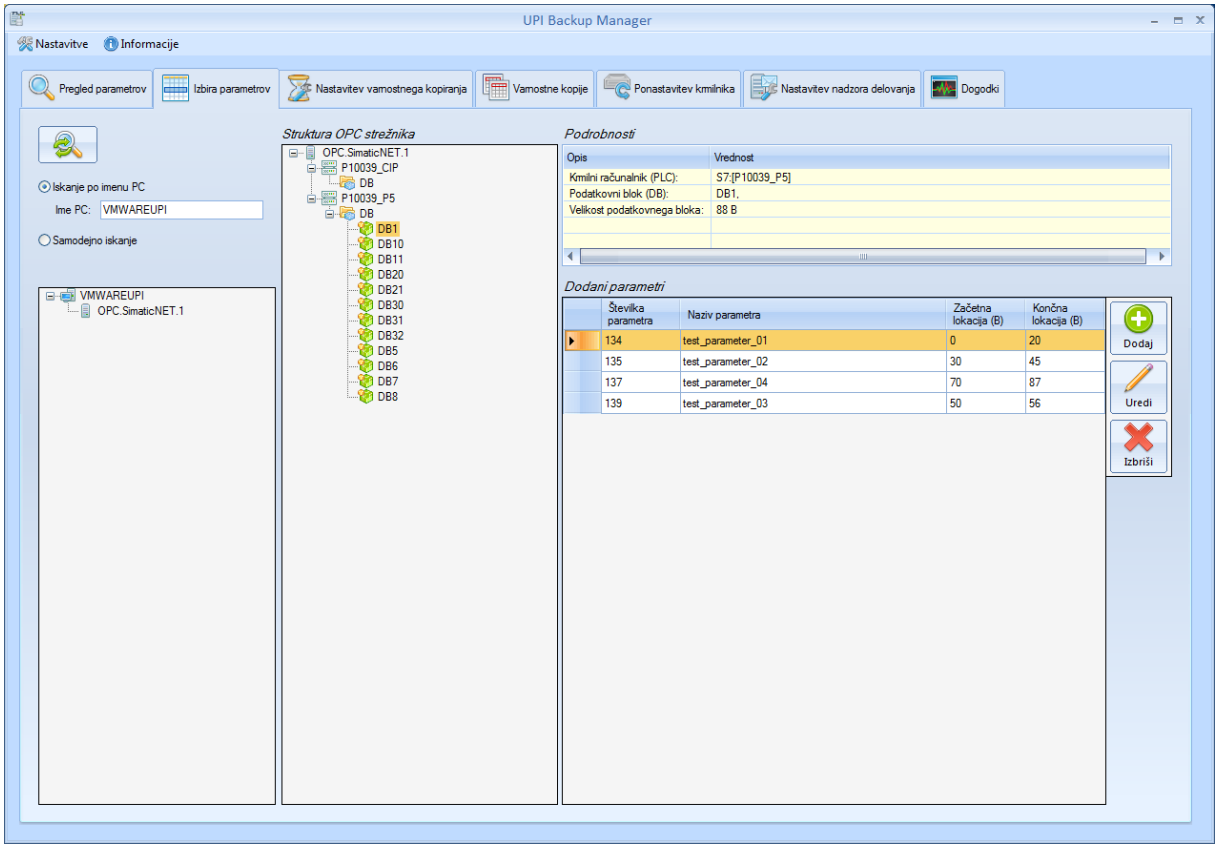
Slika 20: Zavihek za pregled parametrov.

Z uporabo gumba *Izbriši* lahko uporabnik hitro odstrani izbrani parameter iz tabelaričnega prikaza in posledično tudi seznama za varnostno kopiranje (podatki shranjeni v bazi podatkov v tabeli *DB_Parameter*). Ker pa bi z brisanjem zapisa parametra iz podatkovne baze onemogočilo vzpostavitev relacije z morebitnimi že obstoječimi vrednostmi varnostne kopije teh podatkov, pa smo v bazi podatkov oziroma tabeli *DB_Parameter* uvedli parameter *dbp_status*, ki določa vidnost in uporabljenost parametra pri sledečem varnostnem kopiranju (za več informacij glej poglavje 5.3.1.1). Pri kliku na gumb *Izbriši* tako ne izbrišemo parametra, ampak mu v tabeli *DB_Parameter* na bazi podatkov postavimo vrednost *dbp_status* na 0. Zaradi preprečevanja kopičenja neaktivnih parametrov pa ob koncu brisanja, oziroma spreminjanja statusa parametra preverimo ali je bil izbrani parameter že uporabljen pri kakšnem varnostnem kopiranju in ali ima pripadajočo vrednost v tabeli *DB_Parameters_Data*. V primeru, da parameter še nima nobene pripadajoče vrednosti ga

odstranimo. Iskanje in odstranjevanje neaktivnih parametrov izvedemo s klicem metode *findAndDeleteUnusedParameters*.

5.4.1.2 Izbira parametrov

V glavnem meniju je kot drugi na voljo zavihek izbira parametrov. Kot smo že opisali, je ta del aplikacije namenjen predvsem izbiri parametrov za varnostno kopiranje, ter njihovemu urejanju.



Slika 21: Zavihek za izbiro parametrov.

Uporabniški vmesnik tega dela aplikacije je sestavljen iz dela za iskanje in povezavo na izbrani strežnik OPC DA, dela za pregled strukture izbranega strežnika, tabele za pregled podrobnosti podatkovnega bloka, ter dela za pregled in upravljanje izbranih parametrov, ki pripadajo določenemu podatkovnemu bloku. Uporabnik preko iskalnega mehanizma (samodejno ali po izbranem imenu računalnika) zažene iskanje strežnikov OPC DA na lokalnem omrežju. V primeru ugodnih iskalnih rezultatov se v drevesni strukturi (komponenta *TreeView*) prikažejo najdeni strežniki OPC DA. Za iskanje strežnikov OPC DA uporabimo komponento OPC DA .Net, oziroma njen razred *OpcServerBrowser*. Preko metode *GetServerList*, ki jo kličemo preko objekta tega razreda pridobimo seznam imen vseh najdenih strežnikov OPC DA in pripadajočih GUID vrednosti, oziroma globalno enoličnih

identifikacijskih oznak programske opreme, ki jih uporabimo za prepoznavanje tipa strežnika OPC DA.

```
OpcServerBrowser opBr = new OpcServerBrowser(pcName);
opBr.GetServerList(out OPCServerList, out guId);
```

Ob dvojnem kliku na najdeni strežnik OPC DA ustvarimo objekt tipa *OpcServer*, katerega metoda *Connect* omogoča vzpostavitev povezave z izbranim strežnikom OPC DA, s posredovanjem njegovega imena ter imena računalnika.

```
OpcSrv = new OpcServer();
int connRes = OpcSrv.Connect(compName, opcServerName);
```

Po uspešni povezavi na izbrani strežnik izpišemo njegovo strukturo, oziroma vsebino (formatirano na želeno obliko) v obliki drevesne strukture. Za prikaz drevesne strukture ponovno uporabimo komponento *TreeView*, ki jo napolnimo s klicem metode *ShowBrowseTreeList* preko objekta tipa *OpcServer*.

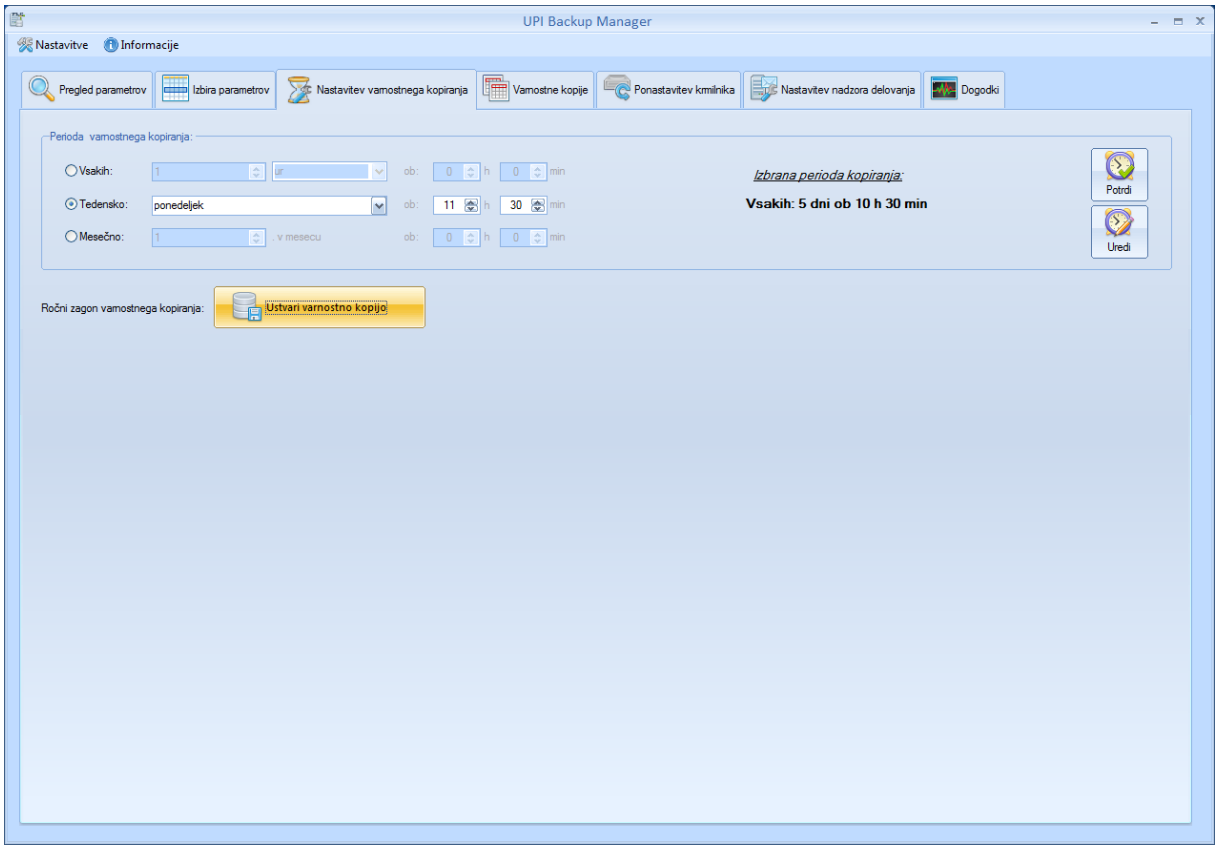
```
TreeView realTree = new TreeView();
ShowBrowseTreeList sBrowse =
OpcSrv.ShowBrowseTreeList(realTree,new ListView());
```

Ko uporabnik v tako pridobljeni strukturi izbere podatkovni blok iz vozlišča DB, se v tabelaričnem prikazu v desnem zgornjem kotu aplikacije prikažejo osnovni podatki tega podatkovnega bloka. Prav tako se v tabeli nižje prikažejo vsi parametri (že dodani) izbrani za varnostno kopiranje, ki jih na podlagi izbranega strežnika, krmilnega računalnika in podatkovnega bloka preberemo iz baze podatkov. Z gumbom *Dodaj* lahko uporabnik doda nov parameter tako, da v temu namenskem oknu, ki se odpre po kliku na gumb, vnese naziv parametra in začetno ter končno lokacijo, ki predstavlja blok bajtov, ki bodo varnostno kopirani v okviru parametra. Gumb za urejanje parametrov prav tako odpre to okno s podatki izbranega parametra in omogoča spreminjanje in potrditev spremembe. Odvečne, oziroma neželene parametre pa lahko uporabnik odstrani z uporabo gumba Izbriši, ki izbrani parameter odstrani s seznama parametrov za varnostno kopiranje. Mehanizem brisanja parametrov je enak mehanizmu opisanemu na koncu poglavja 5.4.1.1.

5.4.1.3 Nastavitev varnostnega kopiranja

V tem delu aplikacije je uporabniku omogočeno enostavno nastavljanje periode varnostnega kopiranja in ročni zagon le-tega. Periodo kopiranja je mogoče nastaviti na urno izvajanje, od ene do oseminštirideset ur. Prav tako je na voljo dnevno in tedensko izvajanje kopiranja ob poljubno izbrani uri, ter mesečno izvajanje, kjer lahko določimo kateri dan v mesecu in ob kateri uri se bo izvedlo varnostno kopiranje. Velikokrat se izkaže, da ne želimo čakati na iztek

periode, ampak želimo takoj izdelati varnostno kopijo izbranih parametrov. V ta namen smo v tem delu aplikacije dodali tudi gumb za takojšnji ročni zagon varnostnega kopiranja.



Slika 22: Zavihek za nastavitev varnostnega kopiranja.

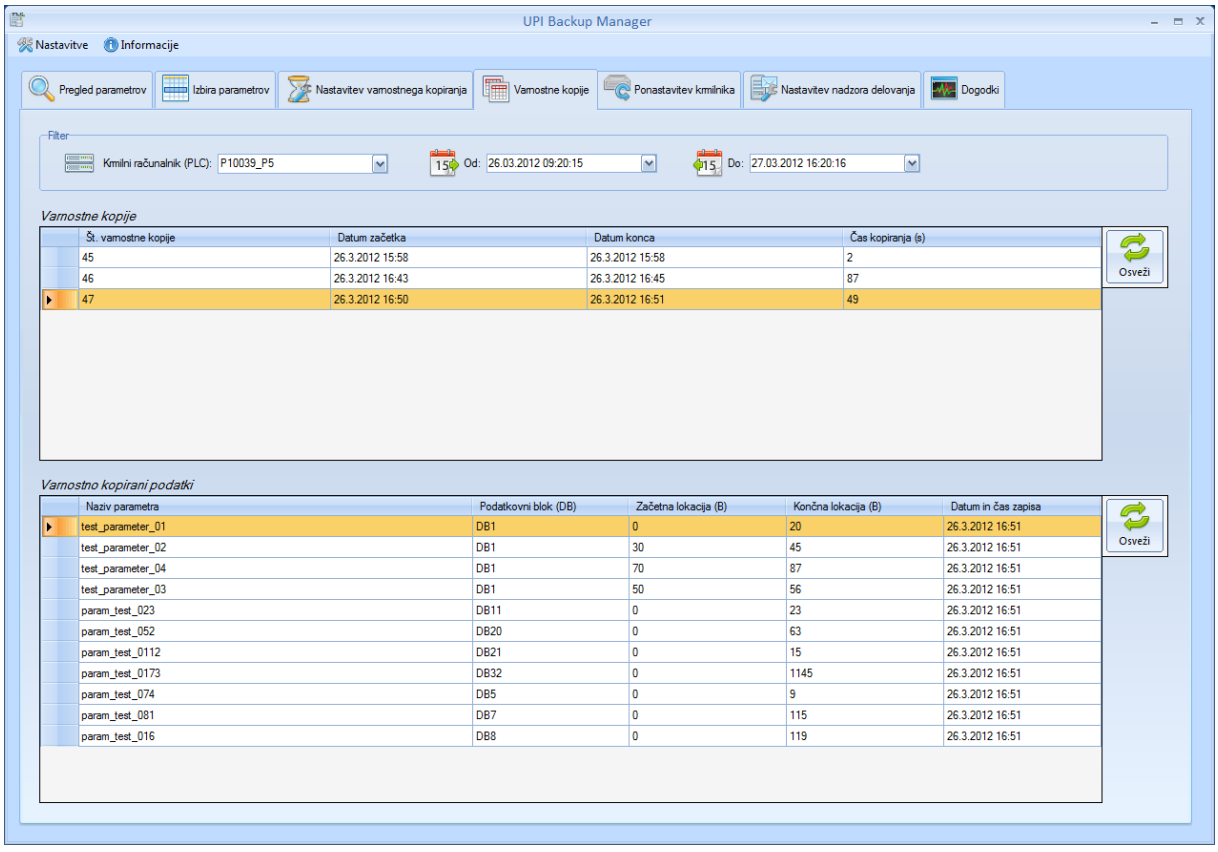
Tako nastavljanje periode varnostnega kopiranja, kot funkcija ročnega zagona, spremenjene podatke, oziroma zahteve posredujeta drugemu delu programske rešitve-servisni aplikaciji. Servisna aplikacija na podlagi zahtevane periode nastavi časovnike v teku ali na podlagi zahteve za takojšnje varnostno kopiranje le-to tudi izvede. Omenjene podatke o izbrani periodi kopiranja aplikacija v tem delu zapisuje v tabelo na bazi podatkov, imenovano *Backup_Settings*, kjer je rezervirana prva vrstica, ki služi za izmenjavo podatkov, oziroma sporazumevanje med obema deloma razvitega sistema za varnostno kopiranje. V primeru ročnega zagona varnostnega kopiranja v tem delu aplikacije, zapišemo vrednost 1 v stolpec *bs_backup_now*. S tem servisni aplikaciji, ki spremlja vrednost tega stolpca sporočimo, da želimo izvesti takojšnje varnostno kopiranje. Servisna aplikacija tako prebere ta podatke, izvede varnostno kopiranje, ter pred tem v stolpec *bs_backup_in_progress* zapiše vrednost 1. Postavljanje te vrednosti nakazuje na trenutno izvajanje varnostnega kopiranja ter predstavlja blokado pred ponovnim poskusom izvedbe te operacije. V primeru, da servisna aplikacija ne deluje, smo v tem delu uvedli mehanizem, ki po nekaj sekundah preveri podatek stolpca *bs_backup_now* in v primeru nespremenjene vrednosti (če je vrednost polja različna 0) uporabniku javi sporočilo o verjetnem nedelovanju servisne aplikacije.

Za nastavljanje periode varnostnega kopiranja, pa uporabljamo tudi nekatere ostale stolpce tabele *Backup_Settings*, katerih pomen je podrobneje razložen na koncu poglavja 5.3.1.1.

5.4.1.4 Varnostne kopije

Ta del aplikacije služi za pregled in iskanje obstoječih varnostnih kopij, ter kopiranih parametrov. Izpis varnostnih kopij omejimo z uporabo filtra, ki omogoča filtriranje varnostnih kopij glede na izbrani krmilni računalnik (na voljo za izbiro so le krmilni računalniki z obstoječimi varnostnimi kopijami vsebine) ter časovnega razpona v katerem je bila ustvarjena varnostna kopija. Podatke, s katerimi napolnimo vsebino kombinirane izvlečne liste krmilnih računalnikov, preberemo z baze podatkov s klicem metode *fillCmbPlcPrev*.

Uporabniški vmesnik je poleg filtra sestavljen še iz tabele za prikaz zapisov varnostnih kopij in tabele za prikaz podatkov o parametrih, kopiranih v okviru izbrane varnostne kopije iz prve tabele. Vsebina tabel se avtomatično generira, oziroma prebere z baze podatkov, glede na izbrani naziv krmilnega računalnika v predelu filtra.

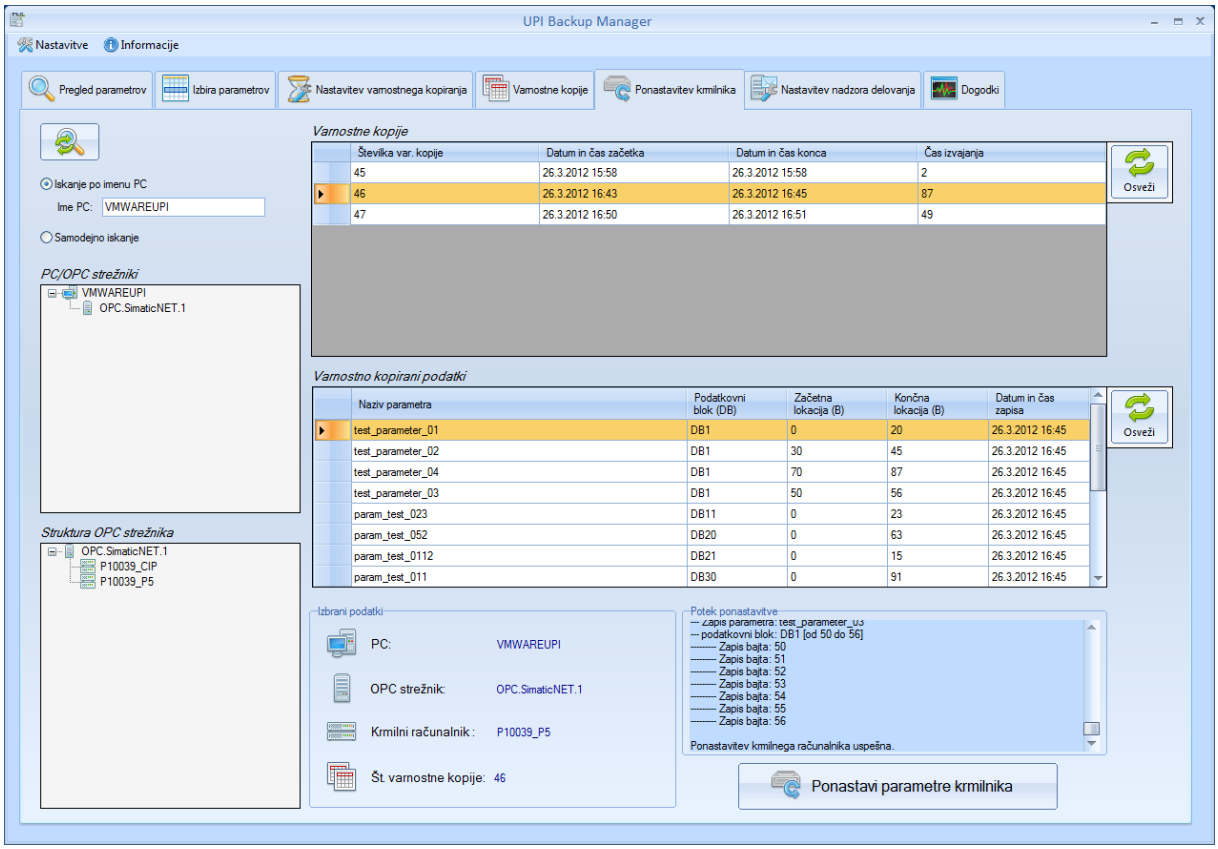


Slika 23: Zavihek za pregled varnostnih kopij.

5.4.1.5 Ponastavitev krmilnega računalnika

Vsaka dobra aplikacija za varnostno kopiranje podatkov ima omogočeno tudi ponastavitev, oziroma restavriranje varnostno kopiranih podatkov. To funkcionalnost smo v našem primeru dodali aplikaciji za interakcijo z uporabnikom in tako onemogočili možnost oddaljene ponastavitve podatkov krmilnega računalnika, ki bi jo uporabnik imel v primeru implementacije v servisnem delu. V tem delu aplikacije je tako potrebno, podobno kot pri delu aplikacije za izbiro parametrov, z iskalnim mehanizmom poiskati, ter izbrati želeni krmilni računalnik. Prav tako je omogočeno avtomatično iskanje strežnikov OPC in po imenu računalnika. Bistvena razlika iskalnika, je le v prikazu rezultatov iskanja, saj so prikazani le tisti strežniki OPC, kateri omogočajo povezavo do krmilnih računalnikov, katerih zapisi varnostnih kopij so najdeni na bazi podatkov. Preverjanje obstoja varnostnih kopij, ki se navezujejo na posamezen strežnik OPC, preverimo v metodi *existsBackupData.*

Po izbiri krmilnega računalnika z drevesne strukture, se podobno kot pri zavihku za pregled varnostnih kopij v tabeli prikažejo obstoječe varnostne kopije za izbrani krmilni računalnik in v sosednji tabeli še podatki o kopiranih parametrih, v okviru posamezne varnostne kopije. Za lažje sledenje in pregled izbranih elementov, smo dodali skupno polje z naslovom Izbrani podatki, ki prikazuje trenutno izbrani računalnik, strežnik OPC, krmilni računalnik in pripadajočo varnostno kopijo.



Slika 24: Zavihek za ponastavitev krmilnega računalnika.

Po izbiri vseh elementov se uporabniku omogoči izbira gumba za zagon ponastavitve izbranega krmilnega računalnika s podatki parametrov izbrane varnostne kopije. Ob zagonu ponastavitve, najprej z baze podatkov preberemo podatke vseh parametrov izbrane varnostne kopije, na katero ponastavljamo krmilni računalnik. Po uspešnem branju podatkov preverimo vrednost stolpca *bs_backup_in_progress* v tabeli *Backup_Settings* na podatkovni bazi, ki služi kot indikator varnostnega kopiranja v teku. V primeru varnostnega kopiranja v teku ustavimo postopek ponastavitve in obvestimo uporabnika. Tako se izognemo ogrožanju integritete novonastale varnostne kopije. V nadaljevanju postopka ponastavitve, prebrane parametre varnostne kopije zapišemo na krmilni računalnik. Pri zapisu parametrov se v zanki sprehodimo skozi vse prebrane parametre, katerih podatke zapišemo na način, kot ga prikazuje sledeča izvorna koda.

```
// povezava na OPC DA strežnik
OpcServer OpcSrv = new OpcServer();
int res = OpcSrv.Connect(pcName, opcSrvName);

if (res == 0)
{
    for (int i = fromB; i <= toB; i++)
    {
      SyncIOGroup srwGroup = new SyncIOGroup(OpcSrv);
```

```
            addCurrentOperationTxt("--------- Zapis bajta: " + i);
            int rtc = srwGroup.Write("S7:[" + plcName + "]" + dbTitle
            + "BYTE" + i, dataBytes[i - fromB]);

             if (HRESULTS.Failed(rtc)) //preverimo uspešnost pisanja
             {
                 addCurrentOperationTxt("--------- Napaka pri
                 zapisovanju na krmilni računalnik (PLC).");

                 MessageBox.Show("Napaka pri zapisovanju na krmilni
                 računalnik (PLC).\nPreverite povezavo na krmilni
                 računalnik (PLC).", "Napaka",
                 MessageBoxButtons.OK, MessageBoxIcon.Error);

                 DB.WriteToSqlLog(3, "Napaka pri zapisovanju na
                 krmilni računalnik (PLC).");
                 errorReported = true;
                 break;
            }
            srwGroup.Dispose();
            OpcSrv.Disconnect(); // prekinemo povezavo z OPC DA

}
```

Na začetku zapisovanja podatkov na krmilni računalnik, najprej izdelamo nov objekt tipa *OpcServer* (razred *OpcServer* pripada razvojni komponenti OPC DA .Net) in vzpostavimo povezavo s strežnikom OPC DA. To storimo s klicem metode *Connect,* ki ji podamo ime računalnika in strežnika OPC, na katerega se povezujemo. Sledi zapisovanje posameznega bajta podatkov, ki ga izvršimo preko objekta tipa *SyncIOGroup*, ki omogoča sinhrono zapisovanje podatkov na krmilni računalnik (več informacij o takšnem načinu dostopa do podatkov, je podanih v poglavju 3.1.1.1). Podatke zapišemo s klicem metode *Write,* ki ji podamo podatke o predmetu, ki ga zapisujemo (naziv krmilnega računalnika, naziv podatkovnega bloka, lokacija) in vrednost.

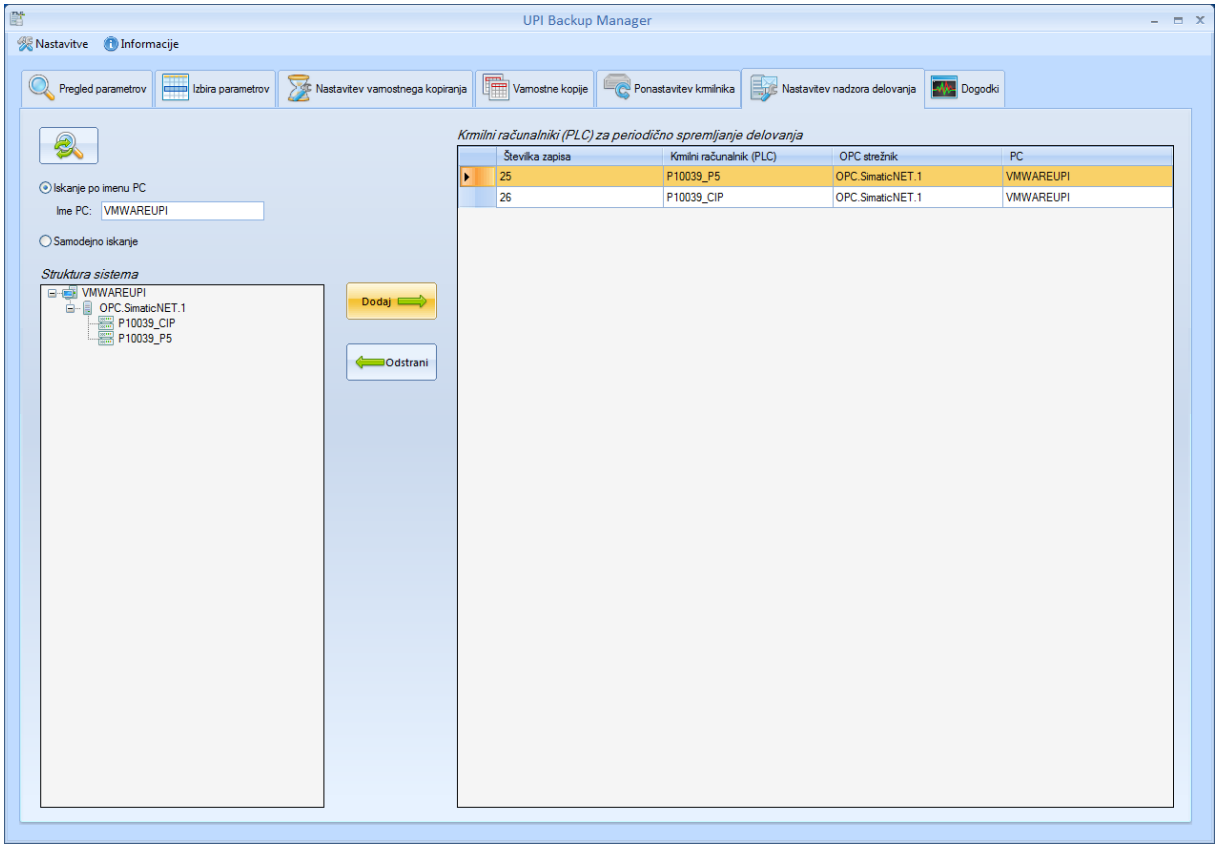
Posamezne dogodke znotraj ponastavitve beležimo, oziroma izpisujemo v skupno polje *Potek ponastavitve* s klicem metode *addCurrentOperationTxt.*

5.4.1.6 Nastavitev nadzora delovanja

Servisni del našega sistema za varnostno kopiranje, periodično in na zahtevo izvaja varnostno kopiranje. Zaradi možnosti večje časovne razlike, med posameznim izvajanjem kopiranja, smo v servisnemu delu sistema dodali še funkcionalnost periodičnega preverjanja delovanja strežnikov OPC DA in krmilnih računalnikov. Prednastavljena perioda preverjanja delovanja je trideset minut. V aplikacijo za interakcijo z uporabnikom, smo dodali opcijo nastavitve nadzora delovanja, ki omogoča pregled in iskanje dosegljivih krmilnih računalnikov in

dodajanje le-teh na seznam preverjanja, ki določa delovanje katerih krmilnih računalnikov bo periodično preverjala servisna aplikacija. Iskalni mehanizem je zgrajen podobno, kot pri delu za izbiro parametrov in ponastavitev krmilnih računalnikov, le da so v skupni drevesni strukturi prikaz vsi dosegljivi strežniki OPC in krmilni računalniki. Prikaz podrobnejše strukture krmilnih računalnikov v tem delu ni potrebna.

Ob izbiri krmilnega računalnika iz drevesnega prikaza ima uporabnik ponujeno možnost izbire gumba dodaj, ki doda podatke izbranega krmilnega računalnika v sosednjo tabelo, ki prikazuje podatke vseh že izbranih krmilnih računalnikov. V primeru izbire podatkov krmilnega računalnika iz tabelaričnega prikaza, pa je uporabniku omogočena izbira gumba za odstranitev zapisa iz seznama.



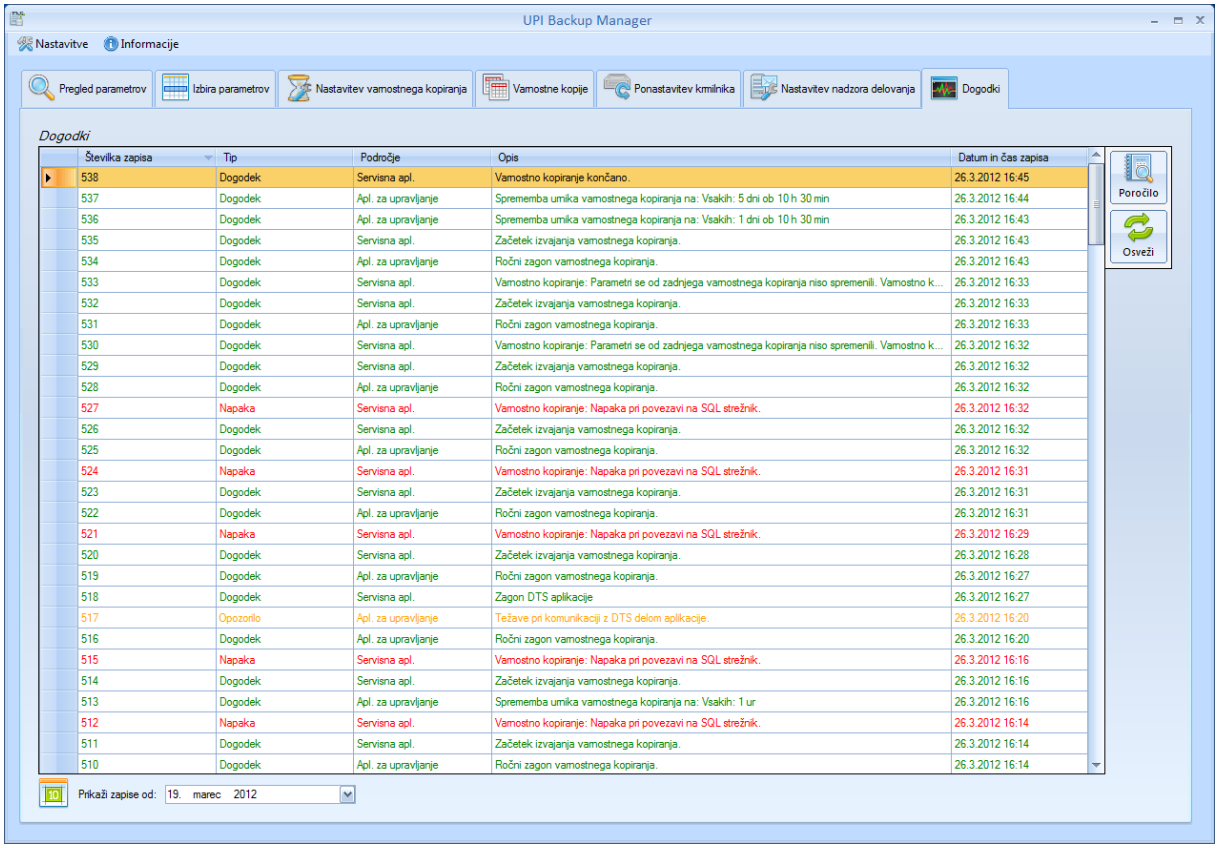
Slika 25: Zavihek za nastavitev nadzora delovanja krmilnih računalnikov.

5.4.1.7 Dogodki

V tem delu aplikacije smo uporabniku omogočili pregled vseh vrst dogodkov, ki so se zgodili med delovanjem sistema za varnostno kopiranje. Zapise vseh dogodkov prikazujemo v tabeli, ki med drugim za vsak dogodek vsebuje podatek o področju (servisna aplikacija, aplikacija za interakcijo z uporabnikom), tipu (dogodek, napaka, opozorilo), času zapisa in opisu dogodka. Vse podatke v tem delu preberemo iz tabele *Log* na bazi podatkov, kamor servisna aplikacija

in aplikacija za interakcijo z uporabnikom zapisujeta dogodke med delovanjem. Dogodke preko celotnega sistema zapisujemo s klicem metode *WriteToSqlLog*, ki pripada razredu *DB* (statični razred, ki vsebuje lastno razvite metode za izmenjavo podatkov z bazo podatkov).

Zapise prikazane v tabeli pa lahko uporabnik tudi filtrira, glede na datum pojavitve dogodka. Izbrani filter se prav tako upošteva pri generiranju poročila, ki ga generiramo z uporabo komponente *Crystal Reports*.



Številka zapisa	Tip	Področje	Opis	Datum in čas zapisa
538	Dogodek	Servisna apl.	Varnostno kopiranje končano.	26.3.2012 16:45
537	Dogodek	Apl. za upravljanje	Sprememba urnika varnostnega kopiranja na: Vsakih: 5 dni ob 10 h 30 min	26.3.2012 16:44
536	Dogodek	Apl. za upravljanje	Sprememba urnika varnostnega kopiranja na: Vsakih: 1 dni ob 10 h 30 min	26.3.2012 16:43
535	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:43
534	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:43
533	Dogodek	Servisna apl.	Varnostno kopiranje: Parametri se od zadnjega varnostnega kopiranja niso spremenili. Varnostno k...	26.3.2012 16:33
532	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:33
531	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:33
530	Dogodek	Servisna apl.	Varnostno kopiranje: Parametri se od zadnjega varnostnega kopiranja niso spremenili. Varnostno k...	26.3.2012 16:32
529	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:32
528	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:32
527	Napaka	Servisna apl.	Varnostno kopiranje: Napaka pri povezavi na SQL strežnik.	26.3.2012 16:32
526	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:32
525	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:32
524	Napaka	Servisna apl.	Varnostno kopiranje: Napaka pri povezavi na SQL strežnik.	26.3.2012 16:31
523	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:31
522	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:31
521	Napaka	Servisna apl.	Varnostno kopiranje: Napaka pri povezavi na SQL strežnik.	26.3.2012 16:29
520	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:28
519	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:27
518	Dogodek	Servisna apl.	Zagon DTS aplikacije	26.3.2012 16:27
517	Opozorilo	Apl. za upravljanje	Težave pri komunikaciji z DTS delom aplikacije.	26.3.2012 16:20
516	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:20
515	Napaka	Servisna apl.	Varnostno kopiranje: Napaka pri povezavi na SQL strežnik.	26.3.2012 16:16
514	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:16
513	Dogodek	Apl. za upravljanje	Sprememba urnika varnostnega kopiranja na: Vsakih: 1 ur	26.3.2012 16:16
512	Napaka	Servisna apl.	Varnostno kopiranje: Napaka pri povezavi na SQL strežnik.	26.3.2012 16:14
511	Dogodek	Servisna apl.	Začetek izvajanja varnostnega kopiranja.	26.3.2012 16:14
510	Dogodek	Apl. za upravljanje	Ročni zagon varnostnega kopiranja.	26.3.2012 16:14

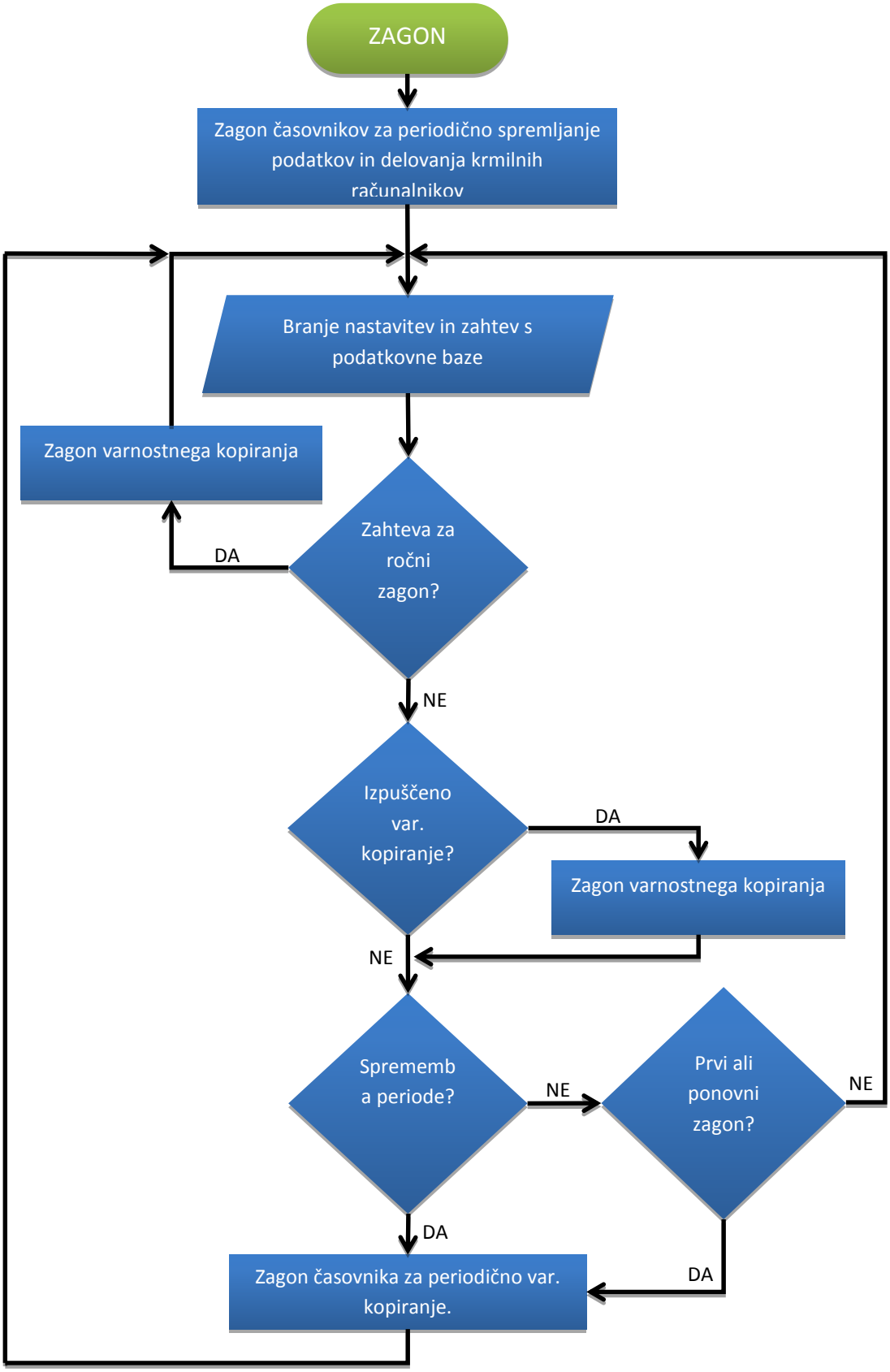
Slika 26: Zavihek za pregled dogodkov.

5.4.2 Razvoj servisne aplikacije

Drugi del programske rešitve predstavlja servisna aplikacija, ki v ozadju skrbi za periodično izvajanje varnostnega kopiranja ter izvajanje nadzora delovanja krmilnih računalnikov, dodanih na seznam v okviru aplikacije za interakcijo z uporabnikom (več o delu programa za nastavitev nadzora delovanja v poglavju 5.4.1.6).

Kot smo že omenili v fazi načrtovanja, smo ta del programske rešitve izdelali kot servisno aplikacijo, saj je potrebno zagotoviti neprekinjeno-periodično izvajanje varnostnega kopiranja. Poleg periodičnega izvajanja, je glavna naloga servisne aplikacije tudi spremljanje podatkov v tabeli *Backup_Settings* na bazi podatkov in prilagoditev izvajanja, v primeru spremembe podatkov o zahtevani periodi izvajanja kopiranja (*bs_type*, *bs_backup_period*, *bs_backup_time*) ali podatkov, ki predstavljajo zahtevo za ročni zagon kopiranja (*bs_backup_now*). Zaradi kompleksnega izvajanja servisne aplikacije, smo izdelali diagram poteka, ki je prikazan na sliki Slika 27 in prikazuje posplošen potek, oziroma postopek izvajanja servisne aplikacije.

Ob zagonu servisne aplikacije, najprej ustvarimo dva časovnika (komponenta *Timer*), prvega za periodično spremljanje podatkov o nastavitvah na podatkovni bazi in drugega za prav tako periodično preverjanje delovanja krmilnih računalnikov s seznama, ki smo ga definirali z uporabo aplikacije za interakcijo z uporabnikom. Po vsakem branju nastavitev tekom izvajanja prvega časovnika preverimo, ali je prišlo do izdane zahteve za ročni zagon varnostnega kopiranja. V primeru, da je bila ta zahteva izdana izdelamo varnostno kopijo. V nasprotnem primeru preverimo ali je interni podatek, oziroma vrednost spremenljivke o času naslednje varnostne kopije (inicializiramo jo ob vsakem zagonu časovnika za periodično kopiranje) prekoračena kar pomeni, da smo izpustili periodo kopiranja (vzrok je lahko nedelovanje aplikacije ali nedelovanje računalnika). V primeru izpustitve izdelamo varnostno kopijo in nadaljujemo s preverjanje spremembe periode. V primeru, da je uporabnik v aplikaciji za interakcijo spremenil nastavitve varnostnega kopiranja, izvedemo zagon časovnika, ki bo po nastavljeni novi periodi izvajal varnostno kopiranje. Če pa ni prišlo do spremembe, potem preverimo ali gre za prvi, oziroma ponovni zagon aplikacije, čemur sledi zagon časovnika, oziroma ponovitev celotnega cikla s časovnim premorom.



Slika 27: Posplošen diagram poteka za izvajanje servisne aplikacije.

Ob vsakem izteku časovnika za periodično varnostno kopiranje, začasno prekinemo glavni časovnik, ki spremlja spreminjanje nastavitev, ter preberemo podatke o vseh izbranih parametrih iz tabele *DB_Parameter* na podatkovni bazi. Po uspešni pridobitni podatkov se sprehodimo skozi vse prebrane parametre, ter za vsakega izmed njih s podatkovne baze preberemo podatke za povezavo na krmilni računalnik in strežnik OPC DA kateremu pripada. Za tem izvedemo povezavo na pripadajoči strežnik OPC DA, ter izvedemo branje podatkov. Podatke preberemo v obliki bloka posameznih bajtov. Rezultate branja posameznega parametra shranjujemo v podatkovni slovar tipa *Dictionary*, ki nam omogoča shranjevanje podatkov v obliki <*identifikator_parametra, prebrani_blok_podatkov*>. Po uspešno izvedenem branju primerjamo prebrane podatke s podatki zadnje zabeležene varnostne kopije s podatkovne baze. V primeru odkrite razlike pri primerjavi, shranimo podatke v obliki nove varnostne kopije na podatkovno bazo. Če razlike med podatki ni, potem v tabelo *Log* na podatkovni bazi zabeležimo dogodek o preskoku varnostne kopije, zaradi enakosti podatkov. Tekom servisne aplikacije v tabelo *Log* zapisujemo vse dogodke, opozorila in napake.

5.5 Težave pri izdelavi aplikacije

Pri izdelavi aplikacije nam je glavno oviro predstavljala časovna omejitev, ki nam je v nekaterih delih onemogočila dodatno razširitev, oziroma implementacijo dodatnih funkcionalnosti. Tako smo bili omejeni na izdelavo aplikacije, ki nakazuje na možnost razvoja obsežnega sistema za varnostno kopiranje s krmilnih računalnikov in hkratno izvajanje nadzora delovanja krmilnih računalnikov in podprte opreme.

Poleg omenjenega smo bili pri razvoju omejeni tudi z razpoložljivostjo strojne opreme, ki je nujna za preizkus delovanja funkcionalnosti varnostnega kopiranja, ponastavitve, izbire parametrov in nekaterih ostalih funkcionalnosti aplikacije. Pri razvoju aplikacije smo tako uporabljali zgolj krmilne računalnike proizvajalca Siemens in tako poizkušali v danem času izdelati aplikacijo kar najbolj kompatibilno, oziroma nadgradljivo s strani podpore različnim krmilnim računalnikom.

5.6 Možne izboljšave

Izdelana aplikacija predstavlja začetno verzijo sistema za varnostno kopiranje. V tem poglavju smo predstavili nekatere možne izboljšave, oziroma nadgradnje, ki jih zaradi časovne omejitve nismo uspeli realizirati in predstavljajo dobro smernico pri nadaljnjem razvoju.

5.6.1 Izboljšava komunikacije

Trenutno za komunikacijo med obema deloma aplikacije, uporabljamo bazo podatkov, kjer imamo rezervirano vrstico v tabeli *Backup_Settings*, preko katere aplikaciji izmenjujeta zahteve za izvajanje različnih operacij. Zaradi takšnega načina komunikacije je oteženo preverjanje delovanja servisne aplikacije iz aplikacije za interakcijo, ter počasna izmenjava zahtev (ročni zagon in nastavitve periode varnostnega kopiranja), katere hitrost je odvisna od frekvence branja podatkov z rezervirane tabele. Ena izmed možnih rešitev je uporaba vtičnikov (angl. Sockets), ki nam v višjem programskem jeziku C# omogočajo realizacijo komunikacije med različnimi aplikacijami. V tem primeru bi servisna aplikacija delovala kot strežnik z vtičnikom, ki bi na določenih vratih (angl. Port) omogočala komunikacijo z aplikacijo za interakcijo z uporabnikom, ki bi v tem primeru predstavljala odjemalca z vtičnikom.

5.6.2 Razvoj podpore različnim krmilnim računalnikom

Pomembna izboljšava se pri razviti aplikaciji ponuja predvsem s strani zagotavljanja kompatibilnosti z različnimi krmilnimi računalniki, ki bi poleg Siemens krmilnih računalnikov (in ostalih s podobno naslovno hierarhijo) podpirala tudi ostale manj razširjene. Pri razvoju smo zato poizkušali izdelati enostavno nadgradljivo aplikacijo s strani zagotavljanja kompatibilnosti, kar bo v prihodnosti omogočilo hitro in enostavno dopolnitev na tem področju.

5.6.3 Kompatibilnost z ostalimi strežniki OPC

Izdelana aplikacija podpira komunikacijo le z vmesniki OPC DA, ki so danes še vedno najpogosteje uporabljeni vmesniki za dostop do nižje-nivojskih naprav. Ker se danes uveljavlja že nova različica vmesnikov OPC, poznana kot OPC UA je ena izmed potrebnih izboljšav aplikacije tudi zmožnost dostopa do ostalih, oziroma novejših strežnikov OPC. Na tem področju je možnih veliko rešitev, saj vrsta podjetij ponuja tako imenovane ovojnice (angl. Wrapper), ki omogočajo dostop do ostalih strežnikov OPC brez večjih sprememb v obstoječih aplikacijam (odjemalcih OPC). Ena izmed omenjenih je komponenta OPCDA.NET-UA, ki jo ponuja podjetje Advosol [11].

5.6.4 Hitrejše izvajanje varnostnega kopiranja

Pri testiranju aplikacije se je pokazala večja časovna potratnost pri ustvarjanju varnostnih kopij, ki zajemajo večje število krmilnih računalnikov in pripadajočih podatkovnih blokov. Po temeljitem pregledu izvorne kode servisne aplikacije smo ugotovili, da je možna izboljšava pri branju podatkov izbranih parametrov. Ocenili smo, da bi s predhodnim vzpostavljanjem povezave (ne več pri vsakem branju podatkov parametra) na vse različne strežnike OPC DA, občutno pospešili generiranje obsežnih varnostnih kopij.

5.6.5 Nadgradnja obveščanja o zaznanih nepravilnostih v delovanju

Servisna aplikacija trenutno omogoča periodično spremljanje delovanja izbranih krmilnih računalnikov in posledično tudi strežnikov OPC DA. V primeru odkritih težav pri komunikaciji to tudi zabeleži v tabelo dogodkov. Kot izboljšavo obstoječega sistema izvajanja nadzora, bi lahko dodali še obveščanje v obliki elektronskih sporočil ali kratkih sporočil SMS z uporabo GSM modema in sms prehoda kot je Ozeki SMS Gateway [19], kar bi omogočalo hitro obveščanje in reakcijo odgovornih, v primeru odkritih nepravilnosti.

5.6.6 Izvoz varnostnih kopij na različne medije

Trenutna izvedba aplikacije zapisuje vse izdelane varnostne kopije na bazo podatkov. Za zagotavljanje splošne varnosti in obstojnosti izdelanih varnostnih kopij, bi lahko v aplikacijo dodali možnost izvoza in uvoza varnostnih kopij, na različne medije za shranjevanje podatkov.

5.6.7 Možnost medsebojne primerjave obstoječih varnostnih kopij

Uporabniku bi pri tej nadgradnji omogočili avtomatično vsebinsko primerjavo dveh izbranih varnostnih kopij. Aplikacija bi izbrani varnostni kopiji vsebinsko primerjala, ter uporabniku izpisala podatke o vseh vsebinskih razlikah. S tem bi zagotovili možnost hitre primerjave ter ugotavljanja razlik, v varnostno kopiranih podatkih.

6 ZAKLJUČEK

Z razvito aplikacijo, ki jo zajema to diplomsko delo, smo nakazali možnost in začrtali pot za nadaljnji razvoj celovitega sistema, za varnostno kopiranje izbranih podatkov, s krmilnih računalnikov. Kot smo to opisali že v poglavju 5.6, je možno razvito aplikacijo nadgraditi in s tem izboljšati še na mnogih področjih. V fazi načrtovanja in med samim razvojem smo ugotovili, da bi bila poleg nekaterih optimizacijskih posodobitev zelo dobrodošla razširitev na področju izvajanja nadzora delovanja, kar bi razviti programski rešitvi dodalo več funkcionalno in dodano praktično vrednost.

V nekem proizvodnem procesu, ki predstavlja ciljno delovno okolje aplikacije, bi lahko tako razširjena aplikacija predstavljala nepogrešljivo in celo standardno programsko opremo za zagotavljanje varnosti, pred izgubo podatkov ter izvajanje nadzora delovanja izbranih komponent v tem okolju. Naša aplikacija bi tako poleg standardnega in periodičnega varnostnega kopiranja izvajala tudi vsesplošno obveščanje o napakah in težavah na povezovalnih komponentah, kot so strežniki OPC in krmilnih računalnikih, ki so danes ključni del praktično vsakega avtomatiziranega sistema v proizvodnem procesu. Menim, da smo pri razvoju uspeli v relativno kratkem času realizirati praktično vse zastavljene cilje in izdelati lahko nadgradljivo programsko rešitev, ki nudi neprekinjeno, zanesljivo ter relativno hitro varnostno kopiranje, pregled, upravljanje in ponastavitev ključnih podatkov, za delovanje proizvodnega procesa.

KAZALO SLIK

KAZALO TABEL

VIRI IN LITERATURA

[1] (2012) The Data Recovery Solution. Dostopno na: http://www.ontrack.com/library/ontrack02wp.pdf

[2] (2012) The Cost Of Lost Data. Dostopno na: http://gbr.pepperdine.edu/2010/08/the-cost-of-lost-data

[3] (2012) Backup types. Dostopno na: http://www.backup4all.com/kb/backup-types-115.html

[4] (2012) All about Backup & Security. Dostopno na: http://www.backupschedule.net/databackup/databackupmethods.html

[5] (2012) How to backup. Dostopno na: http://howtobackup.net

[6] (2012) Backup. Dostopno na: http://en.wikipedia.org/wiki/Backup

[7] (2012) The history of backup. Dostopno na: http://www.backuphistory.com

[8] (2012) Automation. Dostopno na: http://en.wikipedia.org/wiki/Automation

[9] (2012) OPC Foundation. Dostopno na: http://www.opcfoundation.org

[10] Frank Iwanitz, Jürgen Lange, *OPC: Fundamentals, Implementation, and Application*, Heidelberg: Hünting, 2002.

[11] (2012) Advosol. Dostopno na: http://www.advosol.com

[12] (2012) PLC Manual. Dostopno na: http://www.plcmanual.com

[13] (2012) SIEMENS Automation Technology. Dostopno na: http://www.automation.siemens.com

[14] (2012) C Sharp (programming language). Dostopno na: http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29

[15] (2012) Microsoft Visual Studio. Dostopno na: http://www.microsoft.com/visualstudio/en-us

[16] (2012) Sybase PowerDesigner. Dostopno na: http://www.sybase.com/products/modelingdevelopment/powerdesigner

[17] (2012) Microsoft SQL Server. Dostopno na: http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[18] (2012) SQL Server Management Studio. Dostopno na: http://en.wikipedia.org/wiki/SQL_Server_Management_Studio

[19] (2012) Ozeki SMS Gateway. Dostopno na: http://www.ozekisms.com/