UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Fleischmann

Gradniki vmesniškega podsklopa sistema za procesno dokumentacijo

Visokošolski strokovni študij

Diplomsko delo

Mentor: doc. dr. Janez Demšar

Ljubljana, 2008

Zahvala

Zahvaljujem se dr. Antonu Ružiču za številne podatke, nasvete in strokovno pomoč pri diplomskem delu. Hvala tudi Odseku za avtomatiko, robotiko in biokibernetiko Instituta Jožef Stefan, za pogoje, ki so mi jih nudili pri izdelavi diplomskega dela.

Zahvaljujem se tudi mentorju doc. dr. Janezu Demšarju, za pomoč in napotke pri izdelavi diplomskega dela.

Naloge, ki sem jih opravil za diplomsko delo so del večjega sistema. Zahvaljujem se sodelavcem Odseka, ki so delali na drugih sklopih tega sistema za informacije in pomoč, ki so bile nujne za usklajeno delo.

Vsem članom družine se zahvaljujem za vse, s čimer so pripomogli k mojemu uspehu.

Kazalo

Seznam uporabljenih kratic in simbolov

Kratice

Kratica	*Pomen*
PLK	programirljivi logični krmilnik (angleško Programmable Logic Controller, PLC)
SCADA	sistem za nadzor, vodenje in zajemanje podatkov (angleško Supervisory Control And Data Acquisition)
VBA	programski jezik Microsoft Visual Basic for Application
DL	risalna plast v programih zbirke Microsoft Office (angleško Drawing Layer)
CAD	računalniško podprto načrtovanje (angleško Computer-Aided Design)
CAE	računalniško podprto inženirstvo (angleško Computer-Aided Engineering)
MCAD	mehansko oziroma strojniško računalniško podprto načrtovanje (angleško Mechanical Computer-Aided Design)
API	programski vmesnik, to je množica funkcij, procedur, metod in razredov, s katerimi računalniški programi dostopajo do funkcionalnosti operacijskega sistema (angleško Application Programming Interface)
I/O	vhodno/izhodno (angl. Input/Output)
OLE	programska tehnologija podjetja Microsoft za predmetno povezovanje in vdelovanje (angl. object linking & embedding)

Pisave

Tekst pisan s poudarjenimi črkami, se nanaša na imena menijev, gumbov in ostalih stvari, na katere lahko kliknite, kot npr. gumb **V redu (OK)**

Tekst znotraj oklepajev se nanaša na tipke tipkovnice (npr. <Enter>).

Povzetek

V nalogi smo opravili nekatera dela potrebna za razvoj sistema za projektiranje in analizo procesnih sistemov. Namen takega sistema je omogočiti hiter opis raznolikih sklopov iz katerih sestoji nek konkretni avtomatizirani sistem za nadzor nekega tipično proizvodnega sistema in hiter opis povezav med temi komponentami.

Pregledali smo bolj znane dostopne komercialne sisteme, namenjene projektiranju in drugim povezanim nalogam, to je sistem ePLAN, WSCAD, SEE Electrical/CADdy++ ter AutoCAD Electrical. Analizirali smo njihove lastnosti.

Glede na želen namen in uporabnost, usmerjeno predvsem v podporo razvoju programskih in materialnih delov avtomatiziranega sistema, smo utemeljili smiselnost razvoja lastnega sistema.

Opisali smo koncept celovite zgradbe sistema za podporo razvoju avtomatizaciji procesnih sistemov. Zaradi lažjega razvoja, dopolnjevanja in poznejše uporabe je zasnovana zgradba modularna. Iz istih razlogov postavljeni koncept predvideva uporabo bodisi odprtih orodij, bodisi razširjenih de-facto standardov.

Opisali smo dva gradnika celotnega sistema, ki smo ju razvili v okviru te naloge.

Ključne besede

orodja za projektiranje, orodja za dokumentacijo, uporabniški vmesniki, grafi

Abstract

In this work we have developed some components of a custom developed system for process control system documentation. The overall goal of such documentation system is to give a user the capability to describe various components and aspects about an automated process control and to describe relations between them. After that, the user should be able to inquire and get the required information in various forms.

We overviewed some known commercial systems dedicated to the design and documentation of electrical and electromechanical systems. We briefly evaluated ePLAN, WSCAD, SEE Electrical/CADdy++ and AutoCAD Electrical.

We presented arguments and needs to develop an own program, aimed specifically to the documentation of various software, hardware and material aspects of a process control system.

In our work, we have described the concept and overviewed the structure of such a support system for development of automated process control. The modular structure allows and

facilitates gradual development and later upgrading. For the same reasons it is based on open source and de-facto standard software tools as much as possible.

We have described two components of the system that we have developed.

Key words:

design tools, documentation software, user interfaces, graphs

1 Uvod

V okviru diplomskega dela smo opisali nekatere dostopne sisteme za računalniško podprto projektiranje in razvili nekaj modulov, ki bodo del lastnega programskega sistema za projektiranje.

Naloga izhaja iz nekaterih konkretnih potreb osrednjega nacionalnega raziskovalnega instituta oziroma odseka, ki se raziskovalno in razvojno ukvarja z avtomatizacijo in robotizacijo [1]. Pri raziskovalnem delu razvijajo lastne naprave, na primer:

- različne robote za testiranje športne obutve in opreme,
- merilne komore in pripadajoče krmilnike za testiranje in izvajanje eksperimentov za določanje človeških fizioloških značilnosti,
- naprave in krmilnike za merjenje in testiranje ergonomskih značilnosti opreme.

Omenjeni odsek izvaja tudi raziskovalno-razvojne projekte za naročnike iz gospodarstva. Pri tem pogosto razvijejo in predajo v uporabo zaključene sisteme, na primer:

- testne naprave,
- naprave za avtomatizirano izvajanje proizvodnih procesov,
- robotizirane celice,
- celotne proizvodne linije.

Takšne sisteme je potrebno dokumentirati iz praktičnih razlogov, včasih pa tudi iz zakonskih razlogov, na primer:

- za nemoteno in učinkovito razvojno fazo,
- v obliki navodil za uporabo, ki jih potrebuje uporabnik,
- v obliki dokumentacije, ki je potrebna za morebitno dovoljenje za uporabo,
- v obliki tehnične dokumentacije ki se preda uporabniku za namen vzdrževanja in lastnega servisiranja,
- v obliki lastne razvojne dokumentacije, namenjene posegom pri iskanju nepravilnosti v delovanju in nadaljnjem razvoju, spremembam ter dopolnjevanju sistema.

Vsak sistem večje velikosti je tipično sestavljen iz več gradnikov, na primer:

- mehanskih komponent,
- senzorjev in aktuatorjev,
- krmilnikov, komandnih plošč, zaslonov za uporabniške vmesnike,
- komunikacijske in omrežne opreme,
- programske opreme.

Obstajajo tipični načini za dokumentacijo nekaterih od zgornjih komponent. Mehanski sistemi in razporedi naprav v okviru neke linije, celice ali postrojenja se pogosto dokumentirajo s klasičnimi računalniško podprtimi sistemi za strojniško načrtovanje (MCAD), ki vključujejo funkcije za računalniško podprto inženirstvo (CAE). Danes so ti sistemi pogosto integrirani z ali v nekatere druge širše sisteme za računalniško podprto načrtovanje, na primer v robotske

simulacijske sisteme, diskretne simulacijske sisteme. Za razvoj večjih programskih paketov obstajajo različni paketi za računalniško podprto programsko inženirstvo (CASE), ki so skupek metod in avtomatiziranih orodij za podporo teh procesov.

Če je naša naloga zelo strojniško-konstrukcijsko usmerjena, tipično uporabljamo CAD/MCAD sistem. Za predhodno robotsko programiranje (angleško off-line robot programming) uporabljamo robotske simulacijske oz. programirne sisteme.

V omenjenem odseku je pogosto bistvo projekta v načrtovanju nove rešitve nekega postopka ali avtomatizaciji nekega procesa, ki ni zelo obsežno v nekem posameznem vidiku, temveč je bistvo v nekem novem algoritmu, postopku, konstrukcijski rešitvi in podobno.

Takšne posebne ali »butične« rešitve se največkrat izvajajo na ključ, zato praviloma zajemajo vse vidike projekta. Ti so lahko:

- razmestitev naprav znotraj danih gabaritov,
- modeliranje in simulacija dogodkov in procesov,
- izbira in morebitna modifikacija običajnih komponent avtomatizacije,
- lastno načrtovanje in izdelava posebej razvitih komponent avtomatizacije, vključno z mehansko zgradbo, aktuatorji, senzorji, elektronskimi vezji, morebitnimi vgrajenimi računalniki in programsko opremo,
- izbira krmilnega sistema, ki lahko npr. vključuje več računalnikov PLK (programirljivih logičnih krmilnikov), PC računalnikov, izbira obstoječe in razvoj lastne programske opreme.

Načrtovanje krmilnega sistema vključuje poleg izbire, programiranja računalniških komponent tudi pravilno povezavo ter integracijo vseh teh sistemov.

Končni krmilni sistem tako lahko sestavljajo tesno integrirana:

- programska oprema in algoritmi, implementirani v različnih jezikih ali orodjih (Visual Studio, CodeGear RAD Studio), VBA, jeziki za programirljive logične krmilnike (PLK), programski jeziki robotskih krmilnikov, programi, ki jih nalagamo v ugnezdene računalnike (angleško embedded computer) strojev in naprav ter podobno,
- večje število računalnikov (PC, PLK, ugnezdeni procesni računalniki, robotski krmilniki) na katerih teče ta programska oprema,
- elektronska vezja in naprave,
- številni senzorji in aktuatorji za povezavo računalnikov s krmiljenim procesom,
- ožičenje za napajanja in za komunikacije računalnikov in perifernih senzorjev ter aktuatorjev,
- računalniške mreže med računalniki oz. med računalniki in »pametnejšimi« napravami.

Na omenjenem odseku instituta želijo uporabljati sistem, ki učinkovito podpira ta proces načrtovanja, vzdrževanja in dopolnjevanja krmilno-avtomatizacijskih sistemov, ki poleg programskih komponent zajema tudi povezljivost in druge vidike [1].

Osnovna naloga takega sistema je podpora izvedbe programske opreme. Ker gre za procesno programsko opremo, rabimo pri temu poleg informacije o medsebojni povezavi programskih

gradnikov tudi informacijo o funkcionalnih (in včasih fizičnih) povezavah s programskimi gradniki na drugih računalnikih, povezavah med računalniki, povezavah s senzorji, aktuatorji, stroji in tudi s fizičnimi sistemi.

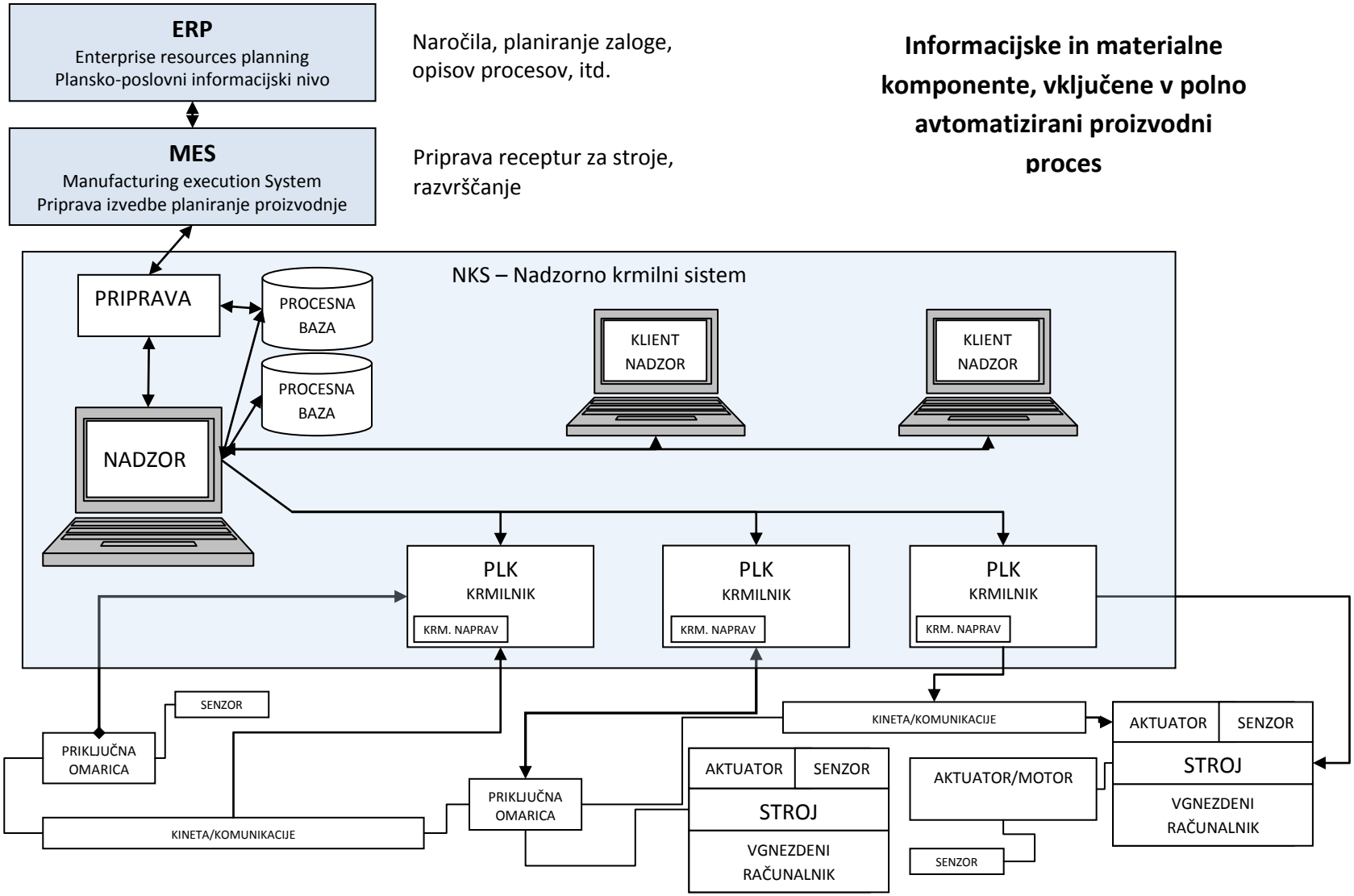
Recimo, da imamo avtomatizirani proces in ustrezni krmilni sistem prikazan na strani 4 spodaj. Pri avtomatiziranem delu se paralelno izvajajo vse naloge. Za izvedbo neke konkretne naloge lahko velja:

- na ERP sistemu je opis izvedbe naloge za neko razpisano proizvodnjo,
- na MES sistemu se ob tem planira način delitve te naloge na več strojev,
- na nadzornem sistemu se za proženje proizvodnje naložijo podatki na osrednji PLK,
- osrednji PLK razdeli podatke na druge PLK-je, ki so povezani z ustrezno periferijo, to je senzorji, aktuatorji in stroji, ki so vključeni pri izvedbi naloge,
- povezava PLK-jev in ustrezne periferije je speljana preko vodil, kinet, priključnih omaric in elektronskih vezij,
- povezava PLK-jev in inteligentnih strojev je izvedena preko običajnih digitalnih povezav (npr. števila digitalnih vhodov/izhodov) ali preko procesnih vodil po nekem protokolu (Ethernet, RS-485/ProfiBus in podobno).

Pri razvoju vseh funkcionalnosti sistema (s programsko opremo za vse vključene računalnike) je pomembno razumevanje povezav med naštetimi komponentami. Še bolj pomembno pa je razumevanje načina delovanja in povezav pozneje pri iskanju vzroka morebitnih napak ali pri dograditvi sistema, saj smo v vmesnem času lahko pozabili vse podrobnosti.

Želimo imeti sistem, s katerim pri razvoju opišemo razvit sistem in nam pri razvoju ali pozneje pomaga pri razumevanju delovanja predvsem tako, da nam na osnovi vnesenih podatkov daje odgovore na vprašanja kot so na primer:

- kateri računalniki in katere programske komponente na teh računalnikih so vključene pri izvajanju neke funkcije delovanja (procesnega krmiljenja),
- želimo opis posameznih nalog posameznih programskih komponent oz. računalnikov pri izvedbi omenjene funkcije,
- s katerimi podatki nekega programskega modula so povezani podatki drugega programskega modula,
- v katerih procesnih točkah oz podatkih (angleško tags) procesne baze nadzornega sistema (SCADA) so shranjeni podatki o stanju senzorja na neki napravi,
- po kateri kineti ali kinetah in morebiti drugih komponentah (na primer priključnih omaricah) poteka signalna povezava med izhodno kartico nekega krmilnika PLK in krmiljenim elektromotorjem,
- za nek izhod krmilnika PLK želimo točen fizični položaj povezanega procesnega aktuatorja (na primer na shemi, narisani v paketu AutoCAD) in tudi fizični položaj na vseh dostopnih fotografijah tega elementa in okolice,
- želimo točen fizični potek povezanih signalov in veličin, tako električnih (digitalnih in morebiti analognih) kot pnevmatskih, med krmilnikom PLK, krmiljenim pnevmatskim ventilom, pnevmatske povezave in strojem, kateremu moramo zagotoviti pravilen dotok pritiska.



Slika 1: Informacijske in materialne komponente, vključene v polno avtomatizirani proizvodni proces

Ker obstaja več komercialnih sistemov, namenjenih projektiranju in dokumentiranju električnih shem, krmilnih omaric, priključnih sponk krmilnikov PLK, pnevmatskih napeljav in podobno, smo želeli najprej ugotoviti, ali je kakšen izmed teh primeren za opis, dokumentiranje in podporo pri pravkar naštetih nalogah, ki nastopajo ob razvoju in vzdrževanju procesne programske opreme in sistemov. V poglavju 2 smo pregledali sisteme AutoCAD Electrical, SEE Electrical Expert, ePLAN electric P8 in WSCAD. Ugotovili smo, da za naš namen ti niso najbolj primerni. Imajo namreč veliko funkcij, ki jih ne potrebujemo, vendar zaradi obsega teh zahtevajo precejšen čas, da jih osvojimo in jih uporabljamo učinkovito, to je zadosti hitro. Po drugi strani ne omogočajo enostavnega opisa povezav med različnimi tipi gradnikov, npr. med programsko opremo, električno materialno opremo, povezavami, pnevmatsko ali hidravlično opremo in podobno.

Zato so se odločili za razvoj novega sistema za dokumentacijo in podporo razvoju in vzdrževanju procesnih programsko-materialnih sistemov. Opis osnovnega koncepta in zgradbe smo podali v poglavju 3. Gre za večji modularni sistem, ki se bo korakoma razvijal. Sestavljen bo iz raznih gradnikov za urejanje sistemov in podsistemov, povezav, opisov, urejanje grafičnih predstavitev, baz in podobno.

Naša naloga v okviru diplomske naloge je razvoj nekaterih izmed gradnikov bodočega sistema.

V danem konceptu je poleg zgradbe predvidena tudi uporaba čim bolj razširjenih orodij, de-facto standardov in kjer je možno odprtokodnih orodij.

V poglavju 4 opisujemo značilnosti osnovnih uporabljenih orodij, to je jezika VBA skupaj s komponento »Drawing Layer« in orodja Graphviz.

Nato v poglavju 5 opisujemo naše delo na gradnikih sistema, to je a) določanje in razvoj osnovnih komponent za uporabniški vmesnik, namenjen vnosu zgradbe sistemov in podsistemov ter povezav med njimi in b) razvoj osnovnih modulov za označevanje in komponent na slikah in shemah.

Sklepne ugotovitve in smernice za nadaljevanje dela smo podali v poglavju 6.

2 Pregled obstoječih sistemov za računalniško podprto projektiranje

V tem poglavju bomo opisali nekaj komercialno dostopnih sistemov, namenjenih različnim nalogam projektiranja električnih naprav in večjih sistemov.

V tem sestavku bomo za tovrstne sisteme, katerih namen smo uvodoma opisali, uporabljali okrajšavo »računalniško podprto projektiranje«. So nek razred iz večje družine programov za računalniško podprto načrtovanje (angleško CAD) [2].

Ob tem naj omenimo, da se generična imena teh programov nekoliko razlikujejo tako v angleških originalih kot v slovenskih prevodih, tudi zaradi tega, ker so namenjena ali izhajajo iz nekoliko različnih področij uporabe. Kot na primer, v literaturi najdemo angleška imena:

- Electrical CAD program,
- Computer-Aided-Design Software for Electrical Design,
- Computer-Aided-Design dedicated to Electrical Engineering, Systems, and Industrial Fluids,
- Electrical CAD dedicated to Industrial Automation,
- Electrical design automation software,
- Electrical design software.

Vsi spodaj opisani sistemi imajo funkcionalne podobnosti. Praviloma je vsak izmed teh sistemov zgrajen iz več modulov, od katerih so posamezni namenjeni specifičnim nalogam, na primer za električno ožičenje, za hidravliko in podobno. Prav tako je največkrat posamezni modul ali paket modulov dostopen v več različicah glede na obseg funkcij in posledično glede na ceno.

Uporabnik tako mora izbrati sistem, ki ustreza njegovim potrebam. Če kasneje zaradi potreb preide na bolj izpopolnjeno različico, naj ne bi imel težav s prenosom vnesenih projektov v dopolnjen sistem.

Pregledali bomo programe AutoCAD Electrical[3], SEE Electrical Expert[4][5][6][7], ePLAN electric P8[8][9] in WSCAD[10][11] [12][13][14][15][16][17][18][19][20].

2.1 AutoCAD Electrical

AutoCAD Electrical je sistem, ki je namenjen predvsem načrtovanju in urejanju elektronskih krmilnih sistemov [3]. Izpeljan je iz osnovne različice AutoCAD-a, ki načeloma tudi omogoča načrtovanje krmilnikov in ostalih procesnih sistemov, vendar je za izdelavo projekta, ki ga razmeroma hitro naredimo v AutoCAD Electrical, potrebno vložiti več truda, saj so v različici AutoCAD Electrical poenostavljene določene operacije, ki bi sicer od uporabnika osnovne različice AutoCAD-a zahtevale veliko dela.

Nekaj glavnih prednosti AutoCAD Electrical pred osnovno različico AutoCAD-a:

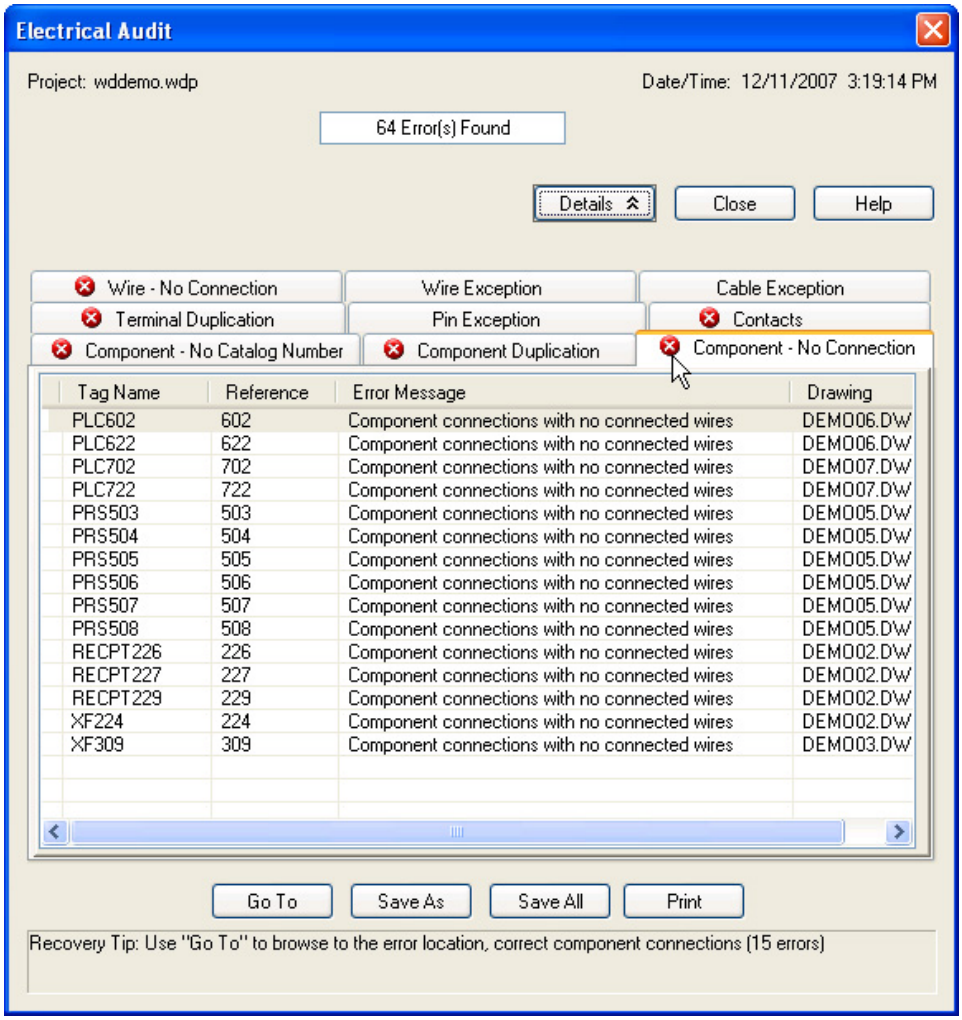
- vsebuje že v naprej izdelane elektrotehnične simbole,
- vstavljeni simboli so avtomatsko poravnani s katerokoli spodaj ležečo žico,

- obstoječe ožičenje se avtomatsko prekine in nato ponovno poveže z vstavljenim,
- avtomatsko oštevilčenje žic ter ostalih komponent.

AutoCAD Electrical vsebuje tudi vmesnik API, ki nam omogoča, da z izbranim programskim jezikom naredimo svojo aplikacijo, ki bo obsegala poljubne funkcije sistema AutoCAD Electrical, oziroma lahko sami napišemo funkcije, ki jih sistem še ne vsebuje ali pa si sestavimo svoj uporabniški vmesnik, ki bo prirejen našim potrebam.

Pomembnejša je tudi funkcija samodejno številčenje žic in označevanje komponent, saj nam ročno upravljanje s temi podatki vzame precej časa, predvsem pa pogosto prihaja do napak. AutoCAD Electrical tako samodejno številči vse komponente in žice na podlagi izbrane konfiguracije in tudi zazna situacijo, v kateri vstavljena številka žice pride v konflikt z nekim elementom. V takšnem primeru samodejno preišče vse elemente vzdolž obravnavane žice za morebitno prosto mesto, v katerega bi se lahko priklopila vstavljena žica.

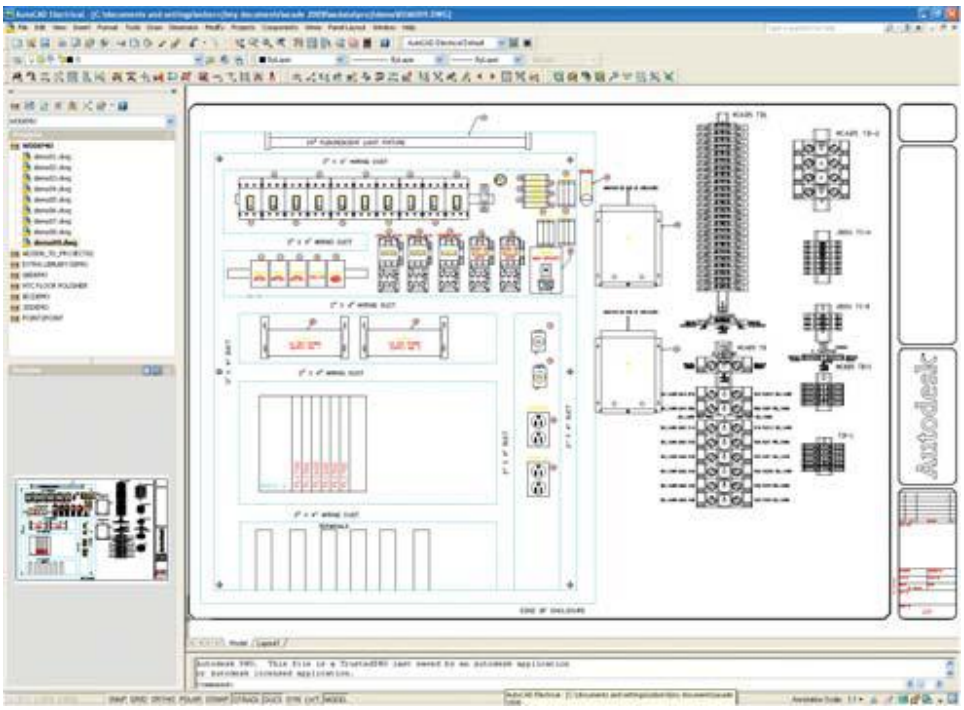
K odpravljanju napak pripomore tudi električno revizijsko poročilo (angleško Electrical Audit Report), ki analizira in izdela poročilo o manjkajočih oziroma nepravilnih številkah žic.



Slika 2: Električno revizijsko poročilo

To programsko okolje vsebuje tudi knjižnico že izdelanih simbolov, ki pomagajo načrtovalcem zmanjšati čas načrtovanja, saj jim ni potrebno ročno načrtovati posameznih simbolov, ampak

jih le dodajo iz seznama v projekt. Posledično se zmanjša tudi možnost napak v projektu. Teh že v naprej sestavljenih simbolov je sicer manj kot v konkurenčnih sistemih. Simboli so inteligentni. Tako na primer, ko med vstavljanjem releja v tokokrog določimo znamko in model releja, sistem ve, koliko stikov določen rele vsebuje in prepreči preveliko število povezav.



Slika 3: Prikaz uporabniškega vmesnika sistema AutoCAD Electrical

Ostale pomembnejše značilnosti sistema:

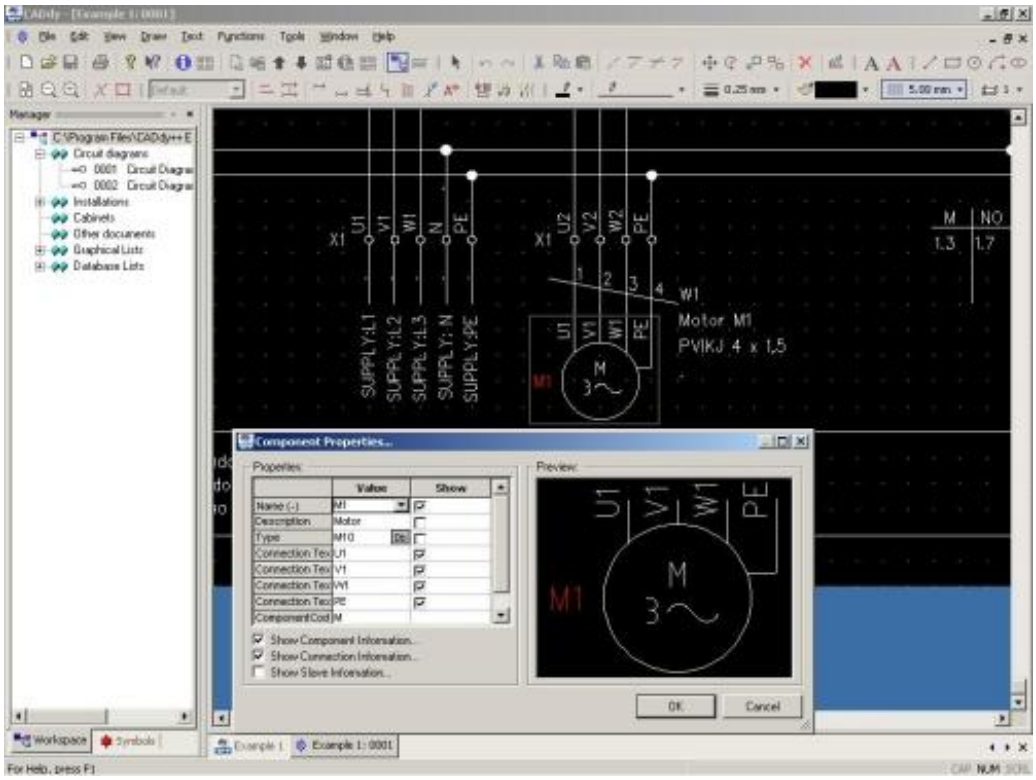
- vsebuje okoli 2000 definiranih standardiziranih simbolov, kot so na primer stikala, varovalke, releji, terminali, itd.,
- avtomatično oštevilčenje žic in poimenovanje komponent, kar pripomore, da projekt vsebuje manj napak,
- avtomatična izdelava poročil projekta, kar načrtovalcem prihrani čas,
- realno-časovno preverjanje napak omogoča odpravo napak že med izdelavo projekta,
- vsebuje celotno funkcionalnost znanega programskega paketa AutoCAD, le da so dodane funkcije, ki so bile razvite posebno za električne kontrolne sisteme,
- avtomatična izdelava PLC V/I shem iz podatkov, ki so shranjeni v razpredelnicah,
- enostavna izmenjava podatkov v široko uporabljenem formatu DWG,
- ponovna uporaba že izdelanih shem. Enostavno lahko kopiramo del sheme z vsemi njenimi značilnostmi ter jo uporabimo v novem projektu.

2.2 SEE Electrical V4R1

SEE Electrical je eden bolj razširjenih programskih paketov za projektiranje v elektrotehniki v srednji Evropi in v državah balkanskega polotoka. Prisoten je v več različnih vejah industrije [21] [5].

Vgrajeni vmesniki nudijo možnost vnašanja dokumentov iz ostalih Windows aplikacij, kot so Microsoft Office, AutoCad, vnos rastrskih in bitnih slik ter izvoz projektnih podatkov v večino

standardnih formatov, ki nam jih nudi Windows okolje. V vgrajenih podatkovni knjižnici je vsebovano tudi že okoli 10.000 elektrotehničnih simbolov.



Slika 4: Prikaz uporabniškega vmesnika sistema SEE Electrical V4R1

Ker je zgradba tega sistema modularna, obstajata tudi dve razširitve in sicer, modul SEE Electrical Cabinet Layout, ki omogoča konstruiranje stikalnih omar v elektrotehniki, ter SEE Electrical House installation, ki predstavlja risarski modul za hišne instalacije.

Najpomembnejše funkcije, ki jih vsebuje sistem SEE Electrical V4R1 so [4] [5]:

- DRC (preračunavanje možnih dovoljenih povezovalnih poti (angleško Design Rule Check)),
- orodja za ožičevanje (angleško rubberband) in samodejno povezovanje (angleško auto connection),
- Uvoz/izvoz baze proizvajalcev preko XML formata,
- »Autodiagraming«.

DRC

Algoritem sistema SEE Electrical omogoča pravokotno povezovanje komponent z uporabo funkcije DRC ter tako preverja možne in dovoljene povezave tako pri enopolnem kot pri več-polnem povezovanju. Funkcija samodejno prepozna število povezav in jih skuša samodejno povezati na želeno mesto.

orodja za ožičevanje in samodejno povezovanje

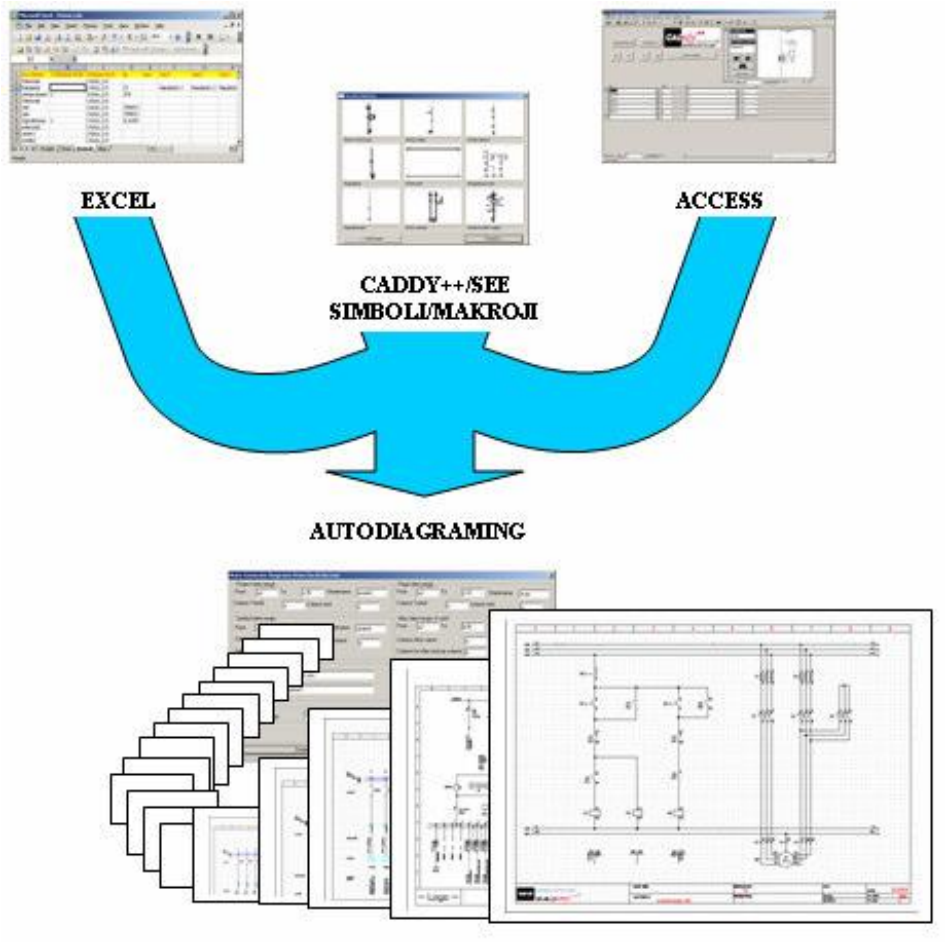
Z vklopom funkcije za ožičevanje dosežemo, da je simbol ob premikanju po delovni predlogi vedno pri ključen na vodnike. Z vklopom funkcije za samodejno povezovanje lahko omogočimo samodejno povezovanje elementov z ustreznim številom žic, kar pripomore k nekoliko hitrejšemu projektiranju.

Uvoz/izvoz baze proizvajalcev preko XML formata

Na voljo je več načinov uvozov/izvozov izdelkov oziroma podatkov baz različnih proizvajalcev. Najbolj uporaben uvoz/izvoz je izveden s pomočjo XML datoteke, s katero dosežemo neposredni uvoz ali izvoz podatkov brez dodatnih nastavitev.

»Autodiagraming«

Kot eno bolj zanimivih lastnosti tega sistema bi omenili metodo »Autodiagraming« [7]. Ta metoda omogoča hitrejšo izdelavo celotne projektne dokumentacije na podlagi podatkov, ki so shranjeni v MS Excel tabeli ali v MS Access bazi ter iz simbolov in makrojev, ki se nahajajo v projektu.



Slika 5: Potek metode »Autodiagraming«

Postopek uporabe te metode se začne z shranjevanjem posameznih delov izdelanega projekta v bazo simbolov, ki predstavljajo skupino elementov, celoten tokokrog ali določen diagram. Pred izdelavo simbola je potrebno definirati začetno in končno točko tokokroga ter opremiti simbole s teksti, ki bodo samodejno prevzemali vsebino iz MS Excel tabele ali MS Access baze.

Ob postavitvi elementov s pomočjo funkcije »autodiagraming« se samodejno vzpostavijo tudi povezave med simboli. Simboli, sponke in kabli lahko po potrebi označujemo samodejno, vrstni red postavljanja pa definiramo v Excelu oziroma Accessu.

Ta metoda je vključena je le v najbolj izpopolnjeno različico (različica Advanced) tega sistema in je primerna za večja podjetja, ki letno pripravijo veliko projektov. Za manjša podjetja bi nakup te različice po vsej verjetnosti predstavljal prevelik začetni vložek, saj je cena različice »Advenced« precej višja od osnovne različice.

Ostale pomembnejše značilnosti sistema:

- avtomatično številčenje komponent,
- urejevanje podatkov o kablih: število žic, barva, radij, tip, itd.,
- urejanje elementov in povezav neposredno v seznamih,
- nadzor podvojenih komponent,
- običajen prikaz bloka sponk vključno z informacijami o spončni letvi in lokaciji,
- upravljanje z releji in navzkrižno označevanje z vsemi podatki,
- vnašanje elementov le od izbranega proizvajalca,
- realno-časovno preverjanje napak, ki omogoča odpravo napak že med izdelavo projekta,
- samodejno številčenje potencialov in povezav,
- baza proizvodov (uvoz proizvodov iz formata ECAD ali Excelove tabele),
- izdelava lastne zbirke komponent,
- spletna tehnična pomoč.

2.3 ePLAN electric P8

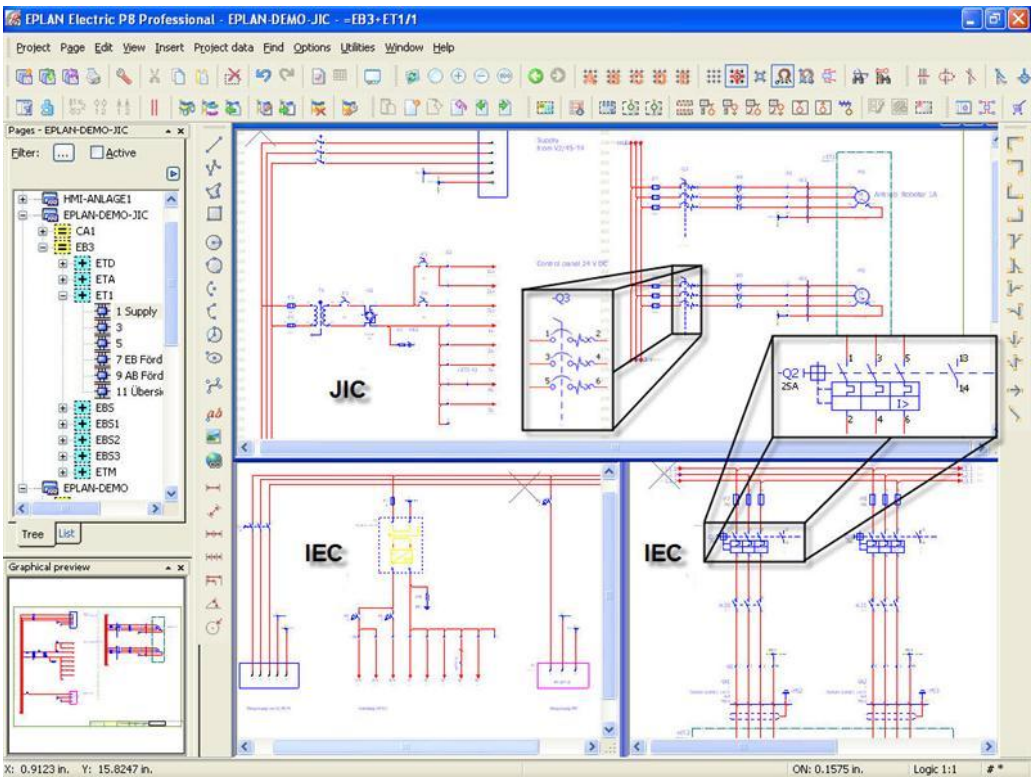
ePLAN Electric P8 je orodje za električno projektiranje. Uporablja se na več različnih področjih, kot so v industriji, energetiki, transportu ter v zgradbah. Sistem temelji na standardizirani strukturi podatkovne baze, poleg tega vključuje še grafični urejevalnik, osrednje upravljanje s pravicami uporabnikov ter pregledovalnik dokumentov [9].

Ker predvsem na področju mehatronike prihaja do pogostih prepletanj različnih inženirskih disciplin, je zahtevano poleg projektiranja elektrike oziroma elektronike tudi projektiranje upravljanja s fluidi, mehanike kot procesnega inženiringa. ePLAN poskuša združiti vse naštete discipline s skupno tehnološko platformo za vse različice tega sistema. Pri skupnem projektiranju so v vsakem posameznem delu pomembni nekateri skupni podatki in povezave med njimi, kot so: baze komponent, grafični urejevalnik, pregledovalniki, delo z revizijami, prevajanje v tuje jezike, administratorske pravice in drugo.

Slika 6: Platforma ePlan

Omenjena platforma omogoča, da so vsi skupni podatki na razpolago na enak način, vendar pa so odvisno od discipline (elektrotehnika, fluidi, mehanika, procesna, itd.) različni samo posamezni deli uporabniškega vmesnika in pripadajoče funkcije. Prednost take platforme in grafično-objektnega inženiringa je, da je ena in ista komponenta vidna v različnih disciplinah in v različnem grafičnem izgledu.

V primeru, da v določenem projektu uporabljamo npr. končni kontakt, je ta kontakt v elektriki signal na PLK-ju in ima določen grafični izgled, obenem pa je ta element tudi kontakt ki »bere« pozicijo določenega cilindra in ima drugačen grafični izgled v fluidni shemi.



Slika 7: Prikaz uporabniškega vmesnika sistema ePlan P8

ePLAN Electric P8 omogoča tudi štiri različne metode projektiranja [8]. Prva najbolj običajna in najbolj pogosto uporabljena metoda je grafična metoda.

Druga metoda je objektno – orientirana metoda. To je novejša in bistveno drugačna metoda, kjer se ne dela s simboli na običajen način, ampak se projektiranje in risanje tokovnih shem izvaja s katalоškimi števili komponent. V pregledniku (navigator-ju) se izbere želeno kataloško število oziroma tip, ki se s funkcijo »povleci in spusti« prenese na stran sheme, pri čemer ga program, ki deluje na podlagi funkcij, pretvori v simbol s kompletno definiranimi podatki, ki se nahajajo v podatkovni bazi. Na ta način naj bi projektiranje postalo enostavnejše, saj je dovolj samo poznavanje komponente, ki jo želimo uporabiti, program pa samodejno poskrbi za vse ostalo.

Tretja metoda se začne s seznamom komponent (kosovnico) in je v praksi dokaj pogosta. Pred vsakim projektom se izdela seznam elementov oziroma materiala, ki se bo uporabil v projektu. Običajno se to naredi v MS Excelu. ePLAN Electric P8 ponuja možnost, da se seznam neposredno uvozi, ter da se komponente neposredno uporabljajo v fazi projektiranja na objektno – orientirani način, kot je bilo predhodno razloženo. ePLAN Electric P8 omogoča da se seznam materiala izdela neposredno iz kataloga, glede na to da so v bazi podatkov vneseni vsi potrebni podatki za nabavo. S to metodo se projektiranje in definiranje začne že pri popisu opreme, in šele kasneje se nadaljuje z risanjem shem iz vnaprej pripravljene tabele v Excelu. Program omogoča tudi preverjanje potrošene količine elementov. Za vsako prekoračitev planiranih količin komponent se prikaže opozorilo, kar omogoča, da bodo vse potrebne komponente na razpolago v trenutku, ko jih potrebujemo.

Četrta metoda se začne z dispozicijo elementov v električni omari. Običajno je navada, da se pred izdelavo shem zaradi različnih potreb najprej definira dispozicija elementov v omari. Glede na to, da je program grafično in objektno orientiran, nam tak način dovoljuje, da se na osnovi izdelane dispozicije elementov preide v fazo izdelave tokovnih shem na osnovi porabljenih elementov v dispoziciji.

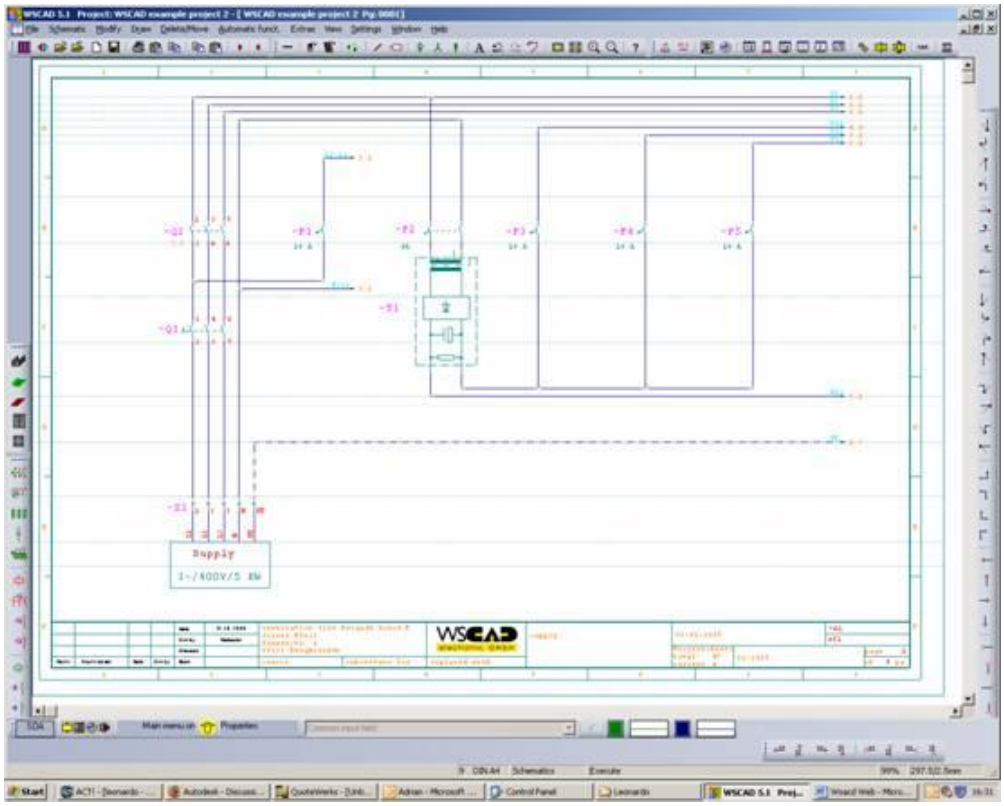
Ostale pomembnejše značilnosti sistema:

- vstavljanje že izdelanih simbolov iz baze simbolov (največ 4000 v enem projektu),
- baza proizvajalcev in podatkov o njihovih proizvodih (140 proizvajalcev s proizvodnimi programi),
- avtomatsko spajanje komponent,
- funkcije med posameznimi vezji,
- spletni izbor komponent,
- spletni prevod dokumentacije v enega ali več tujih jezikov,
- izdelava električne dokumentacije v skladu z vsemi svetovnimi standardi (DIN, JIC, IEC, KKS),
- Windows funkcionalnost (povleci in spusti, kopiraj/prilepi, pisava Unicode, povečava),
- podpira vse formate papirja,
- prilagoditev uporabniškega vmesnika, delovnega okolja, ikon, kratic,
- izmenjava podatkov preko formata XML, TXT, CSV, ODBC, COM,
- DXF/DWG uvoz/izvoz,
- MS Office uvoz/izvoz,
- uvoz/izvoz grafičnih formatov (bmp, jpg, gif, tiff),
- izdelava predlog strani/projektov/elementov,
- baza simbolov po IEC/DIN normi z 8 variantami v enem simbolu,
- izdelava in urejanje simbolov,
- vstavljanje simbola s »povleci in spusti« operacijo,
- izdelava lastnih makrojev,
- prilagodljiv raster,
- neomejeno število slojev (angleško layers),
- prosto grafično risanje (z opcijo kotiranja linij, krogov, kotov, povečevanja, razširjanja, itd.),
- osnovno delo s PLK-ji,
- filtriranje strani, projektov, baz,
- avtomatsko številčenje elementov in strani.

2.4 WSCAD

WSCAD se uporablja za izdelavo sheme vezij na več področjih, kot sta nadzor in avtomatizacija tehnologije, merjenje in regulacija, hidravlika, pnevmatika, elektronika, načrtovanje električnih omaric ter še na nekaterih ostalih področjih. Program je zaradi svoje preprostosti sicer bolj primeren za mala in srednje velika podjetja, vendar ga lahko zasledimo tudi v nekaterih večjih proizvodnih obratih, kjer ga uspešno uporabljajo tudi za načrtovanje proizvodnih linij.

WSCAD je programski paket, v katerem sta združena dva programa, program za projektiranje avtomatizacije in program za projektiranje električnih napeljav. Slednji omogoča boljšo obdelavo različnih tlorisnih risb, saj je to samostojno programsko orodje, ki je grafično močnejše od programa za projektiranje avtomatizacije. Novejše različice tega programskega paketa omogočajo uvoz in izvoz različnih risb v formatu »dxf« in »dwg« in s tem povečuje povezljivost s programom AutoCAD.



Slika 8: Prikaz uporabniškega vmesnika sistema WSCAD

Z uporabo modula električne napeljave je v načrtu električnih napeljav mogoče izrisati dispozicijsko risbo električnih napeljav z uporabo dodatnih instalacijskih simbolov. Kot osnovo za takšno dispozicijsko risbo je mogoče uporabiti gradbeno ali strojno risbo, uvoženo v formatu »dxf« ali »dwg«, jo z WSCAD-om dodatno obdelati in uporabno prirediti za lastno uporabo.

Možna je tudi namestitev še dodatno razširjenega modula za omenjene električne napeljave, ki omogoča samodejno izdelavo shem električnih razdelilcev. Projektant mora v tem primeru v celoti izdelati tlorisno shemo električnih napeljav, posameznim elementom pa tabelarično določiti vse parametre in tudi ustrezne makroje za vsak posamezni električni tokokrog. Samodejni izris električnega razdelilca se lahko izdela v enopolni ali tropolni izvedbi na že v naprej določenem formatu sheme.

Najpomembnejše funkcije, ki jih vsebuje sistem WSCAD so:

- črna skrinjica (angleško black box),
- upravljalec konektorjev,
- upravljalec PLK-jev,
- upravljalec kablov,

- polavtomatsko vodena izdelava krmilne omarice.

Črna skrinjica

Programski paket vsebuje funkcijo imenovano četveropol ali črno skrinjico, ki je primerna za izdelavo elementov z več priključki na enostavnejši in bolj pregleden način. Elementu se s pravokotnikom določi telo, v naslednjem koraku se v telo elementa postavijo priključki, s ponujeno tabelo pa se mu določijo tudi vsi ostali potrebni parametri (referenčno ime, naziv, ID številka, številka in ime priključka). Element se lahko uporabi takoj ali pa se ga shrani v knjižnico simbolov za kasnejšo uporabo. Ob umeščanju četveropola v risbo je tabela z vsemi naštetimi parametri vedno na razpolago tako za pregled podatkov kot tudi za morebitno spreminjanje.

Upravljalec konektorjev

Konektorji so v procesu projektiranja zahtevnejši elementi, saj so sestavljeni iz več različnih delov (tuljave ter pomožnih kontaktov). Programski paket WSCAD 5 ima v najbolj izpopolnjeni različici poseben upravljalec konektorjev, ki omogoča spletno obdelavo konektorskih funkcij ob vsaki umestitvi konektorja v električno povezavo na risbi. V kontaktnem križu se pri kontaktu, katerega smo že umestili na risbo, pojavi lokacijsko besedilo, ki pove, kje je kontakt umeščen.

Upravljalec PLK-jev

PLK moduli se pri projektiranju avtomatizacije lahko izrišejo kot samostojen simbol z vsemi priključki in vsemi povezavami na te priključke. V tem primeru lahko govorimo o kompaktnih PLK-jih, ki so funkcijsko manj zahtevni in imajo tudi manj V/I priključkov. Pri takšnih PLK-jih lahko običajno vse povezave narišemo v nekaj risbah. Kompleksnejši PLK-ji so sestavljeni iz več različnih modulov in imajo lahko tudi več 100 V/I priključkov. Zato jih pri projektiranju ne moremo izrisati kot samostojen modul, temveč so sestavljeni iz večjega števila enakih ali različnih modulov, njihove povezave pa se nahajajo na različnih straneh nekaj 10 ali 100 listnega načrta. Program WSCAD 5 omogoča projektiranje tako s kompaktnimi krmilniki kot tudi z modularnimi krmilniki. V prvem primeru je za to potreben samo en grafični simbol kot glavni element in še nekaj pomožnih grafičnih simbolov, ki imajo značilnost stranskega elementa. V drugem primeru pa je v knjižnici in bazi WSCAD 5 na razpolago večje število različnih modulov, katere projektant naniza v določenem zaporedju na risbo, jih poveže z napajanjem, vse povezave z V/I priključki pa izvede preko virtualnih V/I simbolov, ki so lahko nameščeni na različnih straneh načrta. Povezavo med virtualnimi V/I simboli in umeščenimi V/I moduli zagotavljata tako imenovani tekst PLK in lokacijski tekst (stran/pozicija), ki določata kateri virtualni V/I simbol je povezan na določen priključek nameščenega V/I modul in kakšna je njegova funkcija.

PLK upravljalec omogoča v posebnem pogovornem oknu v pregledni drevesni strukturi enostaven vpogled v vsak umeščeni V/I modul in omogoča spreminjanje pozicije V/I virtualnih simbolov s funkcijo »zagrabi–povleci«.

Upravljalec kablov

Pri programu WSCAD se kabli vrisujejo s pomočjo posebnega kabelskega simbola, ki omogoča izdelavo pomožne črte preko vseh kabelskih vodnikov izbranega kabla ter posebnega pogovornega okna upravljavca kablov, v katerem izberemo nov kabel iz obstoječe baze ter mu določimo način označevanja njegovih vodnikov. Upravljavec kablov vodi evidenco o vseh že umeščenih kablih in njihovih še prostih vodnikih, tako da vse kable, ki imajo porabljene že vse vodnike obarva rdeče in izpiše, da je število prostih vodnikov enako 0, ostale kable pa pusti obarvane črno ter izpiše dejansko število še prostih vodnikov.

Polavtomatska vodena izdelava načrta krmilne omarice

Programski paket WSCAD vsebuje polavtomatsko vodeno izdelavo načrta razmestitve elementov krmilne omarice iz spiska vgrajenega materiala v načrtu. Predhodno mora projektant dokončati projektni seznam materiala ter odpreti posebno vrsto sheme »izgradnja omarice«. V to risbo najprej s posebnimi grafičnimi simboli v ustreznem merilu umesti montažno ploščo, plastične kanale za žice, pritrdilne letve za opremo in napajalne vodnike, potem pa preko posebnega pogovornega okna izvede izdelavo spiska vgrajenega materiala. Iz spiska s pomočjo posebnih funkcij za centriranje in ponavljanje posamično izbira in vgrajuje vsak element posebej iz obstoječega spiska materiala. Vsak element, ki ima v bazi tudi sliko izgleda, se na montažno ploščo umesti v obliki in gabaritih, ki jih določajo podatki iz baze. Element, ki v bazi nima slike izgleda, pa se lahko umesti na montažno ploščo z gabariti (dolžina, širina, višina), ki jih projektant določi kar med projektiranjem in se ne shranijo v obstoječo bazo programa. Vsi že vgrajeni elementi se v spisku pogovornega okna obarvajo rdeče, ostali pa so obarvani črno.

Ostale pomembnejše značilnosti sistema:

- podpira tuje jezike ter omogoča prevajanje,
- omogoča tiskanje nalepk opreme, kablov in sponk,
- omogoča delovanje s starim standardom DIN 40719 in z novim standardom IEC 61346,
- avtomatsko številčenje simbolov,
- revizija zgodovine,
- DXF / HPGL izvoz/izvoz,
- VNS Izvoz,
- ASCII,Access, Excel, dBase, Uvoz / Izvoz (Podatki),
- UGL / UGS izvoz,
- omogoča uvoz podatkov preko eCAD standardnega vmesnika.

3 Razlogi za razvoj lastnega sistema in koncept sistema za podporo razvoja procesnih programskih sistemov

V tem poglavju bomo opisali razloge iz katerih smo se odločili za razvoj lastnega sistema za dokumentacijo, ne glede na dostopnost komercialnih sistemov, opisanih v predhodnem poglavju.

Nato bomo na kratko opisali nekatere vidike koncepta programa za dokumentacijo procesnih krmilnih sistemov koncepta, kot je pomembno za razumevanje vloge modulov, ki smo jih razvili. Več o konceptu programa najdemo v [1].

3.1 Razlogi za razvoj lastnega sistema

Ker so si zgoraj opisani komercialni sistemi po načinu uporabe precej podobni med seboj, smo se odločili pridobiti podrobnejše informacije o možnostih, ki jih ponujajo opisani sistemi, z namenom, da bi ugotovili, ali je glede na naše specifične potrebe upravičen razvoj lastnega sistema za dokumentacijo in načrtovanje. V ta namen smo obiskali slovensko podjetje, ki se ukvarja z distribucijo enega od opisanih sistemov in se pri strokovnjakih dodatno pozanimali o lastnostih tega sistema, saj iz brošur in tehničnih opisov ni mogoče razumeti vseh možnosti.

Iz daljše predstavitve njihovih strokovnjakov, smo prišli do zaključka, da takšen sistem iz več razlogov ne bi bil primeren za naše potrebe.

Poudarjamo, da so vsi naprej našteti ocene in razlogi podani glede na naše specifične zahteve. Opisan sistem oz. drugi sistemi so za določene druge namene zelo dobra izbira.

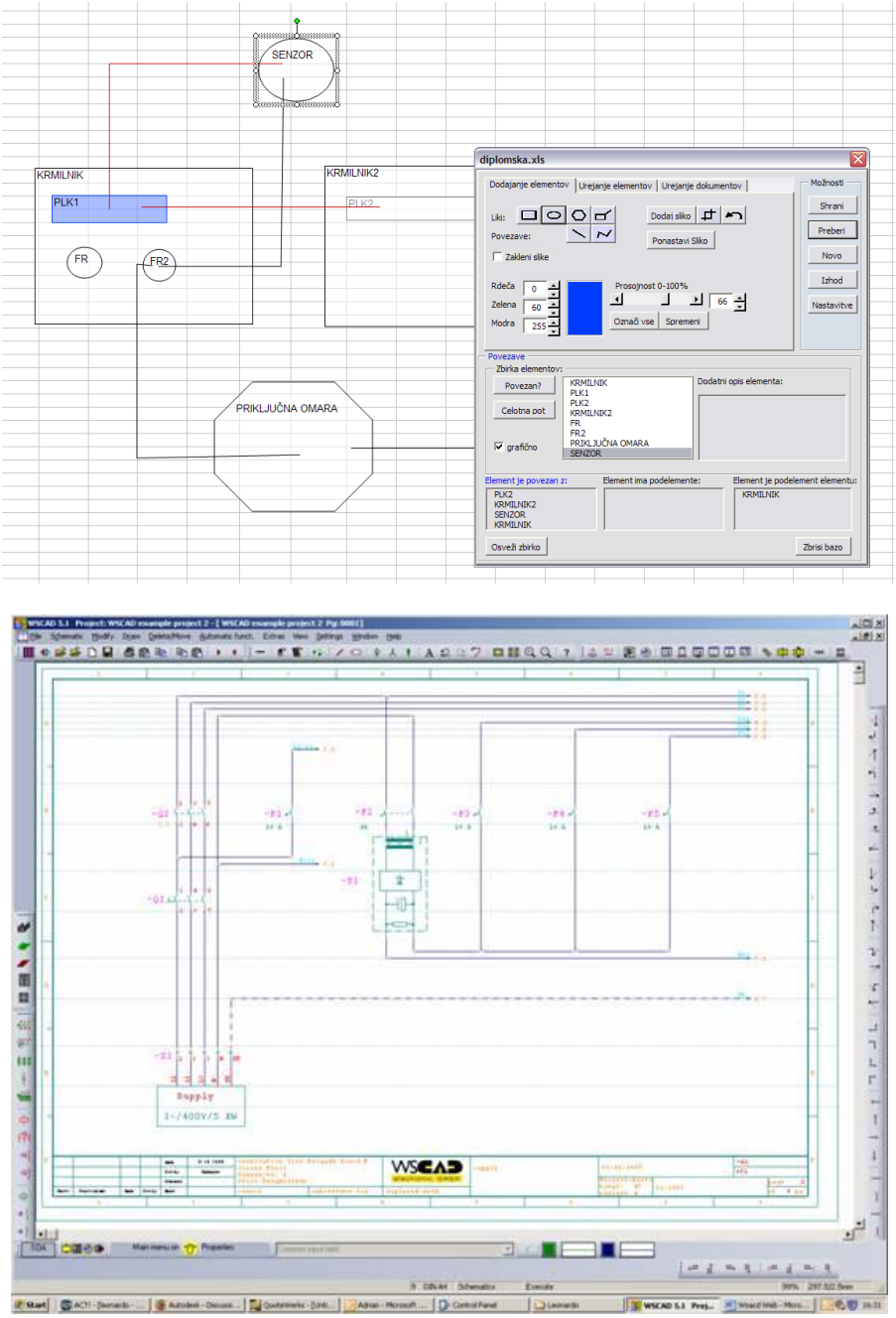
Eden glavnih je ta, da obstoječi sistemi zahtevajo že v samem začetku zelo podrobno definicijo proizvodnega sistema, kar pa ni vedno lahko delo. To se izkaže tudi v našem primeru, saj je proizvodnji sistem že izdelan, dokumentacija pa obstaja le v pisni obliki. Težko se je prebiti skozi vse sheme in tako do zadnje žice vpisati celotno linijo v sistem, saj niti ne potrebujemo pri vseh delih sistema takšno natančnost. To bi predstavljalo ogromno vnesenih podatkov, čeprav jih veliko večino sploh ne bi potrebovali.

V obstoječih sistemih je potrebno vsak nov dodani element definirati zelo natančno, od pozicije elementa, število konektorjev, ki jih vsebuje, do načina povezave, kar nam vzame veliko časa. Ti podatki za nas niti niso pomembni, za nas je bistveno le to, s čim je nek element v proizvodnem sistemu povezan.

Druga pomembna pomanjkljivost obstoječih programskih sistemov je ta, da ne podpirajo vnosa na nižjem nivoju kot je žica, torej podatkov ki se prenašajo po žicah. V našem primeru so potrebne tudi vrednosti določenih spremenljivk, ki jih v našem sistemu lahko zelo enostavno vnesemo, na povsem enak način, kot bi dodali nov element v našo dokumentacijo.

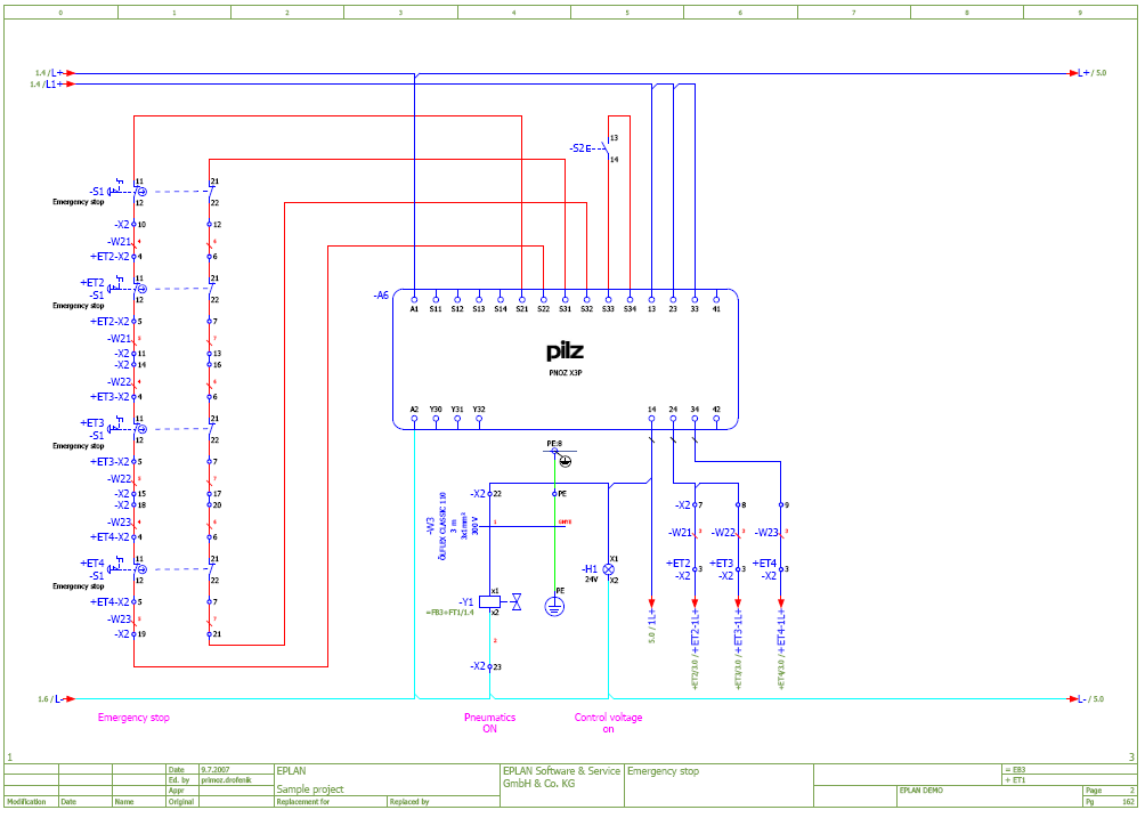
Slabost obstoječih sistemov je tudi dokaj zahteven in obsežen uporabniški vmesnik. Tako bi moral uporabnik opraviti kar nekaj usposabljanj, da bi ga lahko brez težav uporabljal, medtem ko smo v našem sistemu uporabili le orodja in funkcije, ki so primerna za naše potrebe in posledično je tudi obsežnost takega vmesnika neprimerno manjša. Spodaj je primerjava našega

uporabniškega vmesnika za grafično urejanje, ki je še vedno v prototipni fazi ter vmesnika sistema WSCAD.

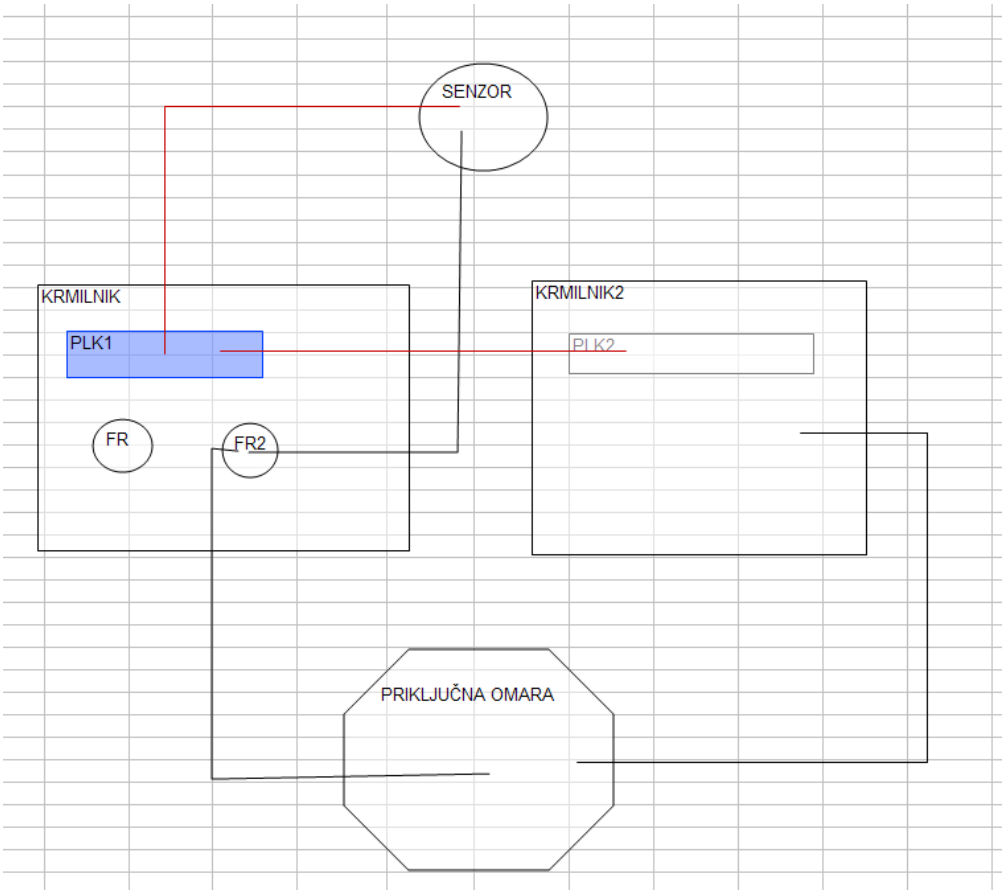


Slika 9: Primerjava našega uporabniškega vmesnika z vmesnikom v WSCAD-u

Naš sistem je zasnovan tako, da lahko po želji na začetku določimo le dele glavnega sistema, podrobnejšo vsebino pa pustimo prazno, ki jo lahko kasneje po želji dopolnjujemo. Povezave v sistemu ne predstavljajo fizično žice ampak le abstraktne povezave elementov, ki povedo, kako so elementi med seboj povezani. Spodaj je prikazana primerjava zahtevnosti povezovanja elementov med seboj v ePLAN-u ter v našem sistemu.



Slika 10: Povezovanje elementov v sistemu ePlan



Slika 11: Povezovanje elementov v našem sistemu

Kot je razvidno iz primerjave, je naš način predstavitve povezav med elementi v proizvodnem sistemu precej bolj enostaven in razumljiv, posledično tudi ne zahteva veliko truda za novega uporabnika, da mu koncept izdelanih shem na ta način, postane razumljiv.

Naš sistem predstavlja tudi zelo enostavno rešitev v primeru razširitve proizvodnega sistema, saj ni potrebno veliko sprememb v dokumentaciji. Enostavno le dodamo nov element ter določimo povezavo, s katerim elementov je povezan. Sistem bo sam upošteval hierarhijo vstavljenega elementa ter logične povezave s katerimi je povezan.

3.2 Nekatere značilnosti koncepta našega programa za dokumentacijo procesnih sistemov

V tem poglavju bomo le na kratko opisali nekatere vidike koncepta programa za dokumentacijo procesnih krmilnih sistemov koncepta, kot je pomembno za razumevanje vloge modulov, ki smo jih razvili. Več o konceptu programa najdemo v [1].

Osnovni koncept je naslednji:

- dokumentirani sistem enovito omogoča opis množice različnih tipov elementov, ki nastopajo v nadzorno-krmilnem sistemu nekega procesa, na primer:
    - nadzorni sistem, ki teče na PC računalniku,
    - krmilnik PLK,
    - priključna sponka PLK,
    - priključek na sponki,
    - programski modul,
    - podatkovna baza,
    - tabela podatkovne baze,
    - spremenljivka (kateremkoli programske komponente),
    - kineta, po kateri so speljani kabli,
    - vodniki in kabli, ki povezujejo elemente,
    - priključne omarice,
    - komunikacijski protokol,
    - naprava ali stroj,
    - ugnezdeni krmilnik stroja,
    - senzor,
    - aktuator,
    - in tako naprej.

  Po našem konceptu torej lahko preskočimo od opisovanja programskega modula na opis pnevmatskega ventila, ne da bi pri tem morali pognati nek drugi program.

- homogena zgradba, ne glede na tip opisanega elementa,
- podatkovna struktura, ki odraža stanje nadzorno-krmilnega sistema in procesa je homogena, ne glede na tip elementov.

To dosežemo z naslednjo zgradbo:

- osnovni gradnik je sistem; ta ima:

- imenovane lastnosti; vrednost lastnosti je enojna vrednost, ali polje vrednosti; vrednost je lahko osnovna vrednost ali kazalec,
- lahko vključuje podsisteme, ki imajo isto zgradbo kot sistem (rekurzivni opis),
- povezave z drugimi sistemi (ki ne morajo biti istega tipa).

Z dodeljevanjem imenovanih lastnosti določimo tip elementa. Na primer, če želimo opisati krmilnik PLK, lahko vnesemo:

Ime lastnosti	Vrednost
Tip gradnika	krmilnik_PLK
Proizvajalec	Mitsubishi
Tip	Q2ASH-CPU
…	…
…	…

ali

Ime lastnosti	Vrednost
Tip gradnika	spremenljivka_procesne_baze
Tip baze	iFix_real_time_base
Tip spremenljivke	analogna
…	…
…	…

ali

Ime lastnosti	Vrednost
Tip gradnika	priključna omarica
Proizvajalec	Rittal
Tip	RT-23h-8
…	…
Slike znotraj sistema	[polje kazalcev na slike priključne omarice v procesnem okolju]
…	…
…	…

Pri tem je zelo pomembno, da smo konsistentni. Na primer, za krmilnike PLK izberemo gradnik, ki mu damo isti nabor imenovanih lastnosti. Pri tem bo prva lastnost vedno »Tip gradnika« z vrednostjo »krmilnik_PLK«, ostali nabor imenovanih lastnosti bo isti, vendar bo imel (verjetno) različne vrednosti.

Če želimo, lahko opišemo posamezne module, vključene v krmilnik PLK, na primer modul CPU, vhodno-izhodne module, komunikacijske module in podobno. To bomo naredili tako:

- vsakega izmed teh modulov bomo opisali kot ločen sistem
- vključili jih bomo kot podsisteme sistema za nek konkretni krmilnik PLK

Nek sistem (ki je logično lahko tudi podsistem, če smo ga postavili znotraj nekega drugega sistema) lahko povežemo z nekim drugim sistemom (oz. podsistemom).

Povezava je lahko poljubno natančna. Lahko na primer opišemo, da je en krmilnik PLK povezan z drugim krmilnikom PLK. Prav tako lahko opišemo, da je nek modul nekega krmilnika PLK (ali tudi nek priključek nekega modula nekega krmilnika PLK) povezan z nekim senzorjem v krmiljenem procesu.

Ni nujno, da povezujemo isti tip elementov. Če nas zanima samo razvoj nadzornega sistema, lahko na primer spremenljivko podatkovne baze nadzornega sistema povežemo direktno z nekim senzorjem. Čeprav gre v tem primeru dejanska povezava preko večjega števila vmesnih računalnikov, ta za naš trenutni namen ni pomembna, zato natančen opis izpustimo.

Opisovanje tako gradnikov kot povezav med njimi je torej »skalabilno«, lahko opisujemo tako natančno, kot je pomembno za naš namen kot je razvidno tudi iz slike na strani 21 zgoraj.

V nadaljevanju opisujemo naše delo na programskem modulu, ki omogoča osnovni opis zgradbe sistemov in podsistemov in povezav med njimi. V naslednjem poglavju opisujemo značilnosti osnovnih uporabljenih orodij, to je jezika VBA in orodja Graphviz. Nato opisujemo module, ki smo jih za omenjene naloge razvili s temi orodji.

4 Opis osnovnih uporabljenih orodij

V tem poglavju opisujemo osnovna orodja, uporabljena za razvoj vmesniških gradnikov. To so Visual Basic for Applications (VBA) in Drawing layer, ki nastopata v programih zbirke Microsoft Office in odprtokodni program Graphviz. Program, napisan v VBA se prenese na drug računalnik enostavno s prenosom gostiteljske datoteke, na primer Excel datoteke.

Pomemben razlog za uporabo VBA je tudi ta, da ima v okviru gostiteljskih programov, na primer zbirke MS Office na MS Windows operacijskem sistemu že vgrajenih in dostopnih veliko uporabnih funkcij, kot je na primer risalno orodje, ki nam omogoča prikaz raznih grafičnih elementov in povezav našega sistema.

4.1 Značilnosti jezika Visual Basic for Application in komponente Drawing Layer

4.1.1 Osnovne značilnosti orodja Visual Basic for Application

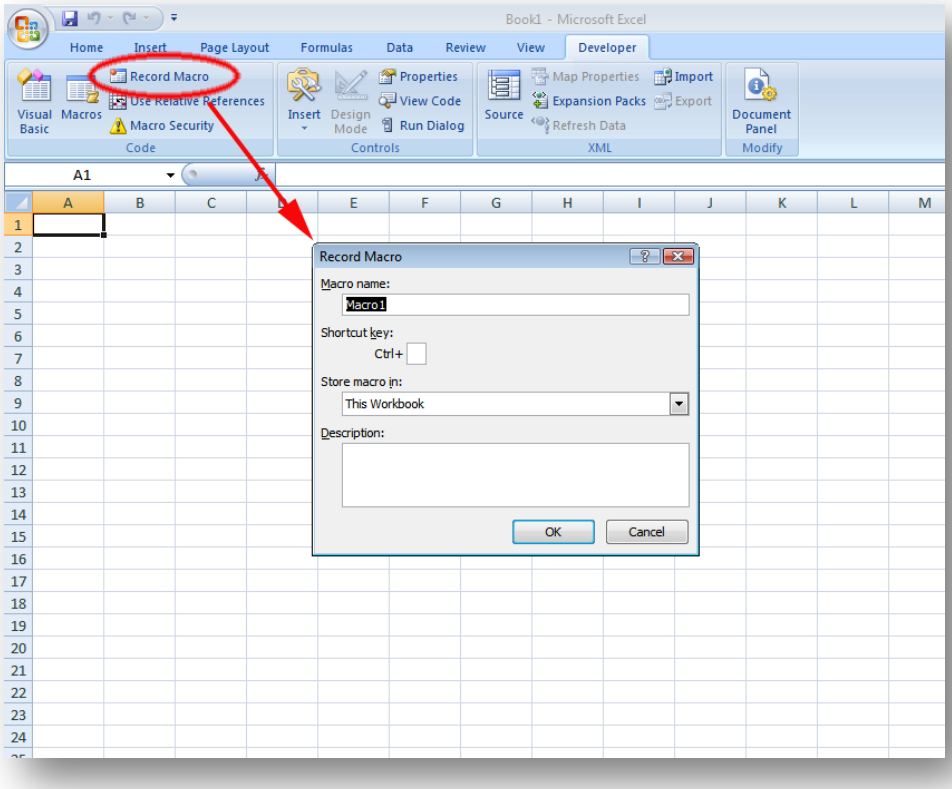
VBA (Visual basic for application) je posebna izvedba programskega jezika Microsoft Visual Basic z že vsebovanim razvojnim orodjem (angleško Integrated development environment, IDE). Ker se koda VBA lahko izvaja le znotraj gostiteljske aplikacije, pomeni, da ni možno izdelati aplikacije, ki bi delovala samostojno brez programa gostitelja. VBA je vključen v večino aplikacij programskega paketa Microsoft Office. Najdemo ga lahko tudi v nekaterih drugih programskih paketih, kot so na primer Microsoft MapPoint, Microsoft Visio, AutoCAD, WordPerfect in CorelDraw. Z njim lahko izvedemo skoraj vse operacije, ki so na voljo v gostiteljevi aplikaciji, zato je glavni namen tega jezika olajšati izvajanje zamudnih operacij, avtomatizirati izvajanje izbranih dogodkov (akcij), ki se zaporedno ponavljajo ter razširjajo funkcionalnosti v gostiteljevi aplikaciji. S pomočjo VBA jezika je možno tudi spreminjati vsebino menijev in orodnih vrstic po naših potrebah ter dodajanje lastnih uporabniških oken.

Ker VBA vsebuje tudi OLE (predmetno povezovanje in vdelovanje; OLE, angl. object linking & embedding) avtomatizacijo, pomeni, da lahko izvajamo in nadzorujemo neko drugo aplikacijo, ki ni del gostiteljeve aplikacije. Na ta način lahko na primer samodejno izdelamo poročilo v Wordu iz podatkov, ki so zapisani v Excelu, medtem ko se naša aplikacija nahaja v Excelu [22].

Za pisanje našega sistema v programskem paketu Microsoft Office s pomočjo programskega jezika VBA, smo se odločili zato, ker zbirko Microsoft Office danes najdemo nameščeno že na številnih računalnikih, kar olajša prenos našega sistema z enega na drug računalnik, saj je celotni sistem zajet v navadni »doc« oziroma »xls« datoteki. Drugi razlog je ta, da ima programski paket Microsoft Office implementiranih veliko uporabnih funkcij, ki jih lahko enostavno uporabljamo v naši aplikaciji s pomočjo jezika VBA. Najbolj pomembna funkcija za nas je, da Microsoft Office vsebuje risalno plast (angleško Drawing layer), ki nam omogoča grafičen prikaz raznih elementov in povezav v našem sistemu.

Procedura ali funkcija zapisana v VBA jeziku se imenujejo makro. Makre lahko shranjujemo in jih lahko prenašamo v ostale dokumente paketa Office, v katerih želimo opraviti zapisane operacije. Lahko pa enostavno posnamemo nek dogodek s funkcijo »posnemi makro«

(angleško record macro), ki se nahaja v skoraj vseh programih v paketu Microsoft Office. V razvojnem okolju VBA se nam nato samodejno prikaže nova procedura z imenom »makro 1«, ki vsebuje vse dogodke, ki smo jih naredili med snemanjem makra. Takšni makri predstavljajo veliko prednost tudi za povsem običajnega uporabnika programskega paketa Office, ki ne pozna programske kode, niti ne ve kam se shranjuje, želi pa si enostavno ponoviti sicer dolgotrajne in ponavljajoče se postopke.



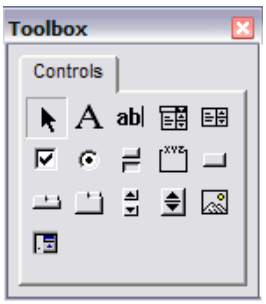
Slika 12: Okno, ki se nam prikaže pred začetkom snemanja makra.

Jezik VBA primarno ni namenjen za pisanje obsežnega programskega sistema, zato se včasih zgodi da naletimo na kakšno oviro. Eno izmed ovir smo opisali tudi v poglavju »Težave pri doseganju nekaterih funkcionalnosti na prehodu na novejša orodja«

V tem poglavju opisujemo najprej težave, ki so se pojavile pri interakciji našega programa z dogodkovnim sistemom operacijskega sistema Windows ter način reševanja. Nato podajamo še potrebne spremembe zaradi prehoda na novejšo zbirko z uporabljenim orodjem.

4.1.2 Razvojno okolje za uporabo orodja VBA

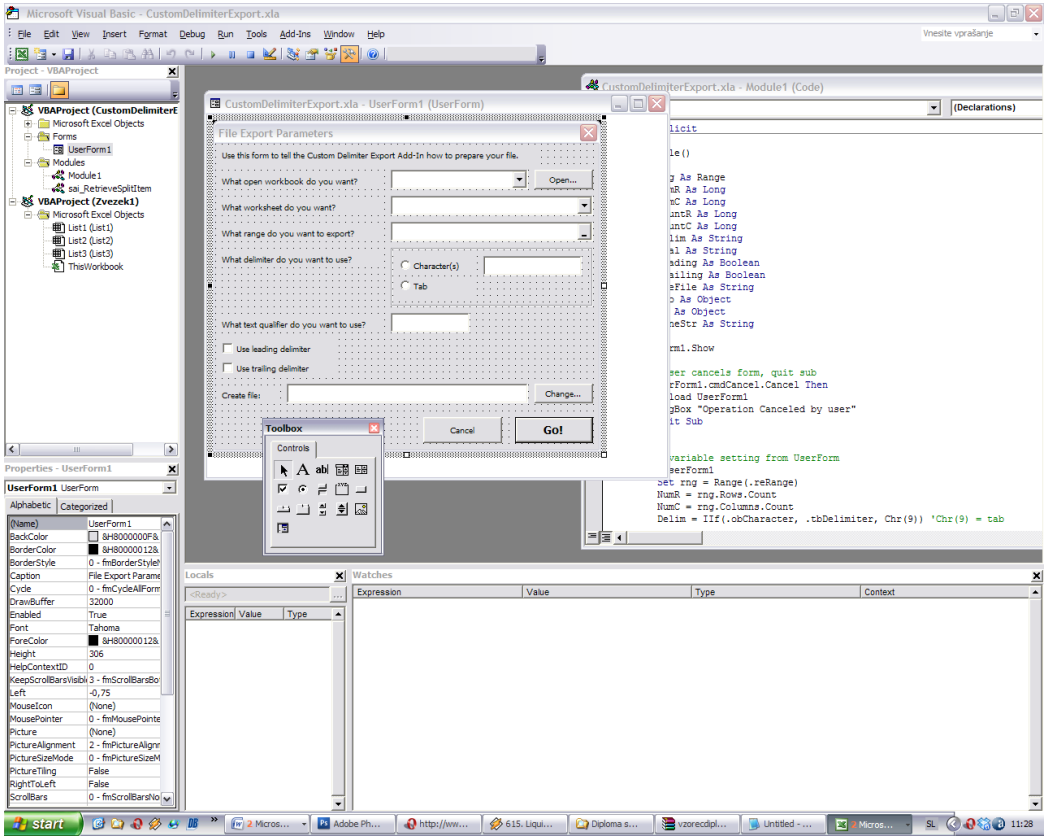
Kompleksnost izdelave programov v jeziku VBA lahko precej povečamo, saj v posebnem urejevalniku, do katerega lahko dostopamo v Wordu, Excelu ali Accessu preko menija ali z kombinacijo tipk <alt> in <F11>, dodajamo tudi uporabniška okna ter vse ostale pomembnejše kontrole, kot so »DropBox«, »ComboBox«, »CheckBox«, »ToggleButton«, »CommandButton«, »TabStrip«, »MultiPage«, »ScrollBar«, »SpinButton«, »Image«, »RefEdit« ter »Label«. V kolikor nam osnovne kontrole ne zadoščajo, obstaja tudi možnost dodajanja novih z vključevanjem raznih knjižnic, ki so del drugega razvojnega okolja.



Slika 13: Okno z osnovnimi orodji, ki jih lahko uporabljamo v razvojnem orodju VBA.

Razvojno okolje VBA sestavlja več oken, izmed katerih ima vsak svoj namen:

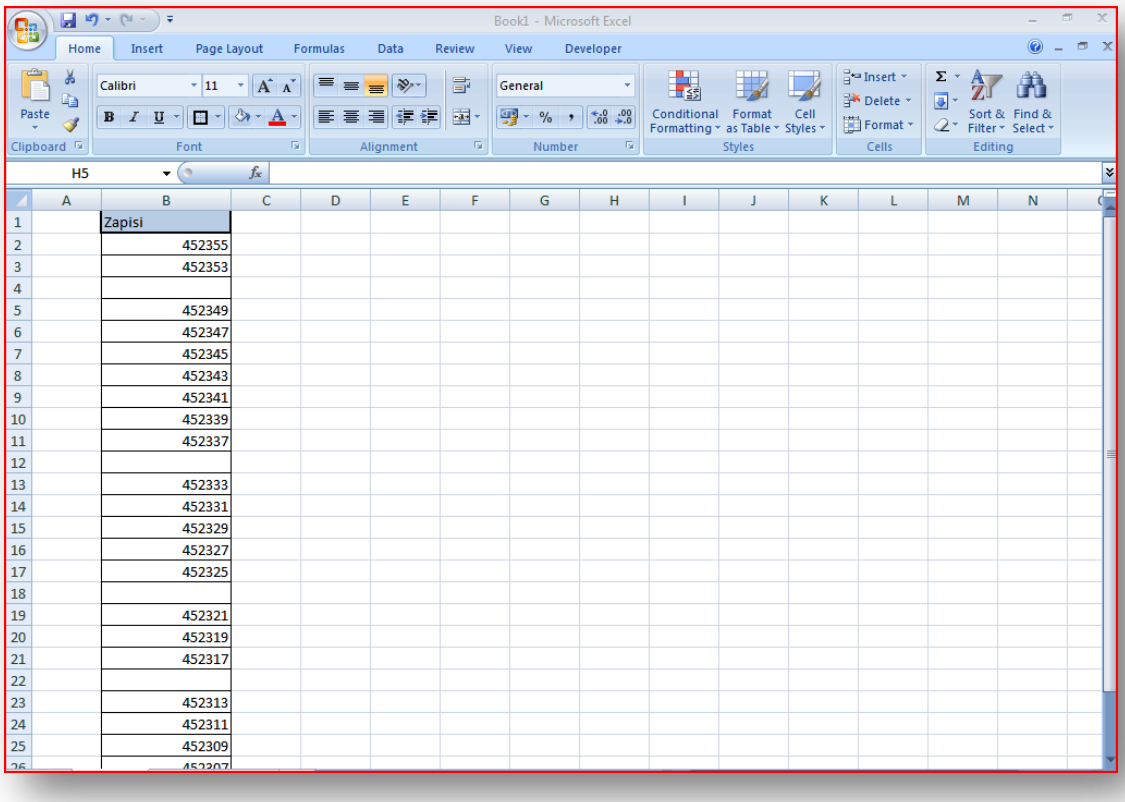
- *okno »Project Explorer«* predstavlja seznam objektov, ki so zajeti v aplikaciji. V tem seznamu najdemo Excelove delovne zvezke, delovne liste, uporabniška okna ter module, kamor običajno shranjujemo kodo VBA,
- *okno lastnosti* (angleško properties) za posamezni objekt prikazuje njegove lastnosti, kjer jih lahko po želji tudi spreminjamo.
- *okna »Immediate Window«, »Locals« in »Watches Window«* so namenjna testiranju novo napisane kode oziramo napak v samem programu,
- *objektni raziskovalec* (angleško Project Explorer) omogoča prikaz vseh razredov, lastnosti, metod, dogodkov in konstant, ki so dosegljive v objektni knjižnici in procedurah objekta.



Slika 14: Prikaz vseh možnih oken, ki jih lahko zasledimo v uporabniškem vmesniku razvojnega orodja VBA.

4.1.3 Primer uporabe makra

Vzemimo za primer, da imamo Excelov dokument z nekaj deset tisoč zapisi, izmed katerih pa je nekaj tisoč zapisov praznih. Nato želimo te prazne vrstice odstraniti.



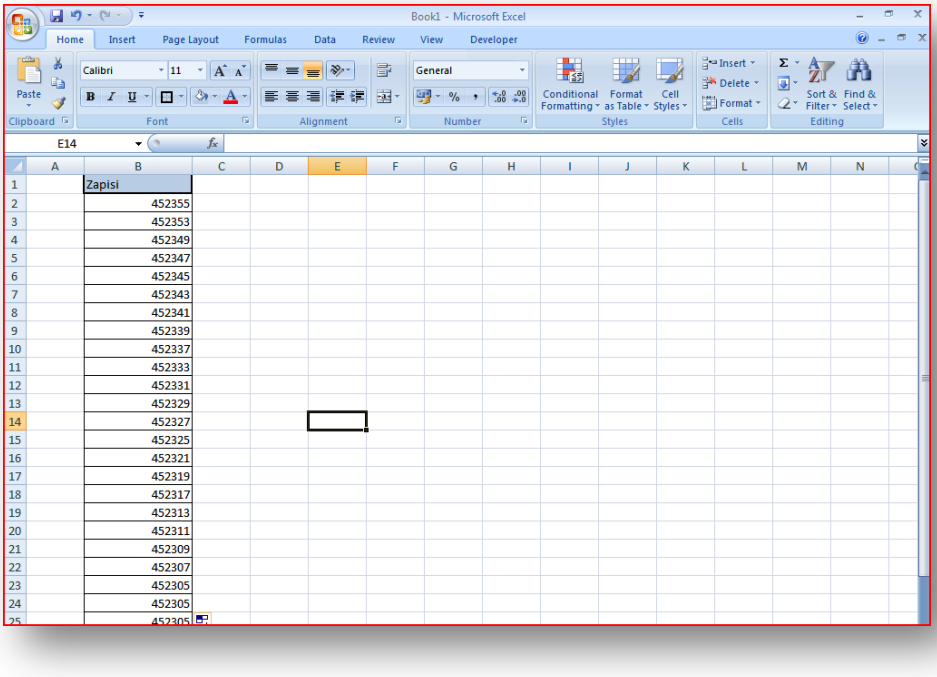
Slika 15: Excel z različnimi zapisi, pred uporabo makra za brisanje praznih vrstic

Z nekaj vrsticami spodnje kode, lahko sicer zelo dolgotrajen postopek odstranjevanja praznih vrstic, skrajšamo le na čas, ki je bil potreben za zapis kode ter na čas, ki je potreben za izvedbo te kode.

```
Public Sub DeleteBlankRows()
    Dim dbMaxRow As Double, dbMinRow As Double, i As Double
    Dim dbMaxCol As Double
    Dim rng As Range

    On Error Resume Next
    Set rng = Selection.Parent.UsedRange
    dbMaxRow = rng.Rows.Count
    dbMinRow = rng.Cells(1, 1).Row
    dbMaxCol = rng.EntireColumn.Count
        For i = dbMaxRow To dbMinRow Step -1
            If IsError(rng.Cells(1, 1).Offset(i - 1, 0).EntireRow. _
                SpecialCells(xlCellTypeBlanks).Count) Then
            Else
                If rng.Cells(1, 1).Offset(i - 1, 0).EntireRow. _
                    SpecialCells(xlCellTypeBlanks).Count = dbMaxCol
    Then
```

```
                    rng.Cells(1, 1).Offset(i - 1, 0).EntireRow.Delete
                End If
            End If
        Next i
    Set rng = Nothing
End Sub
```



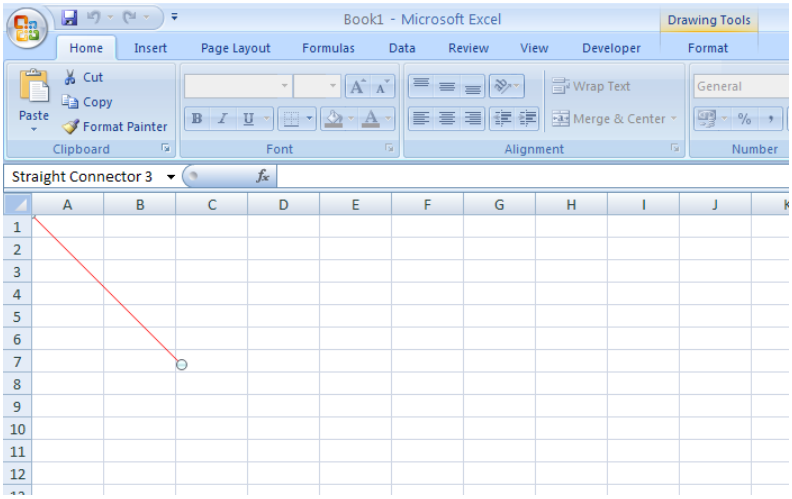
Slika 16: Excel z različnimi zapisani po uporabi makra za brisanje praznih vrstic

Z makri lahko simuliramo skoraj vse operacije, ki jih je mogoče izvajati v večini programov v programskem paketu Office. V našem primeru uporabljamo risalno orodje v Excelu, saj ima izdelane že vse komponente, ki jih potrebujemo v našem sistemu. Zato lahko brez težav dodajamo izbrane grafične elemente na delovni list Excela ter jim spreminjamo velikost, pozicijo, prosojnost, barvo, opis, itd., oziroma lahko preberemo le lastnosti že dodanih elementov na delovnem listu Excela.

Preprost primer kako deluje uporaba makra na risalnem orodju, prikazuje naslednji primer.

Spodnja koda nam doda na Excelov odprti delovni list črto rdeče barve s točkama oglišč (0,0) ter (100,100). Lastnosti koordinatnega sistema v Microsoft Excelu so opisane v prilogi B.

```
Sub dodajRdecoCrto()
    Worksheets.Item(1).Shapes.AddLine(0, 0, 100, 100).Select
    ActiveWindow.Selection.ShapeRange.Line.ForeColor.RGB = RGB(255, 0,
    0)
End Sub
```



Slika 17: Dodana rdeča črta na delovnem zvezku Excela po uporabi makra

Podrobnejšo razlago uporabe grafičnih elementov na risalni plasti ter samo risalno plast smo opisali v poglavju »Risalna plast – »DrawingLayer«« na strani 31.

4.1.4 Objektni model programskega paketa Office in hierarhija objektov

Microsoftov objektni model vsebuje večje število objektov, katere lahko upravljamo preko kode VBA. Skoraj vse funkcionalnosti, ki jih lahko dosežemo preko grafičnega vmesnika v izbranem orodju paketa Office, lahko dosežemo tudi z uporabo teh objektov preko kode VBA. Lahko pa seveda uvedemo še nove funkcionalnosti, ki jih paket Office ne vsebuje, za nas bi bile pa mogoče zelo uporabne.

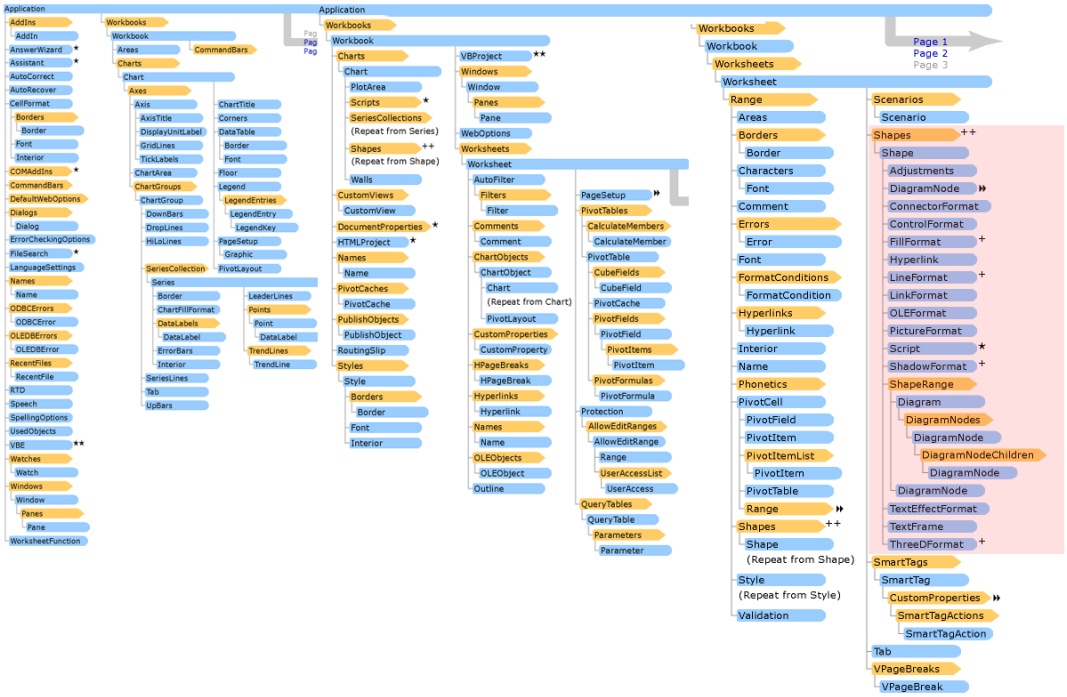
Vsako orodje v paketu Office ima svojo zgradbo objektnega modela, vendar so si precej podobni med seboj. Tako na primer Excel vsebuje objekt »Workbook«, ki ga v Wordu ne bomo našli. Namesto tega Word vsebuje objekt »Document«. To posledično pomeni, da se večina objektov, ki so izpeljani iz različnih glavnih objektov, razlikuje. Objekt »Shapes«, ki ga mi najbolj uporabljamo v našem sistemu je pa v večini orodij Office, zelo podoben, zato tudi prenos našega sistema v drugo okolje paketa Office ne bi bil težaven, izgubili bi le del funkcionalnosti. Težavo prenosa našega sistema iz Excel-a v Word smo opisali v prilogi B.

4.1.4.1 Objektna zgradba Excela

Na sliki spodaj je predstavljena objektna zgradba Excela s pomočjo diagrama. Aplikacija ali komponenta programabilno razširja svojo funkcionalnosti s pomočjo teh objektov. Z objekti delamo na način, da uporabljamo njihove lastnosti in metode.

Objekti so urejeni v hierarhični zgradbi. Najvišji objekt, iz katerega izhajajo vsi ostali, se običajno imenuje objekt »Application«. Ta objekt predstavlja celotno aplikacijo v kateri se nahajamo in vsi ostali objekti so del tega objekta. Skupina podobnih objektov je lahko združena

v hierarhiji kot zbirka objektov.



Slika 18: Naše področje uporabe Excelovega objektnega modela

Lastnost objekta »Application« uporabljamo za pridobitev reference do tega glavnega objekta. Ko enkrat dobimo to referenco, lahko enostavno dostopamo do vseh ostalih objektov, ki so nižje od tega glavnega objekta v hierarhični zgradbi. Sledeči primer prikazuje, kako nastaviti prvo celico na delovnem listu Excela na vrednost 20.

```
Application.Workbooks(1).Worksheets(1).Cells(1, 1) = 20
```

Da pridemo do lastnosti celice, začnemo izpeljavo pri objektu »Application«, nadaljujemo z objektom »workbook«, nato izberemo »worksheet« (delovni list), katerega uporabljamo, na koncu pridemo do celice (angleško Cells), ki ji lahko nato enostavno določamo vrednosti, uporabljamo vse funkcije računanja na njej, oblikujemo besedilo, obrobe, itd..

4.1.5 Risalna plast – »DrawingLayer«

Delovni list Excela je sestavljen iz številnih plasti. Glavni dve plasti sta besedilna plast (»text layer«) in risalna plast (»drawing layer«). Vsako od teh dveh glavnih plasti sestavljajo objekti, preko katerih lahko s pomočjo jezika VBA (z makri) tudi programsko uporabljamo vse operacije, ki so na voljo preko uporabniškega vmesnika in tudi operacije, ki jih lahko naredimo le z uporabo makrov.

Ko začnemo uporabljati Excel, uporabljamo le besedilno plast, dokler sami ne izberemo nekega objekta, ki nas nato samodejno prestavi v neko drugo plast. Na primer, če dodamo grafični element, Excel samodejno preklopi v risalno plast.

Vse plasti so prosojne, razen na mestih kjer so vstavljeni grafični elementi ter tako omogočajo pogled na celoten delovni list, kjer so združeni elementi vseh plasti skupaj. Tako uporabnik niti ne ve, ali obstaja več plasti saj jih ni mogoče videti.

Risalna plast deluje kot ena zelo »debela« plast, ki je sestavljena iz toliko podplasti, kot jih potrebujemo. Lahko izdelamo številne objekte, ki so lahko ločeni med seboj ali pa jih po želji združimo v skupine, ter jih na ta način lahko skupaj premikamo po risalni plasti, ne da bi se med seboj razdružili. Možno je tudi premikanje objektov med posameznimi plastmi. Na ta način lahko spreminjamo vrstni red objektov, ter tako določimo ali se bo nek objekt nahajal nižje oziroma višje v zbirki vseh plasti v risalni plasti. Na primer, lahko vstavimo sliko, ki se nahaja za celicami delovnega lista, v isti plasti kot same celice, ali pa pred celicami, ki nato zastira pogled do celic, ki se nahajajo pod to sliko. Če vstavimo še eno sliko, oziroma še kakšen drugi grafični element, pa lahko določamo, kateri element bo višje v risalni plasti in kateri nižje ter na ta način lahko tudi prekrivamo objekte med seboj [23][24].

Do risalne plasti dostopamo preko objektov skupine »Shapes«, ki jih lahko tudi podrobneje vidite na sliki na strani 31. Območje uporabljenih objektov za risalno plast je označeno z rdečo barvo.

Grafični elementi risalne plasti so predstavljeni s tremi različnimi glavnimi objekti:

- objekti zbirke »Shapes«, ki služi za izdelavo grafičnih elementov ter njihov pregled,
- objekti zbirke »ShapeRange«, ki omogoča spreminjanje več elementov hkrati, na enak način kot lahko to storimo preko uporabniškega vmesnika,
- objekt »Shape«, ki se uporablja za oblikovanje in spreminjanje le enega elementa.

4.1.5.1 Spreminjanje lastnosti grafičnih elementov na risalni plasti

Veliko oblikovalnih lastnosti elementov ni nastavljeno tako, da bi se nanašale neposredno na objekt »Shape« oziroma objekt »ShapeRange«. Namesto tega so te lastnosti elementa grupirane pod sekundarnimi objekti, kot je objekt »FillFormat«, ki vsebuje vse lastnosti, ki se nanašajo na polnila elementov [25].

Spodnji primer prikazuje, kako elementu na prvem delovnem listu, z zaporedno številko ena, spremenimo polnilo na rdečo barvo:

```
Worksheets(1).Shapes(1).Fill.ForeColor.RGB = RGB(255, 0, 0)
```

Objekt »Shape« vsebuje metode, ki omogočajo dodajanje novih elementov oziroma le del elementov na risalno plasti. Te metode so:

»AddCallout«, »AddConnector«, »AddCurve«, »AddLabel«, »AddLine«, »AddOLEObject«, »AddPolyline«, »AddShape«, »AddTextbox« in »AddTextEffect«.

Grafični elementi, ki vsebujejo objekt »ShapeNodes«

Obstaja tudi posebna skupina objektov, ki jim ne moremo prirejati objekt »FillFormat«, niti ne moremo dodajati besedilo. Sem spadajo na primer element črta (angleško line) in prostoročni element (angleško freeform). Tem elementom lahko prirejamo objekt »ShapeNodes«, katerega pri ostalih elementih ne moremo.

»ShapeNode« predstavlja zbirko objektov v določenem prostoročnem elementu kot sta na primer krivulja in večtočkovna črta. Vsak element, ki podpira tip »ShapeNode«, vsebuje vozlišče, ki je med dvema segmentoma v elementu, lahko pa predstavlja tudi kontrolno točko v zavitem segmentu prosto oblikovnega elementa, kot je krivulja.

Slika 19: Dve kontrolni točki v krivulji

.

Spodnji primer prikazuje kako iz zgornjega grafičnega elementa izbrišemo zbrišemo tretje vozlišče – kontrolno točko.

```
Set myDocument = Worksheets(1)
myDocument.Shapes(1).Nodes.Delete 3
```

Slika 20: Krivulja po odstranitvi ene kontrolne točke

Kot vidimo, se ukrivljenost krivulje spremeni, saj ne obstaja več vmesne kontrolne točke, ki bi določala, do kamor bi moral potekati prvi del krivulje. Tako se krivulja prilagodi tako, da se poveže na naslednjo najbližjo kontrolno točko.

Spreminjanje več grafičnih elementov na enkrat

V uporabniškem vmesniku obstajajo operacije, ki lahko spremenijo lastnosti več grafičnim elementom na enkrat. Na primer, lahko označimo več elementov in vsem na enkrat spremenimo polnilo na izbrano barvo. Obstajajo pa tudi operacije, ki jih lahko uporabljamo le na enem elementu na enkrat. Takšen primer je recimo urejanje besedila v grafičnem elementu.

To omejitev lahko premagamo z izdelavo makra v jeziku VBA, saj lahko uporabimo zanko, ki vsakemu elementu posebej doda oziroma uredi besedilo, ki se nahaja v njem.

```
Set myDocument = Worksheets(1)
For Each sh In myDocument.Shapes
    If sh.Type = msoAutoShape Then
         Sh.select
         Selection.Characters.Text = strBesedilo
    End If
Next
```

4.1.6 Upravljanje Excela iz druge aplikacije s pomočjo OLE avtomatizacije

VBA vsebuje tudi OLE avtomatizacijo, ki omogoča, da lahko izvajamo in nadzorujemo neko drugo aplikacijo, ki ni del gostiteljeve aplikacije. Na ta način lahko na primer samodejno izdelamo poročilo v Wordu iz podatkov, ki so zapisani v Excelu, medtem ko se naša aplikacija nahaja v Excelu [22].

Sledeči primer prikazuje, kako v neki drugi aplikaciji lahko ustvarimo Excelov objekt »workbook«, ki nato odpre nov delovni list v Excelu.

```
Set x = CreateObject("Excel.Sheet")
```

Preko objekta »x« lahko nato dostopamo do vseh ostalih objektov Excela, kar nam omogoča uporabo vseh njegovih funkcij in posledično postane tudi programiranje povsem enako kot, če bi program pisali v razvojnem okolju VBA, ki se nahaja v Excelu.

4.1.6.1 Implicitna in eksplicitna referenca objekta

Ko delamo z lastnostmi in metodami naše aplikacije z uporabo VBA-ja znotraj Excela, nam je objekt »Application« že samodejno navoljo. To se imenuje *implicitna referenca* objekta. V primeru, če delamo z Excelom iz druge aplikacije, moramo ustvariti objekt, ki predstavlja Excelov objekt »Application«. To se imenuje eksplicitna referenca objekta.

Na primer, sledeči proceduri vračata ime trenutno aktivnega delovnega lista v Excelu. Procedura »ShowNameFromInsideXL« je napisana tako, da deluje znotraj Excela in uporablja implicitno referenco objekta »Application«, medtem ko je procedura »ShowNameFromOutsideXL« napisana tako, da lahko teče zunaj Excela, zato vsebuje *eksplicitno referenco* objekta »Application«.

Procedura »ShowNameFromInsideXL()«, napisana na način, da lahko deluje znotraj Excela:

```
Sub ShowNameFromInsideXL()
   MsgBox "'" & ActiveSheet.Name & "' is the currently active worksheet."
End Sub
```

Procedura »ShowNameFromOutsideXL()«, napisana tako, da teče zunaj Excela:

```
Sub ShowNameFromOutsideXL()
    Dim xlApp As Excel.Application

    Const XL_NOTRUNNING As Long = 429

    On Error GoTo ShowName_Err
    Set xlApp = GetObject(, "Excel.Application")
    MsgBox "'" & ActiveSheet.Name & "' Trenutno aktiven delovni list."
    xlApp.Quit
    Set xlApp = Nothing

ShowName_End:
    Exit Sub
ShowName_Err:
    If Err = XL_NOTRUNNING Then
        ' Excel is not currently running.
    Set xlApp = New Excel.Application
    xlApp.Workbooks.Add
    Resume Next
    Else
        MsgBox Err.Number & " - " & Err.Description
    End If
    Resume ShowName_End
End Sub
```

4.2 Orodje za delo z grafi Graphviz

V tem poglavju opisujemo Graphviz (Graph Visualization Software), ki je paket odprtokodnih orodij za risanje grafov. Sestavljajo ga opisni format grafa, ki se imenuje format »dot«, in set orodij, ki izdelajo ali procesirajo datoteke zapisane v formatu »dot«[26][27].

Podatke o grafu zapišemo v tekstovni datoteki, ki mora biti napisana v formatu »dot«. V tej datoteki definiramo vozlišča ter povezave med vozlišči. Ko Graphviz prejme te podatke, sam določi pozicijo vozlišč in povezav med njimi. Na kakšen način razporeja vozlišča in povezave med njimi, je odvisno od orodja, ki ga uporabimo. Izbrano orodje nam vrne rezultat v grafični ali tekstovni obliki.

Grafično obliko lahko vrne v več različnih formatih. Podprta je večina slikovnih formatov, »postScript« ter format »pdf«. Med slikovnimi formati je podprt tudi vektorski format »svg«, ki se uporablja za spletno grafiko. Odločili smo se, da bomo uporabljali format »png« za grafični izhod, saj datoteka zasede malo prostora in tudi v Excel ga lahko brez težav uvozimo, za tekstovni izhod pa bomo uporabljali format »dot«.

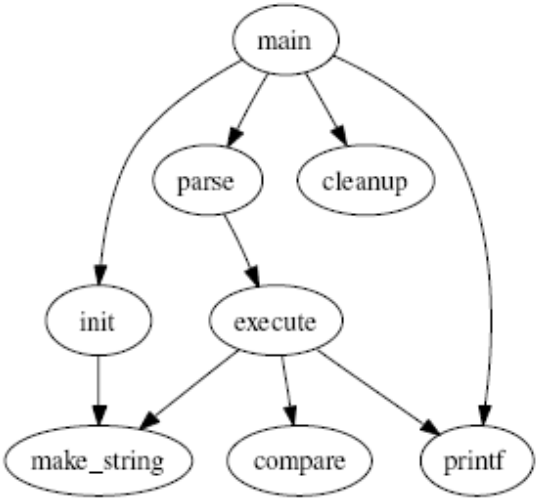
Graphviz vsebuje tudi knjižnico funkcij, kar omogoča, da ga lahko uporabljamo kar iz naših lastnih aplikacij, ne da bi pri tem uporabnik vedel, da za izris grafa skrbi povsem samostojno orodje.

4.2.1 Osnovna zgradba programskega paketa Graphviz

Graphviz je sestavljeno iz več različnih orodij, katero ima vsaka svoje posebne lastnosti in tudi specifičnen izris grafa[28][29]. Na voljo imamo naslednja orodja:

- Dot,
- Neato,
- Twopi,
- Circo,
- Fdp.

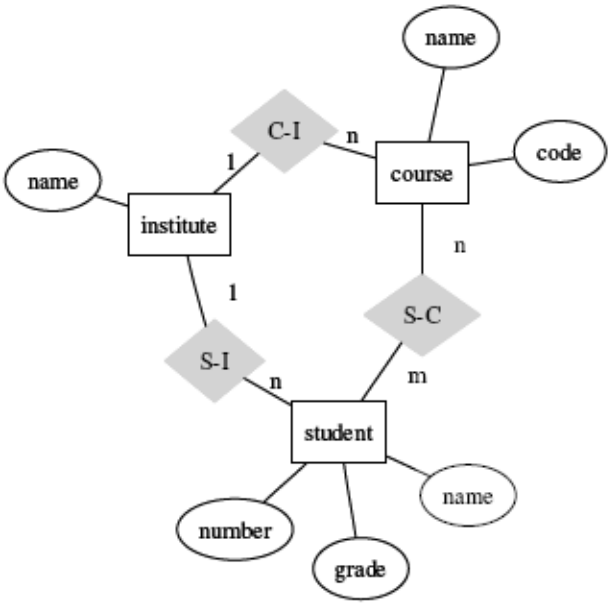
Orodje »Dot« omogoča risanje usmerjenih hierarhičnih grafov. To orodje usmerja vse robove vozlišč v isto smer (od zgoraj navzdol oziroma iz leve proti desni) in pri tem poskuša, da ne bi prišlo do prekrivanja robov[30].



Slika 21: Graf izdelan z orodjem »Dot«

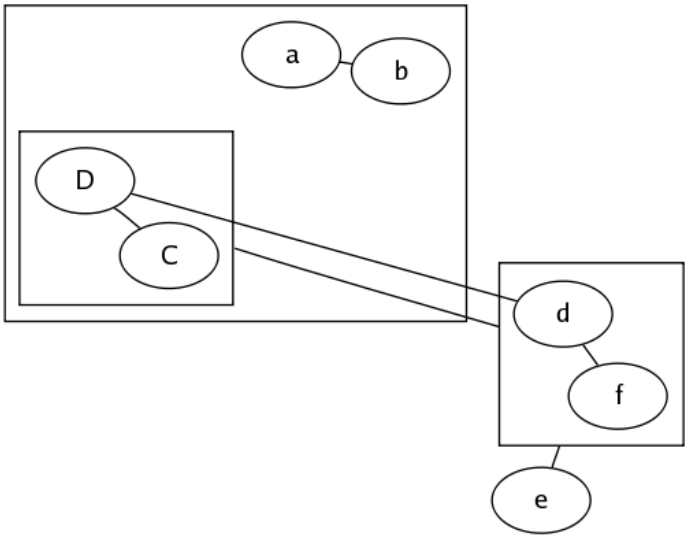
Orodje »Neato« je namenjen risanju neusmerjenih grafov, ki se običajno uporabljajo v telekomunikacijah in v programiranju. Postopek risanja grafa s tem orodjem se začne z gradnjo virtualnega fizičnega modela, nato pa po njem algoritem iterativno išče nizko-energijsko konfiguracijo[31].

Nizko-energijska konfiguracija pomeni, da je med vsak par vozlišč postavljena navidezna vzmet. Njena dolžina je nastavljena na najkrajšo razdaljo med obema vozliščema. Vzmeti potisnejo vozlišča tako, da se geometrijske razdalje v grafu med vsemi vozlišči enakomerno ujemajo.



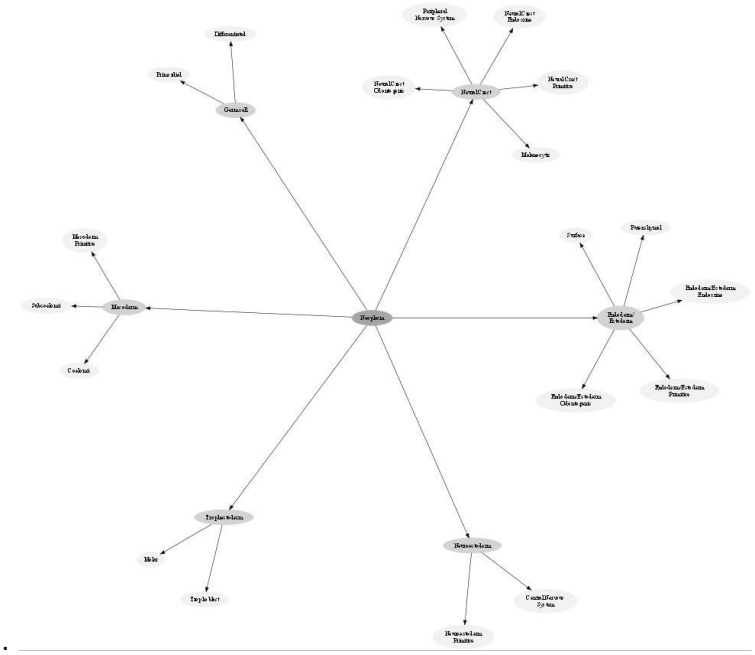
Slika 22: Graf izdelan z orodjem »Neato«

Orodje »Fdp« je soroden orodju »Neato« in je namenjen obsežnim grafom in gnezdenju neusmerjenih grafov. Razlika med tema dvema orodjema se razlikuje po tem, da orodje »Fdp« odlikuje možnost dodajanja podgrafov ter da podpira tudi usmerjene grafe.



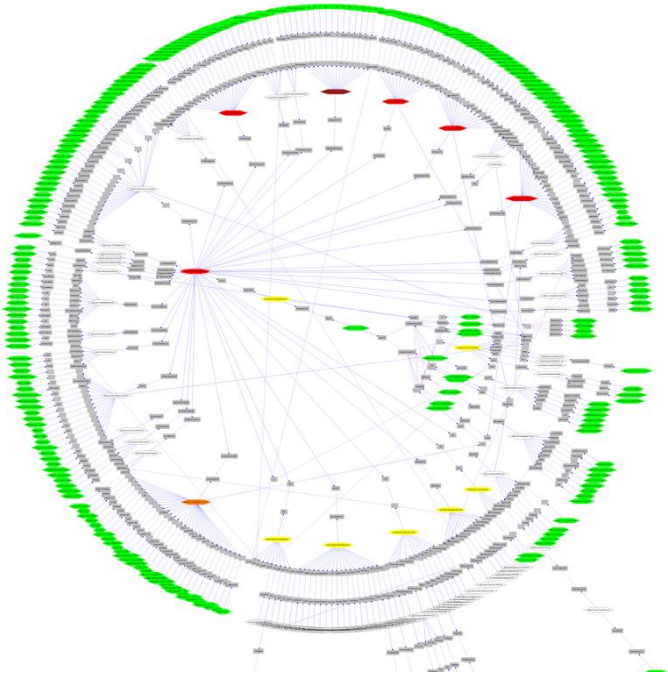
Slika 23: Graf izdelan z orodjem »Fdp«

Orodje »Circo« je primeren za diagrame, ki vsebujejo več cikličnih struktur, kot je na primer prikaz telekomunikacijskega omrežja.



Slika 24: Graf izdelan z orodjem Circo

Orodje »Twopi« prikaže graf v krožni obliki. Vozlišča so nameščena na koncentričnem krogu v različnih razdaljah od središča, ki je odvisna od tega, kolikšna je razdalja med korenskim vozliščem.



Slika 25: Graf izdelan z orodjem »Twopi«

V našem sistemu smo se sprva osredotočili na orodje »Dot«, vendar smo kmalu ugotovili njegovo pomanjkljivost, saj nima možnosti dodajanja podgrafov. Ker pa mi izrisujemo elemente tako, da ti elementi vsebujejo tudi podelemente in podelementi vsebujejo spet svoje podelemente in tako dalje, je možnost risanja podgrafov za nas operacija, ki jo mora naš sistem

nujno vsebovati. Zaradi tega razloga smo se odločili za orodje »fdp«, katero pa nima težav s tem in tudi najbolj ustreza predstavitvi naših elementov v proizvodnem sistemu.

4.2.2 Prikaz delovanja orodja Graphviz

Paket orodij Graphviz izriše graf na podlagi vhodnih podatkov, ki morajo biti zapisani v formatu »dot«. Te podatke mu pošljemo s pomočjo tekstovne datoteke. Nato izbrano orodje v Graphvizu procesira podano vhodno datoteko in nam vrne grafično in tekstovno obliko rezultata.

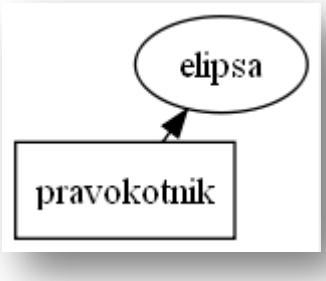
Grafična oblika je v našem primeru potrebna za vizualno predstavo dela oziroma celote nekega proizvodnega sistema, katerega predstavimo uporabniku našega sistema. Tekstovno obliko pa potrebujemo, da lahko na podlagi podatkov, ki so zapisani v njej, razberemo tudi pozicijo in velikost elementov na sliki, ki jo izdela orodje »fdp«.

Za primer vhodne datoteke vzemimo spodnjo vsebino, ki jo pošljemo v Graphviz. Vsebina predstavlja definiranje glavnih lastnosti grafa.

```
digraph graf1
{
13685 [shape="box", label=pravokotnik]
13686 [shape="oval", label=elipsa]
13685 -> 13686
}
```

»Digraph« pomeni, naj izriše usmerjeni graf. Nato sledijo definicije dveh vozlišč, katerima lahko določimo obliko in dodamo besedilo. Drugi del definicije grafa predstavlja seznam povezav med vozlišči. V spodnjem primeru je dodan ukaz, ki vozlišče z oznako 13685 poveže z vozliščem 13686. Simbola »->« in »<-« kažeta smer proti kateremu vozlišču naj bo usmerjena povezava. Na voljo je še simbol »--« , ki predstavlja neusmerjeno povezavo.

Nato v Graphvizu procesiramo podano vhodno datoteko in vrne nam grafično in tekstovno obliko rezultata. Kot grafični izhod dobimo naslednjo sliko:



Slika 26: Grafični rezultat procesiranja vhodne datoteke

Tekstovni rezultat procesiranja vhodne datoteke:

```
digraph graf1 {
node [label="\N"];
graph [bb="0,0,111,85"];
```

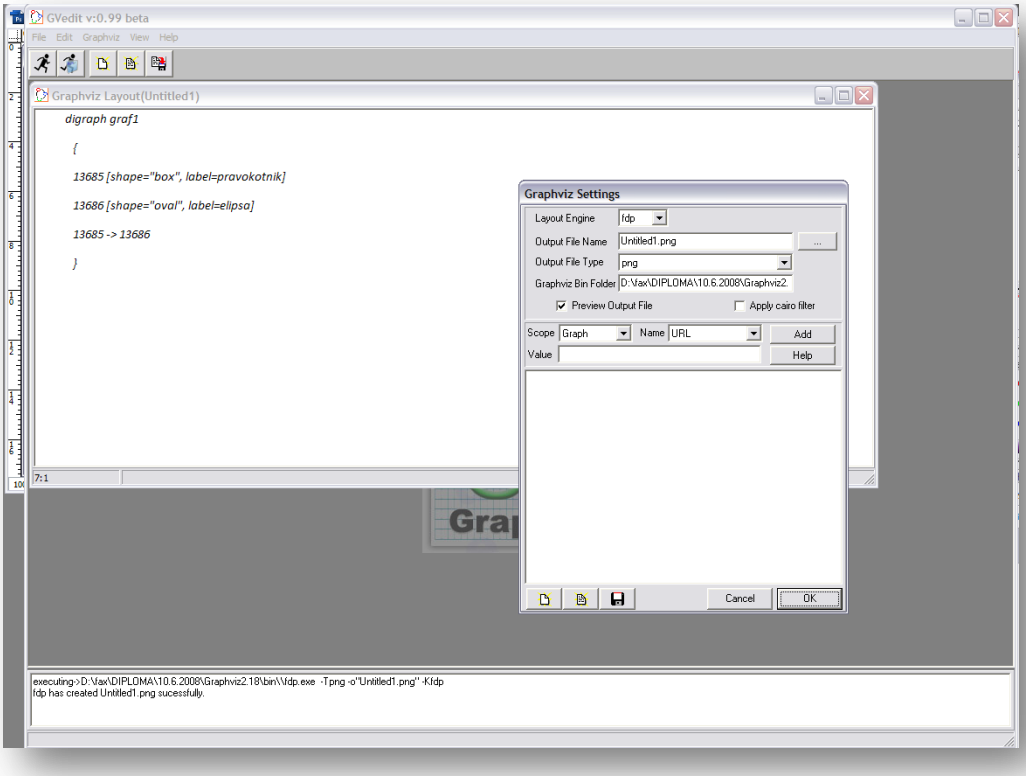
```
13685 [label=pravokotnik, shape=box, pos="42,19",
width="1.14",height="0.50"];
13686 [label=elipsa, shape=oval, pos="78,66", width="0.89",
height="0.50"];
13685 -> 13686 [pos="e,65,49 56,37 57,38 58,40 59,41"];
}
```

Kot vidimo iz zgornjega primera, ostane zapis po izdelavi tekstovne izhodne datoteke precej podoben v primerjavi z vhodno tekstovno datoteko, vse kar je spremenjeno, je to, da Graphviz doda pozicijo in velikost vsakega elementa, kot so predstavljeni na grafu izdelane slike. Ravno s temi podatki, lahko natančno določimo pozicijo naših elementov na odprtem dokumentu v Excelu, ki nato uporabniku omogoča interakcijo z grafom.

Na primer, uporabniku omogoča klik na izdelano sliko, naš program pa sam razbere kateri element je uporabnik izbral. Na podlagi uporabnikove izbire program ustrezno reagira.

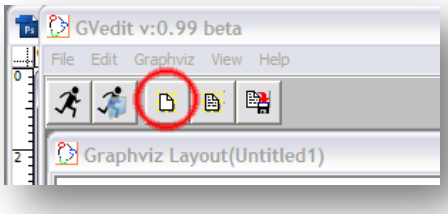
Za preizkušanje izrisa grafov z različnimi orodji ima Graphviz vgrajen tudi grafični uporabniški vmesnik »GVedt«.

»GVedit« je orodje v Graphvizu, ki je namenjeno ustvarjanju, pregledovanju, urejanju ter procesiranju »dot« datotek. V njem lahko v posebnem oknu dodamo opis grafa v jeziku »dot«, izberemo orodje, ki ga želimo uporabiti za izris grafa ter na koncu izberemo še želeni izhodni format grafa.

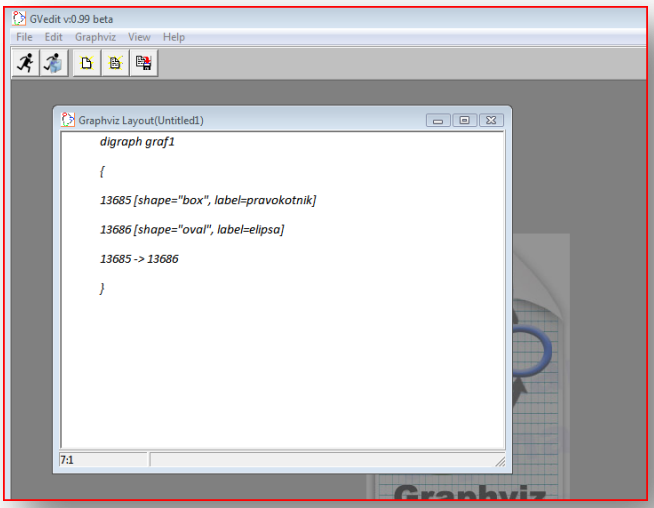


Slika 27: Uporabniški vmesnik orodja »GVedit«

Uporaba grafičnega urejevalnika je precej preprosta. Najprej izberemo nov dokument.

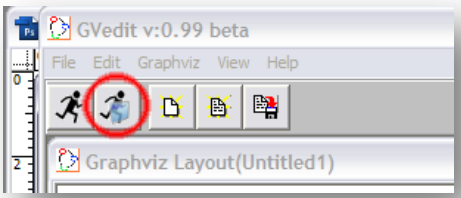


Slika 28: Izbira novega dokumenta

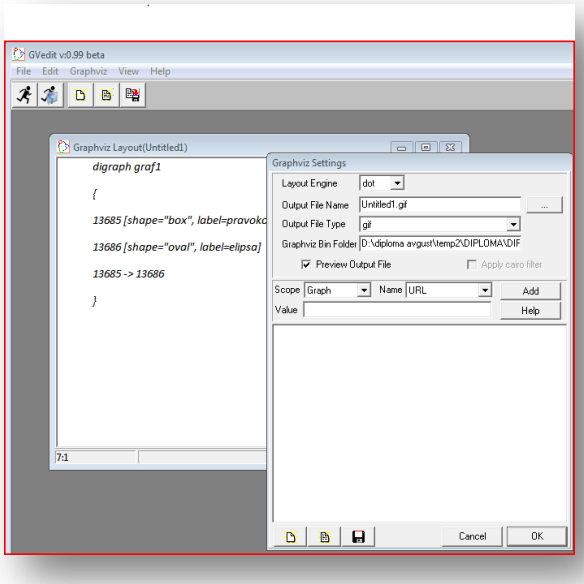


Slika 29: Okno z dodanim opisom grafa v jeziku »dot«

Odpre se nam manjše okno, v katerega napišemo opis grafa v formatu »dot«. Nato sledijo nastavitve, ki so povezane s samo izdelavo grafa.



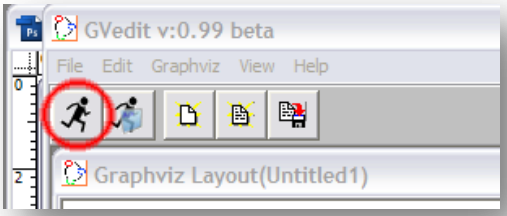
Slika 30: Izbira pregleda nastavitev



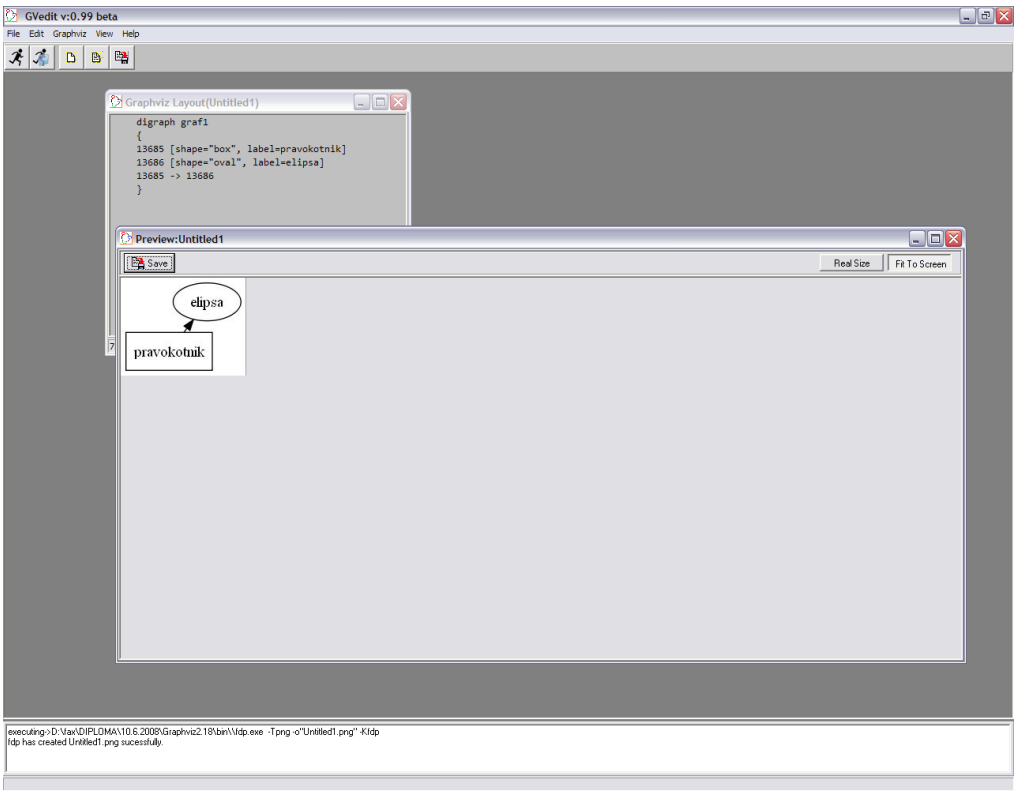
Slika 31: Okno z vsemi možnimi nastavitvami

Najbolj osnovne nastavitve, ki jih lahko izbiramo so orodje izrisa, tip izhodnega formata, lokacija izhodne datoteke ter ime izhodne datoteke.

Nato poženemo izdelavo grafa in v novem oknu se nam prikaže rezultat.



Slika 32: Izbira začetka procesiranja dokumenta



Slika 33: Končan postopek izdelave grafa v urejevalniku GVedit

Za izdelavo grafov lahko uporabljamo tudi ukaze v lupini. Sicer je ta način precej uporabniku neprijazen, saj tu ne vidimo neposredno rezultata, ki ga je izdelal Graphviz, je pa zato toliko bolj primeren za avtomatizacijo izdelave grafa s pomočjo klicev iz našega lastnega programa, ki uporablja Graphviz kot generator grafa. Na takšen način končni uporabnik niti ne opazi, da za izris grafa poskrbi povsem samostojno orodje.

V našem primeru najprej v programu izdelamo datoteko tipa »dot«, ki vsebuje vse potrebne informacije o vozliščih ter povezavah med njimi. Nato s klicem v lupini kličemo Graphviz ter počakamo, da nam vrne želeno izhodno datoteko.

Primer klica v lupini, ki kot rezultat izdela le tekstovno datoteko, v kateri so zapisane pozicije vozlišč in povezav:

```
fdp.exe D:\Grafi\test.dot -o D:\Grafi\test.txt
```

Klic je sestavljen iz imena orodja, ki smo ga izbrali, nato sledi pot do vhodne datoteke, določitev oblike izhodne datoteke ter pot in ime izhodne datoteke. Pri tej metodi se je pojavila težava v našem sistemu, saj naš sistem ni mogel vedeti, kdaj je Graphviz zaključil zahtevano operacijo. Tako je prišlo do situacije, da je včasih našel zahtevano datoteko, včasih pa ne. Problem smo rešili s proceduro, ki preverja procese v operacijskem sistemu in sporoči, kdaj se določeni proces tudi zaključi. Klice v lupini tako kličemo preko te omenjene procedure, ki zaustavi delovanje programa za toliko časa, dokler se ne sprosti zahtevani proces, ki služi za izdelavo grafa v Graphvizu.

5 Opis razvitih modulov

V tem poglavju opisujemo dva razvita modula, na koncu pa še ločeno nekaj težav pri doseganju nekaterih funkcij in pri prehodu na novejšo različico uporabljenih orodij.

Zgradba našega sistema je modularna. Razlog za takšno odločitev je v tem, da želimo najprej razviti osnovne funkcionalnosti, ki so potrebne za vse naloge sistema. Šele ko bodo enkrat razvite vse potrebne funkcionalnosti oziroma vsi moduli, bomo lahko s pomočjo modularne zgradbe, enostavno sestavili vse dele sistema skupaj ter v sistem vključili še primeren uporabniški vmesnik.

Prednost modularne zgradbe je tudi pri morebitnih razširitvah sistema, saj nam skoraj ni potrebno narediti veliko sprememb v obstoječi kodi, ampak le dodamo nov modul, ki vsebuje že vse potrebne funkcije in procedure.

Za razvoj smo uporabili orodja VBA in Graphviz. VBA je dostopen na vseh računalnikih, ki ima nameščeno zbirko MS Office. Prav tako se avtomatsko namesti z namestitvijo številnih drugih programov, ki ga uporabljajo kot skriptni jezik (npr CorelDraw, AutoCAD itd.). Graphviz je odprtokodno orodje, dostopno tako za MS Windows kot za različne druge operacijske sisteme.

Pred razvojem sistem smo morali najprej določiti nekaj osnovnih gradnikov, ki bodo sestavljali naš sistem, tako programskih kot vizualnih. Tako smo za osnovne grafične gradnike izbrali naslednje grafične elemente:

- Slike, ki izboljšajo razumevanje izrisanih elementov določenega proizvodnega sistema.
- Pravokotnik, elipsa, oblaček (angleško callout) in šestkotnik, ki predstavljajo različne oblike, ki bodo v končnem sistemu predstavljale vsaka svojo skupino elementov v proizvodnem sistemu. Na primer, pravokotniki bodo predstavljali PLK in krmilnik, medtem ko bodo elipsa predstavljala senzor.
- Črta in prostoročni elementi, ki služijo za povezovanje ostalih omenjenih grafičnih elementov med seboj.

V urejevalniku VBA smo definirali tudi svoj podatkovni tip, ki ga imenujemo »tipLik«. Ta podatkovni tip nam služi za začasno shranjevanje lastnosti grafičnih elementov, ki jih potrebujemo za normalno delovanje našega sistema.

Sestava podatkovnega tipa »tipLik«:

```
Public Type tipLik
  strOblika As String
  intX As Integer
  intY As Integer
  intSirina As Integer
  intVisina As Integer
  dblKot As Double
  sngVozlisca() As Single
  strText As String
  dblTransparent As Double
  lngBarva As Long
```

```
    lngBarvaCrte As Long
    strImeDat As String
    sngLeviCrop As Single
    sngDesniCrop As Single
    sngZgCrop As Single
    sngSpodCrop As Single
    intPovezanZ() As Integer
    lngPodelementi() As Long
    blnImaPovezave As Boolean
    blnImaPodelement As Boolean
    intSklop As Integer
    lngId As Long
    lngPredhodnik As Long
    strOpis As String
End Type
```

Kot vidimo »tipLik« vsebuje več spremenljivk različnih tipov, ki so potrebne za zapis vseh lastnosti grafičnih elementov. Podatkovni tip »tipLik« lahko definiramo tudi kot tabelo tipa »tipLik«, ki nam omogoča, da vanjo shranimo celotno vsebino risalne plasti.

Gradniki, tako grafični kot programski, so podrobneje opisani v prilogi A.

Ker podatkovna baza še ni dokončno izdelana, je večina modulov zgrajena tako, da jih je možno uporabljati s testno podatkovno bazo, kot tudi brez nje.

Razvili smo dva modula:

- modul za shranjevanje in ponovno izrisovanje grafičnih elementov in ugotavljanje povezav med njimi,
- modul za izris grafa na podlagi izbranih podatkov.

Na koncu tega poglavja smo opisali še težave, ki so se pojavile pri prehodu na novejšo različico paketa Microsoft Office, saj je različica 2007 doživela korenite spremembe predvsem v samem uporabniškem vmesniku, nekaj sprememb pa je doletelo tudi nekatere grafične elemente.

5.1 Modul za shranjevanje in ponovno izrisovanje grafičnih elementov in ugotavljanje povezav med njimi

Del modula, ki je namenjen shranjevanju elementov uporabljamo, ko končamo z grafičnim vnašanjem dokumentacije oziroma s projektiranjem.

S pomočjo uporabniškega vmesnika na delovni list Excela nanašamo poljubne grafične elemente ter morebitne podelemente. Nato jih poljubno povežemo med seboj z grafičnim elementom, kot je na primer črta ali z prostoročnim elementom. Ko zaključimo z delom, se sprožijo funkcije, ki najprej preberejo lastnosti vseh dodanih grafičnih elementov, nato poiščejo morebitne podelemente in na koncu preverijo še za povezave med vsemi elementi.

5.1.1 Shranjevanje lastnosti grafičnih elementov

Te elemente lahko zapišemo v podatkovno bazo ali v tabelo našega lastnega tipa »tipLik«.

Spodnja koda prikazuje način zbiranja podatkov iz risalne plasti v tabelo tipa »tipLik«.

```
Dim shpLik() as tipLik

    If (shpLik.AutoShapeType = msoShapeRectangle) Then
        With udtLiki(i)
        .intVisina = shpLik.Height
        .dblKot = shpLik.Rotation
        .strOblika = "stirikotnik"
        .intSirina = shpLik.Width
        .intX = shpLik.Left
        .intY = shpLik.Top
        .dblTransparent = shpLik.Fill.Transparency()
        .lngBarva = shpLik.Fill.ForeColor.RGB
    .lngBarvaCrte = shpLik.Line.ForeColor.RGB
    .lngId = shpLik.ID
On Error Resume Next      'Glej opombo 1!
    .strText = shpLik.TextFrame.Characters().Text
End With
```

Kot vidimo je potrebno shraniti podatke o poziciji, velikosti, barvi, prosojnosti, besedila ter edinstveno »ID« številko grafičnega elementa. Te podatke rabimo za več namenov, na primer za ugotavljanje povezav med elementi, ugotavljanje podelementov, spreminjanje lastnosti, opisa elementov, itd.

5.1.2 Iskanje podelementov

Ko so grafični elementi shranjeni, se začne postopek s katerim najprej preverimo če kateri grafični element vsebuje tudi kakšen podelement.

Napisali smo algoritem, ki preverja pozicijo ter velikost vsakega elementa posebej in nato preveri, če obstaja kakšen element, ki je na skoraj enaki poziciji, s tem da je manjši kot trenutno obravnavani element. V kolikor so izpolnjeni ti pogoji, je algoritem našel podelement, ki ga nato zapišemo v želeno zbirko podatkov.

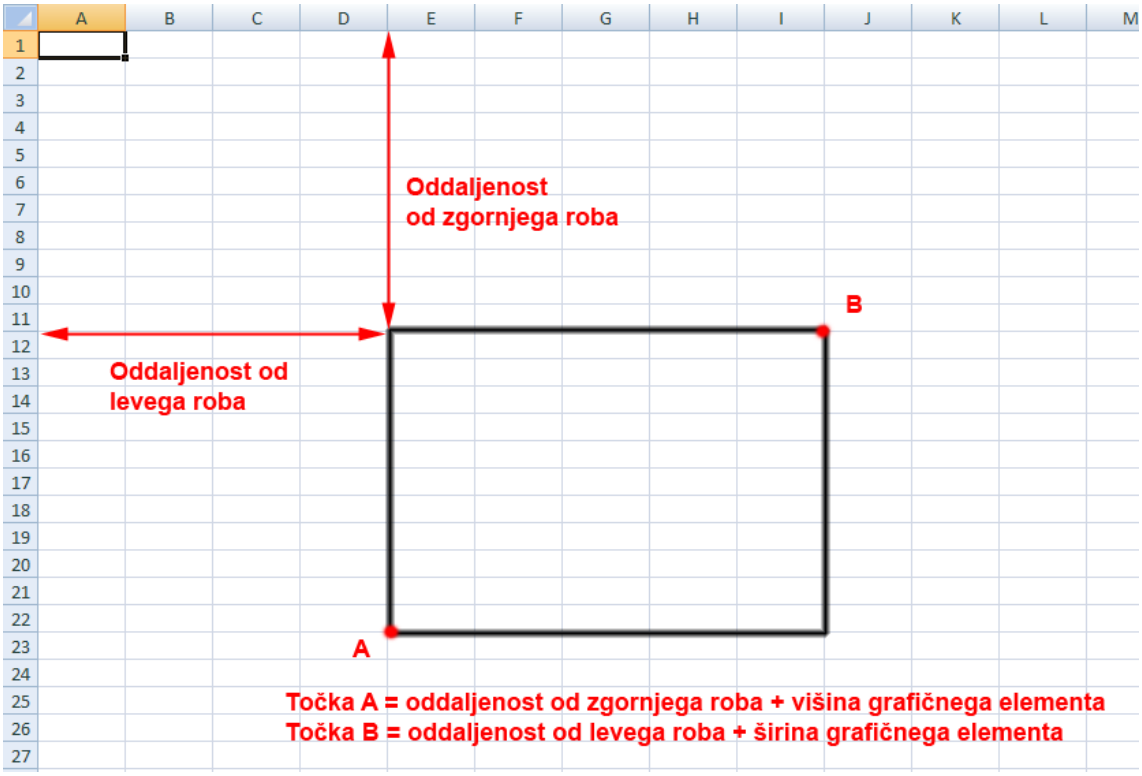
Prikaz dela algoritma, ki preverja morebitne podelemente:

```
For i = 1 To UBound(udtLiki)
    For j = 1 To UBound(udtLiki)
        If (udtLiki(i).intY < udtLiki(j).intY _
            And udtLiki(i).intX < udtLiki(j).intX _
            And (udtLiki(i).intY + udtLiki(i).intVisina) > _
            (udtLiki(j).intY + udtLiki(j).intVisina) _
            And (udtLiki(i).intX + udtLiki(i).intSirina) > _
            (udtLiki(j).intX + udtLiki(j).intSirina)) Then

                    .
                    .
                    .
```

Algoritem preverja, če se nek element nahaja znotraj drugega elementa. Kot prvo preveri, če se element nahaja v območju razdalje, ki je večja kot je razdalja od levega roba delovnega lista ter do oddaljenosti do levega roba grafičnega elementa na delovnem listu Excela in manjša od

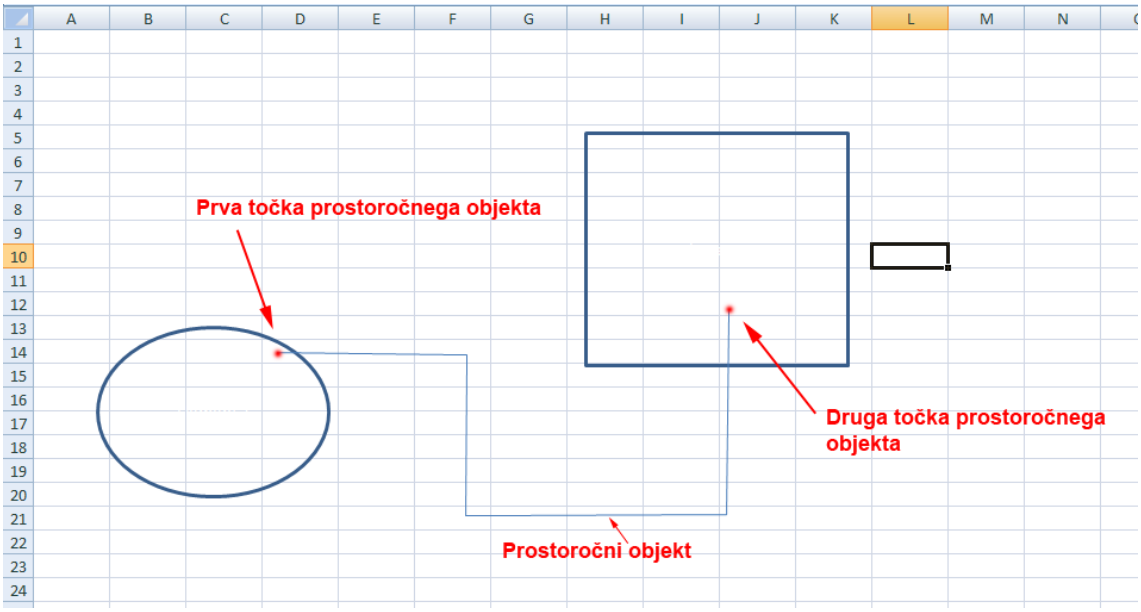
razdalje od levega roba delovnega lista pa do skrajne desne točke grafičnega elementa. Skrajno desno točko elementa dobimo tako, da razdalji od levega roba delovnega lista prištejemo še širino elementa, za katerega preverjamo, če vsebuje podelemente. Algoritem na enak način preverja vertikalno pozicijo elementa, le da tu vzame razdaljo od zgornjega roba delovnega lista pa do zgornje točke grafičnega elementa.



Slika 34: Predstavitev izračuna razdalje do točke A in do točke B

5.1.3 Iskanje povezav med grafičnimi elementi

Po iskanju podelementov, se sproži procedura, ki preveri vse povezave med grafičnimi elementi. Povezave preverja tako, da primerja pozicijo začetne in končne točke vsakega grafičnega elementa, ki je namenjen povezovanju (črta in prostoročni element). Če se pozicija prve točke nahaja v območju, ki ga zavzema kakšen drug grafični element, procedura na enak način preveri še končno točko črte ali prostoročnega elementa. V primeru, da je v obeh primerih rezultat iskanja pozitiven, dobimo povezavo dveh elementov, ki jo zapišemo v izbrano podatkovno zbirko.



Slika 35: Začetna in končna točka prostoročnega objekta

5.1.3.1 Težave s povezavami v paketu Microsoft Office 2007

Program smo začeli razvijati že pred prihodom nove različice Microsoft Excela 2007 in je bil prilagojen različicam vse do Excela 2003. Prihod različice 2007 je prinesel kar nekaj korenitih sprememb, tudi glede risalne plasti. Tako ni bilo več mogoče poganjati aplikacije na sistemih, ki so imeli nameščeno najnovejšo različico programskega paketa Office. Morali smo narediti več sprememb.

Glavna sprememba za nas, ki so jo naredili razvijalci Office paketa 2007 je, da črta, ne spada več pod prostoročni objekt, kar pomeni, da ne moremo dobiti koordinate obeh vozlišč črte. Ker pa so črte pri našem programu izredno pomembne, saj nam služijo za povezovanje elementov med seboj, smo morali poiskati drugo rešitev.

Ugotovili smo, da se da natančno pozicijo črte dobiti z »left«, »top«, »height« in »width« lastnostmi grafičnega elementa.

V VBA-ju »left« lastnost pomeni, oddaljenost zgornje leve točke elementa od levega roba dokumenta, »top« pomeni oddaljenost leve zgornje točke elementa od zgornjega roba dokumenta, »height« in »width« lastnosti pa pomenita višino in širino elementa. Pozicijo obeh točk dobimo tako, da za zgornjo levo točko vzamemo lastnost »left«, ki je enakovredna vrednosti na »X« osi koordinatnega sistema delovnega dokumenta, ter »top«, ki predstavlja vrednost na »Y« osi. Za spodnjo desno točko vzamemo oddaljenost od roba (»width«) in k njej prištejemo širino elementa (»width«). To predstavlja točko koordinate »X«. Za »Y« koordinato te točke naredimo podobno, le da vzamemo razdaljo od zgornjega roba dokumenta – »top« ter prištejemo višino elementa – »height«.

Opazili smo tudi, če je lastnost »rotation«, ki nam pove kot pod katerim je narisana črta, enaka 90 ali 270 stopinj, zgoraj opisana metoda določevanja točk ne deluje, saj Excel obravnava črto, kot da bi bila vedno obrnjena 90 stopinj v levo.

Tako lahko naredimo preizkus in postavimo črto v zgornji levi kot dokumenta, kjer bi morale biti »top« in »left« lastnosti enaki 0, in vidimo, da se to ne zgodi. »Left« lastnost celo dobi negativno število, zato moramo upoštevati dejstvo, da tudi če mi ne obrnemo črto preko vertikalne ali horizontalne osi, program v določeni situaciji, ki je odvisna od kota narisane črte, sam določi vrednost lastnosti »FlipVertical« in »FlipHorizontal«. Zato moramo spremljati vrednost teh dveh lastnosti, ki nam povesta, če je črta obrnjena preko osi »Y« ali preko osi »X«.

Spodnja funkcija predstavlja končno rešitev problema našega problema. Razvoj te funkcije je bil ključen za prehod našega sistema na novejšo različico paketa Microsoft Office.

```
Function ReturnCoordinates(element As Shape) As Integer()

   Dim intPoints() As Integer
   Dim intX As Integer
   Dim intY As Integer
   ReDim intPoints(1 To 2, 1 To 2)

       If (element.Rotation = 90 Or element.Rotation = 270) Then
           If element.HorizontalFlip = True And element.VerticalFlip = False Then
               intX = element.Left + element.Width / 2 - element.Height / 2
               intY = element.Top - (element.Width / 2 - element.Height / 2)
               intPoints(1, 1) = intX
               intPoints(1, 2) = intY
               intPoints(2, 1) = intX + element.Height
               intPoints(2, 2) = intY + element.Width
           ElseIf element.HorizontalFlip = True And VerticalFlip = False Then
               intX = element.Left + element.Width / 2 - element.Height / 2
               intY = element.Top - (element.Width / 2 - element.Height / 2)
               intPoints(1, 1) = intX
               intPoints(1, 2) = intY + element.Width
               intPoints(2, 1) = intX + element.Height
               intPoints(2, 2) = intY
           ElseIf element.HorizontalFlip = False And element.VerticalFlip = False
Then
               intX = element.Left + element.Width / 2 - element.Height / 2
               intY = element.Top - (element.Width / 2 - element.Height / 2)
               intPoints(1, 1) = intX
               intPoints(1, 2) = intY + element.Width
               intPoints(2, 1) = intX + element.Height
               intPoints(2, 2) = intY
           ElseIf element.HorizontalFlip = False And element.VerticalFlip = True
Then
               intX = element.Left + element.Width / 2 - element.Height / 2
               intY = element.Top - (element.Width / 2 - element.Height / 2)
               intPoints(1, 1) = intX
               intPoints(1, 2) = intY
               intPoints(2, 1) = intX + element.Height
               intPoints(2, 2) = intY + element.Width
           End If
       Else
           If element.HorizontalFlip = False And element.VerticalFlip = True Then
               intX = element.Left
               intY = element.Top
               intPoints(1, 1) = intX
               intPoints(1, 2) = intY + element.Height
               intPoints(2, 1) = intX + element.Width
```

```
                intPoints(2, 2) = intY
            ElseIf element.HorizontalFlip = False And element.VerticalFlip = False
Then
                intX = element.Left
                intY = element.Top
                intPoints(1, 1) = intX
                intPoints(1, 2) = intY
                intPoints(2, 1) = intX + element.Width
                intPoints(2, 2) = intY + element.Height
            End If
        End If
    ReturnCoordinates = intPoints()
End Function
```

5.1.4 Modul za izris elementov iz izbranih podatkov na delovni list Excela

Drugi del modula predstavlja izrisovanje že shranjenih grafičnih elementov na delovni list Excela.

Naš sistem je zgrajen tako, da lahko v podatkovno bazo shranjujemo več različnih shem, katera vsaka vsebuje svoje grafične elemente. Nato s pomočjo uporabniškega vmesnika izberemo shemo, ki želimo, da se ponovno izriše.

Za izris se uporabljajo funkcije in procedure, ki se nahajajo v modulu za izris elementov, ki za svoje delovanje lahko uporablja podatke, ki so shranjeni v podatkovni bazi ali pa v podatkovnem tipu »tipLik«.

Da procedure in funkcije v modulu lahko pravilno izrišejo vse shranjene grafične elemente, morajo biti ti shranjeni v pravilni obliki oziroma, morajo vsebovati vse potrebne lastnosti, iz katerih je mogoče razbrati pravilno pozicijo, rotacijo ter samo grafično podobo elementa.

Nekaj najbolj pomembnih lastnosti, katere potrebujemo za pravilno obliko izrisa :

- »top« - pomeni razdaljo od zgornjega roba delovnega lista in do zgornjega roba grafičnega elementa,
- »left« - pomeni razdaljo od levega roba delovnega lista pa do levega roba grafičnega elementa,
- »height - višina grafičnega elementa,
- »width« - širina grafičnega elementa,
- »rotation« - kot pod katerim je obrnjen grafični element,
- »characters« - besedilo v grafičnem elementu,
- »FlipVertical« - ugotovimo, če je element obrnjen preko »X« osi,
- »FlipHorizontal« - ugotovimo, če je element obrnjen preko »Y« osi.

Modul vsebuje tudi proceduro »dodajElementNaDokument«, ki na podlagi njenih parametrov izriše poljubne elemente na delovni list Excela. Ta procedura je uporabna, kadar v uporabniškem vmesniku dodajamo nove grafične elemente, ki še niso bili predhodno nikjer shranjeni.

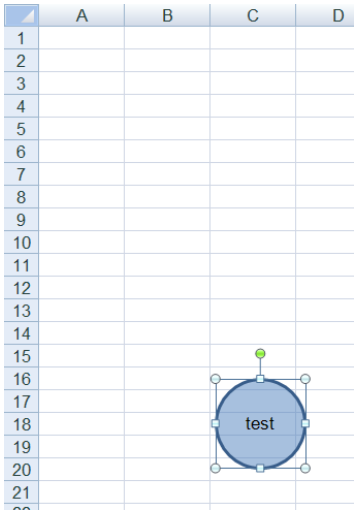
Deklaracije procedure z njenimi parametri, ki določajo lastnosti grafičnih elementov:

```
Public Sub dodajElementNaDokument(oblika As String, _
```

```
                                    strVelikost As String _
                                    Optional strBesedilo As String ="", _
                                    Optional strD_opis As String = "", _
                                    Optional X As Integer = 50,
                                    Optional Y As_ Integer = 50)
```

Primer uporabe te metode, če za parametre vzamemo naslednje podatke:

```
Oblika=”elipsa”
strVelikost=”srednja”
strBesedilo=”test”
x=100
y=200
```



Slika 36: Primer uporabe metode »dodajElementNaDokument«

5.2 Modul za izris grafa na podlagi izbranih podatkov

Modul je sestavljen iz procedur in funkcij, ki so namenjene izdelovanju datoteke tipa »dot«, branju datoteke »dot«, uporabi orodja Graphviz, vstavljanju izdelane slike na delovni list ter interakciji z vstavljenim grafom.

Graf nam služi za lažjo predstavo o medsebojni povezanosti različnih elementov v proizvodnem sistemu. V primeru, če obstaja zelo veliko povezav, postanejo tudi podatki, ki so predstavljeni kot besedilo, nepregledni. Sprva smo sami razvijali modul, ki je izrisal hierarhični graf na podlagi vhodnih podatkov, vendar smo ga kasneje zaradi različnih razlogov opustili in se raje osredotočili na odprtokodno orodje Graphviz, ki je zelo prilagodljivo in vsebuje že veliko uporabnih funkcij, ki jih ne bi bilo smiselno na novo odkrivati v našem sistemu. Opis opuščenega modula za izris grafa smo opisali v prilogi B.

5.2.1 Izdelava datoteke tipa »dot«

Za izdelavo datoteke tipa »dot« skrbi procedura, ki kot vhod dobi dva parametra in sicer spremenljivko tipa »tipLik«, ki vsebuje podatke o vseh grafičnih elementih, iz katerih želimo izdelati graf ter spremenljivka tipa »string«, ki predstavlja ime izhodne datoteke.

Spodnji primer prikazuje postopek, ki iz podatkov v podatkovni bazi, ki so nastali na podlagi uporabe modula za branje elementov in povezav, izdelamo datoteko, ki je zapisana v formatu »dot«.

Tabela elementov

Element

opis	ID	Lik	ID_ELEMENT	x	y	width	height	podelemen
Test1	23	1	13255	106,0714	163,9286	80,35716	85,71425	☐
Test2	24	1	13256	375	200,3572	138,2142	101,7857	☐
test 3	25	9	13257	213,2142	310,7143	96,42858	99,64291	☐
povezava	27	-2	13258	139,8215	242,6785	128,5714	97,5	☐
povezava	29	-2	13259	264,6428	228,2142	166,0714	122,1429	☐
	0		(New)	0	0	0	0	☐

Tabela povezav med elementi

Element | element_povezava

id_element	id_element	id_crte	dokument
13256	13257	29	
13257	13255	27	
0	0	0	

Slika 37: Podatki zapisani v podatkovni baz

Vsebina izdelane datoteke tipa »dot« na podlagi podatkov iz podatkovne baze:

```
graph graf1
{
13255 [shape="box", label=Test1]
13256 [shape="box", label=Test2]
13257 [shape="ellipse", label=test 3]
13255 -- 13257
13257 -- 13256
}
```

5.2.2 Uporaba orodja Graphviz

Graphviz uporabljamo za izdelavo grafov in izdelavo tekstovne datoteke, v kateri so zapisane koordinate posameznih vozlišč v tem grafu. Delovanje Graphviza smo podrobneje opisali že v poglavju »Prikaz delovanja orodja Graphviz« na strani 39.

Za komunikacijo z programskim paketom Graphviz uporabljamo proceduro, katera nato uporablja ukazno vrstico v kateri definiramo obliko izhodno datoteke ter lokacijo, kjer želimo, da se datoteka shrani. V Graphvizu izdelan graf nato uvozimo v Excel, kjer uporabniku služi za lažjo predstavo proizvodnega sistema oziroma le dela proizvodnega sistema.

Ker pa procedura kliče proces zunaj našega sistema, se je pojavila težava, saj naš sistem ni vedel, kdaj se je zunanji proces zaključil. Posledica tega je bila, da je enkrat sistem deloval pravilno, drugič pa ni našel zahtevane datoteke, saj jo Graphviz še ni pravočasno izdelal. Kot rešitev te težave smo naknadno uvedli funkcijo, ki zaustavi naš sistem, dokler se ne zaključi zunanji proces, v našem primeru ta proces predstavlja izbrano orodje v Graphvizu.

Spodaj je prikazan primer procedure, ki s pomočjo orodja Graphviz izdela graf na podlagi podatkov, ki so že shranjeni v datoteki v formatu »dot«. Vhodni parameter predstavlja tako ime vhodne datoteke, kot tudi ime izhodne datoteke, ki kasneje dobi končnico »png«.

Po zaključku te procedure tako obstajata dve datoteki z enakim imenom, le z drugačnimi končnicami, saj ena predstavlja podatkovno datoteko v formatu »dot«, druga pa slikovno datoteko formata »png«.

```
Public Sub izdelajSliko(strDatoteka As String)
    Dim ustvariDot As String
    strDatoteka = strPath & "\" & strDatoteka & ".dot"
    If (FileThere(strDatoteka & ".png")) Then
        Kill strDatoteka & ".png"
    End If
    If FileThere(strDatoteka) Then
        ustvariDot = strPath & "\Graphviz2.18\bin\" &strEngine & ".exe _
        " & strDatoteka & " -O -Tpng " & strDatoteka & ".dot"
        ShellAndWait ustvariDot, vbNormalFocus, True
    Else
        MsgBox ("Napaka! Datoteke " & strDatoteka & " ni mogoče najti!")
    End If
End Sub
```

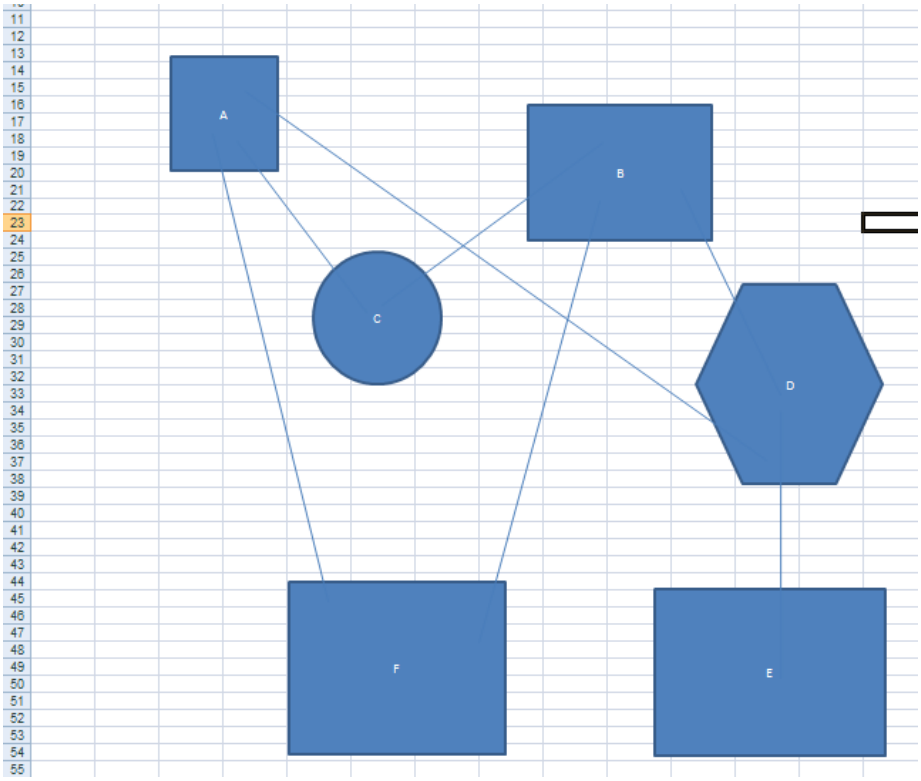
5.2.2.1 Postopek spremembe slikovnega grafa v interaktivni graf

Modul za izris grafa vsebuje tudi proceduro ki prebere datoteko tipa »dot«. Kot smo omenili v poglavju Prikaz delovanja orodja Graphviz, nam Graphviz izdela tako grafično kot tekstovno datoteko. Tekstovno obliko potrebujemo za to, da lahko ugotovimo natančne koordinate elementov na prav tako izdelanem grafu s pomočjo Graphviza. S pomočjo teh koordinat smo dosegli, da uporabnik lahko interaktivno izbira elemente na izdelani sliki, procedura, ki se prav tako nahaja v tem modulu, pa lahko iz uporabnikovega klika ugotovi, kateri element je uporabnik izbral in nato ustrezno reagira.

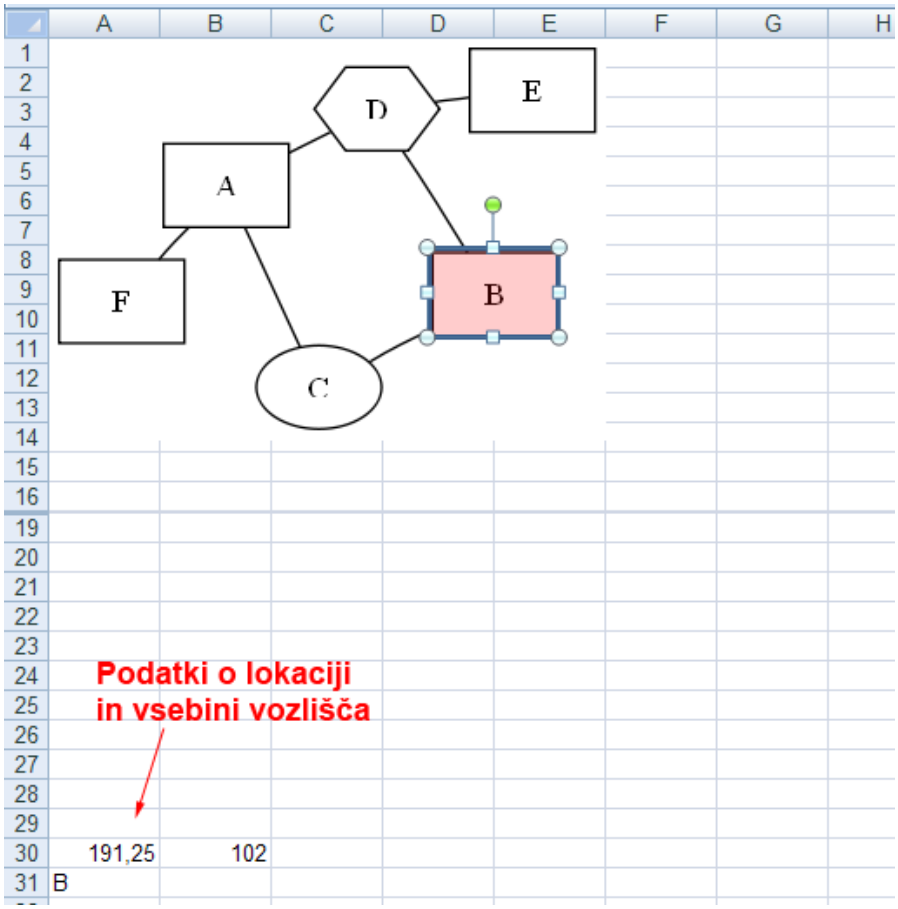
Napisali smo proceduro, ki prebere vsebino izhodne tekstovne datoteke v tabelo našega tipa »tipLik«. Nato procedura poskrbi za usklajevanje koordinat ter velikosti oblik, ki jih zavzema posamezni element na sliki, s koordinatami Excelovega odprtega delovnega dokumenta.

Tako dobimo vse potrebne podatke o lokaciji in velikosti izdelanih elementov v naši tabeli. Na vsak uporabnikov klik na procedura preveri, če se lokacija uporabnikovega klika ujema s kakšnim od shranjenim elementom v napolnjeni tabeli. Na ta način pridemo do interaktivnega grafa.

Spodaj je prikazan primer preproste sheme s testnimi vrednostmi (Slika 38), na podlagi katere je nato Graphviz izdelal graf (Slika 39). Ko uporabnik klikne na izbrano vozlišče na izdelanem grafu, se to vozlišče obarva rdeče ter pod grafom se izpišejo podatki o vsebini v vozlišču ter njegova lokacija. To služi le kot demonstracija, saj smo na tej točki zaključili z razvojem našega sistema.



Slika 38: Testna shema izdelana v Excelu



Slika 39: Demonstracija interakcijskega grafa

5.3 Težave pri doseganju nekaterih funkcionalnosti na prehodu na novejša orodja

V tem poglavju opisujemo najprej težave, ki so se pojavile pri interakciji našega programa z dogodkovnim sistemom operacijskega sistema Windows ter način reševanja. Nato podajamo še potrebne spremembe zaradi prehoda na novejšo zbirko z uporabljenim orodjem.

5.3.1 Težave z dogodki miške na delovnem listu Excela

Pri izdelavi interakcijskega grafa smo ugotovili, da VBA ne omogoča dogodkov na grafičnih elementih (angleško shapes). Dogodki so mogoči le na celicah. To nam je povzročalo zelo veliko težav, saj je uporabniški vmesnik precej bolj okoren, če ne omogoča uporabniku neposrednega izbiranja elementov. Tako bi bilo potrebno vse lastnosti dodanih elementov spreminjati iz drugega uporabniškega okna in še tam bi bilo potrebno ročno iskanje izbranega elementa, saj sistem ne more vedeti, kateri element imamo v mislih ko ga izberemo, saj ne pozna pozicije miške, kjer se trenutno nahaja.

Problem smo poskušali rešiti na več načinov:

»doEvents«

Sistem je bil najprej zasnovan tako, da je ves čas tekel v neskončni zanki, saj smo samo tako lahko preverjali, če je prišlo do spremembe na določenem elementu. Sistem je deloval, če smo dodali v zanko stavek »doEvents« nato pa se je pojavil problem, saj vedno ko smo odprli Excel in pognali naš sistem, Excel ni hotel označevati elementov. Elemente smo lahko premikali ampak žal jim zaradi neznanega vzroka ni bilo mogoče spreminjati velikosti ali jih obračati. Sistem je začel normalno delovati le, če smo v VBA urejevalniku pritisnili **break**, **reset** gumb ter nato še enkrat zagnali sistem. Vendar, ko smo naslednjič zopet odprli Excel, sistem ponovno ni deloval.

»mouseHook«

Zaradi slabega delovanja neskončne zanke, smo poskušali problem rešiti s tem, da bi dobili dogodke miške na delovnem dokumentu preko sistemskih klicev, kajti v VBA-ju so vgrajeni samo dogodki miške na uporabniškem oknu ter na celicah delovnega lista. Sistem je lepo deloval, če smo uporabili modalno uporabniško okno. Ko pa smo okno spremenili v nemodalno je sistem deloval nepravilno[32][33].

Nemodalno okno omogoča menjavo fokusa med dvema oknoma, ne da bi morali pred tem zapreti osnovno okno. Uporabnik lahko tako nadaljuje svoje delo v dveh različnih oknih ene aplikacije.

Skoraj vedno je prišlo do tega, da se je sistem nehal odzivati. Le v redkih primerih je normalno deloval, ponavadi spet samo takrat, če smo v urejevalniku VBA pritisnili **break** in **reset** ter ponovno pognali sistem.

Časovnik (angl. Timer)

Naslednja rešitev, ki smo jo uporabili je bila poskus z uporabo časovnika, ki na vsakih 300 milisekund pokliče proceduro, ki preverja pozicijo miške in spremembe na delovnem

dokumentu Excela. Kasneje smo tudi ta način opustili, saj je tudi v tem primeru prihajalo do težav, sicer v manjši meri kot v zgoraj opisanih postopkih, vendar še vedno se ni dalo povsem predvideti kdaj se bo Excel nenadzorovano zaključil.

Uporaba oznake (angleško label)

Nato smo našli način, ki je dobro deloval v vseh različicah Excela. Ugotovili smo, da lahko dodamo na delovni list Excela komponente kot so gumb, oznaka, polje z besedilom, itd.. Te komponente omogočajo zaznavanje miške, vendar le na področju, ki ga zavzema posamezna komponenta. Tako smo čez celotno površino delovnega lista dodali in raztegnili komponento oznaka. V nastavitvah smo nastavili na popolno prosojnost te komponente, tako da je postala nevidna. Ker pa še vedno ni bilo mogoče označiti že narisane elemente pod njo, saj so se nahajali pod to dodano komponento, smo določili, da ima vedno zadnjo pozicijo izmed vseh elementov na delovnem listu. Na ta način smo dobili vse dogodke miške, ki smo jih potrebovali.

5.3.2 Težave pri prehodu na sistem Microsoft Office 2007

Sistem smo začeli razvijati že pred prihodom nove različice Microsoft Excela 2007 in je bil prilagojen različicam vse do Excela 2003. Prihod različice 2007 je prinesel kar nekaj korenitih sprememb, tudi glede DL, tako ni bilo več mogoče poganjati aplikacijo na sistemih, ki so imeli nameščeno najnovejšo različico programskega paketa Office. Morali smo narediti več sprememb.

5.3.2.1 Vozlišča narisane črte

Glavna sprememba za nas, ki so jo naredili razvijalci Office paketa 2007 je, da črta, ne spada več pod prostoročni objekt, kar pomeni, da ne moremo dobiti koordinate obeh vozlišč črte. Ker pa so črte pri našem sistemu izredno pomembne, saj nam služijo za povezovanju elementov med seboj, smo morali poiskati drugo rešitev. Rešitev te težave smo že opisali v poglavju Težave s povezavami v paketu Microsoft Office 2007 na strani 49.

5.3.2.2 Razlikovanje med kontrolami VBA in med oblikami na dokumentu

Excel 2007 drugače pojmuje elemente, ki jih dodamo na delovni list. Na primer, v različici 2003 je tudi dodano kontrolo VBA (gumb, oznaka, itd.) obravnaval enako kot ostale oblike. Ko hočemo v Office 2007 izbrano VBA orodje poslati v ozadje za vsemi ostalimi elementi na delovnem listu, še vedno ostane prvi v vrsti. Tako se pojavijo novi problemi, ker smo v nekaterih primerih na ta način dobili pozicijo miške na delovnem listu, saj ne obstajajo vgrajeni dogodki miške na grafičnih elementih. Ker so s tem problemom povezane tudi nekatere druge procedure in funkcije, je sprememba vplivala tudi na funkcijo, ki je namenjena preštevanju vseh elementov na Excelovem dokumentu, saj rezultat ni upošteval več kontrol VBA. Tako nam je funkcija vračala napačno vrednost, kar je povzročalo nepravilno delovanje sistema.

5.3.2.3 Spremenjena začetna pozicija vstavljene slike

Pri dodajanju slike na Excelov dokument, Excel 2007 ne upošteva več zadnjo označeno celico, ampak vstavljeno sliko vedno doda v zgornji levi kot delovnega lista Excela. Pri različici 2003 je Excel sliko dodal na mesto, kjer je bila označena celica. Če vzamemo primer, da imamo označeno celico »A30«, bi prejšnje različice Excela dodale sliko v spodnji levi kot delovnega dokumenta.

5.3.2.4 Snemanje makrov na oblikovnih elementih ni več mogoče

Snemanje makrov na oblikovnih elementih kot so črta, pravokotnik, elipsa ter ostale oblike, v različici 2007 ni več mogoče. To sicer ne predstavlja kakšne večje ovire, ampak nam je ta možnost v prejšnjih različicah omogočala hitrejši razvoj sistema v primerih, ko se je pojavila težava s pisanjem procedure, ki simulira določeno akcijo v Excelu.

Na primer, če smo hoteli dodati sliko na dokument, smo vključili snemalnik makrov in nato smo ročno dodali novo sliko. V razvijalnem okolju VBA smo lahko videli kodo, ki se je zapisala za ta dogodek. To kodo smo lahko nato z manjšimi spremembami lahko uporabili v našem sistemu.

6 Sklepne ugotovitve

V okviru dela smo razvili nekatere module, ki bodo potrebni v okviru razvoja širšega programa za dokumentacijo procesnih nadzorno-krmilnih sistemov.

Opisali smo specifične značilnosti računalniške avtomatizacije procesov in potreb, ki se pojavljajo tako pri opisu kot pri razvoju teh sistemov.

Pregledali smo nekaj komercialnih sistemov, ki podpirajo različne nabore nalog, ki se pojavljajo pri projektiranju sistemov. Utemeljili smo potrebo po razvoju lastnega sistema.

Razvili smo nekaj modulov, potrebnih pri realizaciji lastnega sistema ter uporabljena orodja.

Med delom na nalogi, se je pojavila težava pri prehodu na novo različico uporabljenega programskega orodja, ki je opustila neke do tedaj uporabljane funkcionalnosti, oziroma jih je vpeljala v drugi obliki. Tu smo videli pomembnost načrtovanja programov, kjer se vsi implementacijski detajli »skrijejo« v ločene procedure ali postopke.

Pričujoče delo je gradnik v konceptu celovite zgradbe širšega sistema za podporo avtomatizacije procesnih sistemov, ki so ga zasnovali na odseku osrednjega nacionalnega raziskovalnega instituta, ki se raziskovalno in razvojno ukvarja z avtomatizacijo in robotizacijo. Nadaljeval se bo z načrtovanjem in razvojem drugih modulov, potrebnih za realizacijo celovitega sistema.

7 Priloga A – Primeri uporabniškega vmesnika modula za definiranje zgradbe sistema

V tej prilogi bomo prikazali vse funkcije izdelanega uporabniškega vmesnika, ki je sicer namenjen le testiranju izdelanih modulov in ne predstavlja izgled končnega sistema.

Kot vidimo na spodnji sliki (Slika 40), je uporabniški vmesnik razdeljen na 3 področja. Prvo področje je namenjeno izdelovanju shem proizvodnih sistemov, v drugem področju analiziramo povezave med elementi v zgrajeni shemi, tretje področje pa predstavljajo osnovne funkcije uporabniškega vmesnika.

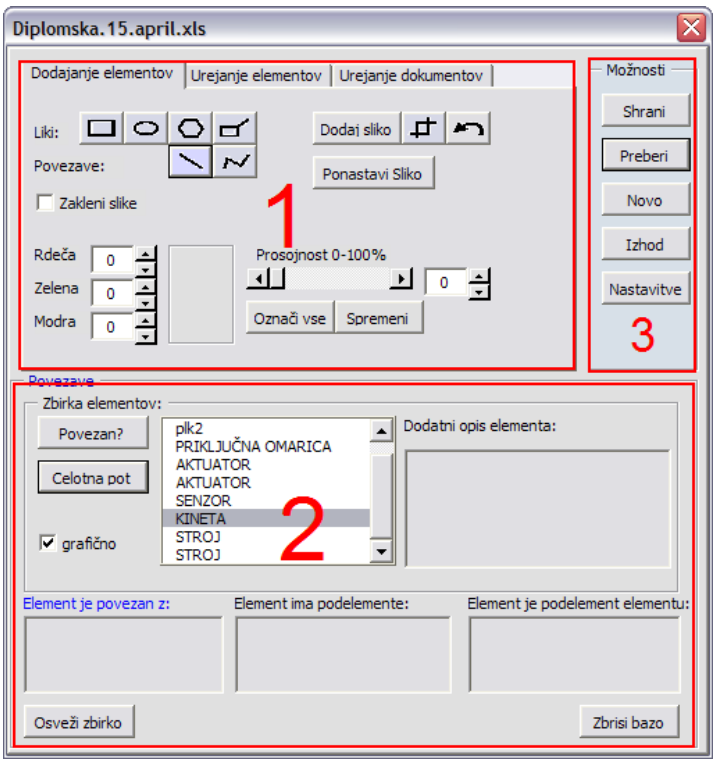
7.1 Prvo področje uporabniškega vmesnika

Prvo območje vsebuje 3 zavihke, izmed katerih je vsak zavihek namenjen različnim nalogam.

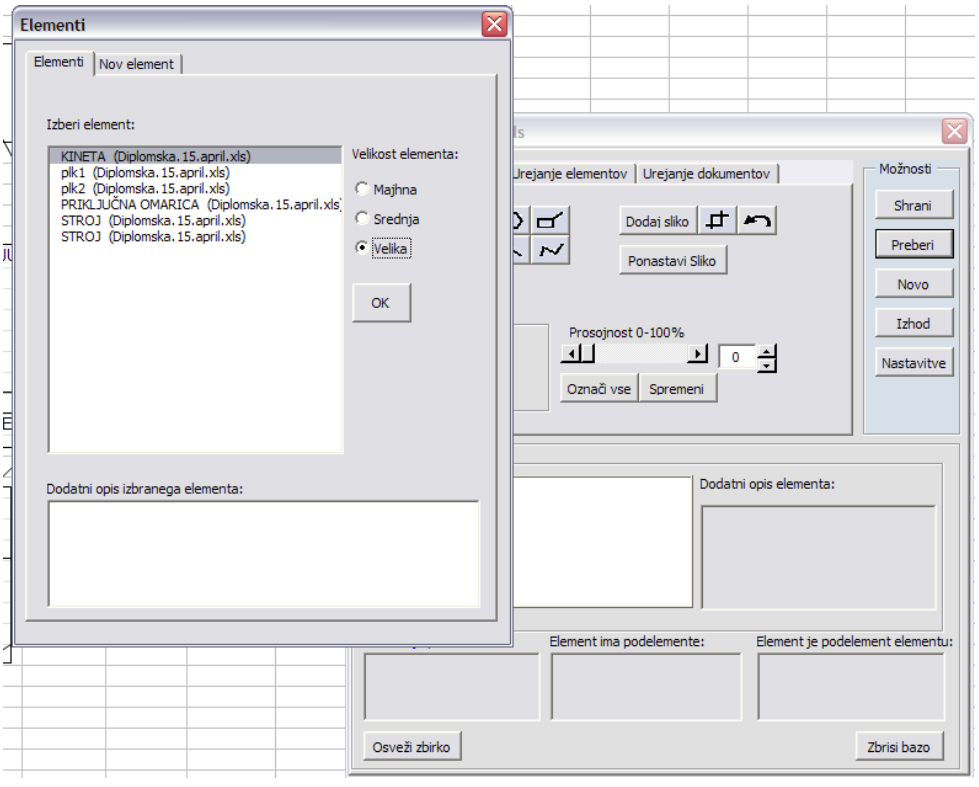
Prvi zavihek se imenuje »Dodajanje elementov«, kjer dodajamo nove grafične elemente na delovni list Excela ter jih tudi vizualno urejamo (prosojnost, barva). S klikom na gumb **Označi vse,** označimo vse grafične elemente na delovnem listu Excela ter jim nato s klikom na gumb **Spremeni** vsem naenkrat priredimo trenutno izbrano barvo ter stopnjo prosojnosti.

Preko tega zavihka lahko dodajamo tudi slike, jim spreminjamo velikost in pozicijo. Če želimo lahko tudi izrežemo le del slike s funkcijo za rezanje slik. S klikom na gumb **Ponastavi sliko**, se na označeni sliki razveljavijo vse spremembe, ki smo jih naredili do sedaj.

Elemente dodajamo tako, da izberemo obliko elementa, kot je pravokotnik, elipsa, šestkotnik ali oblaček (angleško callout). Nato se nam prikaže novo okno, v katerem lahko izberemo že obstoječe elemente, ki spadajo pod izbrano obliko, oziroma lahko dodamo novega. Pri dodajanju novega elementa vpišemo ime elementa, dodatni opis ter izberemo njegovo velikost (Slika 41).

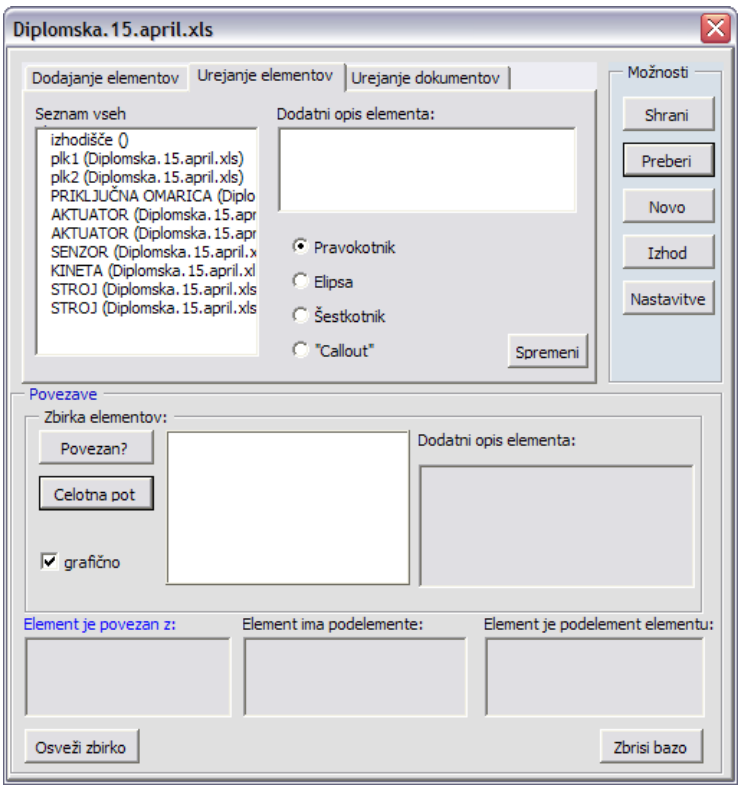


Slika 40: Uporabniški vmesnik, kot ga vidimo takoj po zagonu sistema



Slika 41: Dodajanje novega grafičnega elementa

Drugi zavihek je namenjen urejanju vseh elementov iz različnih dokumentov. Tako lahko naknadno popravljamo njihov dodatni opis ter jim spreminjamo privzeto obliko.

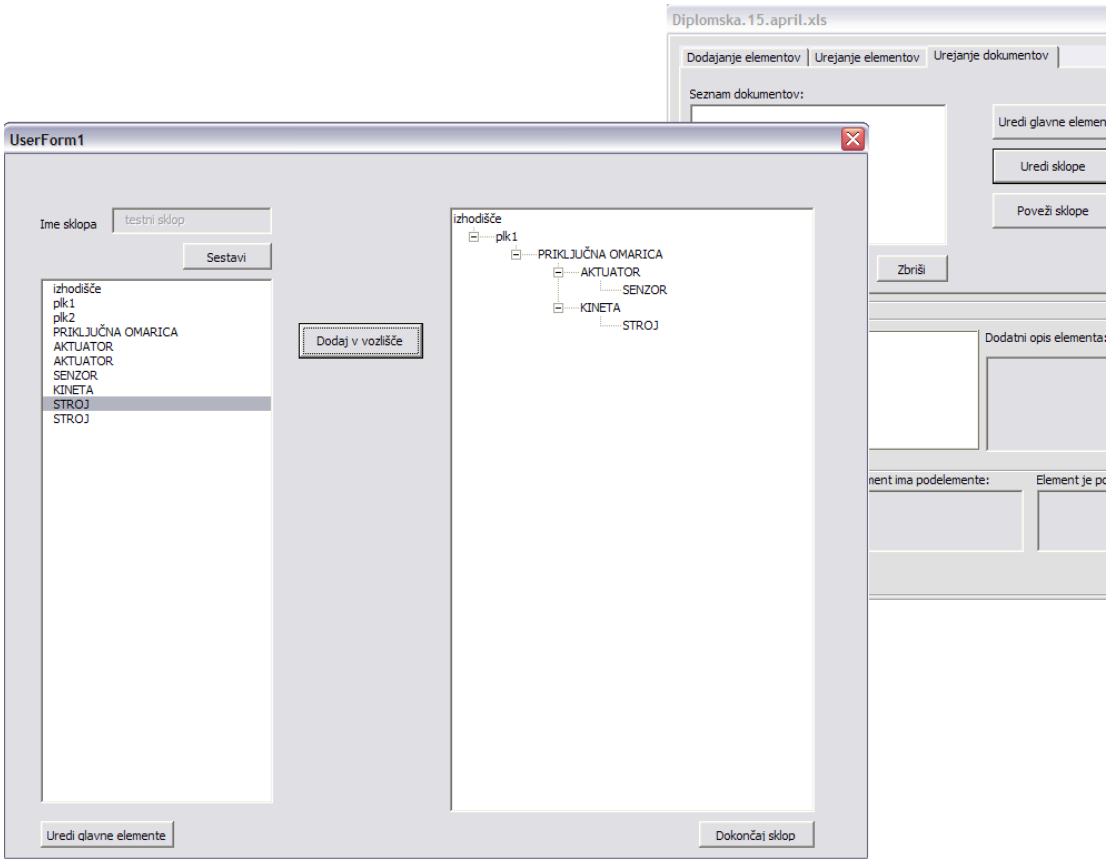


Slika 42: Urejanje vseh elementov

Tretji zavihek pa je namenjen urejanju celotnih dokumentov. Dokument nastane, ko končamo z risanjem sheme in jo shranimo v uporabniškem vmesniku z gumbom **shrani**.

Preko tega zavihka lahko dostopamo tudi do urejevalnika sklopov. Sklop predstavlja glavni element, ki vsebuje podelemente, podelementi lahko vsebujejo svoje podelemente, itd.. Tako urejevalnik sklopov predstavlja gradnjo sklopov, ne predstavlja pa povezav med sklopi. Sklope povezujemo v posebnem urejevalniku, do katerega dostopamo preko gumba **poveži sklope**, ki pa še ni dokončno izdelan. Sklope tako lahko gradimo preko grafičnega vmesnika na delovnem listu Excela ali pa uporabljamo zgoraj opisana urejevalnika.

Urejevalnik sklopov je predstavljen v obliki drevesa, ki ponazarja strukturo izbranega proizvodnega sistema, oziroma le del sistema. Postopek izdelave sklopa se začne z vpisom imena sklopa. Ko vpišemo ime in kliknemo na gumb **Sestavi**, lahko začnemo z gradnjo sklopov. Elemente, ki jih izberemo v levi polovici uporabniškega okna, dodajamo v izbrano vozlišče na desni strani uporabniškega okna z gumbom **Dodaj v vozlišče**. Tako lahko po želji izdelamo strukturo, ki ima povsem enak pomen, kot če bi za to uporabljali grafični urejevalnik. S klikom na gumb **Dokončaj sklop** zaključimo izdelavo sklopa.

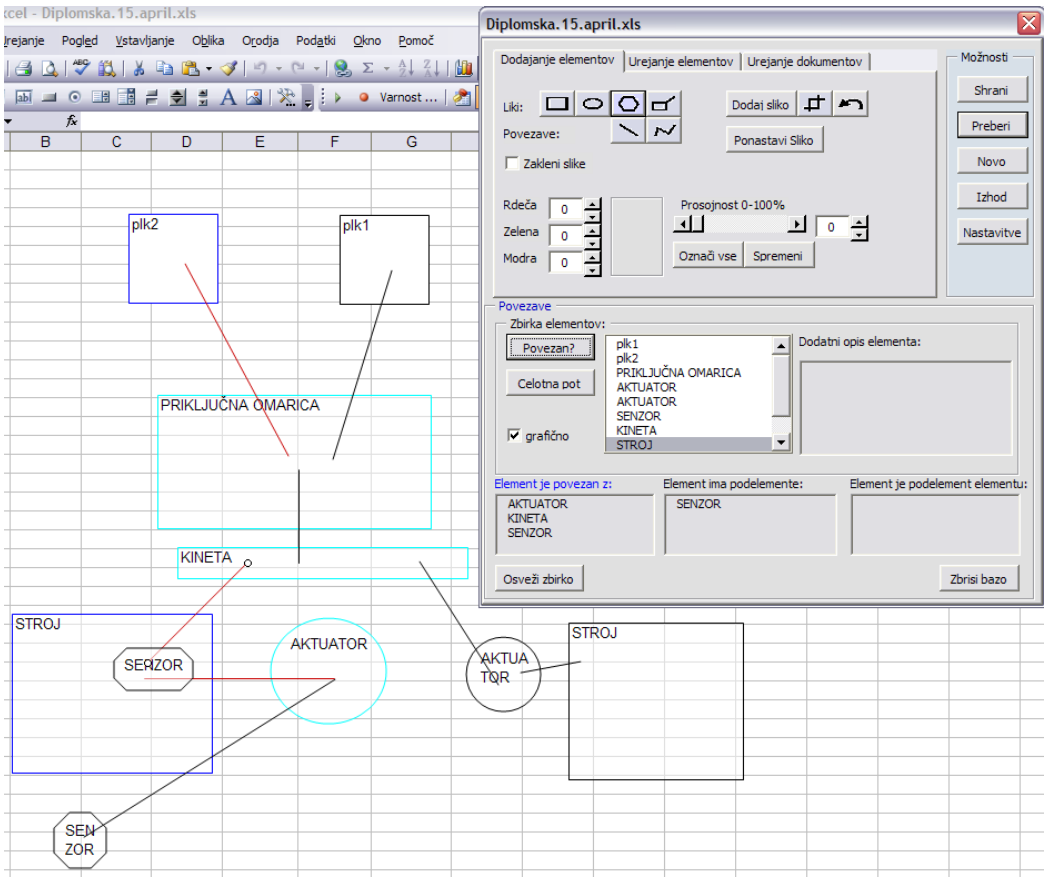


Slika 43: Urejevalnik sklopov

7.2 Drugo področje uporabniškega vmesnika

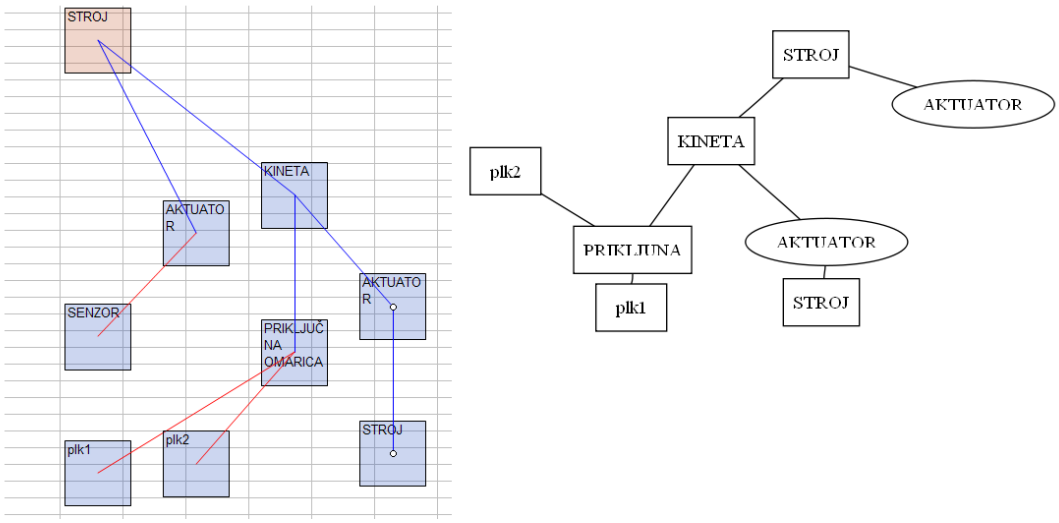
Drugo področje uporabniškega vmesnika predstavljajo komponente, ki so namenjene prikazu načina povezanosti elementov, ki se nahajajo na delovnem listu Excela, med seboj.

Da lahko vidimo s čim je nek element povezan, najprej iz seznama vseh elementov izberemo tistega, za katerega hočemo videti vse njegove povezave. Na koncu kliknemo na gumb **povezan** in pokažejo se nam podatki, s čim je ta element povezan, njegovi morebitni podelementi ter kateremu elementu je izbrani element podelement. Kot smo že omenili je celotni uporabniški vmesnik namenjen le testiranju modulov. Tako so tudi te tri poizvedbe namenjene le temu, da vidimo, če sistem deluje pravilno in posledično uporabniku zaenkrat niso v veliko pomoč.



Slika 44: Prikaz povezav za element STROJ

V tem področju imamo tudi možnost izrisa hierarhičnega grafa za izbrani element. Do te funkcije pridemo s klikom na gumb **celotna pot**. Kot smo že opisali v prilogi B, je kasneje to funkcijo nadomestilo orodje Graphviz. Spodaj je prikazana izdelava grafa zgornje sheme, z opuščenim algoritmom, ter z orodjem Graphviz.



Slika 45: Izris grafa z našim algoritmom ter orodjem Graphviz

7.3 Tretje področje uporabniškega vmesnika

Tretje področje uporabniškega vmesnika večinoma predstavljajo osnovne funkcije uporabniškega vmesnika, kot so:

- **Shrani**, ki shrani trenutno izrisan shemo na delovnem listu Excela v podatkovno bazo,
- **Preberi** analizira celotno narisano shemo na delovnem listu Excela ter napolni seznam vseh elementov v drugem področju uporabniškega vmesnika,
- **Novo**, ki zbriše vsebin delovnega lista ,
- **Izhod** zaključi program, pred tem pa še preveri, če je prišlo do kakšnih sprememb od zadnjega shranjevanja sheme,
- **Nastavitve**, kjer lahko spremenimo privzeto lokacijo podatkovne baze.

8 Priloga B – Gradniki prikazov in njihovi parametri; opuščene funkcionalnosti sistema

V tej prilogi opisujemo nekatere vidike, ki bodo pomembni pri nadaljevanju razvoja sistema za dokumentacijo, na primer pri shranjevanju opisov v datoteke ali baze. Opisujemo tudi nekatere opuščene načine za doseganje funkcionalnosti in omejitve, zaradi katerih smo opustili ta način implementacije.

8.1 Uporabljene okrajšave in dodatne razlage parametrov, ki se bodo uporabljali pri zapisu v podatkovno bazo:

DL – »drawing layer«

Win-X koordinate :

- Horizontalna pozicija elementa:
  - podana je v točkah (angleško points)
  - točka 0 je skrajno leva točka dokumenta
  - meri se od levega roba dokumenta pa do levega roba elementa
- Vertikalna pozicija elementa:
  - podana v točkah
  - točka 0 je zgornja točka dokumenta
  - meri se od zgornjega roba dokumenta in do zgornjega roba elementa

Višina, širina elementa:

- podani v točkah

Kot

- podan v stopinjah (od 0 do 360 stopinj)

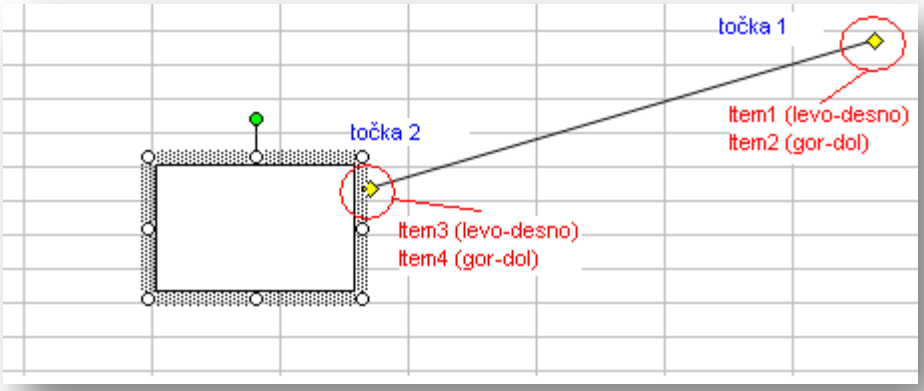
Prosojnost elementa

- od 0 do 100 %

Barva

- tip spremenljivke »Long« (long integer - 4 zloge) število (od 0 do 16777215)

»Item« (pri callout2)



Vsaka točka ima dve vrednosti (angleško item). Prva vrednost določa za koliko je točka horizontalno oddaljena od levega roba »text box-a«, druga vrednost pa za koliko je točka vertikalno oddaljena od zgornjega roba »text box-a«.

- Vrednost 1 je podana kot razdalja od levega roba »text box-a« pa do točke 1. Podana je glede na širino »text box-a«. V zgornjem primeru je vrednost 1 = 3.62 , torej je razdalja enaka 3.62 širinam »text box-a«. Če je črta pritrjena na levo stran »text box-a«, dobi pa tudi predznak »-«.

- Vrednost 2 pa je podana kot razdalja od zgornjega roba »text box-a« pa do točke. Podana je glede na višino »text boxa«. V zgornjem primeru je vrednost 2 = -1, torej je razdalja enaka eni višini »text box-a«. Če točko 1 premaknemo nižje od zgornjega roba »text box-a« dobi razdalja predznak »-«. Kadar je črta pritrjena na levo stran »text box-a«, pa je ravno obratno, torej če je točka 1 višje zgornjega roba je predznak »-«.Razdalja pa se še vedno meri od zgornjega roba »text box-a«.

- Če premaknemo točko 2 se v vrednost 2 in 3 zapišejo vrednosti (razdalja) glede na prejšnji položaj »callouta«.

Točke vozlišč

- če uporabimo »polyline« je število točk neomejeno
- podane so v »Win XY« koordinata
- točka 0 pri X koordinati je skrajno leva točka dokumenta
- točka 0 pri Y koordinati je zgornja točka dokumenta

Levi, desni, zgornji in spodnji odrez (angleško. crop)

- podani v točkah

Opisno ime	Slika	Pravokotnik	Elipsa	Šestkotnik	Callout-2	Črta	Polyline
Tabela							
Polje tabele							
Opis	Katerikoli element v jpg, gif, png, tif formatu	DL Rectangle. Ima lahko DL Text box	DL Oval Ima lahko DL Text box	DL Hexagon Ima lahko DL Text box	DL LineCallout2 Ima lahko DL Text box	DL Line	DL FreeForm
Parameter 1	Ime datoteke. Podano relativno na izhodiščno mapo						
Parameter 2	Hor. pozicija. V Win-X koordinatah.	Hor. pozicija. V Win-X koordinatah.	Hor. pozicija. V Win-X koordinatah.	Hor. pozicija. V Win-X koordinatah.	Hor. pozicija. V Win-X koordinatah.	Hor. pozicija. V Win-X koordinatah.	
Parameter 3	Ver. pozicija. V Win-Y koordinatah.	Ver. pozicija. V Win-Y koordinatah.	Ver. pozicija. V Win-Y koordinatah.	Ver. pozicija. V Win-Y koordinatah.	Ver. pozicija. V Win-Y koordinatah.	Ver. pozicija. V Win-Y koordinatah.	
Parameter 4	Višina elementa	Višina elementa	Višina elementa	Višina elementa	Višina elementa	Višina elementa	
Parameter 5	Širina Elementa	Širina Elementa	Širina Elementa	Širina Elementa	Širina Elementa	Širina Elementa	
Parameter 6		Kot	Kot	Kot	Kot		
Parameter 7		Prosojnost elementa	Prosojnost elementa	Prosojnost elementa	Prosojnost elementa		
Parameter 8		Barva elementa	Barva elementa	Barva elementa	Barva elementa		
Parameter 9		Barva obrobe(lineColor)	Barva obrobe(lineColor)	Barva obrobe(lineColor)	Barva obrobe(lineColor)	Barva obrobe(lineColor)	Barva obrobe(lineColor)
Parameter 10		Vsebina Text boxa v elementu	Vsebina Text boxa v elementu	Vsebina Text boxa v elementu	Vsebina Text boxa v elementu		
Parameter 11					Item(1) - za določitev črte callouta		
Parameter 12					Item(2) - za določitev črte callouta		
Parameter 13					Item(3) - za določitev črte callouta		
Parameter 14					Item(4) - za določitev črte callouta		
Parameter 15						Točke vozlišč	Točke vozlišč
Parameter 16	Levi odrez					Flip vertical	
Parameter 17	Desni odrez					Flip horizontal	
Parameter 18	Zgornji odrez						
Parameter 19	Spodnji odrez						

8.2 Opuščene funkcionalnosti našega sistema

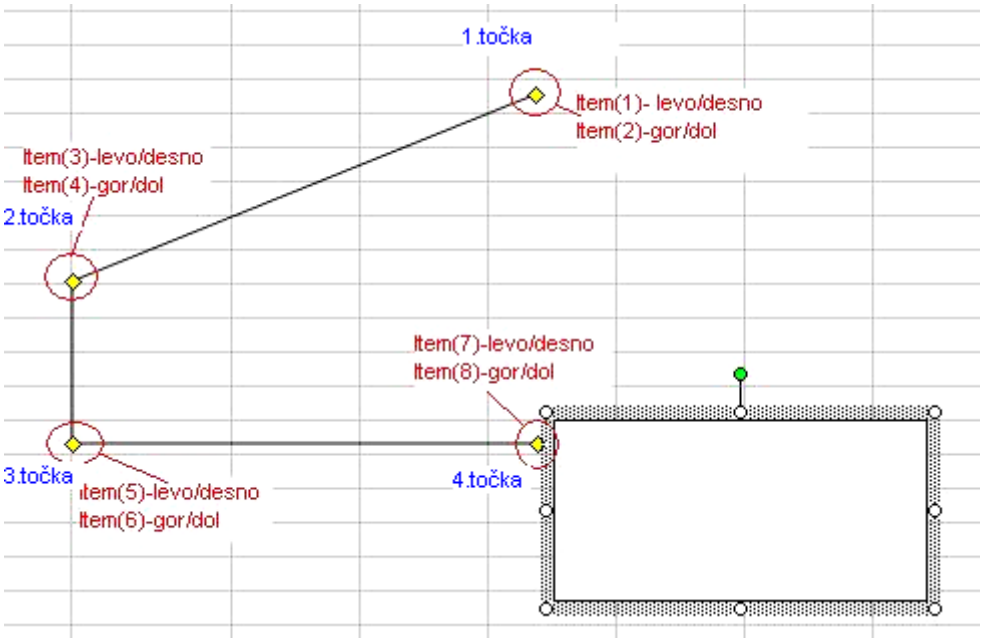
V tem poglavju bomo opisali zastavljene in tudi že delno razvite funkcionalnosti, ki smo jih bili primorani opustiti zaradi nezadovoljivega delovanja oziroma zaradi različnih omejitev paketa Microsoft Office, zaradi katerih ne bi mogli na ta način ustrezno zaključiti funkcionalnosti.

8.2.1 Delovanje sistemu v Word:

Program je bil sprva zasnovan tako, da bi deloval tako v Excelu tako v Wordu, saj imata oba programa skoraj identično risalno orodje. Problem pa se je pojavil pri samih omejitvah Worda, saj je največja širina in višina strani 22 palcev kar pa je premalo. Tako smo se odločili razvijati sistem le v Excelu.

8.2.2 Callout4:

Kot enega izmed grafičnih elementov smo poskušali dodati tudi element »Callout4«. Ko program prebere vrednosti (angleško item) elementa tipa »callout4« in nato na podlagi teh prebranih vrednosti spet nariše »callout4«, se grafična elementa vizualno razlikujeta. Problem je v tem, ker se črta, ki vodi iz »callout-a«, noče avtomatsko pritrditi na drugo stran besedilnega okna. »Callout4« se pravilno nariše samo v primeru, če je črta pritrjena na desno stran besedilnega okna. Ker pa mi potrebujem povsem enak izris sheme, kot je bila narisana, smo opustili ta grafični element.



8.2.3 Procedura preveriSlike():

Procedura se uporablja za preverjanje sprememb (pozicije in razmerja), na slikah ki smo jih dodali na delovni list (angleško worksheet) Excela.

- *Preverjanje razmerja:*

  Tudi, če uporabimo funkcijo »lockAspectRatio«, še vedno lahko spreminjamo slike v nepravem razmerju. Zato pri vsakem klicu te procedure preverjamo razmerje in če pride do razlike med razmerjem, ki smo ga izračunali ob dodajanju nove slike ter med trenutnim razmerjem, se

višina in širina slike spremeni na tisto velikost, ki je še bila v pravem razmerju. Upošteva se tudi uporaba funkcije, ki je namenjena obrezovanju slik (angleško crop). V primeru če jo uporabimo, se zopet spremeni razmerje slike. Zato ga takrat ponovno izračunamo ter ga uporabimo namesto prvotnega razmerja slike.

- *Preverjanje pozicije*

  Če na uporabniškemu vmesniku odkljukamo »zakleni slike« se pri vsakem naslednjem klicu procedure preveri, če je prišlo do sprememb v poziciji slik od zadnjega klica te procedure in če je, se slike postavijo nazaj na tiste pozicije, ki so bile v trenutku, ko smo odkljukali »zakleni slike«

  Težava pri uporabi te procedure

  Pri uporabi te procedure je bilo kar nekaj problemov, saj ne obstaja nobenega dogodka, ki bi se sprožil na delovnem dokumentu Excela, ki bi ga lahko uporabili za sprožitev te procedure. Večina dogodkov se nanaša le na celice in grafe. Mi bi potrebovali npr. dogodek premika miške (angleško mouse move) ali dogodek, ki se sproži na levi gumb miške (angleško left mouse-down), ki je vezan na celotni delovni list. Če pa smo program naredili tako, da je ves čas tekel v zanki z uporabo klica »doEvents«, potem ni deloval pravilno.

  Funkcija »DoEvents« se uporablja za zahtevo po procesorskem časa (ponavadi 50ms). Običajno se uporablja za nemoteno delovanje naše aplikacije, kadar se v njej odvija še ena, zelo zahtevna operacija, ki bi v normalnih okoliščinah povzročila nedelovanje aplikacije, dokler se ta operacija izvaja.

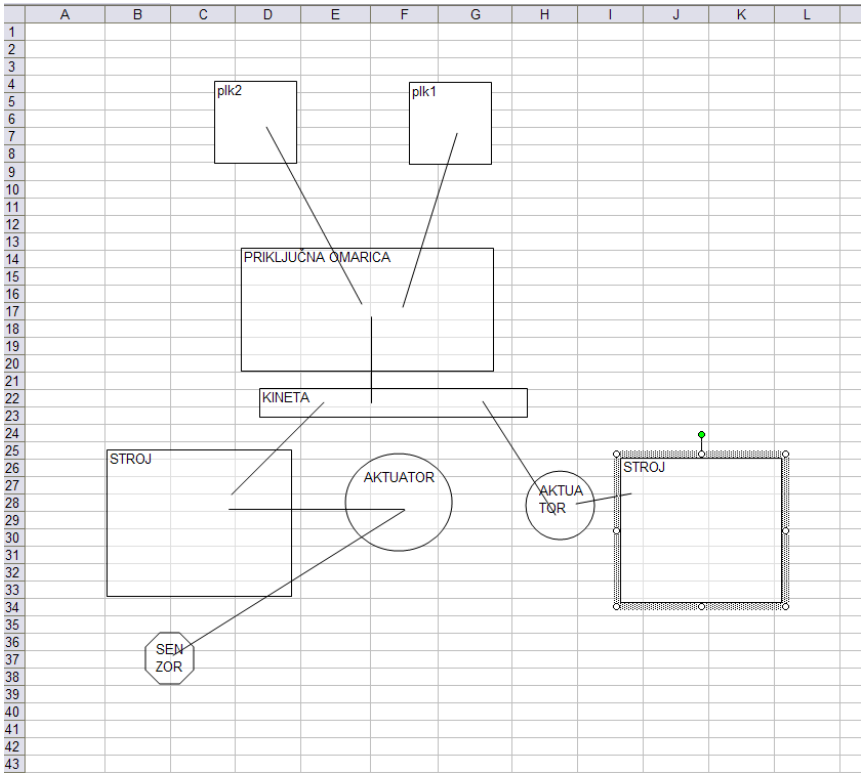
  Procedura »preveriSlike« je bila začasno opuščena zaradi nestabilnega delovanja Excela zaradi uporabe neskončne zanke.

8.2.4 Izris hierarhičnega grafa

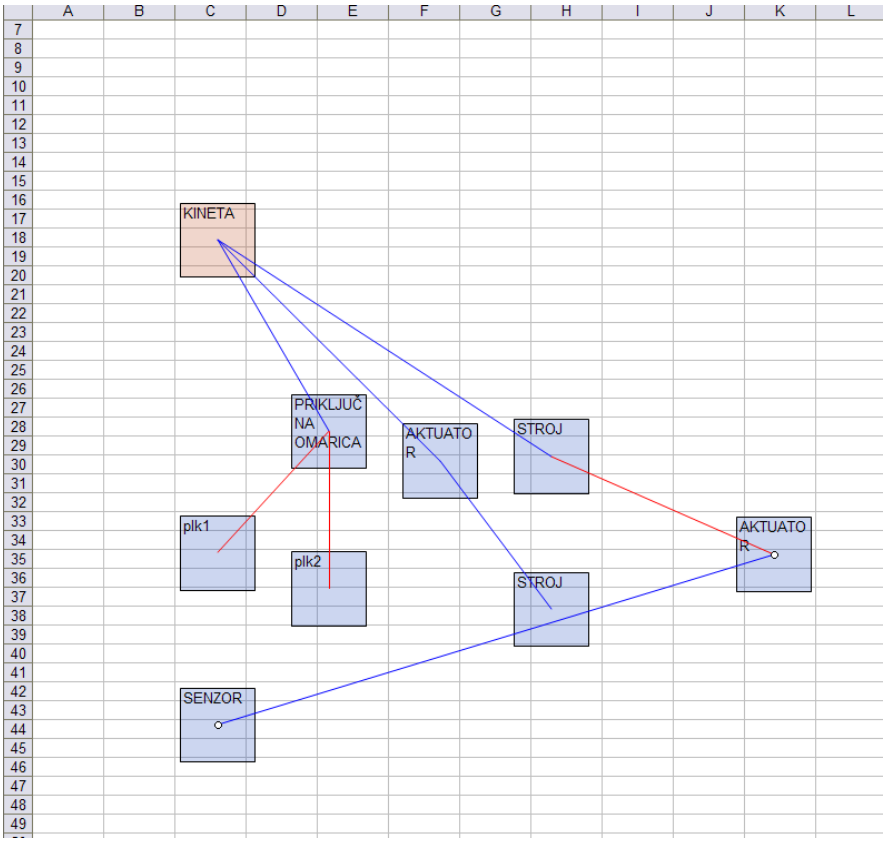
Napisali smo proceduro, ki iz podatkov, ki se nahajajo v podatkovni bazi samodejno izriše hierarhični graf vseh elementov, ki se nahajajo na delovnem listu Excel.

Procedura deluje na ta način, da kot vhodni parameter dobi izhodiščni element, preko katerega bo pregledala vse nadaljnje povezave, ki sledijo iz tega izhodiščnega elementa. Razvoj procedure smo opustili, saj je včasih pri izrisu prišlo do prekrivanja povezav in elementov med seboj. Želeli smo tudi več različnih načinov prikaza grafa in tudi možnost prikaza podelementov, zato bi razvoj algoritma, ki bi upošteval vse te zahteve, vzel preveč časa. Namesto te procedure smo našli odprtokodno orodje Graphviz, ki je zelo izpopolnjeno, vsebuje več različnih načinov izdelave, ima možnost risanja podgrafov ter omogoča mnogo oblikovnih nastavitev prikaza samega grafa.

Primer delovanja našega algoritma na podlagi predhodno izdelane sheme z našim sistemom:



Slika 46: Testna shema



Slika 47: Hierarhični graf izdelan z našim algoritmom na podlagi zgornje sheme. Kot izhodiščni element smo izbrali element »kineta«.

9 Seznam uporabljenih virov

[1] A. Ružič, "Zasnova zgradbe programa za dokumentacijo procesnih nadzorno-krmilnih sistemov," Institut Jožef Stefan Delovno poročilo IJS DP-10036, 2008.

[2] R. Robbins. (2008, Jan.) Control Engineering. [Online]. http://www.controleng.com/article/CA6567373.html

[3] Autodesk. (2008) AutoCAD Electrical - Product Overview Brochure. [Online]. http://images.autodesk.com/adsk/files/autocad_electrical_overview_brochure.pdf

[4] A. Orešnik, "Caddy++/SEE electrical V4R1 Novosti," *Avtomatika*, št. 78, str. 37-39, 2007.

[5] A. Orešnik, "Caddy++/SEE electrical V4R1 Novosti," *Avtomatika*, št. 79, str. 37-40, 2008.

[6] A. Orešnik, "Prihranek časa v vseh fazah priprave elektrotehniške projektne dokumentacije," *Avtomatika*, št. 80, str. 37-38, 2008.

[7] A. Orešnik, "Samodejno kreiranje elektro-načrtov v E-CAD programski opremi SEE Electrical," *Avtomatika*, št. 82, str. 36-38, 2008.

[8] D. Novak, "EPLAN Electric P8 Novi trendi in metode," *Avtomatika*, št. 75, str. 30-32, 2007.

[9] EPLAN Software & Services LLC - Computer Aided Engineering Solutions Experts. [Online]. http://www.eplan.org/

[10] WSCAD electronic GmbH. WSCAD The Software for Automation. [Online]. http://www.wscad.de/

[11] WSCAD electronic GmbH. WSCAD The Software for Automation. [Online]. http://www.wscad.de/website_2004/english/pdf/WSCAD5_english_brochure.pdf

[12] V. Miklič, "Izdelava novega grafičnega simbola v programu WSCAD 5," *Avtomatika*, št. 83, str. 39-43, 2008.

[13] V. Miklič, "Izdelava novega simbola v WSCAD5," *Avtomatika*, št. 85, str. 41-45, 2008.

[14] V. Miklič, "Izdelava projektne dokumentacije z WSCAD5," *Avtomatika*, št. 80, str. 42-46, 2008.

[15] V. Miklič, "Sistem označevanja in oštevilčenja v programu WSCAD 5," *Avtomatika*, št. 79, str. 42-47, 2008.

[16] V. Miklič, "WSCAD 5 - programsko orodje za projektiranje avtomatizacije in elektronapeljav," *Avtomatika*, št. 67, str. 44-46, 2006.

[17] V. Miklič, "WSCAD 5 – programsko orodje za projektiranje avtomatizacije in elektronapeljav," *Avtomatika*, št. 66, str. 44-46, 2006.

[18] V. Miklič, "WSCAD 5 – programsko orodje za projektiranje avtomatizacije in elektronapeljav," *Avtomatika*, št. 70, str. 46-47, 2007.

[19] M. Vojko, "Izdelava novega grafičnega simbola v WSCAD 5," *Avtomatika*, št. 84, str. 39-46, 2008.

[20] M. Vojko, "Izdelava novega simbola v WSCAD 5," *Avtomatika*, št. 82, str. 42-48, 2008.

[21] IGE+XAO Group. IGE+XAO software publisher, specialized in Electrical CAD and PLM. [Online]. http://www.ige-xao.com:8080/icons/ige/anglais/pdf/DOC_SEEELEC_AN.pdf

[22] (2008) Wikipedia, the free encyclopedia. [Online]. http://en.wikipedia.org/wiki/Visual_Basic_for_Astrlications

[23] G. Hart-Davis, "Understand How Excel Handles Graphical Objects," in *How to Do Everything with Microsoft Office Excel 2003*. McGraw-Hill Professional, 2003, str. 104-109.

[24] Microsoft Corporation. (2008) Understanding the Excel Astrlication Object. [Online]. http://msdn.microsoft.com/en-us/library/aa141660.aspx

[25] Microsoft Corporation. (2008) Working with Shapes. [Online]. http://msdn.microsoft.com/en-us/library/aa160260(office.10).aspx

[26] (2008) Wikipedia, the free encyclopedia. [Online]. http://en.wikipedia.org/wiki/Graphviz

[27] M. Tsoukalos. (2004, Sep.) Linux Journal. [Online]. http://www.linuxjournal.com/article/7275

[28] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, "Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools," in *Graph Drawing Software*. Springer-Verlag, 2004, str. 127-146.

[29] (2005, Nov.) Linux.com. [Online]. http://www.linux.com/articles/49655

[30] E. Gansner, E. Koutsofios, and S. North. (2006, Jan.) Graphviz - Graph Visualization Software. [Online]. http://www.graphviz.org/pdf/dotguide.pdf

[31] S. C. North. (2004, Apr.) Graphviz - Graph Visualization Software. [Online]. http://www.graphviz.org/pdf/neatoguide.pdf

[32] S. Teilhet, *Subclassing & Hooking with Visual Basic*. Sebastopol, United States of America: O'Reilly & Associates, 2001.

[33] (2005, Jan.) Osix. [Online]. http://www.osix.net/modules/article/index.php?id=653

[34] M. Schmalz, *Integrating Excel and Access*. Sabastopol, CA, USA: O'Reilly Media, Inc., Dec. 2005.

[35] Hiperion d.o.o., "Izdelava, urejanje in vodenje projektne dokumentacije v programu WSCAD 5," *Avtomatika*, št. 77, str. 43-46, 2007.

10 Izjava o avtorstvu

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod vodstvom mentorja doc. dr. Janeza Demšarja. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljubljana, oktober 2008 Aleš Fleischmann