SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 189

Programski sustav za automatiziranje procesa izrade pročelja u građevinarstvu

Ivica Pađen

Zagreb, lipanj 2011.

Sadržaj

1 Uvod

Kao jedna od najstarijih disciplina, građevinska znanost se od svojih početaka širila usporedno sa razvojem ljudskih spoznaja. Danas građevinarstvo predstavlja splet znanosti među kojima sve veći udio zauzima računarska. Automatizacijom procesa te primjenom odgovarajućih algoritama se omogućuje brže i kvalitetnije izvođenje radova. Stoga ne čudi interes brojnih tvrtki za razvojem računalnog *softwarea* koji ne samo da će ubrzati njihovo poslovanje, već će im i omogućiti prednost u odnosu na konkurente.

U ovom radu će biti objašnjeni procesi stvaranja metalnih pročelja te razmotren računalni pristup njihovom automatiziranju. Cilj praktičnog dijela rada je razviti sustav koji će imati ulogu potpore procesu. Kao ponajbolja tehnologija u kojoj se može ostvariti programsko rješenje se nametnuo *Windows Presentation Foundation* (WPF). Tehnologija, koja je ujedno i dio .NET 4.0 radnog okruženja, svojim bogatim knjižnicama omogućava jednostavniji rad sa geometrijskim i fizičkim podacima.

Jedna od najvažnijih stavki izrade pročelja je proizvodnja elemenata uz najmanju količinu otpada. Tako velika domena potencijalnih rješenja zahtjeva stohastičku metodu za učinkovito pronalaženje optimuma. Jedna od mogućih metoda je algoritam koji svojim izvođenjem imitira prirodni evolucijski proces.

Prvo poglavlje ovog rada opisuje proces izrade pročelja na onakav način kako mu pristupaju građevinski inženjeri. U sljedećem poglavlju su opisani i analizirani zahtjevi na sustav. Potom je dan opis WPF tehnologije s naglaskom na elemente korištene u praktičnom dijelu rada. Treće poglavlje sadrži slojeviti opis arhitekture. Zatim slijedi opis omogućene funkcionalnosti, a potom i primijenjenog evolucijskog algoritma. Na posljetku je dan osvrt na ostvareno rješenje, proces razvoja i postignute rezultate.

2 Proces izrade sastavnih elemenata pročelja

Izrada pročelja je vrlo kompliciran proces te se može reći kako ne postoje dva identična slučaja. Pročelje se sastoji od velikog broja elemenata koje se može grupirati u vidljivi dio, podkonstrukciju te pomoćne i završne elemente. Svi elementi su na određeni način povezani. Prethodno izradi pročelja se očekuje kako će biti izgrađena betonska konstrukcija koja će uz tlo biti potpora i nosač cjelovitog pročelja.

U izradi pročelja sudjeluje veliki broj ljudi. Sve započinje u uredima arhitekata koji na temelju svoje kreativnosti dobivaju ideje u vezi vanjskog, odnosno vidljivog, djela pročelja. Prilikom oblikovanja se uvijek nastoji postići rješenje koje će djelovati vizualno primamljivo, što zbog primjene raznih boja i materijala, što zbog postizanja svakojakih formi koje ponekad djeluju kao da prkose zakonima fizike. Nakon što arhitekti stave svoje ideje na papir, građevinski projektanti ih analiziraju te koristeći svoje znanje raščlanjuju na sastavne elemente. Na taj način nastaju detaljni nacrti na temelju kojih se u proizvodnom pogonu oblikuju potrebni elementi. Novonastali elementi se prenose na mjesto gradilišta gdje će ih građevinski radnici postaviti. Cijeli proces zahtjeva neprekidnu komunikaciju na relaciji arhitekt – projektant – građevinski radnik, a nerijetko se javlja potreba za naknadnim kompromisnim rješenjima.

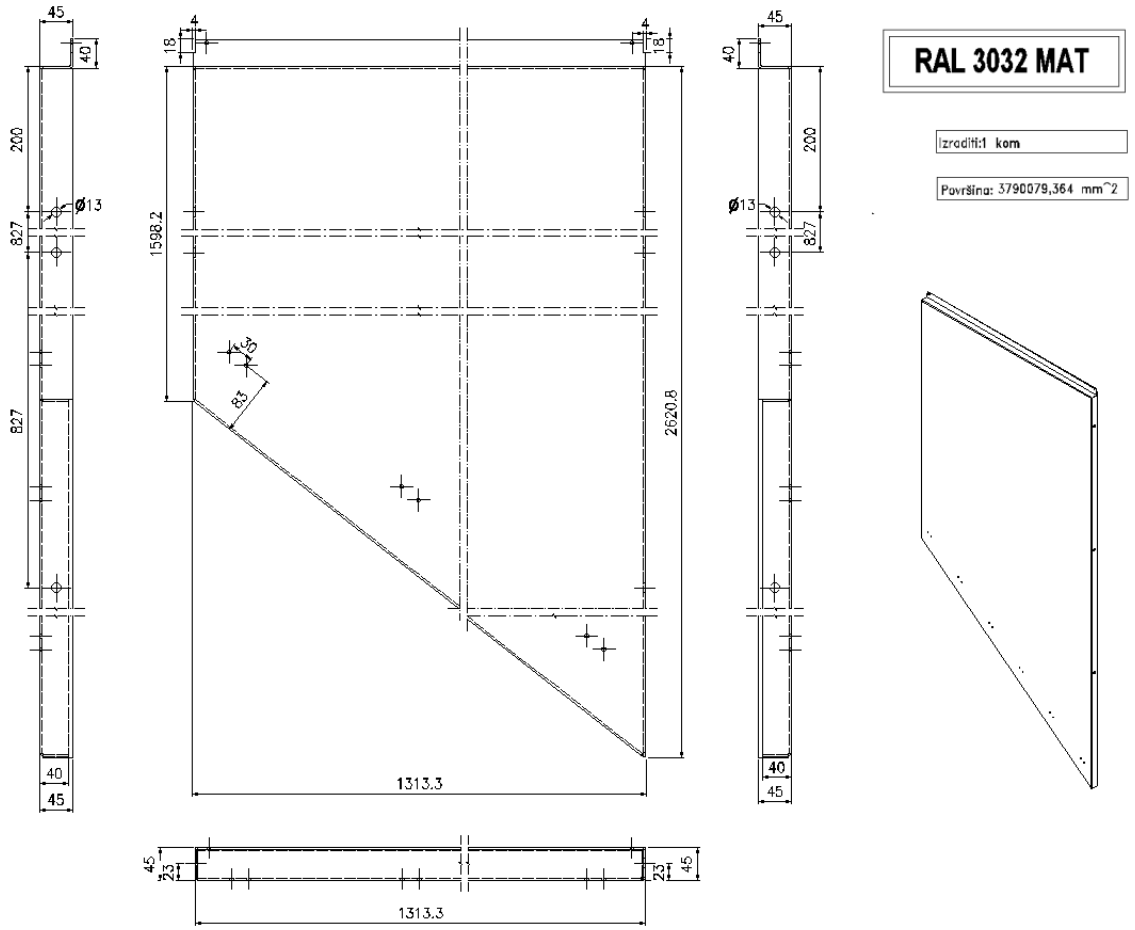
Zbog velike količine detalja i beskonačno velikog broja oblika koje pročelje može poprimiti, proces nije moguće automatizirati. Međutim, u određenim slučajevima, postoje neki segmenti stvaranja koji se izvode tipski te bi ih računalni software mogao sam izvoditi uz pomoć definiranih parametara.

Kod pročelja poput onih prikazanih slikom 2.1 može se primijetiti pravilna mreža elemenata. Definiranjem visine i širine pročelja te podjela moguće je prepustiti računalu da generira nacrte elemenata. Također, u slučaju pročelja kompliciranije forme, na opisani način je moguće generirati značajni udio elemenata. Potom se s obzirom na danu mrežu te dodatne parametre mogu definirati završni elementi krovišta te vrata i prozora. Osim vidljivih vanjskih elemenata moguće je generirati i elemente podkonstrukcije.

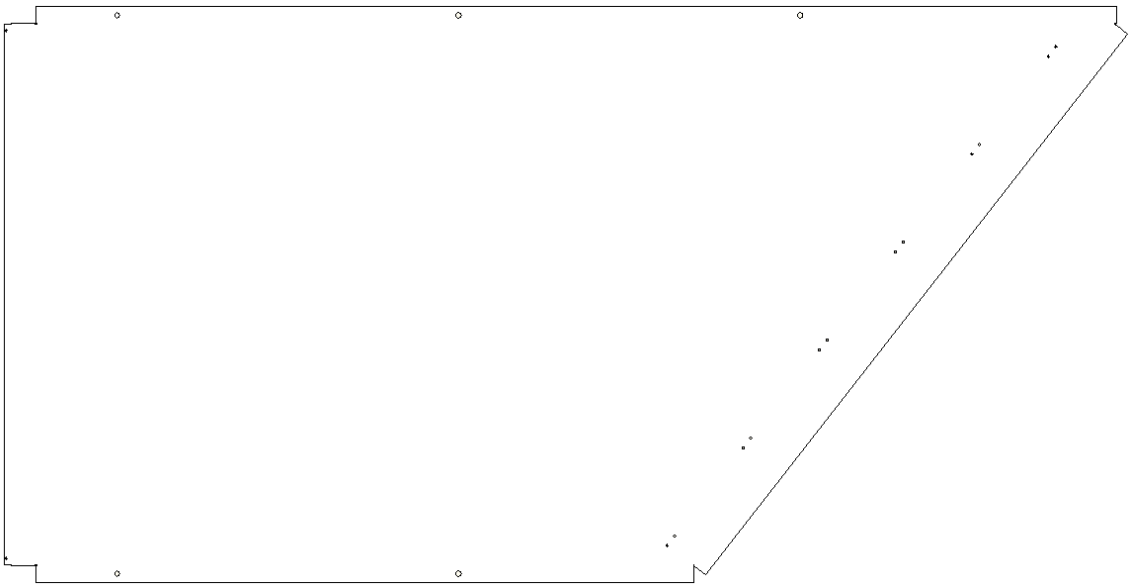
Slika 2.1 Primjer pročelja sa pravilnim rasporedom elemenata pogodno za automatsko generiranje

2.1 Definicija elemenata

Svaki element pročelja je definiran nacrtom i radioničkom dokumentacijom. Nacrt se stvara u AutoCAD aplikaciji specijaliziranoj za 2D i 3D modeliranje objekata. Gotovi nacrt se može izravno proslijediti na stroj za obradu koji će iz materijala izrezati zadani oblik laserskom zrakom. Radionička dokumentacija sadrži sve informacije bitne za određeni oblik. Na temelju tih informacija će radnik uz pomoć stroja za savijanje znati kako oblikovati neki element. Osim opisne uloge, takva dokumentacija ima i arhivsku ulogu za pohranu podataka o projektu. Primjeri nacrta i radioničke dokumentacije su prikazani slikama 2.2. i 2.3.



Slika 2.2 Primjer radioničke dokumentacije nepravilnog plošnog elementa. Vidljivi su prednji, bočni te nacrt. Navedene su pripadne mjere pomoću kojih će se oblikovati element. Dodatno je navedena boja elementa, broj identičnih komada za izrade, površina elementa te 3D skica gotovog oblika.



Slika 2.3 Primjer nacrta u DXF formatu pogodnom za obradu u AutoCad alatu. Nacrt se može izravno poslati na stroj za rezanje koji će prema njemu izrezati element.

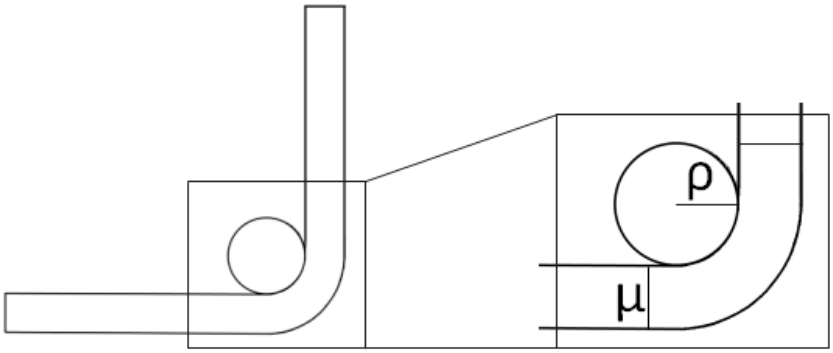
Elementi se mogu izrađivati od različitih vrsta materijala. Standardno se koriste Aluminij, Inox, toplovaljani čelik (TV) i pocinčani čelik (VZ). Ovisno o debljini, materijal ima drugačija svojstva. Prilikom izrade nacrta potrebno uzeti u obzir da će oblik nakon savijanja imati promijenjene dimenzije. Promjene su na razini nekoliko milimetara, ali se nikako ne smiju zanemariti.

Kada se materijal savine na području konveksije se dogodi proširenje te stanjivanje. Proširenje se može iskazati sljedećom formulom**:**

$$\Delta l = \pi \cdot \left(\frac{\beta}{180}\right) \cdot (\rho + \kappa + \mu)$$

Formula 2.1 Izraz za izračun proširenja materijala na području konveksije.

Kao što je prikazano slikom 2.4 parametar ρ označava radijus valjka kojim se materijal savija. Parametar μ označava debljinu materijala dok je parametar κ empirijski dobivena konstanta koja ovisi o debljini i vrsti materijala (tablica 2.1).

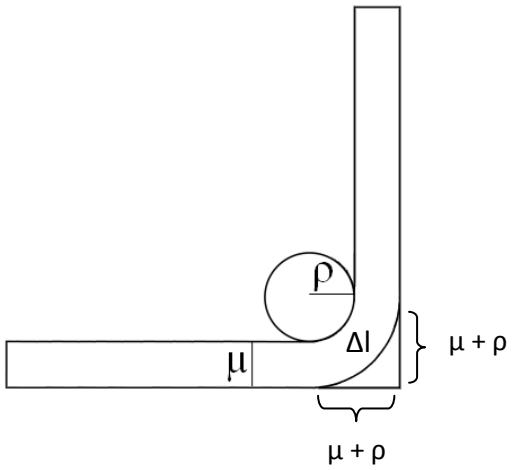


Slika 2.4 Parametri za izračun proširenja materijala prilikom savijanja

Aluminij		**TV/VZ**		**Inox**	
debljina(mm)	**κ faktor**	**debljina(mm)**	**κ faktor**	**debljina(mm)**	**κ faktor**
1	0.1459	0.8	0.19	1	0.135
1.5	0.2671	1	0.08225	1.25	0.065
2	0.275	1.25	0.168	1.5	0.13971
2.5	0.13475	1.5	0.2671	2	0.04
3	0.2608476	2	0.1	3	0.009
4	0.24338	3	0.155	4	0.215
5	0.2711	4	0.19	5	0.08
		5	0.18		
		6	0.215		
		8	0.257		
		10	0.155		

Tablica 2.1 Vrijednosti k faktora raspoređene prema vrsti i debljini materijala. Za toplovaljani i pocinčani čelik se uzimaju iste vrijednosti.

S obzirom na opisano proširenje od početnih dimenzija nacrta je potrebno oduzeti određenu vrijednost. Na primjer, nacrt ima visinu *a* i širinu *b*, te podrazumijeva jedno savijanje po širini. Od visine je potrebno oduzeti dva puta debljinu materijala, te dva puta radijus valjka za savijanje. Potom treba dodati duljinu proširenja materijala nastalu nakon savijanja. Tako da je konačna visina elementa jednaka $a - 2 * \mu - 2 * \rho + \Delta l$ (slika 2.5).

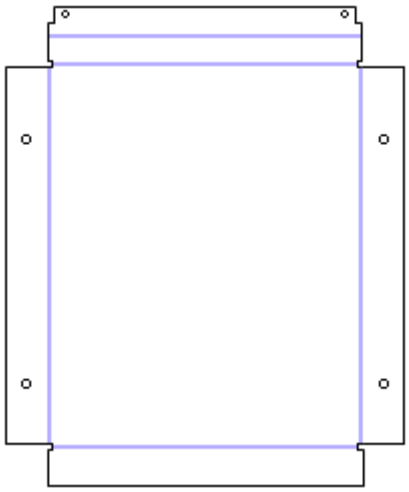


Slika 2.5 Shematski prikaz kao objašnjenje dobivenog izraza za visinu elementa

2.2 Vanjski plošni elementi

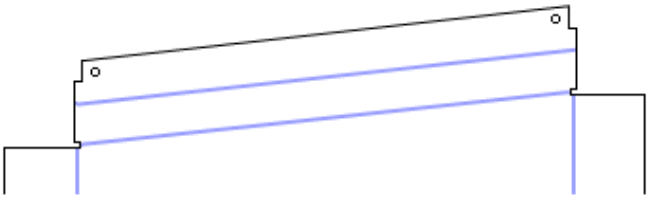
Vanjski plošni elementi su najvažniji dijelovi pročelja za postizanje vizualnog dojma. Kod ovih elemenata je osim vidljive plohe potrebno definirati proširenja. Pojam proširenja se u ovom kontekstu ne odnosi na proširenje materijala, već na ekstenzije elementa koje omogućavaju postavljanje na pročelje. Proširenja koja se nalaze sa strane služe vješanju elementa na podkonstrukciju. Svako proširenje sa lijeve i desne strane elementa sadrži rupe kroz koje prolaze vijci za pričvršćivanje. Donje i gornje proširenje omogućavaju slaganje elemenata jednog na drugi. Gornje proširenje dodatno ima i dvije rupe za sporedne radnje poput vješanja kod bojanja elementa. Zbog lakšeg poimanja plošnog elementa, slikom 2.6 je dan primjer sa gradilišta. Slikom 2.7 je ilustriran razvijeni element.

Slika 2.6 Na slici su vidljivi nepostavljeni plošni elementi (na stalku) te pripadno pročelje u pozadini. Na bližim elementima je vidljiva vanjska strana dok se na nešto daljim elementima može vidjeti unutrašnjost. Valja obratiti pozornost na proširenja koja se vide prilikom bočnog pogleda na elemente.



Slika 2.7 Razvijeni oblik plošnog elementa nakon izrezivanja iz materijala, prije savijanja.

Prema kompliciranosti izvedbe se vanjski elementi mogu podijeliti na pravilne i nepravilne oblike. S obzirom na velik broj mogućih nepravilnih plošnih elemenata, u obzir je uzet najčešći oblik koji se sastoji od četiri strane od kojih je jedna kosa kao što je prikazano slikom 2.8.

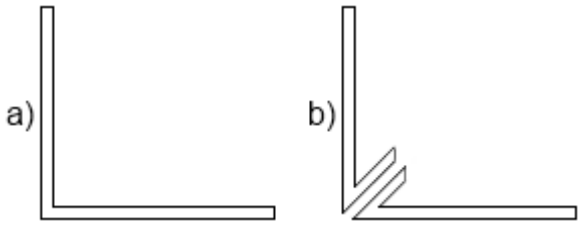


Slika 2.8 Najčešći oblik nepravilnog plošnog elementa s jednom kosom stranom.

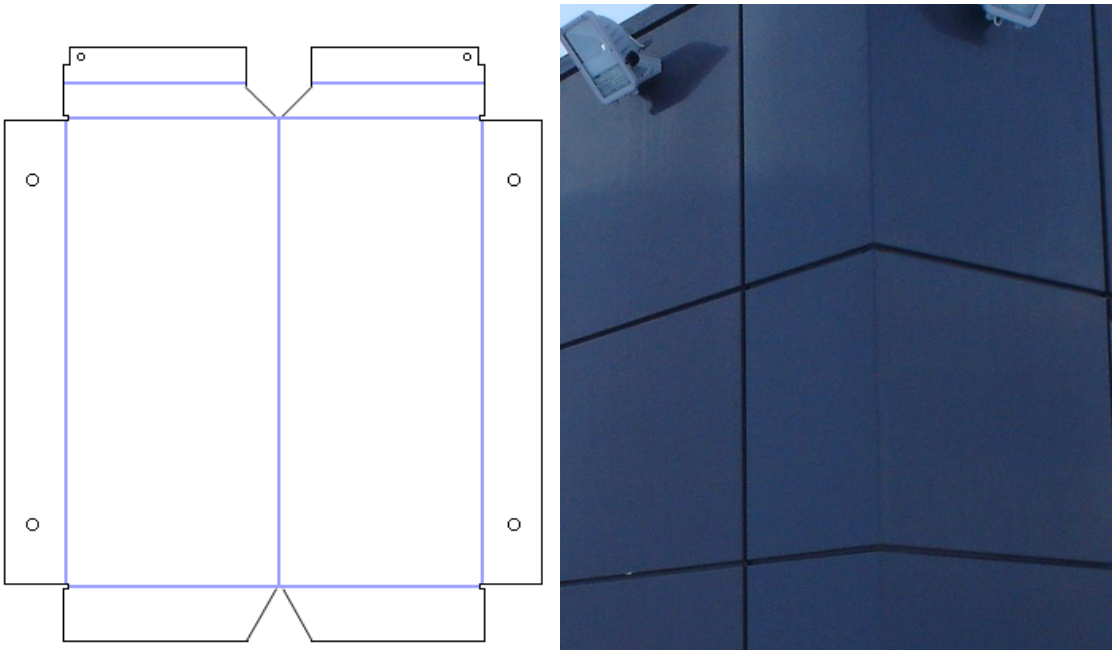
Kod elemenata ovakvog tipa je potrebno odrediti duljinu kosine. Potrebna duljina se može dobiti primjenom pitagorina poučka. Dužu katetu pravokutnog trokuta čini širina elementa. Kraću katetu čini razlika u visini između lijeve i desne strane elementa. Kosina, odnosno hipotenuza, je jednaka korijenu zbroja kvadrata kateta.

2.2.1 Kutni elementi

Kod vertikalnih granica pročelja je potrebno obaviti kutnu obradu vanjskih elemenata. Obrada se može izvršiti na dva načina. Prvi je taj da se vanjski plošni element protegne svojom širinom kroz dva pročelja koja imaju zajednički kut (slika 2.9 – a, slika 2.10). Drugi način je da granica elementa prati granicu pročelja ali s time da se odgovarajuće proširenje savija pod manjim kutom, kako bi bilo dovoljno mjesta za proširenje drugog elementa koji se nastavlja u susjednom pročelju(slika 2.9 – b).



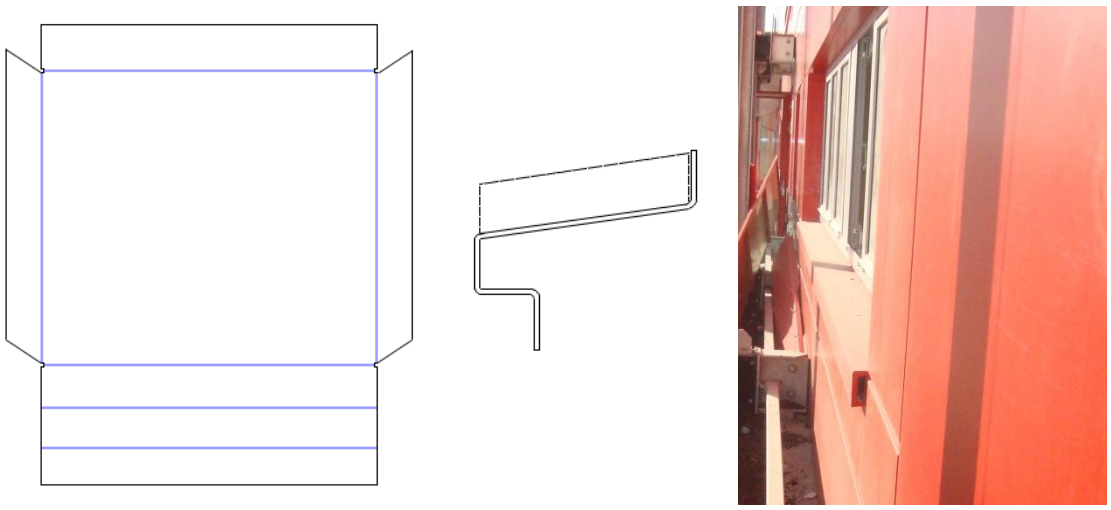
Slika 2.9 Pogledi odozgora na elemente sa kutnom obradom. a) Element koji se svojom širinom proteže kroz dva susjedna pročelja. b) Element koji prati granicu pročelja i zbog toga ima prilagođeno proširenje.



Slika 2.10 Razvijeni oblik elementa koji se proteže kroz dva susjedna pročelja te pogled nakon postavljanja na pročelje.

2.2.2 Obrada prozora i vrata

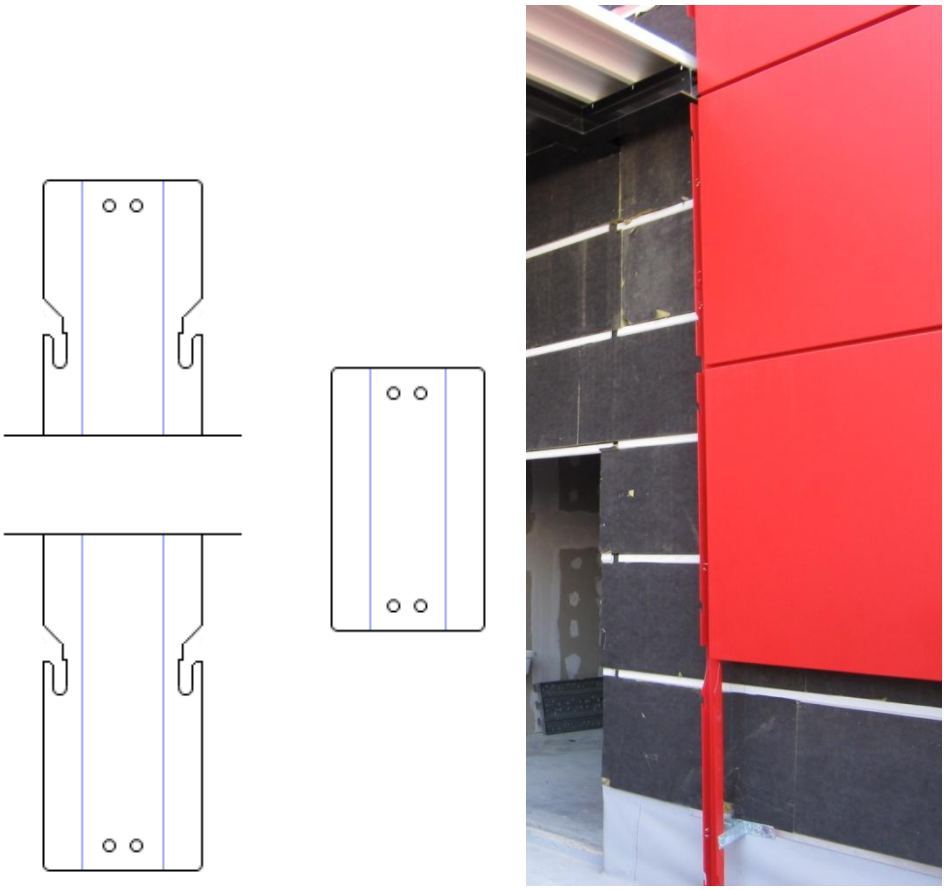
Prostor za postavljanje prozora i vrata je potrebno obraditi na poseban način. Okolni plošni elementi trebaju produžiti svoja proširenja kako bi se mogli pričvrstiti na umetnuti prozor odnosno vrata. Duljina proširenja ovisi o dubini prozora odnosno vrata koja se umeću. Plošni element koji se nalazi iznad treba imati produljeno donje proširenje. Element sa desne strane treba imati produljeno lijevo proširenje, a lijevi element desno proširenje. Ukoliko se umeće prozor, potrebno je nadodati još jedan pomoćni element na donji plošni element (slika 2.11). Taj je element potreban jer plošni elementi na gornjoj strani imaju specifično proširenje na koje se prozor ne može osloniti.



Slika 2.11 Razvijeni oblik potpornog elementa prozora, bočni presjek i fotografija sa gradilišta. Kao što se može vidjeti na bočnom presjeku, element sadrži stranu pod kutem i bočna proširenja kako bi se odvodila voda koja se nakuplja oko prozora.

2.3 Elementi podkonstrukcije

Podkonstrukcija je potpora vanjskim vidljivim elementima. Posebno oblikovani nosači omogućuju da se elementi pričvrste na točno određenu poziciju. Dimenzije ploča iz kojih se izrezuju nosači su 1000mm sa 2000mm, 1250mm sa 2500mm te 1500 sa 3000mm potrebno je spojiti više elemenata u cjelinu kako bi se moglo postaviti elemente u najvišim predjelima pročelja. Spajanje u cjelinu omogućuju spojnice koje se vijcima pričvršćuju za nosače. Nosač i spojnica su prikazani sljedećom slikom.



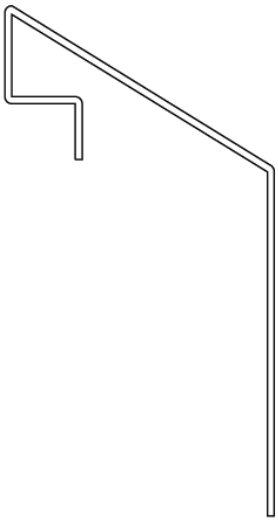
Slika 2.12 Razvijeni podkonstrukcijski nosač (skraćeni prikaz) i spojnica te fotografija nosača sa gradilišta.

Spojnica je jednostavan element koji na sebi ima tek rupe za vijke. Nosač, osim rupa za vijke pomoću kojih se pričvršćuje za spojnicu, sadrži rupe za vijke kojima se pričvršćuje na betonsku konstrukciju te izreze u koje se trebaju umetnuti vijci postavljeni na vanjske plošne elemente. Pozicija izreza određuje i poziciju na kojoj će se nalaziti vanjski plošni elementi.

2.4 Završni krovni elementi

Svako pročelje treba na vrhu imati posebne elemente koji će spriječiti prodiranje vode u unutrašnjost pročelja. Voda ponajprije može prodrijeti u unutrašnjost kao posljedica padalina. Takvi elementi se trebaju uklapati u vizualni dojam kojeg stvaraju vanjski plošni elementi (slika 2.13). Njihov razvijeni oblik predstavlja običnu traku koja ima specifična savijanja po širini (slika 2.14).

Slika 2.13 Pročelje sa postavljenim završnim krovnim elementima



2.14 Bočni presjek završnog krovnog elementa

3 Prikupljanje i analiza zahtjeva

Razvoj sustava započinje preciznom identifikacijom zahtjeva. Ovo poglavlje sadrži *design story* iz kojeg su „destilirani“ zahtjevi. Potom su na temelju zahtjeva određeni slučajevi korištenja.

3.1 Design story

Sustav za automatizaciju procesa izrade pročelja treba služiti kao podrška za izradu relativno jednostavnih tipiziranih pročelja. Najvažnije funkcionalnosti sustava su manipulacija elementima pročelja unutar aplikacije, pohrana elemenata pročelja u DXF formatu prikladnom za otvaranje u AutoCad sustavu te usporedno generiranje dokumentacije za svaki element.

Definiranje projekta

Korisnik, projektant u tehničkom uredu, nakon pokretanja aplikacije stvara novi projekt na kojem želi raditi. Projekt se sastoji od više pročelja čiji je broj i nazive potrebno unesti na početku. Potom se za svako pročelje definira mreža prema kojoj će se generirati vanjski plošni elementi. Mreža se može unesti na dva načina. Prvi je da se unesu visina i širina pročelja te broj podjela po visini i širini na jednake dijelove. Drugi način unošenja mreže je taj da se izravno unose visine redaka i širine stupaca. Prilikom definiranja mreže potrebno je navesti vrijednost razmaka između elemenata odnosno vrijednost fuge. Sve vrijednosti se unose u milimetrima. Potom je potrebno definirati vrstu materijala te generičke parametre prema kojima će se stvarati elementi pročelja. Definiciju navedenih svojstava je potrebno razdijeliti na posebne definicije za vanjske plošne elemente, elemente podkonstrukcije, te završne krovne elemente. Potrebni parametri za definiciju su navedeni u prilogu. Mogući materijali za izradu elemenata uključuju Aluminij, Inox, toplovaljani čelik (TV) i pocinčani čelik (VZ). Tablica materijala sa mogućim debljinama i pripadnim vrijednostima K faktora bitnim prilikom izračuna razvijenog oblika je također dana u prilogu.

Uređivanje elemenata

Nakon definicije projekta potrebno je prikazati vizualnu reprezentaciju generiranih vanjskih plošnih elemenata raspoređenih po pripadajućim pročeljima. Elementi se

prezentiraju onako kako bi izgledali prilikom frontalnog pogleda na pročelje te isprva svi imaju oblik pravokutnika. Potrebno je omogućiti uređivanje prikazanih elemenata. Isprva se mogu pomicati linije rubova pročelja što ima za posljedicu širenje, smanjivanje te preoblikovanje rubnih elemenata. Nakon što se uredi granica pročelja. Svaki vanjski plošni element pojedinačno se može urediti na sljedeće načine. Element može biti proširen ili smanjen po visini odnosno širini. Također, kutne točke elementa se mogu pomicati u smjerovima: gore, gore-desno, desno, dolje-desno, dolje, dolje-lijevo, lijevo, gore-lijevo; kako bi se dobili oblici sa jednom nakošenom stranicom. Posljedice takvih promjena moraju se uzeti u obzir prilikom kasnijeg generiranja razvijenog oblika elementa. Elemente je moguće i podijeliti na manje elemente te spajati u veće cjeline. Element se dijeli tako da se definira vrijednost visine ili širine prvog novog elementa dok će ostatak biti visina odnosno širina drugog novog elementa. Prilikom generiranja drugog novog elementa se uzima u obzir prostor za fugu. Potrebno je odabrati da li se dijeljenje obavlja po visini ili širini te strana(po visini: gore ili dolje; po širini: lijevo ili desno) od koje će se generirati prvi novi element prema unesenoj vrijednosti. Spajanje se odvija na način da se proizvoljni broj susjednih elementa spoji u cjelinu. Površina spojenog elementa će odgovarati zbroju površina odabranih elemenata uvećanoj za površinu fuge između njih. Dodatno se elementu mogu promijeniti vrsta materijala od kojeg je izrađen, te boja. Boje je potrebno postavljati u RAL industrijski standardnim vrijednostima.

Potrebno je omogućiti i obradu vanjskih plošnih elemenata prema poziciji na kojoj se nalaze. Obrada se može podijeliti na kutnu obradu i na obradu oko otvora.

Za kutnu obradu je potrebno označiti elemente koji se nalaze na vertikalnim rubovima pročelja te odabrati jedan od dva tipa obrade. Prvi tip nalaže obradu za element na način da će se on protezati kroz susjedno pročelje. Kod drugog tipa ne dolazi do produljenja elementa, ali je potrebno uzeti u obzir susjedni element koji će pripadati susjednom pročelju te sukladno tome napraviti prilagodbu.

Obrada oko otvora se obavlja na gotovo jednak način za oba tipa otvora (prozori i vrata). Potrebno je odabrati vanjski plošni element koji će simbolizirati otvor, a susjedne elemente prikladno obraditi. U slučaju obrade prozora, dodaje se još jedan element koji uvijek ima konstantne parametre osim širine koja je ovisna o širini prozora.

Tipovi pozicijskih obrada sa parametrima unosa su definirani u dodatku.

Nakon uređivanja vanjskih plošnih elemenata potrebno je generirati krovne elemente i podkonstrukciju na temelju parametara definiranih prilikom stvaranja projekta te trenutnog stanja pročelja. Krovni elementi se rade prema najvišim plošnim elementima pročelja. To znači da ukoliko ti elementi imaju gornju stranicu pod kutom, krovni elementi moraju pratiti tu stranicu. Podkonstrukcija se generira sukladno pozicijama plošnih elemenata i pripadnim vrijednostima parametra pozicije rupe za vijke. Na mjestu rupe za vijke potrebno je postaviti kuke na nosačima. Prilikom generiranja se automatski određuje broj i visine potrebnih nosača. U obzir treba uzeti dimenzije materijala koje je moguće naručiti, a to su ploče dimenzija 1000 mm sa 2000 mm, 1250 mm sa 2500 mm te 1500 mm sa 3000 mm. Spojne elemente nije potrebno vizualno prezentirati jer nema potrebe za izmjenama vrijednosti njihovih parametara.

U svakom trenutku se prate svojstva plošnih elemenata i nosača podkonstrukcije. Ukoliko se element nakon određene izmjene razlikuje od ostalih, to je potrebno naznačiti oznakom drugačijom od oznaka ostalih elemenata.

Izvoz elemenata

Nakon što su uređeni svi elementi pročelja, korisniku mora biti omogućena pohrana njihovog razvijenog oblika u datoteku .dxf formata te pohrana radioničke dokumentacije u .pdf formatu. Ukoliko su neki elementi identični za njih se generiraju zajednička .dxf datoteka i zajednička radionička dokumentacija. Prilikom generiranja razvijenih oblika potrebno je uzeti u obzir proširenja materijala koja nastaju na mjestima savijanja. Izračun proširenja je definiran u dodatku. Radionički dokument treba sadržavati različite poglede na element. Za vanjske plošne elemente trebaju biti prikazani frontalni, profilni i pogled odozgora. Za elemente podkonstrukcije treba prikazati profilni pogled i pogled odozgora. Za krovne elemente je dovoljno prikazati frontalni i profilni pogled. Svi pogledi trebaju imati prikazane mjere prema opisu u dodatku. Uz poglede, potrebno je i navesti boju elementa, broj identičnih elemenata za izradu te svojstva materijala. Dokument dodatno sadrži informacije: Naziv projekta, datum izrade, rok izrade, ime i prezime projektanta, ime i prezime odgovorne osobe, mjesto dostave elementa te druge informacije navedene u dodatku. Te informacije se trebaju moći mijenjati u opcijama Izuzetno, u situacijama kada se vanjski plošni elementi razlikuju samo po visini i širini, spremaju se u jedan dokument. Takav dokument ima generičke poglede te jednu dodatnu tablicu u kojoj se nalaze širine i visine pripadajućih plošnih elemenata.

Također je potrebno pohraniti frontalni pogled na pročelje u .dxf datoteci. Na pogledu moraju biti prikazani elementi sa njihovim oznakama te naziv pročelja.

Datoteke se pohranjuju u mapu na putanji koju korisnik može odrediti u opcijama. Naziv mape treba odgovarati nazivu projekta. U glavnoj mapi se stvaraju nove mape s nazivima pročelja projekta. Te mape sadrže frontalni pogled na pročelje te mape: „Plošni elementi“, „Podkonstrukcija“ i „Krovni elementi“. U navedene mape se spremaju datoteke pripadajućih elemenata. Radionička i .dxf datoteka se imenuju na jednak način, a razlikuje ih samo ekstenzija. Format imenovanja je sljedeći:

<tip elementa><oznaka>_<broj_komada_za_izradu>.ekstenzija

Izračun potrebnog materijala

Kao dodatnu funkcionalnost potrebno je omogućiti izračun optimalne količine materijala potrebnog za izradu elemenata pročelja. Izvješće treba navesti broj i dimenzije potrebnih ploča, prikazati raspored elemenata na pločama te navesti vrijednost iskorištenog materijala u postotcima.

3.2 Zahtjevi

3.2.1 Funkcionalni zahtjevi

1. Stvaranje novog projekta
   - Definiranje naziva projekta
   - Definiranje broja i imena pripadajućih pročelja
   - Definiranje mreže za inicijalno generiranje početnih elemenata
   - Definiranje generičkih parametara i svojstava materijala za plošne elemente, podkonstrukcijske elemente, krovne elemente

2. Uređivanje granica pročelja
   - Pomicanje točaka granice pročelja u smjerovima: gore, gore – desno, desno, dolje-desno, dolje, dolje – lijevo, lijevo, gore-desno
   - Umetanje nove točke granice
   - Automatsko uređivanje graničnih plošnih elemenata

3. Uređivanje plošnih elemenata
    - Promjena oblika prema rubnim točkama
    - Promjena materijala
    - Promjena boje
    - Promjena parametara (duljine proširenja, pozicije rupa za vijke, promjeri rupa...)
    - Brisanje elemenata

4. Dijeljenje plošnih elementa
    - Odabir strane dijeljenja (gornja, donja, lijeva ili desna)
    - Unos vrijednosti duljine ili širine prvog elementa koji će nastati dijeljenjem

5. Spajanje plošnih elemenata
    - Višestruki odabir elemenata za spajanje

6. Preoblikovanje plošnih elemenata u kutne elemente
    - Odabir tipa kutnog elementa
    - Unos vrijednosti kuta ili produljenja (ovisno o odabranom tipu)

7. Preoblikovanje plošnih elemenata u elemente obrade otvora (prozora i vrata)
    - Odabir plošnog elementa koji predstavlja prozor ili vrata
    - Odabir tipova obrade za gornje elemente i elemenata sa strane
    - Generiranje potpornog elementa u slučaju obrade prozora

8. Generiranje elemenata podkonstrukcije
    - Generiranje podkonstrukcijskih nosača
    - Generiranje spojnica podkonstrukcijskih nosača

9. Uređivanje podkonstrukcijskih nosača
    - Promjena boje i materijala nosača

10. Generiranje krovnih elemenata
    - Promjena boje i materijala krovnih elemenata

11. Pohrana razvijenih oblika elemenata i pogleda pročelja u .dxf format

12. Uređivanje podataka radioničke dokumentacije
    - Unos imena i prezimena projektanta
    - Unos imena i prezimena odgovorne osobe
    - Unos mjesta dostave
    - Unos roka izrade

13. Pohrana radioničke dokumentacije elemenata u .pdf formatu

14. Izračun optimalne količine potrebnog materijala

3.2.2 Nefunkcionalni zahtjevi

1. Promjena boje pozadine uređivača pročelja

2. Obavijesti o stanju aplikacije prilikom dugotrajnog izvođenja
    - Poruke o stanju tijekom pohrane u .dxf i radioničke dokumentacije
    - Poruke tijekom izvođenja algoritma za optimizaciju materijala

3. Praćenje promjena nad elementima pročelja tokom rada putem oznaka elemenata

4. Opcija prikaza razvijenog oblika elementa tokom rada

5. Validacija pri unosu vrijednosti

6. Onemogućavanje odabira operacija koje se s obzirom na trenutno stanje aplikacije ne mogu izvesti

3.2.3 Korisnički zahtjevi

1. Onemogućavanje odabira operacija koje se s obzirom na trenutno stanje aplikacije ne mogu izvesti

2. Funkcionalno sučelje tokom dugotrajnih operacija

3.3 Slučajevi korištenja

Slijedi popis slučajeva korištenja grupiranih prema funkcionalnim zahtjevima.

3.3.1 Definicija projekta

1. Stvaranje projekta

1. Korisnik iz glavnog izbornika odabire Datoteka → Novi projekt.
2. Sustav otvara čarobnjak za izradu novog projekta.
3. Korisnik unosi naziv projekta i nazive pripadnih pročelja te odabire opciju „Sljedeći“.
4. Korisnik unosi mrežu za inicijalno generiranje plošnih elemenata na jedan od dva ponuđena načina te odabire opciju „Sljedeći“.
5. Korisnik unosi parametre i svojstva materijala elemenata te odabire opciju „Završi“.
6. Sustav generira elemente pročelja te svako pročelje prikazuje zasebno u drugoj Tab kontroli

3.3.2 Upravljanje granicom pročelja

2. Promjena granica pročelja

1. Korisnik izabire opciju „prikaži granice pročelja“.
2. Korisnik odabire jednu od ponuđenih točaka granica pročelja.
3. Korisnik unosi vrijednost promjene granice te odabire jednu od opcija smjera promjene granice.
4. Sustav mijenja oblike rubnih elemenata kako bi oni bili prilagođeni novoj granici pročelja

3. Unos nove točke granice pročelja

1. Korisnik izabire opciju „Prikaži granice pročelja“.
2. Korisnik izabire ponuđenu opciju „Dodaj novu točku granice pročelja“.
3. Korisnik unosi koordinate nove točke i odabire opciju „U redu“

3.3.3 Upravljanje plošnim elementima

4. Pregled svojstava plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja
2. Sustav prikazuje informacije elementa na kontrolnom pogledu „Opcije pročelja“:
    - Širina
    - Visina
    - Vrsta materijala
    - Debljina materijala
    - Boja

5. Promjena oblika plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja.
2. Korisnik odabire jednu od ponuđenih točaka za promjenu oblika elementa.
3. Korisnik unosi vrijednost promjene te odabire jednu od opcija smjera promjene oblika.
4. Sustav osvježava vrijednost oznake za raspoznavanje različitih elemenata.

6. Promjena boje plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja.
2. Korisnik unosi numeričku vrijednost boje prema RAL standardu.
3. Sustav mijenja boju elementa i osvježava njegovu vizualnu reprezentaciju.

7. Promjena parametara plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju „Svojstva“.
3. Sustav prikazuje dijalog sa ponuđenim parametrima.
4. Korisnik mijenja vrijednosti parametara.
5. Korisnik potvrđuje promjene odabirom opcije „U redu“.

8. Promjena svojstva materijala plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju „Materijal“.
3. Sustav prikazuje dijalog sa svojstvima materijala.

4. Korisnik mijenja vrstu materijala ili neka od svojstva potrebna za računanje produženja materijala.
5. Sustav automatski sa promjenom vrijednosti osvježava vrijednost proširenja materijala prilikom savijanja.
6. Korisnik potvrđuje promjene odabirom opcije „U redu“.

9. Brisanje plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju “Brisanje”

10. Dijeljenje plošnog elementa

1. Korisnik odabire željeni element na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju “Podjela”
3. Sustav prikazuje dijalog sa opcijama podjele.
4. Korisnik odabire horizontalnu ili vertikalnu podjelu.
5. Korisnik odabire stranu podjele te unosi dimenzijsku vrijednost prvog podijeljenog elementa.
6. Korisnik potvrđuje unos odabirom opcije “U redu”.
7. Sustav osvježava stanje na pogledu u skladu sa novonastalim stanjem.

11. Spajanje plošnih elemenata

1. Korisnik odabire željene susjedne elemente na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju “Spoji”.
3. Sustav osvježava stanje na pogledu pročelja u skladu sa novonastalim stanjem.

12. Preoblikovanje plošnih elemenata u kutne elemente

1. Korisnik odabire jedan ili više elemenata za preoblikovanje.
2. Korisnik odabire ponuđenu opciju “Preoblikuj u kutni element”.
3. Sustav prikazuje dijalog sa opcijama kutnog preoblikovanja.
4. Korisnik odabire način preoblikovanja.
5. Korisnik unosi vrijednost za odabrani način preoblikovanja(duljinu za produljenje ili kut za kutno preoblikovanje).
6. Korisnik potvrđuje unos odabirom opcije “U redu”.

13. Preoblikovanje plošnih elemenata u elemente obrade otvora

1. Korisnik odabire plošni element za preoblikovanje u element otvora.
2. Korisnik odabire ponuđenu opciju “Preoblikuj u otvor”.
3. Sustav prikazuje dijalog sa opcijama obrade otvora:
    - Tip obrade gornjeg susjednog elementa
    - Tip obrada straničnih susjednih elemenata
    - Tip donjeg potpornog elementa u slučaju vrste otvora ”Prozor”
4. Korisnik odabire tipove obrade i unosi vrijednost dubine otvora.
5. Korisnik potvrđuje unos odabirom opcije “U redu”.
6. Sustav osvježava stanje na pogledu pročelja u skladu sa novonastalim stanjem.

3.3.4 Upravljanje podkonstrukcijom

14. Generiranje podkonstrukcije

1. Korisnik odabire opciju “Generiraj podkonstrukciju”.
2. Sustav generira podkonstrukcijske elemente.
3. Sustav pretvara pogled pročelja u podkonstrukcijski pogled.

15. Pregled svojstava nosača podkonstrukcije

1. Korisnik odabire željeni element na pogledu pročelja
2. Sustav prikazuje informacije elementa na kontrolnom pogledu „Opcije pročelja“:
    - Širina
    - Visina
    - Vrsta materijala
    - Debljina materijala
    - Boja

16. Promjena boje podkonstrukcijskog nosača

1. Korisnik odabire željeni nosač na pogledu pročelja.
2. Korisnik unosi numeričku vrijednost boje prema RAL standardu.
3. Sustav mijenja boju nosača i osvježava njegovu vizualnu reprezentaciju.

17. Promjena svojstva materijala podkonstrukcijskog nosača

1. Korisnik odabire željeni nosač na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju „Materijal“.
3. Sustav prikazuje dijalog sa svojstvima materijala.
4. Korisnik mijenja vrstu materijala ili neka od svojstva potrebna za računanje produženja materijala.
5. Sustav automatski sa promjenom vrijednosti osvježava prikazanu vrijednost proširenja materijala prilikom savijanja.
6. Korisnik potvrđuje promjene odabirom opcije „U redu“.

3.3.5 Upravljanje krovnim elementima

18. Generiranje krovnih elemenata

1. Korisnik označava željene gornje plošne elemente.
2. Korisnik odabire opciju “Generiraj podkonstrukciju”.
3. Sustav generira krovne elemente.
4. Sustav osvježava stanje na pogledu pročelja u skladu sa novonastalim stanjem.

19. Promjena boje krovnog elementa

1. Korisnik odabire željeni krovni element na pogledu pročelja.
2. Korisnik unosi numeričku vrijednost boje prema RAL standardu.
3. Sustav mijenja boju nosača i osvježava njegovu vizualnu reprezentaciju.

20. Promjena svojstva materijala krovnog elementa

1. Korisnik odabire željeni krovni element na pogledu pročelja.
2. Korisnik odabire ponuđenu opciju „Materijal“.
3. Sustav prikazuje dijalog sa svojstvima materijala.
4. Korisnik mijenja vrstu materijala ili neka od svojstva potrebna za računanje produženja materijala.
5. Sustav automatski sa promjenom vrijednosti osvježava prikazanu vrijednost proširenja materijala prilikom savijanja.
6. Korisnik potvrđuje promjene odabirom opcije „U redu“.

3.3.6 Izvoz razvijenog oblika i radioničke dokumentacije

21. Pohrana razvijenih elemenata i radioničke dokumentacije

1. Korisnik odabire opciju "Izvoz u .dxf i .pdf".
2. Sustav izvještava korisnika o napretku pohrane.
    a. Sustav generira .dxf datoteke sa razvijenim elementima i radioničku dokumentaciju grupirano prema:
        - Plošnim elementima
        - Krovnim elementima
        - Podkonstrukcijskim elementima
    b. Sustav generira poglede pročelja u .dxf formatu.

22. Uređivanje podataka radioničke dokumentacije

1. Korisnik iz glavnog izbornika odabire "Opcije → Radionička dokumentacija"
2. Korisnik mijenja vrijednosti informacija za radioničku dokumentaciju:
    - Broj nacrta
    - Ime i prezime projektanta
    - Ime i prezime odgovorne osobe
    - Mjesto dostave
    - Rok izrade
3. Korisnik potvrđuje unos odabirom opcije „U redu".

3.3.7 Optimizacija potrebne količine materijala

23. Izračun optimalne količine potrebnog materijala

1. Korisnik iz glavnog izbornika odabire „Opcije → Optimizacija materijala".
2. Sustav otvara novi prozor sa parametrima i kontrolama za praćenje izvođenja optimizacijskog algoritma.
3. Korisnik unosi parametre algoritma.
4. Korisnik odabire pročelja čiji elementi sudjeluju u optimizacijskom procesu.
5. Korisnik pokreće izvođenje algoritma odabirom opcije „Pokreni".
6. Sustav izvještava korisnika o napretku u pronalasku optimalnog rješenja:
    a. Porukama o vrijednostima iskoristivosti materijala trenutnih rješenja
    b. Vizualnim prikazom trenutno najboljeg rješenja
7. Korisnik potvrđuje unos odabirom opcije „U redu".

3.3.8 Upravljanje na razini projekta

24. Izmjena putanje za pohranu

1. Korisnik iz glavnog izbornika odabire „Opcije → Generalno“
2. Sustav prikazuje dijalog sa generalnim opcijama.
3. Korisnik unosi putanju ili je definira putem „Browse“ dijaloga.
4. Korisnik potvrđuje unos odabirom opcije „U redu“.

25. Prikaz kontrolnih pogleda

1. Korisnik iz glavnog izbornika odabire „Pogled“
2. Korisnik odabire pogled koji će se prikazati od mogućih:
    - Projekt
    - Opcije pročelja
3. Sustav otvara i/ili stavlja fokus na odabrani pogled.

3.3.9 Upravljanje na razini pročelja

26. Uklanjanje pročelja iz projekta

1. Korisnik iz glavnog izbornika odabire „Pogled → Projekt“.
2. Korisnik iz hijerarhijski uređenog pogleda na projekt i pročelja odabire pročelje za uklanjanje.
3. Korisnik odabire opciju „Obriši“ iz kontekstualnog izbornika.
4. Sustav zahtjeva potvrdu uklanjanja pročelja.
5. Korisnik potvrđuje uklanjanje.

27. Preimenovanje pročelja

1. Korisnik iz glavnog izbornika odabire „Pogled → Projekt“
2. Korisnik iz hijerarhijski uređenog pogleda na projekt i pročelja odabire pročelje za preimenovanje.
3. Korisnik odabire opciju „Preimenuj“ iz kontekstualnog izbornika.
4. Korisnik izravno na pogledu unosi novi naziv pročelja.

28. Dodavanje pročelja u projekt

1. Korisnik iz glavnog izbornika odabire „Novo pročelje“.
2. Sustav dodaje novo pročelje pod defaultnim nazivom u projekt te novu površinu za uređivanje pročelja.
3. Korisnik odabire jednu od dvije ponuđene opcije za generiranje mreže inicijalnih plošnih elemenata te unosi vrijednosti
4. Korisnik potvrđuje unos odabirom opcije „U redu“.

29. Brisanje sadržaja pročelja

1. Korisnik odabire opciju „Brisanje sadržaja“
2. Sustav zahtjeva potvrdu brisanja sadržaja
3. Korisnik potvrđuje brisanje

30. Izmjena boje pozadine uređivača pročelja

1. Korisnik iz glavnog izbornika odabire „Opcije → Generalno“
2. Sustav prikazuje dijalog sa generalnim opcijama.
3. Korisnik odabire opciju sa prikazanom trenutnom bojom pozadine.
4. Korisnik iz ponuđene palete odabire novu boju pozadine.

31. Zoom radne površine pročelja

1. Korisnik regulira zoom vrijednost kotačićem miša

2. Sustav izvodi zoom površine za uređivanje pročelja sa središtem na točki pozicije kursora te osvježava prikazanu zoom vrijednost.

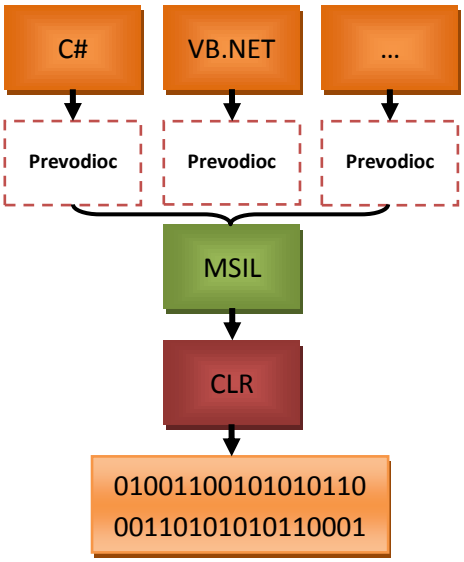
4 .NET Framework i WPF

U ovom poglavlju je dan kratki osvrt na .NET radni okvir te njegov sastavni dio Windows Presentation Foundation. Detaljnije su opisane mogućnosti koje su učestalo korištene tokom izrade praktičnog dijela rada.

4.1 .NET Framework

Razvoj .NET radnog okvira je započet 90-ih godina pod okriljem Microsofta. Cilj je bio proizvesti bogati skup interoperabilnih alata, tehnologija i programskih jezika koji će omogućiti izradu široke palete softwareskih proizvoda. Krajem 2000. godine je dovršeno beta izdanje prve inačice .NET-a. Radni okvir se od tada konstantno nadograđivao, a trenutno je aktualna inačica 4.0.

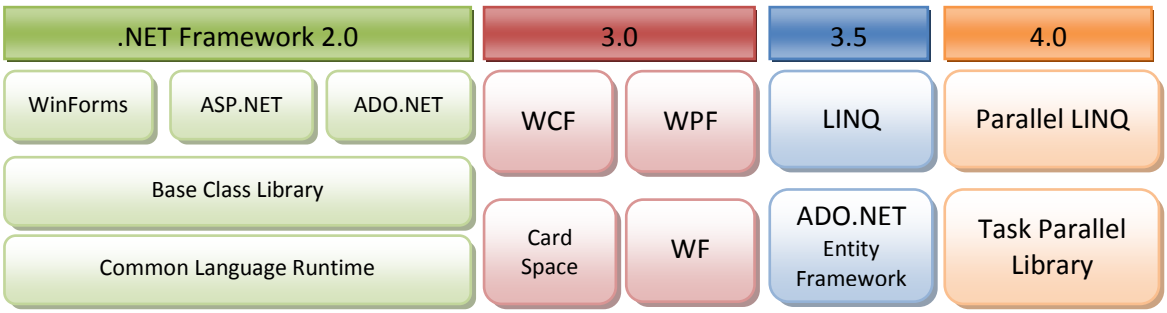
.NET okvir se sastoji od *Common Language Runtime* (CLR) sustava i skupa knjižnica. CLR je sustav koji omogućuje prevođenje višeg programskog jezika u jezik razumljiv računalu te njegovo pokretanje. Za prevođenje u realnom vremenu je zadužen JIT (eng. *Just In Time*) prevodioc. Takvo prevođenje u strojni kôd računala omogućilo je .NETu prelazak na druge operativne sustave uz pomoć third-party sustava(npr. MONO). Međutim, CLR ne obrađuje izravno viši programski jezik, već mu sav kôd dolazi iz MSIL sloja. MSIL, odnosno *Microsoft Intermediate Language* predstavlja jedan zajednički jezik u koji se prevodi kôd pisan u nekom od viših jezika. Tu zadaću prevođenja obavlja IL prevodioc. Microsoft je izdao IL prevodioce za jezike: C#, J#, C++, Visual Basic i Jscript. Osim navedenih, postoje i nezavisno razvijeni prevodioci za jezike: Perl, Python, Cobol, Eiffel i druge. Ovakav način prevođenja omogućuje razvoj aplikacija pisanih u više različitih jezika.



Slika 4.1 Shematski prikaz procesa prevođenja iz višeg programskog jezika u strojni kôd [4]

.NET radni okvir sadrži brojne knjižnice razreda. Osnovna funkcionalnost poput čitanja i pisanja u datoteke, grafičkog renderinga, interakcije s bazom podataka i manipulacije XML dokumentima je sadržana u *Base Class Library* skupu knjižnica. Base Class Library je dostupan svim .NET programskim jezicima. Framework Class Library

obuhvaća sve knjižnice radnog okvira, pa tako uz Base Class Library sadrži i proširene knjižnice WinForms, ADO.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation i druge.



Slika 4.2 Razvoj .NET radnog okvira kroz inačice [4]

4.2 Windows Presentation Foundation

Windows Presentation Foundation, ili skraćeno WPF, je prezentacijski sustav za izradu klijentskih Windows aplikacija sa bogatim korisničkim sučeljima. WPF jezgru čini rezolucijski neovisan vektorski *rendering* sustav koji je napravljen kako bi što bolje iskoristio mogućnosti modernog grafičkog hardvarea. WPF proširuje .NET okvir sa opsežnim skupom razvojnih mogućnosti koje uključuju Extensible Markup Language (XAML), kontrole, podatkovne poveznice, 2D i 3D grafiku, animacije, stilove, predloške, dokumente, audio i video medije te brojne druge. Razredi WPF-a se nalaze unutar Framework Class Librarya unutar System.Windows prostora imena.

WPF aplikacije je moguće razvijati korištenjem XAML *markup* jezika i nekog od .NET viših jezika u pozadini. U XAML-u se oblikuje korisničko sučelje dok se u pozadinskom kôdu definira ponašanje. Na taj način su troškovi razvoja i održavanja smanjeni jer kôd grafičkog sučelja nije usko povezan sa kôdom ponašanja sučelja. Također, razvoj je učinkovitiji jer dizajneri mogu oblikovati izgled sučelja paralelno sa programerima koji implementiraju ponašanje aplikacije. Omogućena je i izrada posebnih alata za oblikovanje korisničkog sučelja (npr. Microsoft Blend). XAML je markup jezik baziran na XML sintaksi. Uobičajeno se koristi za stvaranje prozora, dijaloga, stranica i korisničkih kontrola te za njihovo popunjavanje kontrolama, oblicima i grafičkim elementima. Sljedećim programskim isječkom je prikazana XAML definicija prozora koji sadrži Button kontrolu.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.AWindow"
    Title="Window with Button"
    Width="250" Height="100">

    <Button Name="button" Click="button_Click">Click Me!</Button>
</Window>
```

Programski isječak 4.1 XAML definicija prozora koji sadrži Button kontrolu

Sljedeći isječak prikazuje pozadnski kôd za prethodni slučaj. Implementirano je ponašanje za pritisak na Button kontrolu.

```
public partial class AWindow : Window
{
    public AWindow()
    {
        InitializeComponent();
    }

    void button_Click(object sender, RoutedEventArgs e)
    {
         MessageBox.Show("Hello WPF");
    }
}
```

Programski isječak 4.2 Pozadinski kôd s ponašanjem uslijed pritiska na Button kontrolu.

Slijedi opis mogućnosti WPF-a važnih u kontekstu praktičnog dijela rada.

4.2.1 Kontrole

U skupinu kontrola spadaju svi WPF razredi koje je moguće koristiti unutar prozora ili stranica. Svaka kontrola ima definirano sučelje te ponašanje. Prema namjeni se mogu podijeliti na:

- Button kontrole: Button, RepeatButton
- Prozori dijaloga: OpenFileDialog, SaveFileDialog te PrintDialog
- Kontrole za prikaz podataka: DataGrid, ListView te TreeView
- Kontrole za prikaz i odabir datuma: Calendar te DatePicker
- Kontrole unosa: TextBox, RichTextBox te PasswordBox
- Panel kontrole: Canvas, DockPanel, Grid, StackPanel, WrapPanel...
- Media: Image, MediaElement te SoundPlayerAction
- Menu kontrole: ContextMenu, Menu te ToolBar
- Navigacijske kontrole: Frame, HyperLink, Page, NavigationWindow te TabControl
- Kontrole odabira: CheckBox, ComboBox, ListBox, RadioButton te Slider

- Informacijske kontrole: AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock te ToolTip

Izgled kontrole se može promijeniti izmjenom vrijednost svojstava. Ukoliko bi se željelo postaviti izgled više kontrola na jednom mjestu to se može postići primjenom stilova. Svaka kontrola može u potpunosti promijeniti izgled, pa tako i preuzeti izgled neke druge kontrole. Tako drastična promjena sučelja se postiže postavljanjem ControlTemplate svojstva. Slijedi primjer postavljanja ControlTemplate svojstva Button kontrole.

```
<ControlTemplate TargetType="{x:Type Button}">
    <Grid>
        <Ellipse Fill="{TemplateBinding Background}"
                 Stroke="{TemplateBinding BorderBrush}"/>
        <ContentPresenter HorizontalAlignment="Center"
                          VerticalAlignment="Center"/>
    </Grid>
</ControlTemplate>
```

Programski isječak 4.3 Primjer postavljanja ControlTemplate svojstva

Na ovaj način vanjski izgled Button kontrole poprima izgled elipse.

Slika 4.3 Button kontrola u obliku elipse

Većina kontrola može sadržavati druge kontrole. Tako, na primjer, labela može sadržavati znakovni niz, sliku ili neki drugi element. To se postiže nasljeđivanjem posebnih Content razreda:

- **ContentControl** – Razred predviđen za sadržavanje jedne kontrole. Međutim ta kontrola može biti tipa Panel tako da ustvari može sadržavati i više kontrola. Primjeri: Label, Button i Tooltip.
- **ItemsControl** – Omogućuje sadržavanje više elemenata ili podataka. Primjeri: ListBox, Menu i StatusBar
- **HeaderedContentControl** – Razred sa svojstvima ContentControl razreda koji omogućuje postavljanje naslova. Primjeri: TabItem, GroupBox i Expander.
- **HeaderedItemsControl** – Razred sa svojstvima ItemsControl razreda koji omogućuje postavljanje naslova. Primjeri: MenuItem, TreeViewItem i ToolBar.

Kontrole koje su namjenjene prikazu podataka poput ListBox kontrole, mogu urediti način njihovog prikaza korištenjem DataTemplate razreda. Definiranje DataTemplate elementa je slično definiranju ContentTemplate elementa.

U WPF-u se razlikuju još dva koncepta kontrola: UserControls i CustomControls. UserControls se mogu razmatrati kao kompozicija drugih kontrola. Stvaranje takvih kontrola nalikuje stvaranju prozora. Po kreiranju kontrole se generira XAML datoteka i pripadajući code-behind razred koji nasljeđuje UserControl razred. CustomControls su kontrole koje imaju preuređenu funkcionalnost standardne WPF kontrole ili su potpuno iznova definirane. Za implementaciju Custom kontrole je potrebno da razred nasljeđuje razred neke kontrole (npr. Button), ili nekog nadrazreda ukoliko se definira sasvim novi tip kontrole (npr. FrameworkElement). Unutar razreda je potrebno implementirati DependencyProperties svojstva koja će biti dostupna kroz XAML uređivanje. Na kraju je potrebno definirati generičku temu odnosno defaultni izgled kontrole unutar \Themes\generic.xaml datoteke.

4.2.2 Stilovi

Stilovi omogućuju konzistentnost izgleda korisničkog sučelja. Njima se globalno postavljaju svojstva više od jednog elementa sučelja. Definiranje stilova je nužno kako bi se omogućilo održavanje i dijeljenje vizualnog dojma unutar i između aplikacija. Stil je moguće postaviti za bilo koji element koji nasljeđuje FrameworkElement ili FrameworkContentElement razred. Uobičajeno se postavljaju u *resources* djelu prozora ili nekog drugog elementa sa sadržajem. Sljedeći isječak kôda opisuje postavljanje stila na sve elemente tipa TextBlock.

```
<Window.Resources>
    ...
    <Style TargetType="TextBlock">
    <Setter Property="HorizontalAlignment" Value="Center" />
    <Setter Property="FontFamily" Value="Comic Sans MS"/>
    <Setter Property="FontSize" Value="14"/>
    </Style>
    ...
</Window.Resources>
```

Programski isječak 4.4 Primjer stila

Ukoliko je potrebno premostiti vrijednosti globalno postavljenog stila moguće je kao metu stila postaviti imenovanu kontrolu ili se može na razini kontrole postaviti drugačiji stil.

U slučaju da je potrebno za neke elemente postaviti dodatna svojstva, stilove je moguće i proširiti. Na slijedeći način.

```
<Style BasedOn="{StaticResource {x:Type TextBlock}}"
       TargetType="TextBlock"
       x:Key="TitleText">
    <Setter Property="FontSize" Value="26"/>
    <Setter Property="Foreground">
    <Setter.Value>
        <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
        <LinearGradientBrush.GradientStops>
            <GradientStop Offset="0.0" Color="#90DDDD" />
            <GradientStop Offset="1.0" Color="#5BFFFF" />
        </LinearGradientBrush.GradientStops>
        </LinearGradientBrush>
    </Setter.Value>
    </Setter>
</Style>
```

Programski isječak 4.5 Premošćivanje globalno postavljenog stila

4.2.3 Grafički elementi

WPF sadrži knjižnicu uobičajenih vektorskih 2D elemenata pod nazivom Shapes. Knjižnici pripadaju razredi Ellipse, Line, Path, Polygon, Polyline te Rectangle za iscrtavanje istoimenih elemenata. Moguće ih je postaviti unutar panela i većine kontrola. Osim svojstava prikaza, Shape objekti sadrže i mnoga svojstva koje se nalaze u običnim kontrolama. Tako da je na primjer moguće odgovoriti na klik mišem na pojedini objekt kao u sljedećem XAML odsječku i pozadinskom kôdu.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.EllipseEventHandlingWindow"
    Title="Click the Ellipse">
    <Ellipse Name="clickableEllipse" Fill="Blue"
                    MouseUp="clickableEllipse_MouseUp" />
</Window>
```

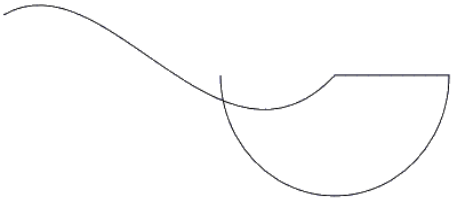
Programski isječak 4.6 Rukovanje događajem koji se javlja na klik mišem

```
void clickableEllipse_MouseUp(object sender, MouseButtonEventArgs e)
{
    MessageBox.Show("You clicked the ellipse!");
}
```

Programski isječak 4.7 XAML omogućavanje odgovora na klik mišem

Pomoću navedenih razreda je jednostavno prikazati osnovne geometrijske oblike. Za iscrtavanje kompliciranih oblika se koristi Path razred. Međutim, instanca tog razreda

nije dovoljna sama za sebe. Kako bi se definirale granice složenog oblika, koriste se Geometry elementi. Razlika između Geometry i Shape objekata je u tome što se Geometry elementi nemogu samostalno prikazati, ali imaju puno raznovrsnije mogućnosti. Tako se pomoću Geometry razreda mogu definirati regije za izrezivanje sadržaja(eng. *clipping*), regije za pronalazak sadržanih elemenata (*hit testing*) itd. Da bi se definirao oblik koji će se prikazati, instanci Path razreda je potrebno za svojstvo Data postaviti instancu Geometry elementa. Ukoliko bi se željelo upotrijebiti više geometrijskih oblika potrebno je Data svojstvu pridružiti PathGeometry objekt. Takav objekt može sadržavati jednu ili više cjelina (PathFigure), a geometrijski oblik se, konačno, definira pomoću segmenata koji mu se pridružuju. Razlikuju se: ArcSegment, BezierSegment, LineSegment, PolyBezierSegment, PolyLineSegment, PolyQuadraticBezierSegment te QuadraticBezierSegment. Uz pomoć navedenih segmenata je moguće definirati proizvoljan oblik poput onog prikazanog slikom 4.4.

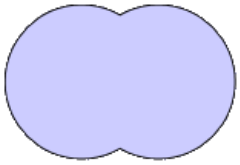


Slika 4.4 Primjer Path oblika

Također je moguće obavljati i geometrijske operacije unije, oduzimanja i presjeka. CombinedGeometry razred ima svojstvo GeometryCombineMode koje služi za postavljanje željene operacije. Za svojstva Geometry1 i Geometry2 se postavljaju vrijednosti Geometry objekata nad kojima se želi obaviti željena operacija. Sljedeći isječak koda definira Uniju dvaju elemenata čiji je rezultat prikazan slikom 4.5.

```
<Path Stroke="Black" StrokeThickness="1" Fill="#CCCCFF">
    <Path.Data>
    <CombinedGeometry GeometryCombineMode="Union">
      <CombinedGeometry.Geometry1>
        <EllipseGeometry RadiusX="50" RadiusY="50" Center="75,75" />
      </CombinedGeometry.Geometry1>
      <CombinedGeometry.Geometry2>
        <EllipseGeometry RadiusX="50" RadiusY="50" Center="125,75" />
      </CombinedGeometry.Geometry2>
    </CombinedGeometry>
    </Path.Data>
</Path>
```

Programski isječak 4.8 XAML kôd za izvođenje operacije unije dvaju grafičkih elemenata



4.5 Rezultat prethodno definirane operacije

4.2.4 Transformacije

Transformacije definiraju kako preslikati točke iz jednog koordinatnog sustava u drugi. Moguće ih je postaviti na kontrole, grafičke elemente te na sve druge objekte izvedbe iz FrameworkElement razreda. Podržane su slijedeće transformacije:

- RotateTransform – Zakreće element za određeni kut
- ScaleTransform – Skalira element za određene vrijednosti po x i y osi
- SkewTransform – Izokreće element za određeni x i y kut
- TranslateTransform – Translatira element za određene x i y vrijednosti

Dodatno postoje još dvije vrste transformacija: TransformGroup i MatrixTransform. TransformGroup objedinjuje više transformacija kako bi se mogle izvesti istovremeno. MatrixTransform je transformacija koja nudi proizvoljno definiranje prema transformacijskoj matrici. Matrica sadrži vrijednosti M11, M12, M21, M22, OffsetX te OffsetY. Navedene vrijednosti definiraju preračunavaju trenutne vrijednosti u vrijednosti nakon transformacije prema sljedećim formulama.

$$X' = M11 * X + M21 * Y + OffsetX$$

$$Y' = M12 * X + M22 * Y + OffsetY$$

Formula 4.1 Transformacijski izrazi

Neutralne vrijednosti matrice su: $\begin{pmatrix} M11 & M12 \\ M21 & M22 \\ OffsetX & OffsetY \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$

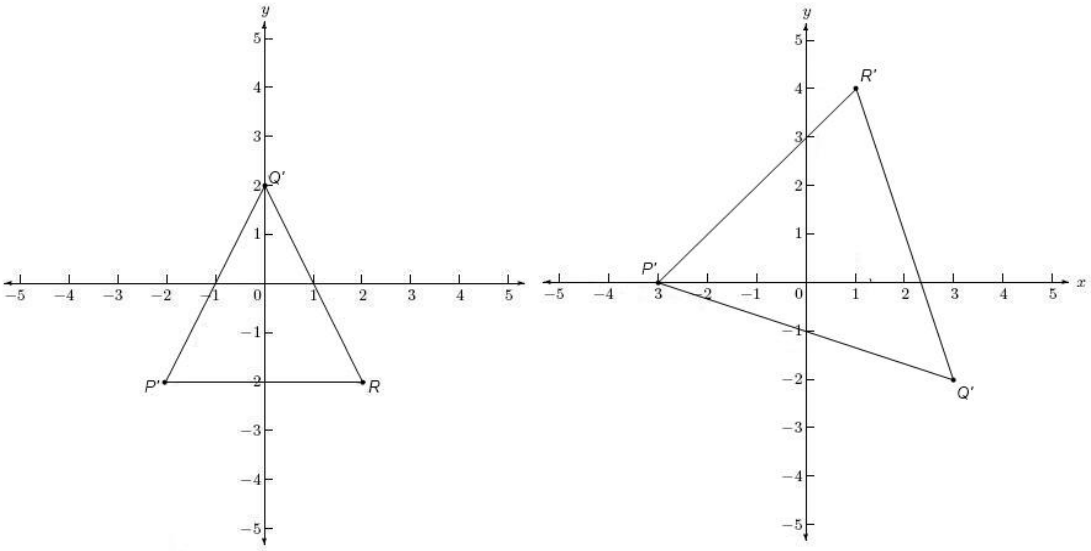
Ako se uzme za primjer trokut sa koordinatama točaka: P(-2,-2), Q(0,2) i R(2,.-2) i primjeni transformacija sa matricom: $\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix}$

Transformacijske formule su sljedećeg oblika:

$$X' = X + Y + 1$$

$$Y' = X - Y$$

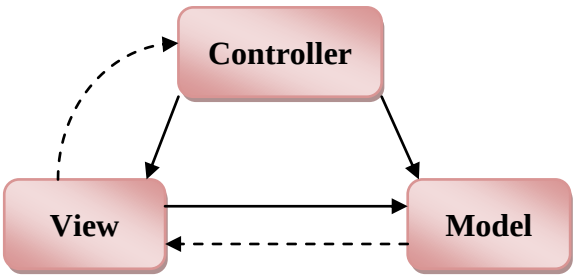
Točke transformiranog trokuta su: P(-3,0), Q(3,-2) te R(1, 4).



Slika 4.6 Primjer matrične transformacije

4.3 MVVM

Sa sve većom primjenom Microsoftovih platformi usmjerenih izradi korisničkih sučelja, kao što su WPF i Silverlight, javila se potreba za odgovarajućim arhitekturnim obrascem. Takav obrazac bi stavio naglasak na lakše oblikovanje sučelja i smanjio međuovisnosti slojeva aplikacije. Tijekom 2005. godine Microsoft je predstavio Model – View – ViewModel obrazac temeljen na Model – View – Controller (MVC) obrascu. MVC izolira logiku domene aplikacije od korisničkog sučelja i time omogućava neovisan razvoj, testiranje i održavanje svake cjeline zasebno. Sloj modela određuje ponašanje i stanje podataka domene aplikacije. Također odgovara na zahtjeve za informacijama o trenutnom stanju i provodi u djelo zahtjeve za promjenom stanja. View sloj prikazuje podatke iz modela na korisničkom sučelju. Controller sloj prihvaća korisničke akcije i na temelju njih šalje zahtjeve u slojeve Model i View. Kako Controller ima asocijacije prema View i Model slojevima, oni ne mogu imati asocijacije prema njemu da bi se izbjegla kružna međuovisnost. Zbog toga se povratna komunikacija odvija uporabom Observer obrasca. Tim obrascem se omogućuje subjektu da održava listu ovisnih elemenata (observera), te ih automatski obavještava o

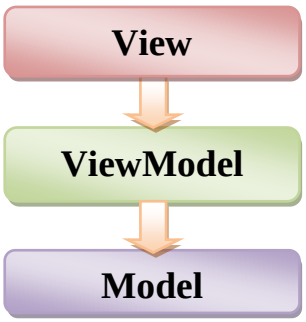


Slika 4.7 Shematski prikaz komunikacije između slojeva MVC obrasca

promjenama. Na taj način pogled iz View sloja održava listu kontrolera(eng. Controllers) koje treba obavijestiti o korisnikovoj akciji, a u Model sloju se nalazi lista pogleda koje treba obavijestiti u slučaju promjene određenih podataka.

MVVM uz uporabu specifičnih funkcionalnosti WPF-a omogućuje potpuno razdvajanje slojeva. S obzirom da nema potrebe za komunikaciju putem *observera*, i View i Model sloj sadrže samo kôd koji pripada tim slojevima. Ovakvo razdvajanje odgovornosti omogućuje:

- Paralelni rad na dizajnu i ostatku aplikacije
- Temeljite *unit* testove
- Ponovno iskoristive komponente unutar i izvan projekta
- Fleksibilnost prilikom izmjene korisničkih sučelja, bez potrebe za izmjenom funkcionalnosti



Slika 4.8 Shematski prikaz komunikacije između slojeva MVVM obrasca

Srednji sloj ima jednosmjernu asocijaciju prema modelu, međutim, za razliku od MVC-a, nema asocijaciju prema gornjem sloju. Njegova uloga je dohvaćanje podataka i njihovo izlaganje View sloju. Također, odgovornost srednjeg sloja je raditi izmjene podataka modela prema zahtjevima koji pristižu iz View sloja.

Gornji sloj nema potrebu za pozadinskim kôdom, već se sve može definirati izravno u XAML dizajnu sučelja. Dvije ključne WPF funkcionalnosti koje omogućuju slabu povezanost View i ViewModel slojeva su *data bindings* i *commands*.

4.3.1 Data Binding

WPF podatkovno povezivanje predstavlja jednostavan i konzistentan način za prikaz te interakciju s podacima unutar aplikacije. Elementi sučelja se mogu povezati sa podacima iz brojnih izvora koji sadrže objekte izvedene iz .NET radnog okvira (Common

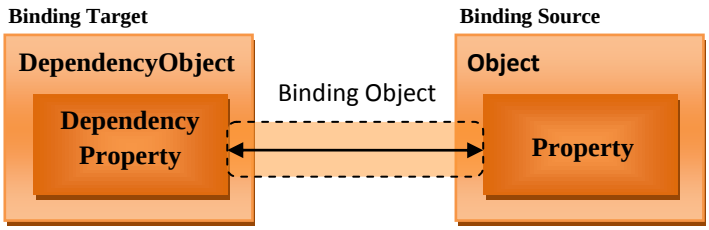
Language Runtime objekti) i XML podatke. Prikaz tako povezanih podataka se može vizualno urediti a moguće je generirati poglede koji će sortirati, filtrirati ili grupirati podatke. Ako je podatkovno povezivanje dobro podešeno i objekti imaju implementiranu funkcionalnost obavještavanja, elementi povezani na podatke prikazuju promjene automatski. Isto tako, ako se vanjska repretentacija podataka promjeni, to će se odraziti na podatke u pozadini.

Obično je za svako povezivanje potrebno definirati sljedeće 4 komponente: element koji je cilj povezivanja, svojstvo cilja za povezivanje, izvor povezivanja i svojstvo izvora za povezivanje (slika 4.9). Svojstvo cilja mora biti *DedependencyProperty*.

DependencyProperty je još jedna odlika WPF-a kojom se omogućuje proširenje funkcionalnosti običnog svojstva CLR objekta. Proširenje donosi sljedeće funkcionalnosti:

- Svojstvo se može postaviti u stilu
- Svojstvo se može postaviti putem podatkovnog povezivanja
- Svojstvo može naslijediti vrijednost roditeljskog elementa
- Za svojstvo se može postaviti animacija
- Svojstvo može dojaviti kada se posljednja vrijednost promijeni
- Svojstvo dojavljuje informaciju da li je potrebno prerazmjestiti elemente sučelja uslijed promjene vrijednosti svojstva

Većina svojstava razreda izvedenih iz UIElement razreda je DependencyProperty i sva, osim *ready-only* svojstava, podržavaju podatkovno povezivanje.



Slika 4.9 Shematski prikaz podatkovnog povezivanja

Mogući načini povezivanja podataka su *OneWay*, *TwoWay* i *OneWayToSource*. OneWay povezivanje omogućuje da se promjene u izvoru ažuriraju na ciljnom elementu, al ne i obratno. Ovakvo povezivanje je praktično prilikom prikazivanja read-only svojstava. TwoWay dozvoljava da se i ciljni element i izvor međusobno ažuriraju. OneWayToSource djeluje na isti način kao OneWay ali u suprotnom smjeru od ciljnog elementa prema izvoru podataka. Dodatno postoji i OneTime povezivanje koje će samo

inicijalno postaviti svojstvo ciljnog elementa na vrijednost izvora. Ostale promjene neće biti zabilježene niti u jednom smjeru toka podataka. Ovakav tip povezivanja je, zbog očuvanja performansi, praktičniji za vrijednosti koje se ne mijenjaju.

Jedno od važnijih svojstava za postavljanje podatkovnog povezivanja je i UpdateSourceTrigger. Svojstvo definira kada se pokreće osvježavanje izvora a može poprimiti 3 vrijednosti: PropertyChanged, LostFocus te Explicit. PropertyChanged osvježavanje se vrši svaki put kada se dogodi promjena unutar Target kontrole. Tako će se, na primjer, izvor osvježiti svakim novim slovom kod unosa u TextBox kontroli. Ako se za UpdateSourceTrigger postavi LostFocus, izvor će se osvježiti tek kada kontrola izgubi fokus. Kod Explicit tipa osvježavanje se vrši samo pozivom UpdateSource() metode. Primjer povezivanja prikazan je sljedećim dvama kratkim isječcima.

```
<TextBox Text="{Binding Name}"/>
```

Programski isječak 4.9 Podatkovno povezivanje - XAML

```
public string Name
{
    get { return _name; }
    set { _name = value; OnPropertyChanged("Name"); }
}
```

Programski isječak 4.10 Podatkovno povezivanje - ViewModel

Za detektiranje promjena izvora podataka, izvor treba definirati mehanizam obavješćivanja. Razred mora implementirati sučelje INotifyPropertyChanged te deklarirati PropertyChanged događaj. Kada se postavi neko svojstvo, potrebno je pozvati događaj. Primjer implementacije mehanizma obavješćivanja je dan sljedećim isječkom.

```
public class Person:INotifyPropertyChanged
{
    private string _name;
    public event PropertyChangedEventHandler PropertyChanged;

    public string Name
    {
        get { return _name; }
        set { _name = value; OnPropertyChanged("Name"); }
    }

    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;
        if (handler != null)
        {
            handler(this,
                  new PropertyChangedEventArgs(propertyName));
        }
    }
    ...
```

Programski isječak 4.11 Implementacija mehanizma obavješćivanja

4.3.2 Commands

Naredbe u WPF-u imaju više namjena. Njihovim korištenjem se odvaja semantika naredbe od objekta odnosno kontrole koja ju pokreće. To dozvoljava više različitih izvora koji mogu pokrenuti istu logiku naredbe te modificiranje logike za različite ciljeve. Tako se na primjer određena naredba može pokrenut pritiskom na Button kontrolu, odabirom iz Menu kontrole te pritiskom kombinacije tipaka na tastaturi. Također, pomoću naredbi se može označiti da li je određena akcija dozvoljena ili nije.

Model naredbi u WPF-u se može rastaviti na četiri glavna koncepta:

- Naredba je akcija koja se treba izvršiti
- Izvor naredbe je objekt koji poziva naredbu
- Cilj naredbe je objekt nad kojim se naredba izvršava
- Povezivanje naredbe spaja naredbu i odgovarajuću logiku

Naredbe se stvaraju uz implementiranje ICommand sučelja. To sučelje izlaže metode Execute() i CanExecute() te događaj CanExecuteChanged. Execute metoda izvodi akciju koja je vezana uz naredbu. CanExecute provjerava da li je zadovoljen uvjet koji omogućuje izvođenje naredbe. Događaj CanExecuteChanged se izvodi kada se dogodi promjena u vrijednosti uvjeta za omogućavanje izvođenja narede. WPF implementacija ICommand sučelja je RoutedCommand razred. Naredbe izvedene iz RoutedCommand

razreda omogućuju podizanje događaja kroz cijelu hijerarhiju od trenutne kontrole preko roditeljske koja ju sadržava do krajnje (eng. bubbling) te obratno od najviše u hijerarhiji prema trenutnoj (eng. tunneling). Događaji putuju kroz hijerarhiju dok ne dođu do kontrole koja implementira povezivanje naredbe odnosno metode za rukovanje podignutim događajima.

U kontekstu MVVM obrasca praktično je kombinirati Commands mehanizam sa DataBinding funkcionalnošću. Time se može postići pokretanje iz gornjeg sloja funkcionalnosti implementirane u srednjem sloju. Primjer takve kombinacije je dan sljedećim dvama isječcima.

```
<MenuItem Header="Spremi" Command="{Binding Path= SaveCommand}"/>
```

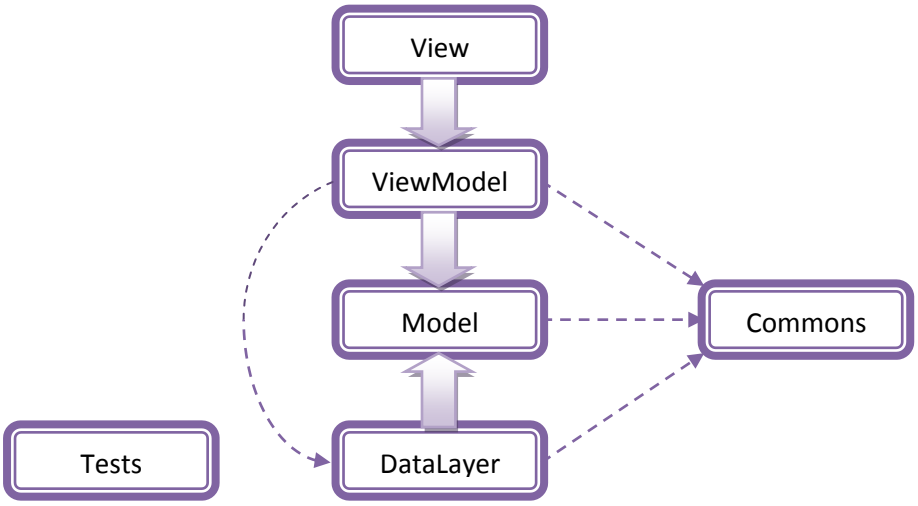
Programski isječak 4.12 Implementiranje naredbe uz pomoć Commands i Data binding mehanizama - XAML

```
private RelayCommand _saveCommand;

public ICommand SaveCommand
{
    get
    {
        if (_saveCommand == null)
            _saveCommand = new RelayCommand(param => this.Save(),
                                            param => this.CanSave());

        return _saveCommand;
    }
}

private bool CanSave()
{
    return CurrentProject != null;
}

private void Save()
{
    ...
}
```

Programski isječak 4.13 Implementiranje naredbe uz pomoć Commands i Data binding mehanizama - ViewModel

5 Dizajn sustava

Sustav je uređen po uzoru na Model – View – ViewModel arhitekturni obrazac. Svaki sloj je izdvojen u zasebni projekt. Osim navedenih slojeva prisutni su i DataLayer, Commons te Tests sloj. Asocijacije među slojevima su prikazane sljedećom shemom.



Slika 5.1 Shematski prikaz slojeva aplikacije i međusobnih poveznica

Sloj DataLayer služi za održavanje trenutnih objekata aplikacije te izvoz elemenata u AutoCad format i pohranu radioničke dokumentacije u PDF formatu. Za pohranu u DXF i PDF format su iskorištene netDxf odnosno TextSharp besplatne dinamičke knjižnice. Realizacija sloja uz korištenje *Repository* obrazaca omogućava jednostavnu i neovisnu naknadnu implementaciju perzistencijskog mehanizma. Sloj Commons sadrži razrede sa funkcionalnošću koja se koristi u više slojeva aplikacije. Tako se u tom djelu nalaze određene „popularne“ geometrijske i matematičke operacije, zajednička baza iznimaka, specifične operacije nad kolekcijama podataka itd. Tests sloj sadrži *unit* testove za testiranje ViewModel, Model i DataLayer slojeva.

Slijedi poglavlje sa opisom modela domene. Veće cjeline modela su prikazane zasebno na kraju poglavlja zbog preglednosti.

5.1 Model domene

Okosnicu modela čini cjelina prikazana slikom 5.5. Pojedini projekt sadrži kolekciju pročelja (Facade). Svako pročelje može imati više plošnih (Pan), krovnih (AtticElement) te potpornih elemenata za prozore (VoidSupportElement). Elementi

podkonstrukcije se generiraju na temelju stanja pročelja te su svojstva tih elemenata iz skupa ograničenih vrijednosti. Zbog toga se takvi objekti tretiraju kao *value* instance, odnosno njihov identitet nije bitan. Tako da razred Facade ima referencu prema *entity* razredu UCarriersAssembly koji predstavlja jednu cjelinu (stupac) povezanih elemenata nosača i spojnica. Razred sadrži kolekciju value objekata tipa UCarrier i UCarrierJuncture. Podkonstrukcijskim objektima se čuva redoslijed i njihove pozicije. Facade razred također sadrži referencu prema singleton razredu TokenProvider. Ovaj razred ima ugrađenu funkcionalnost za održavanje labela elemenata pročelja. Kod stvaranja ili izmjene elementa pročelja, Facade instanca provjerava da li postoje identični elementi. Ukoliko ne postoje, od tokena se dohvaća sljedeći raspoloživi token te se ta vrijednost postavlja za labelu elementa. Kada se element obriše ili promjeni vrijednost labele a da nijedan drugi element više ne sadrži prethodnu vrijednost, Facade instanca vraća oslobođeni token TokenProvider instanci. Na taj je način osigurana konzistentnost labela elemenata.

Plošni element

Najkompliciraniji element pročelja je plošni element. Osim samog po sebi kompliciranog oblika, on može sadržavati i brojne obrade. Slikom 5.6 je prikazan dio modela koji se odnosi na ovaj element. Na prvi pogled bi bilo logično obuhvatiti vidljivi dio i proširenja u jedan razred jer je to fizički jedna cjelina. Međutim, proširenja mogu imati cijeli niz svojstava koji ih razlikuju od vidljivog dijela (rupe za vijke, pomoćne rupe za postavljanje, razni mogući oblici, produljenja koja se nastavljaju na produljenja itd.), tako da su izdvojena u zasebne razrede. Razredi plošnih elemenata i proširenja su čvrsto povezani i čine kompoziciju.

Osim proširenja, plošni elementi mogu sadržavati i obrade vezane uz postavljanje prozora, vrata te kutne obrade. Obrade nisu referencirane iz razreda proširenja, već iz Pan razreda, iz razloga što se jedan tip obrada odnosi na sami vidljivi dio plošnog elementa te se pojedina obrada može odraziti na više proširenja istovremeno.

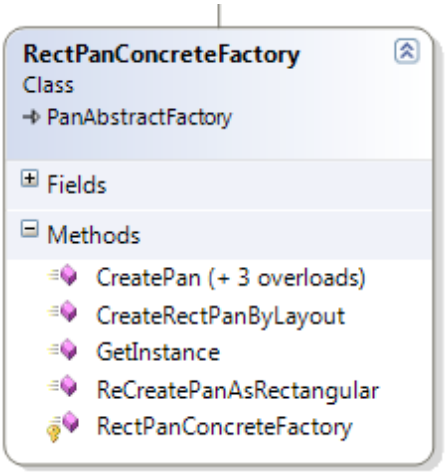
„Proizvodni pogon“

Jedna od najvažnijih cjelina modela domene je tvornica elemenata pročelja. Globalno postoji razred FacadeElementAbstractFactory koji ima svojstva zajednička svim tvornicama elemenata. Potom niže u hijerarhiji slijede specifične apstraktne tvornice:

- PanAbstractFactory – sa zajedničkim svojstvima za stvaranje svih vrsta plošnih elemenata

- UnderconstructionAbstractFactory – sa zajedničkim svojstvima za stvaranje nosača i spojnica podkonstrukcije
- AtticaElementAbstractFacotry – sa zajedničkim svojstvima za stvaranje krovnih elemenata

Zatim slijede konkretne tvornice za svaku pojedinačnu vrstu elementa pročelja: RectPanConcreteFactory, IregPanConcreteFactory, FreeshapedPanConcreteFactory, AtticaElementConcreteFactory, UCarrierJunctureConcreteFactory, UCarrierConcreteFactory, te VoidWindowSupportConcreteFactory. Svaka od konkretnih tvornica je zadužena za proizvodnju pripadnih elemenata. Na ovaj način je izveden AbstractFactory oblikovni obrazac čija je namjena pružiti sučelje za stvaranje kompleksnih objekata bez navođenja njihovih konkretnih razreda. Tako u RectPanFactory postoje metode prikazane sljedećom slikom.

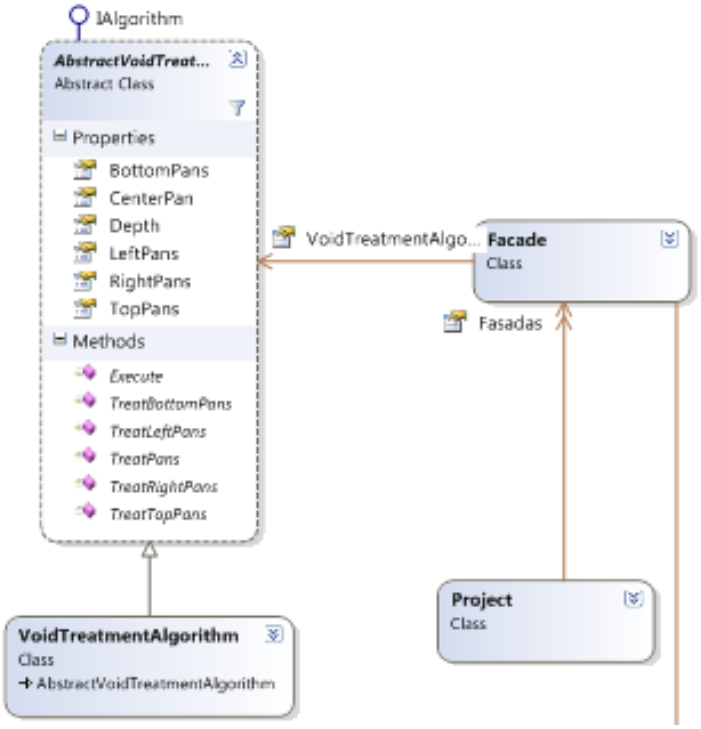


Slika 5.2 RectPanConcreteFactory razred

Ovaj proizvodni pogon se koristi na isti način kao i onaj stvarni. Ako je potrebno neki element promijeniti, uglavnom se ne radi izmjena nad starim elementom već se on odlaže za reciklažu, a proizvodni pogon stvara novi element.

Algoritmi

Opsežnije i kompleksnije funkcionalnosti obrade nad elementima su izdvojene od ostatka modela. Razlog izdvajanja je smanjenje međuovisnosti i olakšano testiranje. Primjeri takvih algoritamskih razreda su VoidTreatmentAlgorithm (za obradu elemenata oko otvora – slika 5.3) te GeneticAlgorithm.



Slika 5.3 Algoritam za obradu elemenata oko prozora i vrata – Strategy obrazac

U slučaju algoritma VoidTreatmentAlgorithm se može očekivati dodavanje novih tipova obrade (npr. obrada elemenata kada prozori ili vrata ne prate mrežu). U modelu je napravljena priprema za takve slučajeve. Definirana je apstraktna klasa AbstractVoidTreatmentAlgorithm preko koje Facade razred pristupa operacijama konkretnih algoritmima obrade (*Strategy pattern*). Ovime je enkapsulirana funkcionalnost te omogućena izmjenjivost algoritama bez da je korisnički razred toga svjestan.

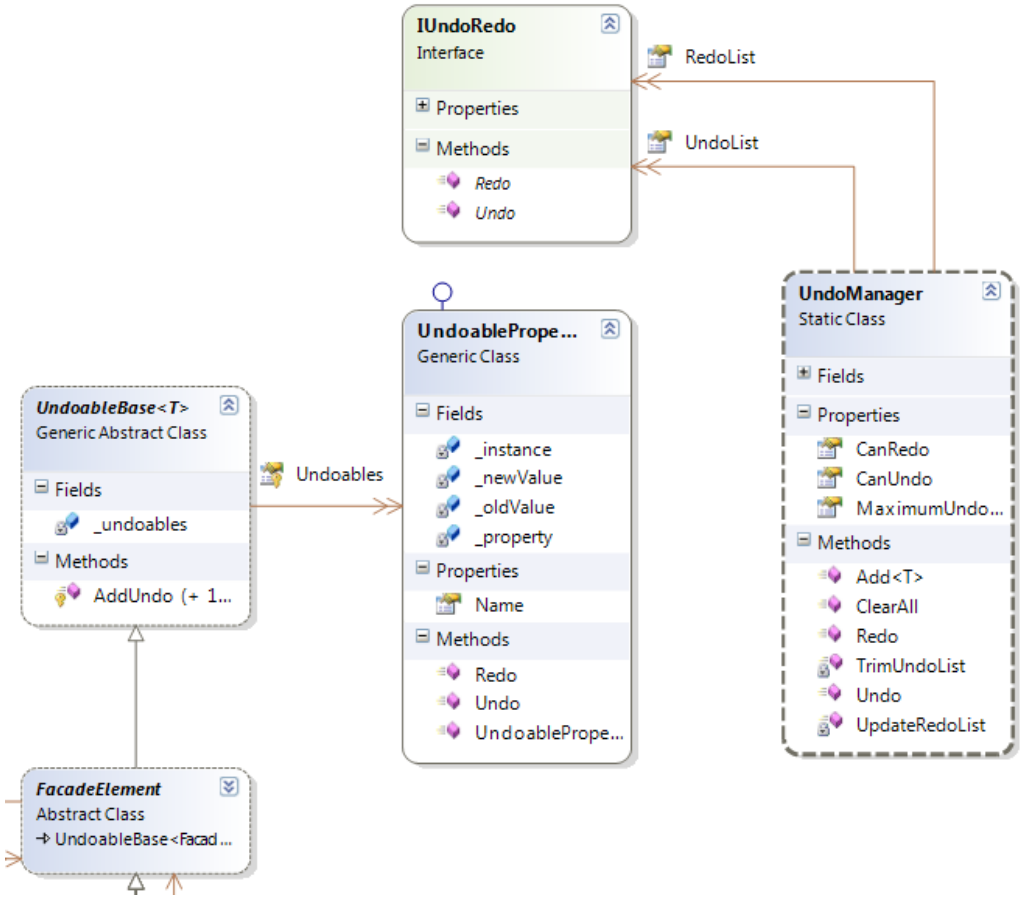
Svi algoritmi su singleton, a njihovo testiranje je detaljnije objašnjeno u poglavlju „Testiranje i verifikacija“.

Izvoz u PDF i DXF format

Radionička dokumentacija se generira na temelju Document razreda. Najviši razred u hijerarhiji sadrži globalna svojstva i globalne metode koje imaju funkcionalnost za potrebe svih vrsta dokumenata (kreiranje tablica, okvira, strelica za mjere, zapis podataka...). Dokumenti trebaju sadržavati skice elemenata koje se sastoje od geometrijskih oblika. Ti oblici se definiraju na različite načine u WPF-u, prilikom pohrane u PDF te prilikom pohrane u DXF. Iz tog razloga su osmišljeni DrawingStub razredi koji na jedinstveni način definiraju grafičke elemente: *Line, Polyline, Arc, Block* te *Circle*. Prema potrebi su definirane metode za dohvaćanje .NET, PDF ili DXF prilagođene definicije.

Undo/redo mehanizam

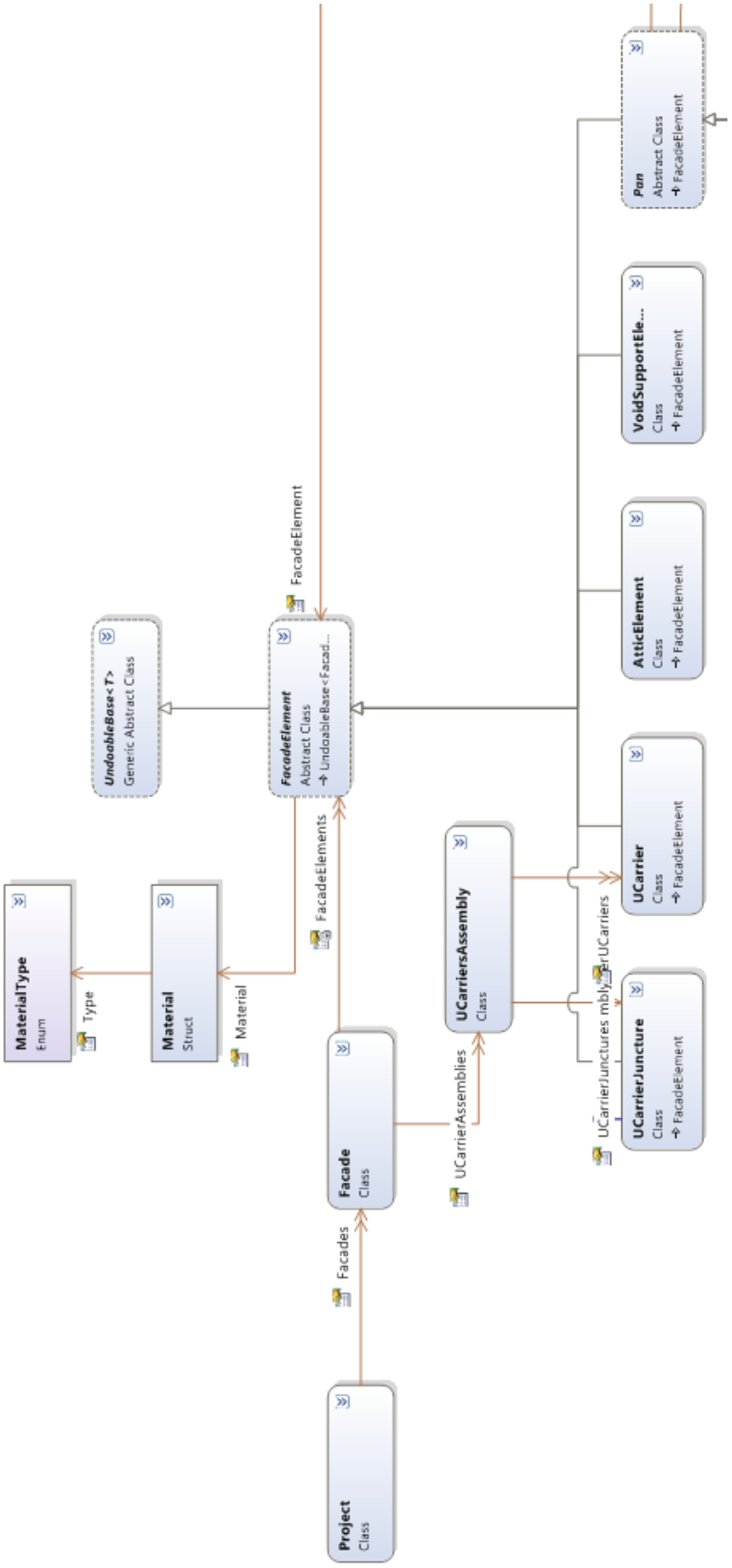
Funkcionalnost poništavanja promjena ostvarena je isključivo za svojstva elemenata pročelja. Implementacija mehanizma je prikazana sljedećom slikom.



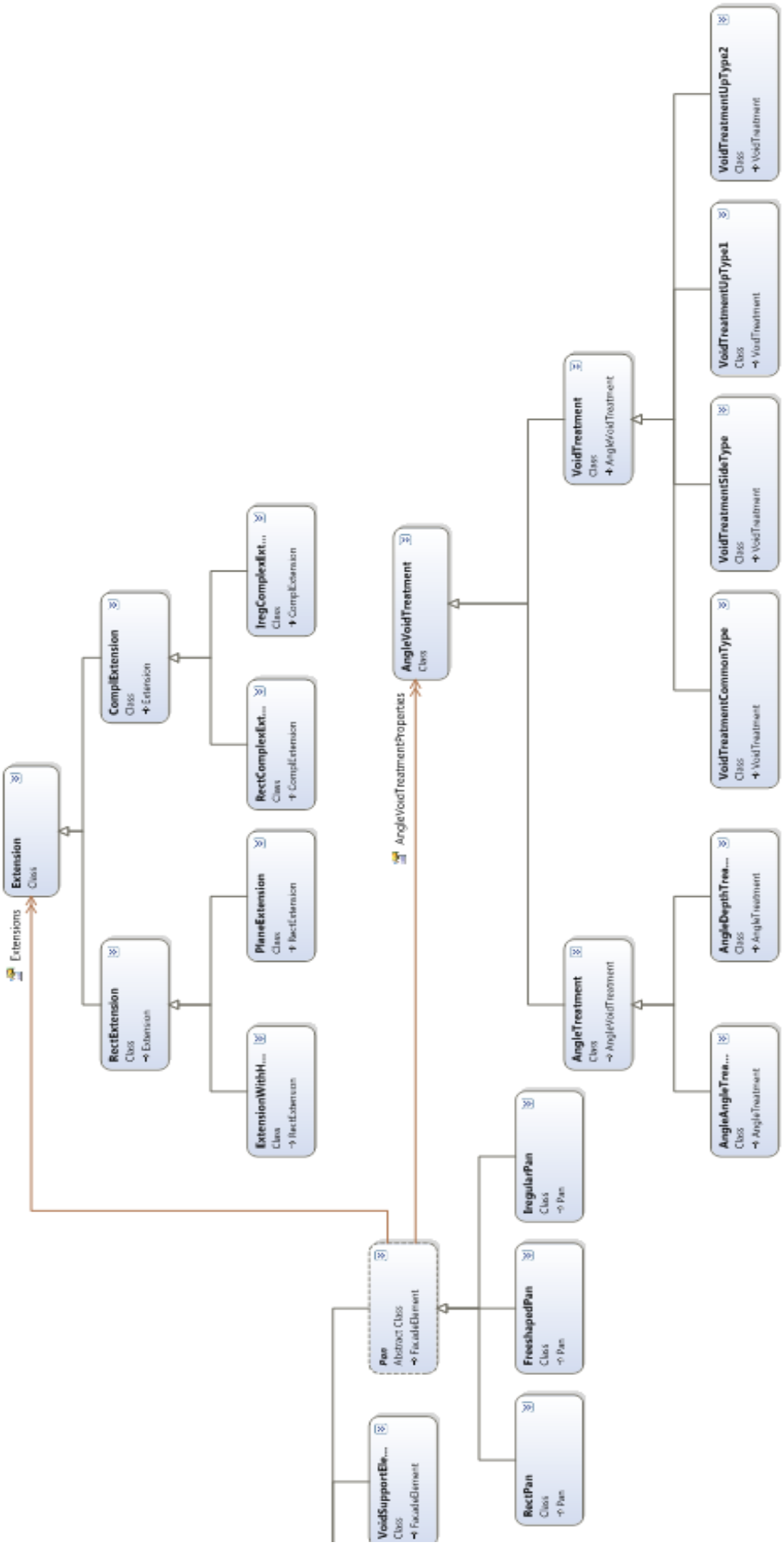
Slika 5.4 Undo/redo mehanizam

Svaki element pročelja nasljeđuje metodu AddUndo() za dodavanje UndoableProperty objekata u kolekciju Undoables. Ova kolekcija služi za grupiranje UndoableProperty instanci prema pojedinim objektima. Prilikom poziva AddUndo() metode, se stvara instanca UndoableProperty sa vrijednostima: instanca kojoj pripada svojstvo, nova vrijednost svojstva, stara vrijednost svojstva i naziv svojstva. Na temelju tih vrijednosti Undo() i Redo() metode mijenjaju svojstva objekata. Prilikom stvaranja UndoableProperty objekata, instance se automatski(iz konstruktora) dodaju u statički razred UndoManager, središte implementacije mehanizma. Unutar navedenog razreda su sadržane undo i redo kolekcije u koje se pohranjuju UndoableProperty instance. Pozivom Undo() i Redo() metoda, manager poziva Undo() odnosno Redo() metode UndoableProperty objekata te izvodi potrebne radnje uređivanja vlastitih Undo i Redo lista.

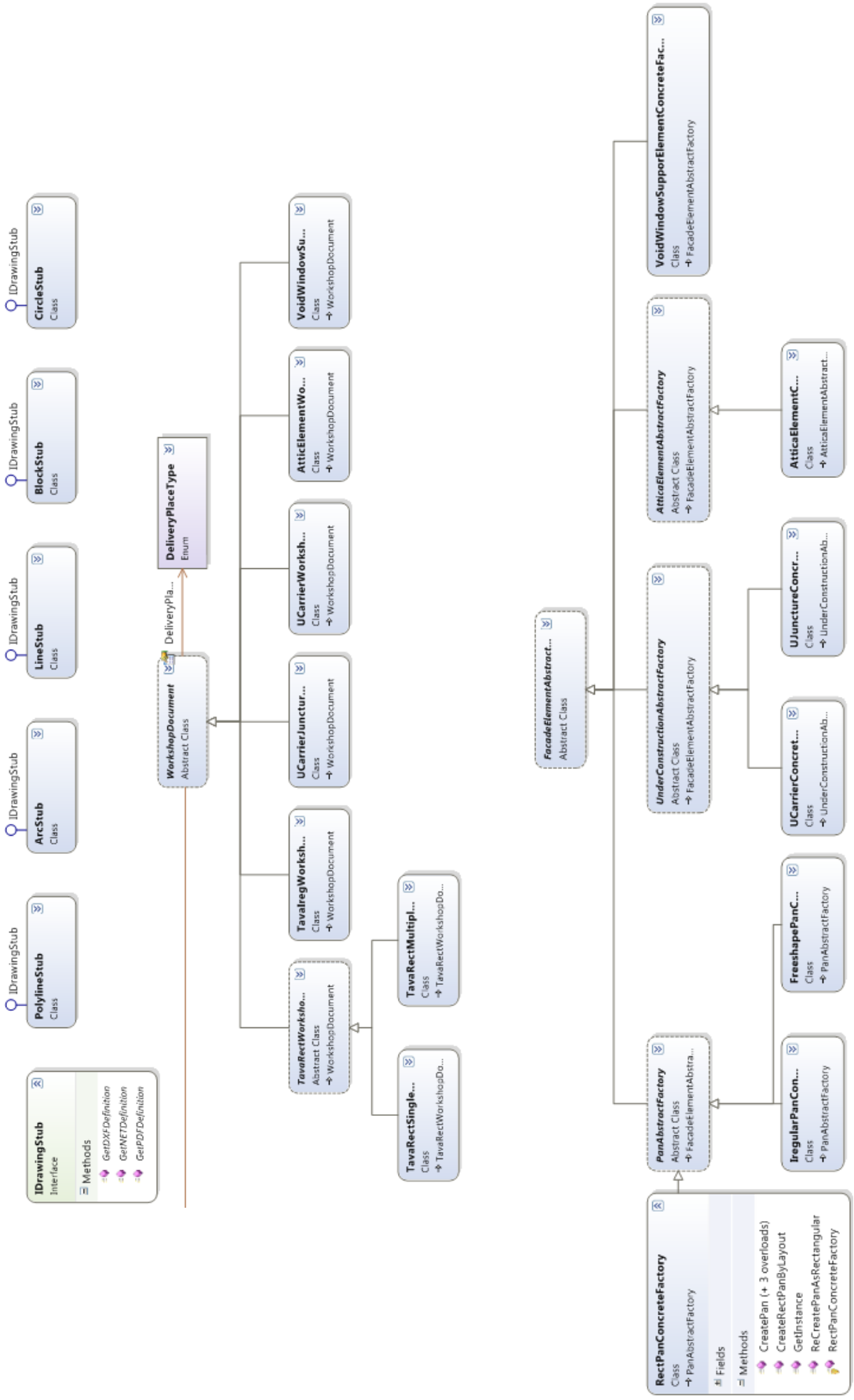
Slijede grafički prikazi većih cjelina modela.



5.5 Osnovica modela. Veze projekta, pročelja i elemenata. Određene veze nisu prikazane zbog preglednosti.



5.6 Kompozicija plošnog elementa (Pan) uključuje proširenja (Extensions)

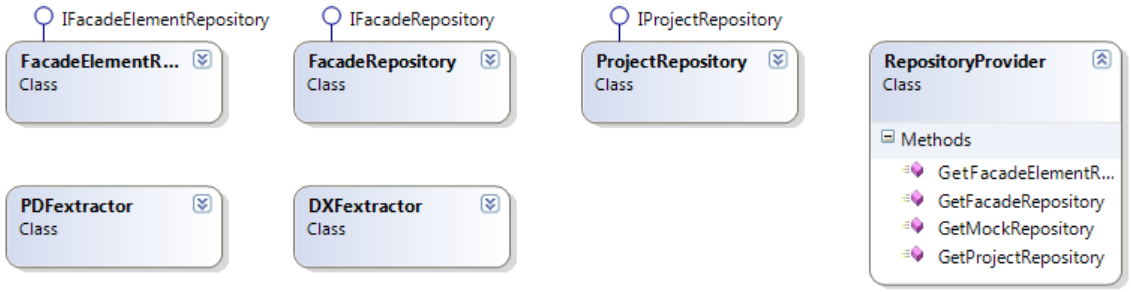


Slika 5.7 Razredi dokumenata te univerzalni grafički elementi (lijevo) i "Proizvodni pogon" sustava (desno)

5.2 DataLayer

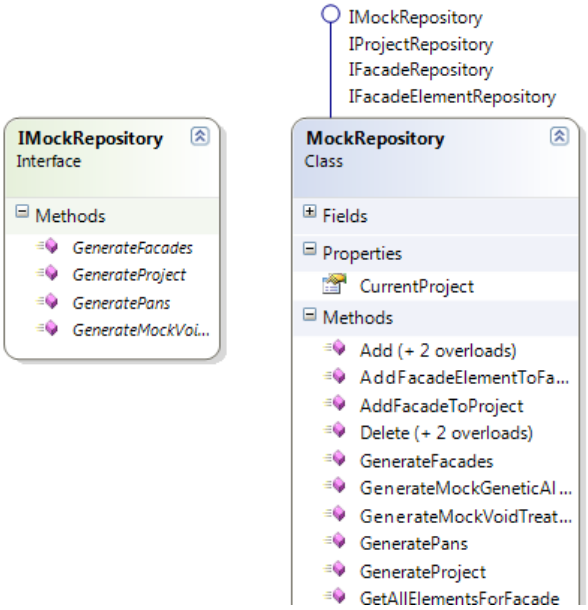
DataLayer cjelina ima namjenu održavanja podataka aplikacije te izvoz u PDF i DXF formate. Trenutno aktualni podaci se čuvaju u repozitorijima. S obzirom da se može očekivati zahtjev za ugradnjom nekog od oblika perzistencije potrebno je smanjiti ovisnost ostale funkcionalnosti o konkretnim razredima repozitorija. Iz navedenog razloga su definirana sučelja za vanjski pristup te RepositoryProvider razred pomoću kojeg se dohvaćaju repozitoriji. Kako su repozitoriji singleton, oni će se prilikom dohvaćanja sami kreirati ukoliko već nisu. Ovime je postignuto centralizirano stvaranje odnosno dohvaćanje repozitorija te neovisnost o konkretnoj implementaciji. Metodama RepositoryProvider razreda se također pristupa putem sučelja čime je ovisnost o podatkovnom sloju svedena na minimum.

Na ovaj način ne samo da je olakšana realizacija perzistencijskog mehanizma, već je olakšana i njegova izmjena. Iz istih razloga je izdvojena funkcionalnost za izvoz u DXF i PDF format u razrede DXFextractor i PDFextractor.



Slika 5.8 Data Layer

Dodatno DataLayer sadrži MockRepository. Uloga Mock repozitorija je stvaranje defaultnih objekata te algoritama. Iako je namijenjen za testiranje, podaci se koriste kako se ne bi trebalo svaki put kreirati objekte kod pokretanja aplikacije. MockRepository implementira sva sučelja ostalih repozitorija tako da ga ViewModel razredi koriste bez da znaju da li je to konkretni ili lažni repozitorij. Implementirane metode pozivaju metode iz odgovarajućih repozitorija.



Slika 5.9 Implementacija "lažnog" - Mock repozitorija

5.3 ViewModel

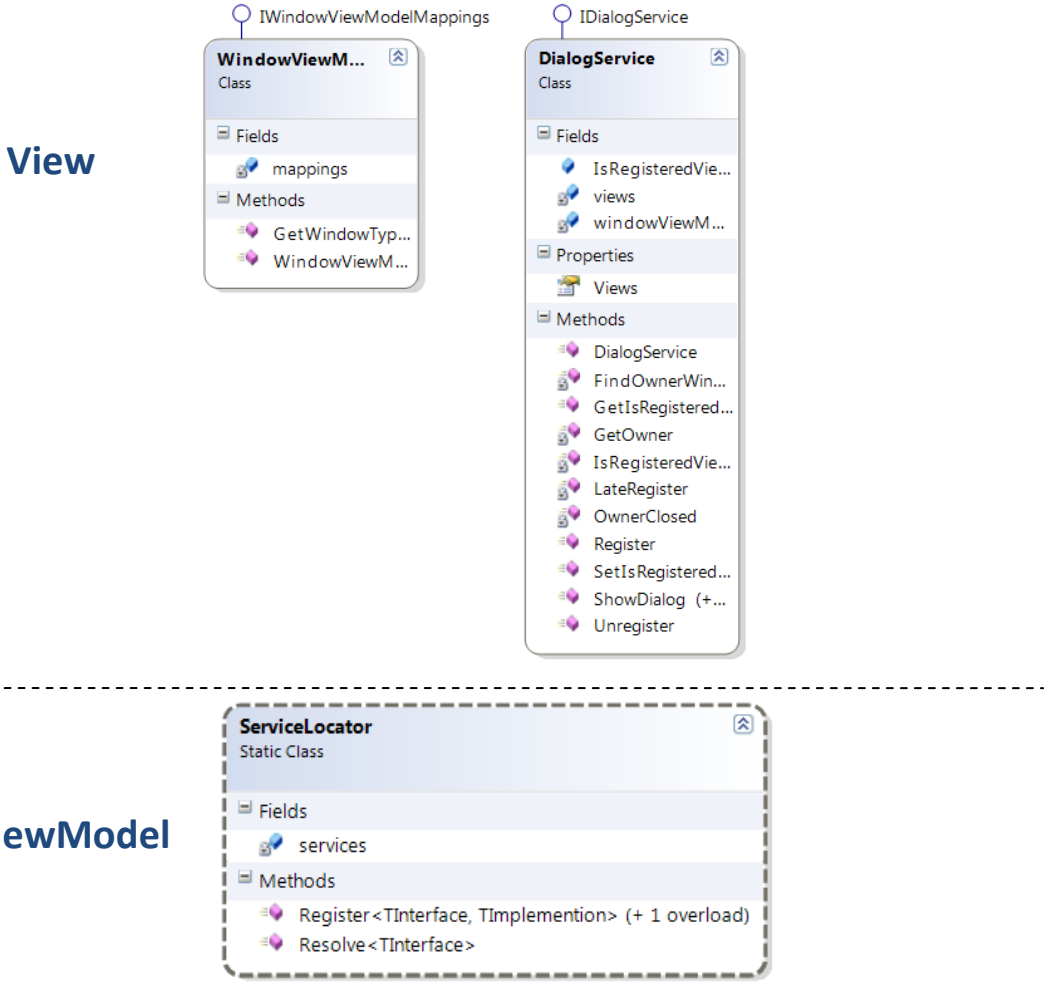
Valja istaknuti kako je upotrjebljen poseban obrazac za otvaranje prozora odnosno dijaloga kojim se postiže *dependency injection*. Pod tim pojmom se podrazumijeva definiranje i izmjena sprege View i ViewModel razreda ovisno o potrebi. Svrha implementacije je postizanje:

- centraliziranog mapiranja View i ViewModel razreda
- dinamičkog povezivanja View i ViewModel razreda
- izbjegavanja pozadinskog kôda
- reduciranja kôda
- izmjene konteksta na relaciji *run time – design time*

Obrazac se sastoji od razreda WindowViewModelMappings, DialogService i sučelja IWindowViewModelMappings koji pripadaju View sloju te razreda ServiceLocator i sučelja IDialogService koji pripadaju ViewModel sloju.

U razredu WindowViewModelMappings se nalazi Dictionary skup podataka sa View i pripadnim ViewModel razredima. Razred DialogService sadrži metode za otvaranje dijaloga (korisničkih i sustavskih), omogućava postavljanje i dohvaćanje WindowOwner instance te definira ovisno svojstvo *(*eng. dependency property*)* isRegisteredView. Preko tog svojstva određeni View razred dozvoljava da ga se iskoristi u mapiranju.

Razred ServiceLocator služi za registriranje sučelja i implementacije razreda WindowViewModelMappings te razreda DialogService. To registriranje se obavlja prilikom pokretanja aplikacije.



Slika 5.10 Implementacija dependency injectiona za dijaloge i viewModele

Nakon ove implementacije je za povezivanje i pokretanje dijaloga dovoljno napraviti sljedeće korake.

1. Registriranje View elemenata

```
public WindowViewModelMappings()
{
    mappings = new Dictionary<Type, Type>
    {
        { typeof(ViewModel.MainWindowViewModel), typeof(MainView) },
        { typeof(ViewModel.FacadeDefineRegularViewModel),
                              typeof(FacadeDefinitionRegularView) },
        ...
```

2. Mapiranje View elementa i ViewModel razreda

```
var fasadaDefineRegularViewModel = new FacadeDefineRegularViewModel();
IDialogService dialogService =
ServiceLocator.Resolve<IDialogService>();
bool? dialogResult = dialogService.ShowDialog( ownerDialogVM,
                                          fasadaDefineRegularViewModel);

if (dialogResult == true)
{
    ...
}
```

3. Poziv iz viewModela

```
<DataTemplate DataType="{x:Type
        ViewModel:ProjectExplorerTabViewModel}">
    <Controls:ProjectExplorerTabView />
</DataTemplate>
```

Povezivanje ViewModel i View razreda koji se ne odnose na dijaloge se navodi u *MainViewResources* XAML kôdu uz pomoć *DataTemplate* razreda. Npr:

```
<Window x:Class="FasadeMVVM.Dialogs.FacadeDefinitionRegularView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Definicija rastera"
    xmlns:Main="clr-namespace:FasadeMVVM"
    Main:DialogService.IsRegisteredView="True"
```

Programski isječak 5.1 Povezivanje View elementa i ViewModel razreda putem DataTemplate razreda

5.4 Testiranje i verifikacija

Aplikacija je testirana brojnim *unit* testovima. Ispitivana je funkcionalnost modela, podatkovnog i ViewModel sloja. Podaci su generirani pomoću prethodno spomenutog Mock repozitorija. Slijedi primjer ViewModel testa.

```
[TestMethod]
public void TestRemoveSelected()
{
    Facade facade = _mockRepository.GetFacade();
    FacadeViewModel facadeViewModel = new FacadeViewModel(facade, null,
                                                          null, false);

    Assert.AreEqual(facadeViewModel.Facade.Pans.Count, 100);

    List<Pan> selectedPans = new List<Pan>() {
                                        facade.Pans[0], facade.Pans[1]};
    facadeViewModel.SelectedPans =
                        new ObservableCollection<Pan>(selectedPans);

   facadeViewModel.RemoveSelectedCommand.Execute();

    Assert.AreEqual(facadeViewModel.Pans.Count, 98);
}
```

Programski isječak 5.2 ViewModel unit test

Zahvaljujući Mock repozitoriju se može dohvatiti algoritam sa postavljenim vrijednostima, što omogućuje pojedinačno mijenjanje vrijednosti i provjeru utjecaja.

```
[TestMethod]
public void TestDepthChange()
{
    IVoidTreatmentAlgorithm voidTreatmentAlgorithm =
        _mockRepository.GenerateMockVoidTreatmentAlgorithm();

    voidTreatmentAlgorithm.Depth = 300;

    voidTreatmentAlgorithm.Execute();

    Pan firstTopPan = voidTreatmentAlgorithm.TopPans[0];
    Extension bottomExtension = firstTopPan.BottomExtension;

    Assert.AreEqual(bottomExtension.Height, 295);
}
```

Programski isječak 5.3 Unit test algoritma

Verifikacija aplikacije se odvija prema planu testiranja. Plan se sastoji od testova funkcionalnosti i preciznih testova operacija nad elementima pročelja. Testovi funkcionalnosti se izvode prema definiranim slučajevima korištenja. Testovi operacija su vrlo važni. Na temelju izvedenih operacija će se izrađivati fizički elementi te je potrebno suziti prostor za pogreške što je više moguće. Testni slučaj i očekivane rezultate definiraju odgovorne osobe.

Slijedi primjer testnog slučaja za operacije.

1. Stvara se default plošni element od Aluminija debljine 1.5 mm
2. Razlika u y koordinatama točaka 1 i 13 razvijenog oblika iznosi 1113.6
3. Materijal plošnog elementa se mijenja u Inox debljine 3 mm
4. Razlika u y koordinatama točaka 1 i 13 razvijenog oblika iznosi 1111.84 mm

Odgovarajući unit test:

```
[TestMethod]
public void TestCase35()
{
    const double verifiedDelta = 1113.6;
    Pan pan = _mockRepository.GeneratePan();

    pan.Material.Type = MaterialType.Aluminum
    pan.Material.Thickness = 1.5;
    Polyline border = pan.GenerateStretchedShape();
    double delta = border.Points[1].Y - border.Points[13].Y;

    Assert.AreEqual(delta, verifiedDelta);

    verifiedDelta = 1111.84;
    pan.Material.Type = MaterialType.Inox
    pan.Material.Thickness = 3;
    Polyline changedborder = pan.GenerateStretchedShape();
    double delta = changedborder.Points[1].Y -
                                changedborder.Points[13].Y;

    Assert.AreEqual(delta, verifiedDelta);
}
```

Programski isječak 5.4 Test operacije nad elementom pročelja

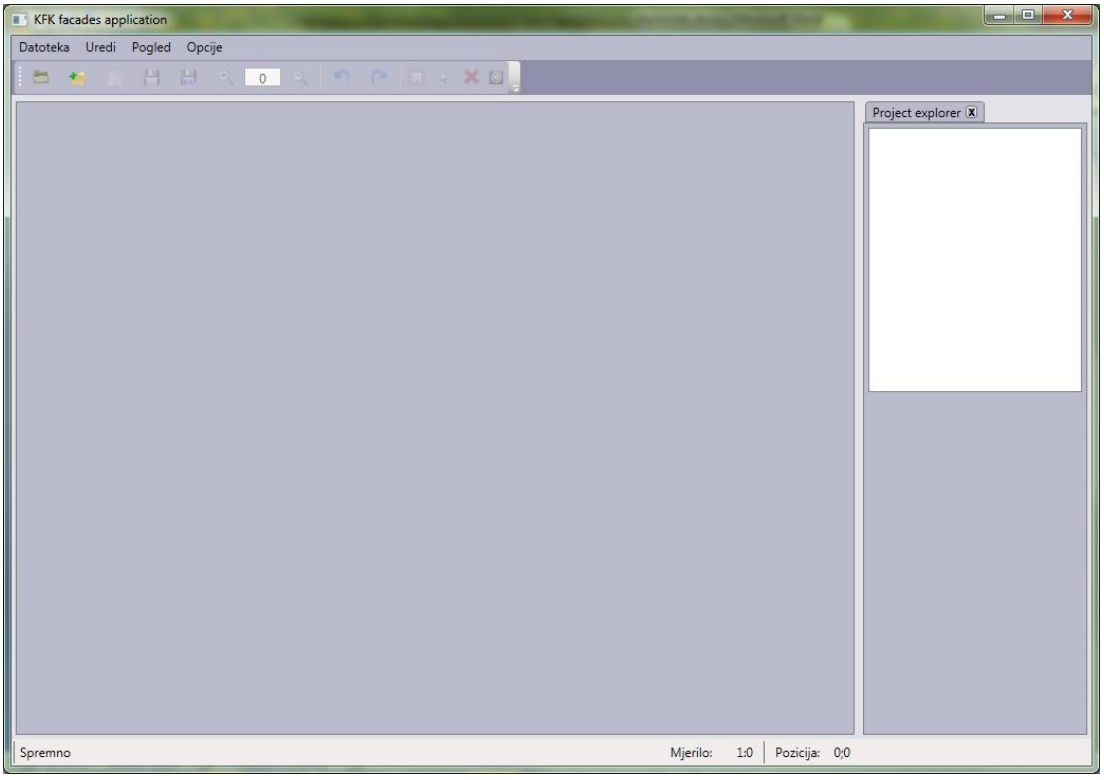
6 Funkcionalnost sustava

Poglavlje sadrži opise funkcionalnosti uz prikaze korisničkog sučelja. Opisi su uglavnom grupirani prema slučajevima korištenja iz poglavlja 3.3.

6.1 Glavni prozor

Glavni prozor programskog ostvarenja se može podijeliti na 5 cjelina:

1. Glavni Menu izbornik
2. Toolbar kontrola sa globalnim opcijama
3. TabControl kontrola za dodavanje radnih površina za uređivanje pročelja
4. TabControl kontrola za dodavanje radnih sučelja sa raznim informacijama i uređivačkim mogućnostima ovisnim o kontekstu
5. StatusBar kontrola sa informacijama o izvođenju aplikacije, mjerilu i poziciji kursora

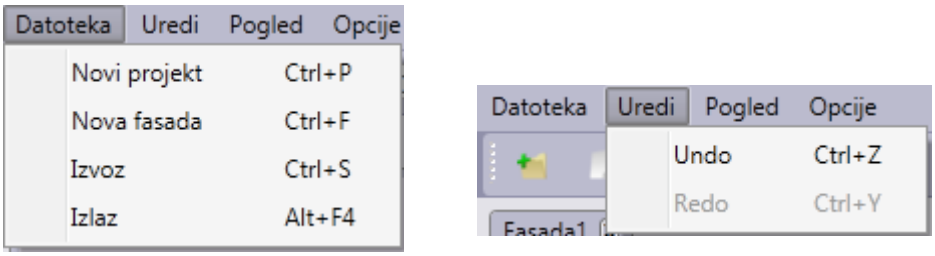


Slika 6.1 Glavni prozor

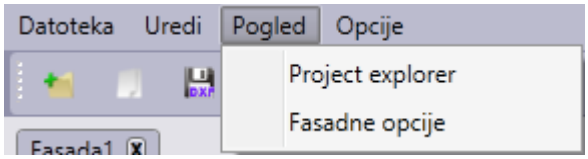
Unutar glavnog izbornika postoje izbornici „Datoteka“, „Uredi“, „Pogled“ te „Opcije“. „Datoteka“ izbornik nudi opcije za stvaranje novog projekta, dodavanje novog pročelja, pohranu (izvoz) nacrta i radioničke dokumentacije, te zatvaranje aplikacije (slika 6.2). Izbornik „Uredi“ sadrži opcije za poništavanje promjena (undo/redo operacije) nad

elementima pročelja (slika 6.2). „Pogled“ izbornik pruža mogućnosti za otvaranje radnih sučelja (slika 6.3). Pod izbornik „Opcije“ spadaju sljedeće opcije (slika 6.4):

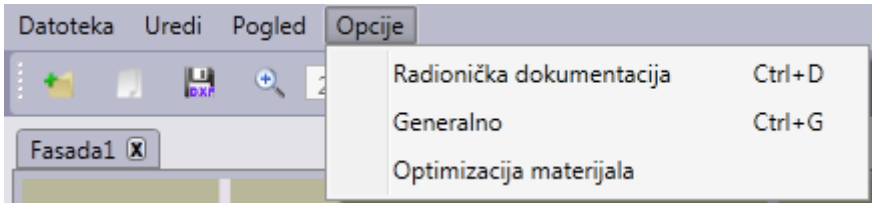
- Radionička dokumentacija – otvara dijalog za uređivanje informacija radioničke dokumentacije
- Generalno – otvara dijalog za uređivanje pozadine radne površine te putanje za pohranu
- Optimizacija materijala – otvara prozor za upravljanje optimizacijom materijala



Slika 6.2 „Datoteka“ i „Uredi“ izbornici



6.3 "Pogled" izbornik



Slika 6.4 "Opcije“ izbornik

Glavna alatna traka ima neke od opisanih opcija iz Menu izbornika uz nekolicinu dodatnih.

Slika 6.5 Glavna alatna traka

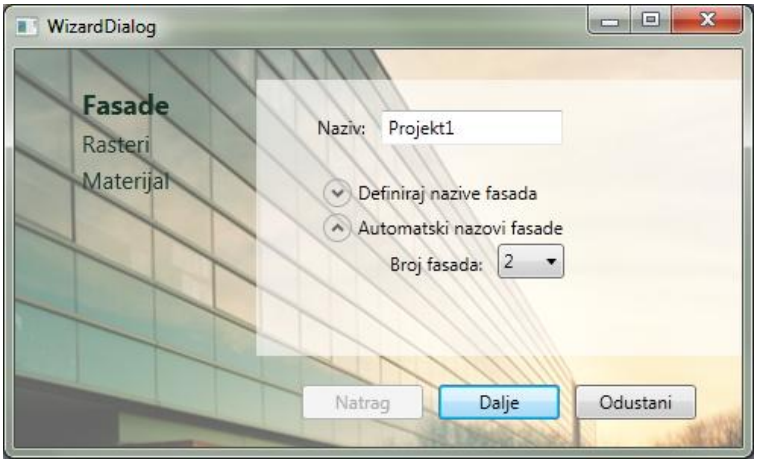
Opcije prikazane prema redoslijedu na slici su: „Novi projekt“, „Dodaj pročelje“, „Zoom in“, „Zoom out“, „Undo“, „Redo“, „Označi više odjednom“, „Označi pojedinačno“, „Očisti“ te „Generiraj podkonstrukciju“.

6.2 Stvaranje projekta

Početno stanje projekta se definira pomoću čarobnjaka kojemu se pristupa kroz glavni Menu izbornik „Datoteka → Novi projekt“. Projekt se definira u tri koraka:

1. Određuje se naziv projekta te broj i nazivi pročelja
2. Definira se inicijalna mreža plošnih elementa po pročeljima
3. Definiraju se parametri i svojstva materijala elemenata pročelja

1. Korak



Slika 6.6 Prvi korak čarobnjaka za stvaranje projekta

Osim automatskog nazivanja pročelja oblika „Fasada#“, moguće je i obaviti unos proizovljnog imena proširenjem „Definiraj nazive fasada“ *expander* kontrole.

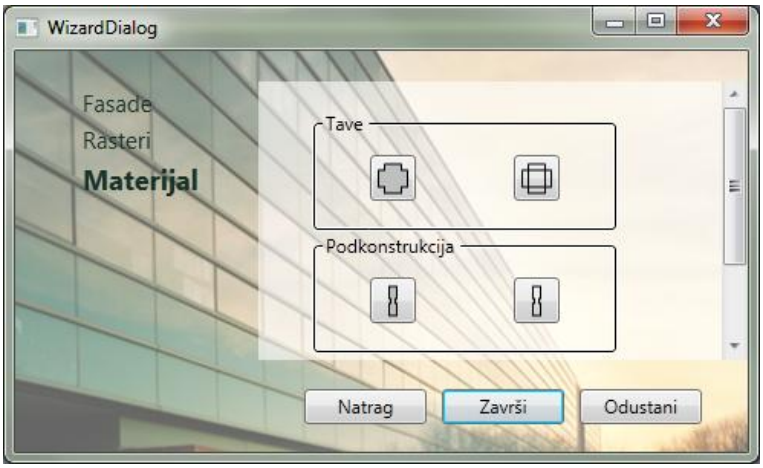
2. Korak

Slika 6.7 Drugi korak čarobnjaka za stvaranje projekta

Definicija mreže početnih plošnih elemenata se obavlja za svako pročelje zasebno. Moguće je odabrati između ponuđena dva tipa definicije. Kod prvog tipa (slika 6.8) je potrebno navesti širinu razmaka između elemenata (širina fuge), ukupni širinu pročelja, ukupnu visinu pročelja, broj podjela po x osi (broj stupaca), broj podjela po y osi (broj redaka). Drugi tip definicije(slika 6.8) zahtjeva posebno unošenje visina redaka i širina stupaca te širinu razmaka između elemenata.

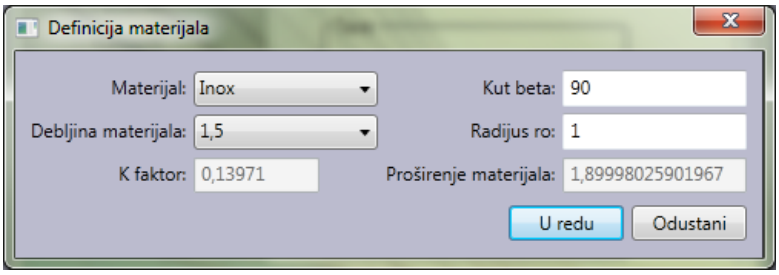
Slika 6.8 Dijalozi za definiciju inicijalne mreže elemenata

3.Korak



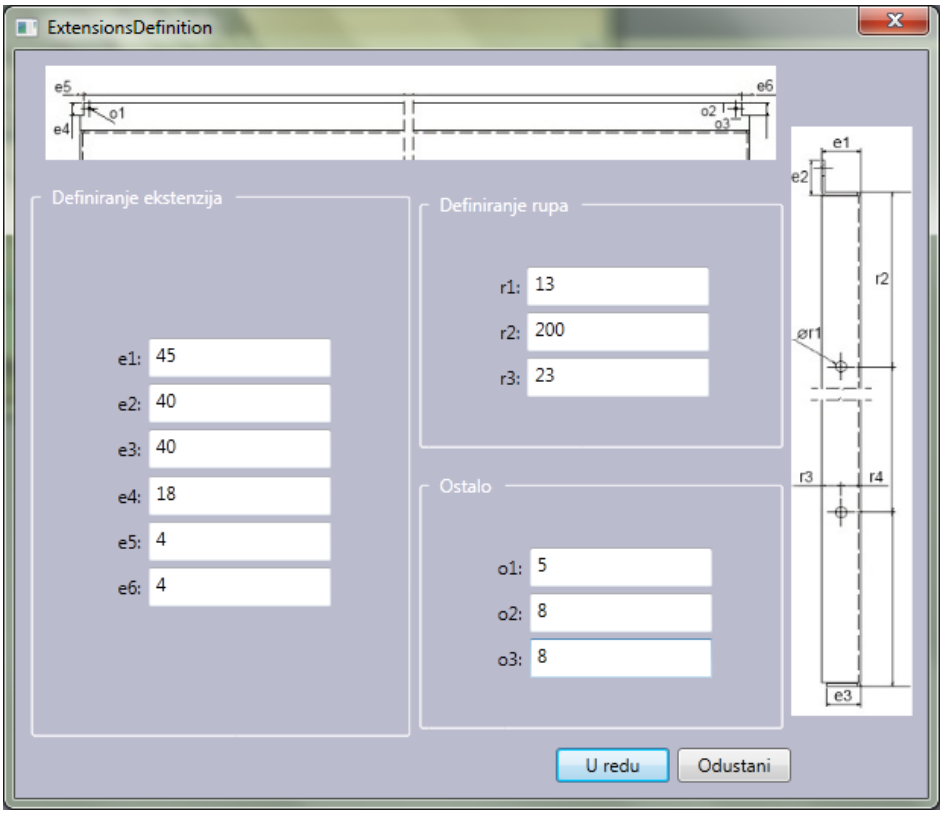
Slika 6.9 Treći korak čarobnjaka za stvaranje projekta

Materijali i svojstva elemenata pročelja se definiraju prema grupama plošnih elemenata, elemenata podkonstrukcije te krovnim elementima. Dijalog s definicijom materijala (slika 6.10) omogućava izbor vrste (Aluminij, Inox, TV te VZ), debljine materijala, defaultni kut pod kojim se materijal savija, te radijus stroja koji obavlja savijanje. Ovisno o odabranim vrijednostima, automatski se prikazuju vrijednosti κ faktor, te proširenje materijala koje će se dogoditi kod savijanja.



Slika 6.10 Dijalog za definiranje materijala elementa

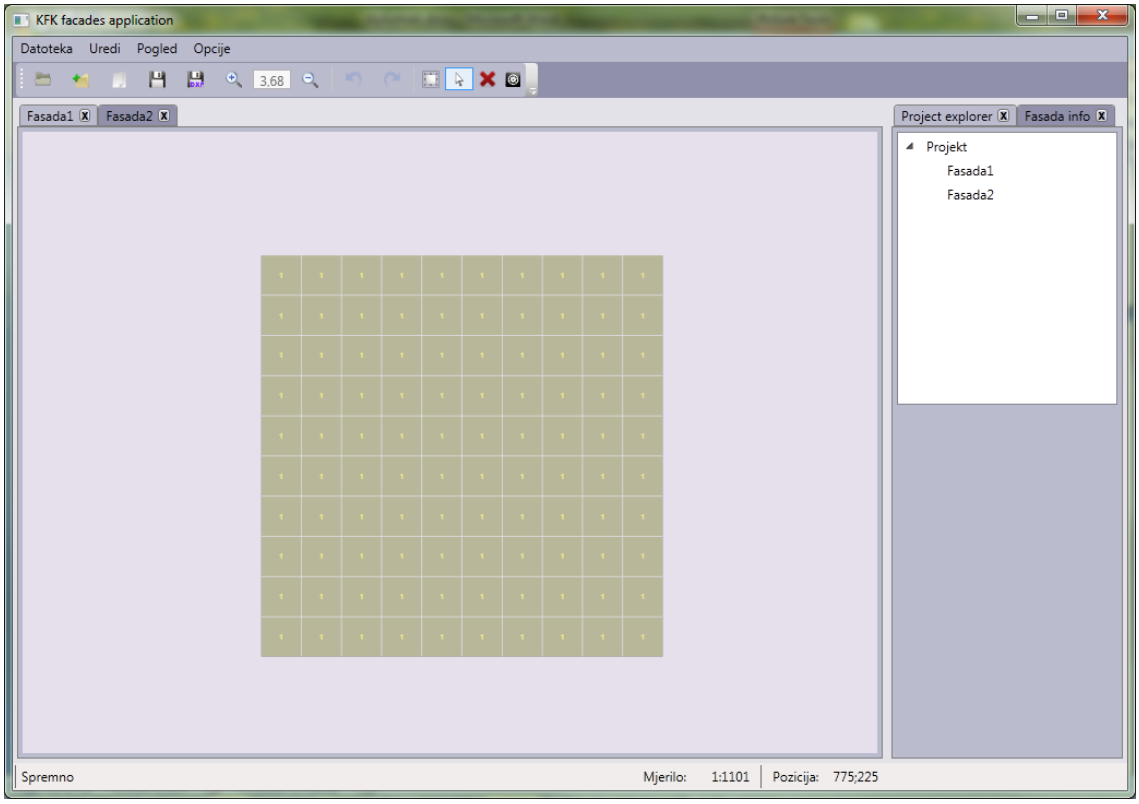
Dijalog s definicijom parametara plošnih elemenata sadrži grafičke prikaze za raspoznavanje značenja pojedinog parametra.



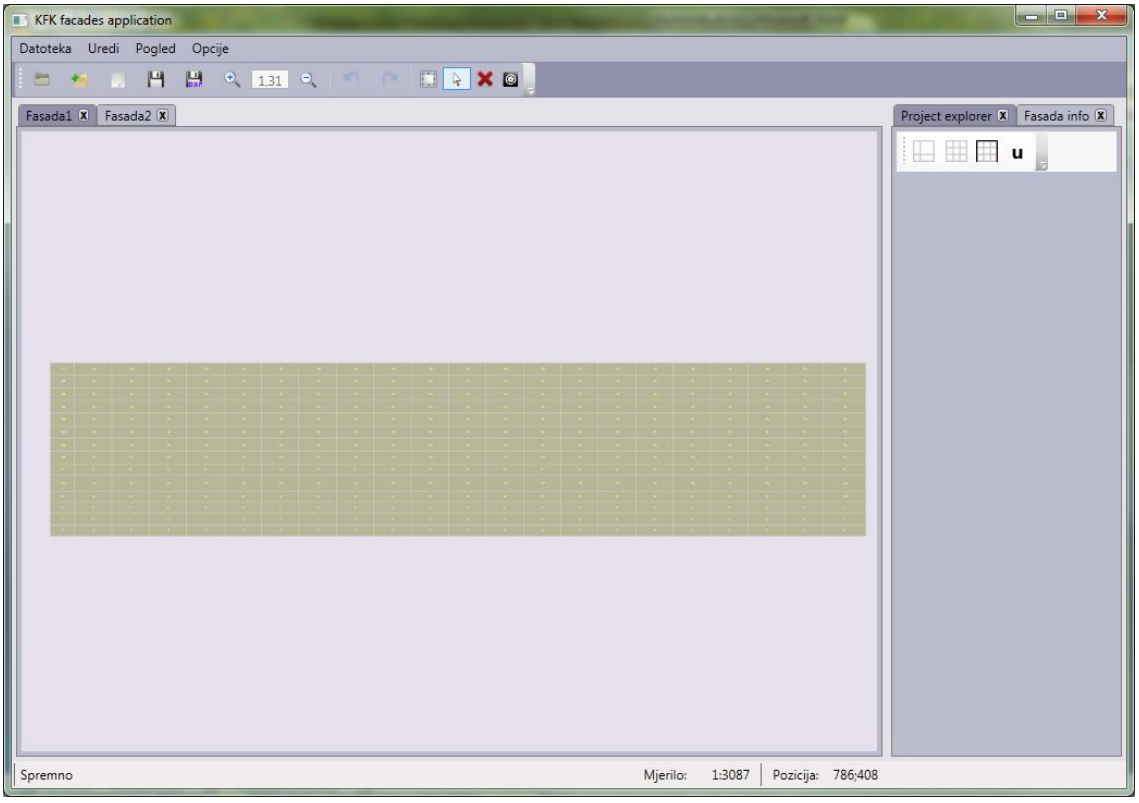
Slika 6.11 Dijalog za definiranje parametara plošnih elemenata

Na sličan način su definirani dijalozi s parametrima podkonstrukcijskih te krovnih elemenata.

Po završetku rada s čarobnjakom se generiraju pročelja. Dodaju se Tab kontrole sa radnim površinama za manipulaciju elementima, te se dodaje Tab kontrola „FasadaInfo“ među radna sučelja. Rezultati prethodnog primjera definicije su vidljivi na sljedećim dvjema slikama.



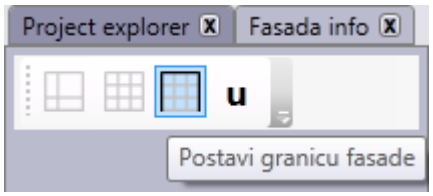
Slika 6.13 Na slici je vidljiva radna površina sa mrežom elemenata definiranih prema prvom tipu. TreeView kontrola radnog sučelja "Project explorer" je osvježena u skladu sa trenutnim stanjem projekta.



Slika 6.12 Na slici je vidljiva radna površina sa mrežom elemenata definiranih prema drugom tipu. Među radnim sučeljima je fokusirana Tab kontrola "FasadaInfo" koja sadrži elemente vezane uz uređivanje trenutno fokusirane radne površine.

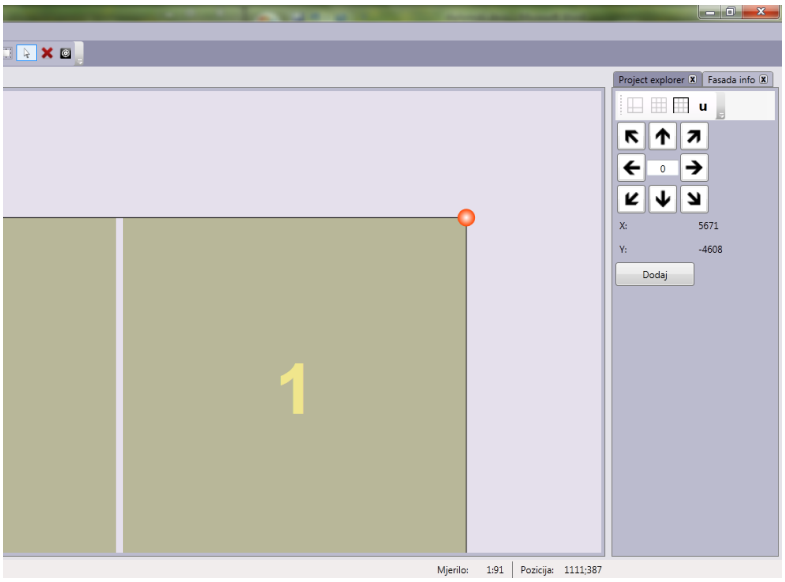
6.3 Promjena granice pročelja

Promjena granice pročelja započinje odabirom opcije „Postavi granicu fasade“.



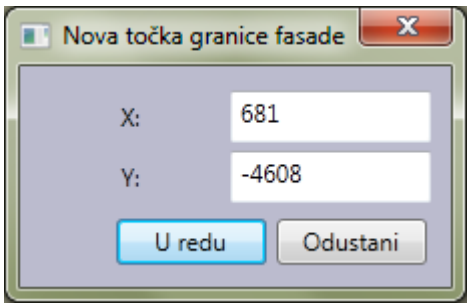
Slika 6.14 Opcija "Postavi granicu fasade"

Tom opcijom se na radnoj površini prikazuju linije granice pročelja sa *resize* točkama. Odabirom neke od točaka se na radnom sučelju pojavljuju kontekstualne opcije. Moguće je unesti vrijednost za odabranu točku te izvesti operaciju odabirom jedne od opcija smjera(slika 6.15).



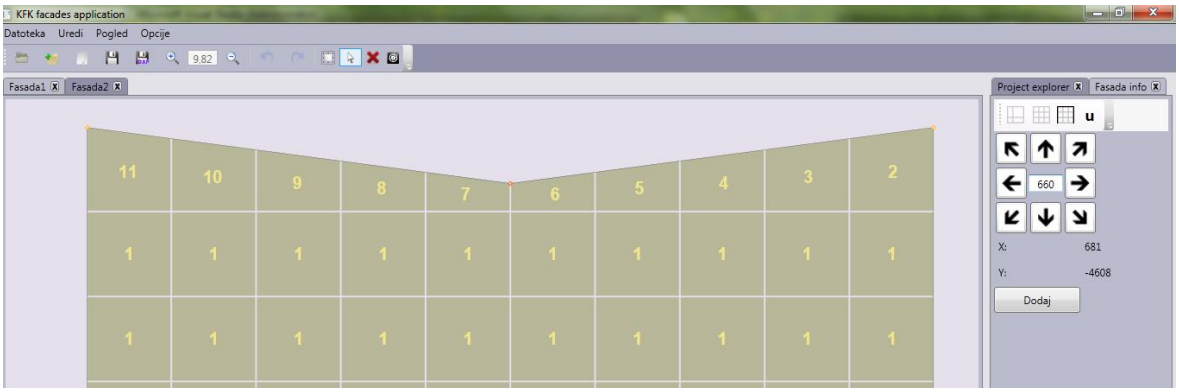
Slika 6.15 Odabir resize točke granice pročelja

Ukoliko se odabere opcija „Dodaj“, korisniku se prikazuje dijalog unutar kojeg se može odediti x i y koordinate nove *resize* točke (slika 6.16).



Slika 6.16 Dijalog za unos x i y koordinata nove *resize* točke

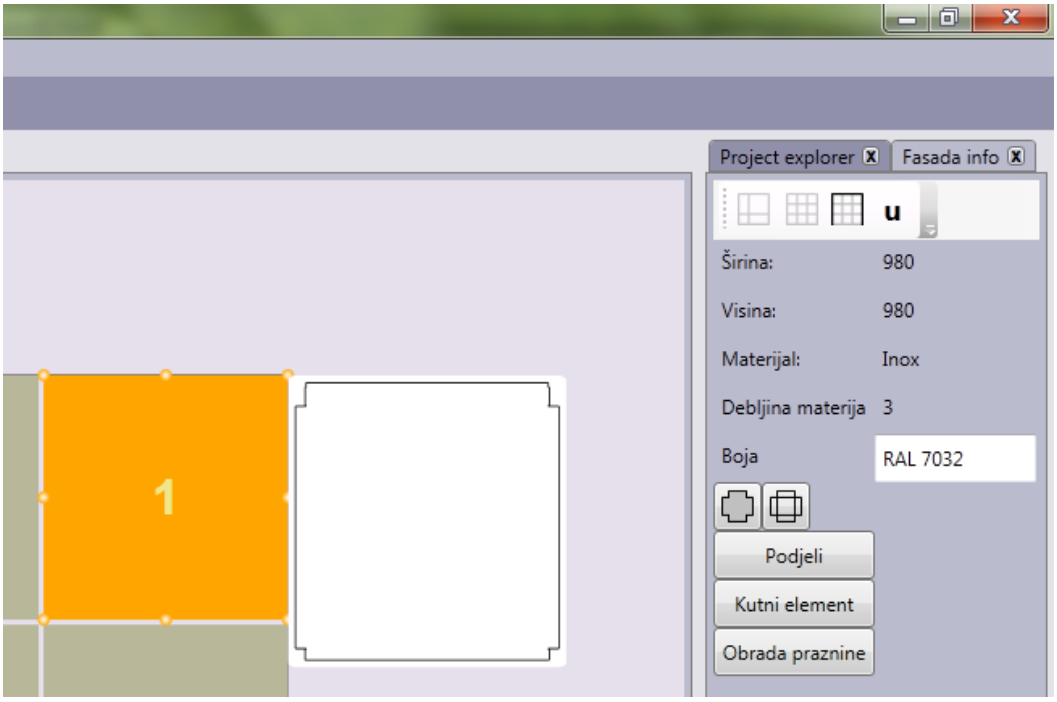
Na ovaj način se mogu postići razni oblici pročelja. Primjer rezultata promjene granice pročelja je prikazan sljedećom slikom.



Slika 6.17 Primjer rezultata promjene granice pročelja

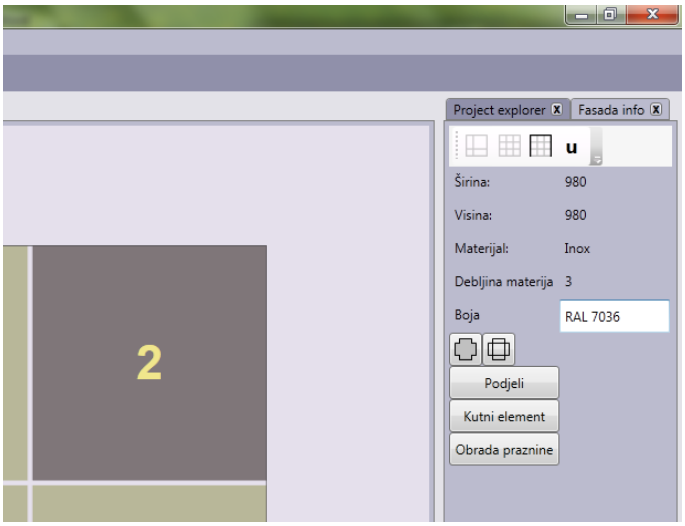
6.4 Upravljanje plošnim elementima

Korisnik može pregledati svojstva određenog plošnog elementa klikom na njega. Na radnom sučelju će se pokazati odgovarajuće informacije te moguće opcije. Također će se prelaskom kursora preko elementa prikazati i njegov razvijeni oblik.



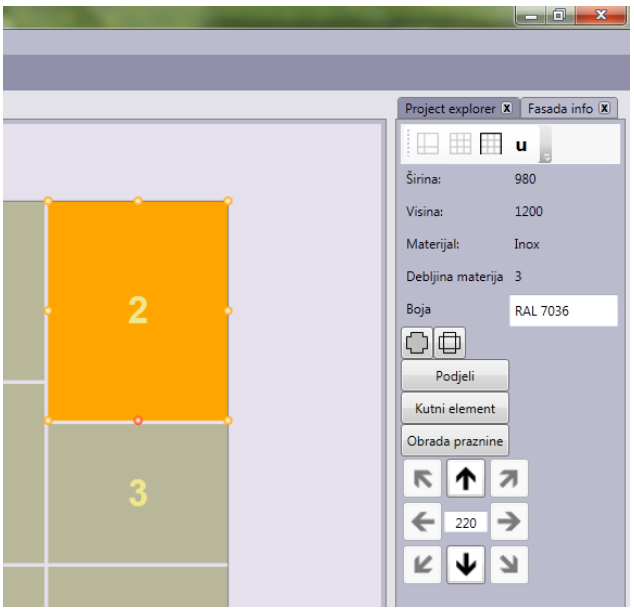
Slika 6.18 Pregled informacija o plošnom elementu

Boju elementa je moguće promijeniti izravno mijenjajući unos u TextBox kontroli te *Enter* potvrdom.



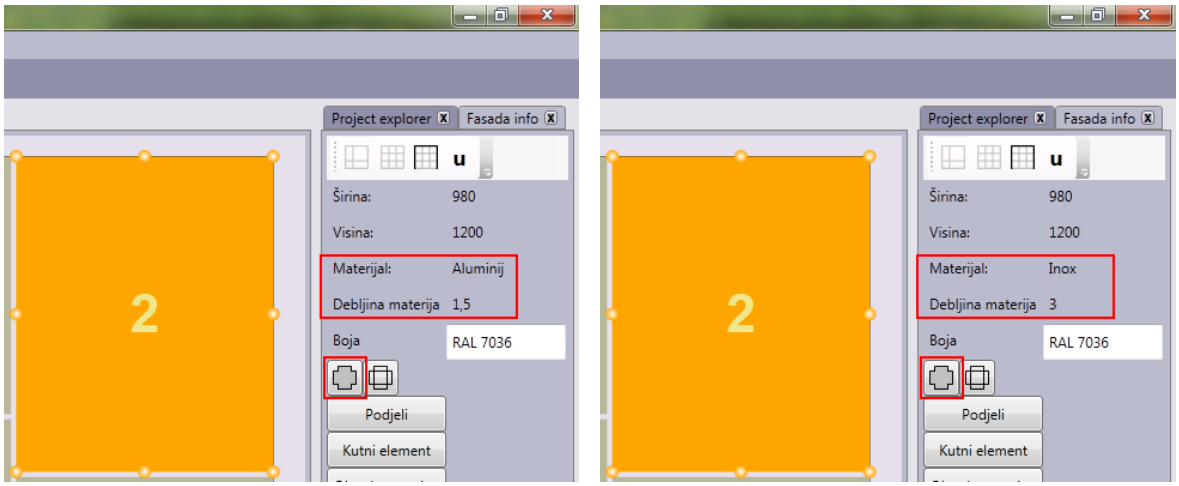
Slika 6.19 Stanje nakon promjene boje plošnog elementa

Oblik se mijenja na sličan način kao što se mijenja granica pročelja. Kako korisnik označi element, pojavljuju se *resize* točke. Odabirom jedne od točaka pojavljuju se *resize* opcije te korisnik odabirom smjera izvodi *resize* na temelju *resize* vrijednosti.



Slika 6.20 Primjer stanja nakon obavljanja resize operacije. U ovom slučaju je prethodno obavljenja resize operacija nad elementom 3 kako nebi došlo do sudaranja sa elementom 2.

Promjenu materijala i parametara plošnog elementa se obavlja odabirom opcija „Materijal“ i „Parametri“. Otvaraju se dijalozi identični onima prilikom definiranja materijala i parametara u čarobnjaku za stvaranje projekta.

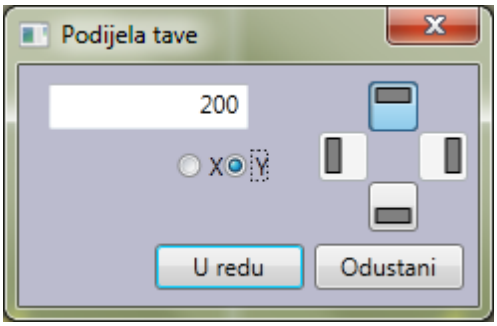


Slika 6.21 Stanje prije i nakon promjene materijala

Na isti način se obavlja promjena parametara.

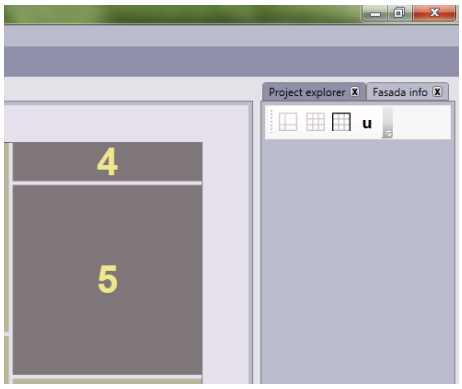
Funkcionalnost brisanja se obavlja pritiskom *Delete* tipke na tastaturi. Brišu se svi označeni elementi.

Elementi se mogu dijeliti i spajati. Prethodno operaciji podjele elementa potrebno je opisati način podjele. Odabirom opcije „Podjeli“ se otvara dijalog prikazan sljedećom slikom.



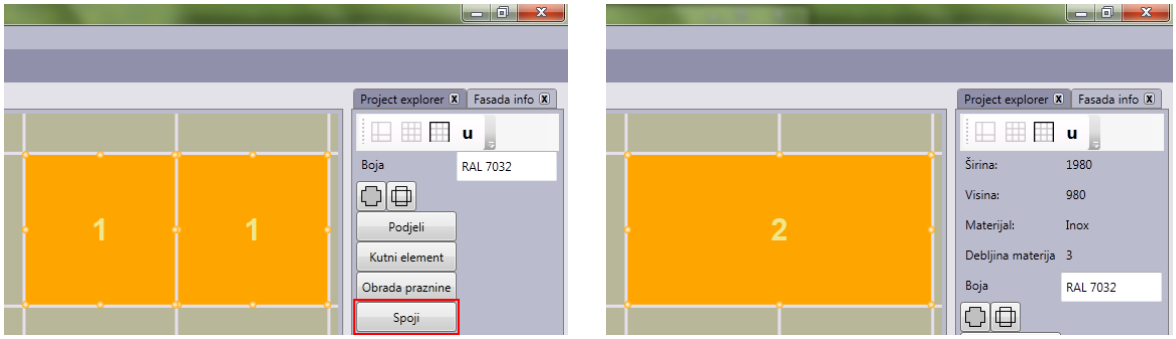
Slika 6.22 Dijalog za definiranje podjele elementa

Moguće je odabrati horizontalnu ili vertikalnu podjelu, te pripadajuću stranu podjele. Potvrdom se izvršava definirana podjela. Rezultat podjele iz prethodnog primjera je prikazan sljedećom slikom.



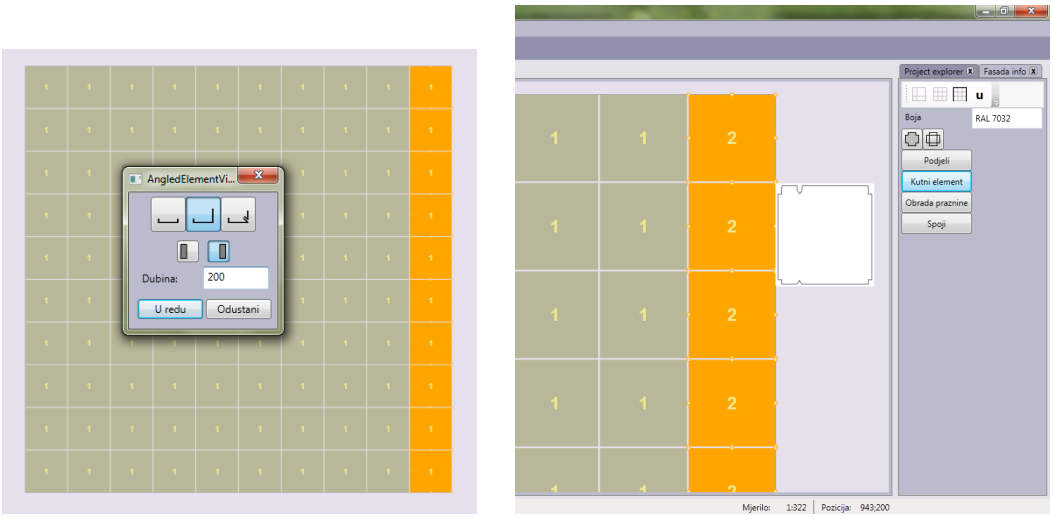
Slika 6.23 Primjer rezultata podjele prema prethodnoj definiciji

Spajanje elemenata se odvija odabirom opcije „Spoji“ koja se pojavljuje prilikom označavanja dva ili više plošnih elemenata. Primjer spajanja je prikazan sljedećim dvjema slikama.



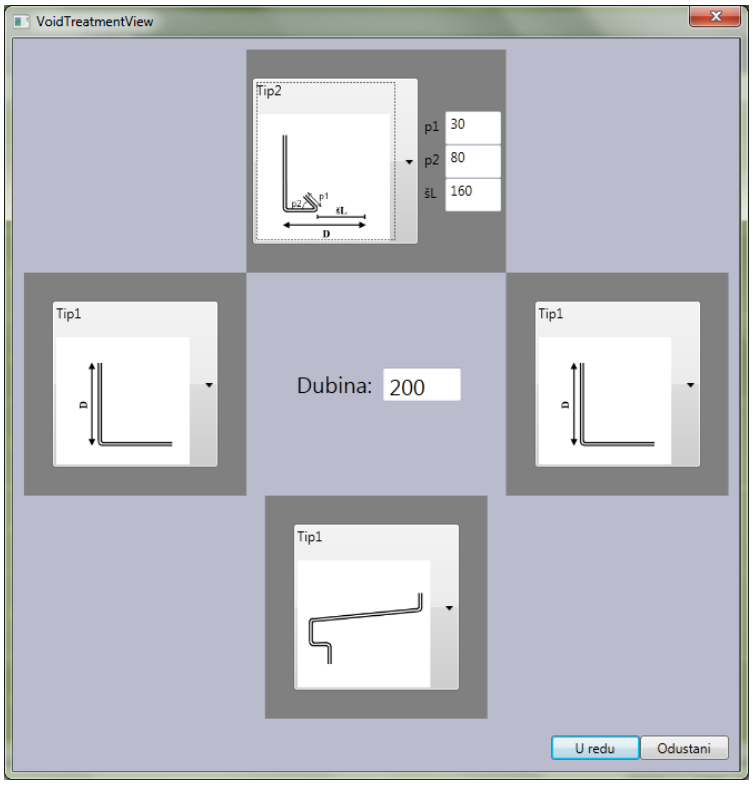
Slika 6.24 Primjer spajanja plošnih elemenata

Preoblikovanje plošnih elemenata u kutne elemente se izvršava odabirom opcije „Kutni element“. Po odabiru opcije se otvara dijalog gdje se bira način kutne obrade te pripadajući parametar. Potvrdom se element preoblikuje u kutni element. Primjer preoblikovanja je dan sljedećim slikama.

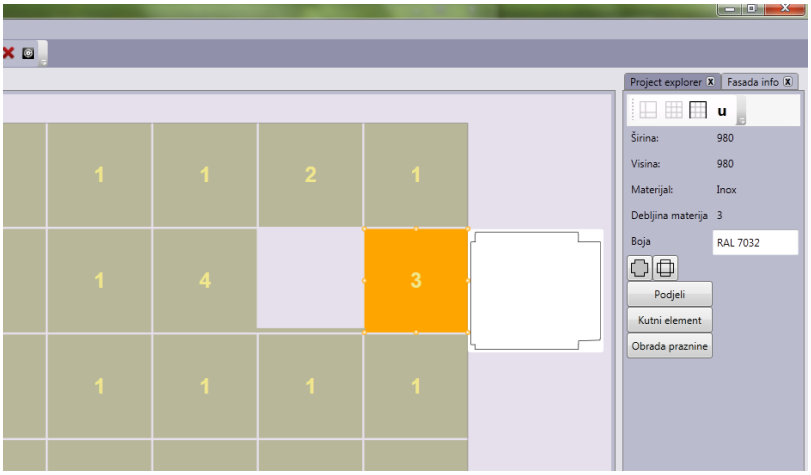


Slika 6.25 Primjer kutne obrade

Preoblikovanje plošnih elemenata u elemente obrade otvora se izvršava odabirom opcije „Obrada praznine“. Po odabiru se pojavljuje dijalog sa mogućnostima odabira tipa obrade. Odabrani element će se smatrati otvorom (prozorom ili vratima), a okolne elemente će sustav sam detektirati te na njih primijeniti željene tipove obrada. Dodatno će se pridodati potporni element u slučaju da je otvor prozor.



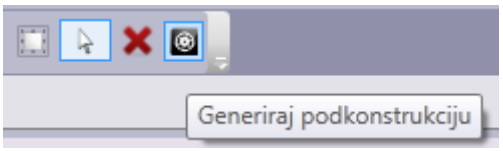
Slika 6.26 Dijalog za umetanje otvora i obradu susjednih elemenata



Slika 6.27 Rezultat umetanja prozora

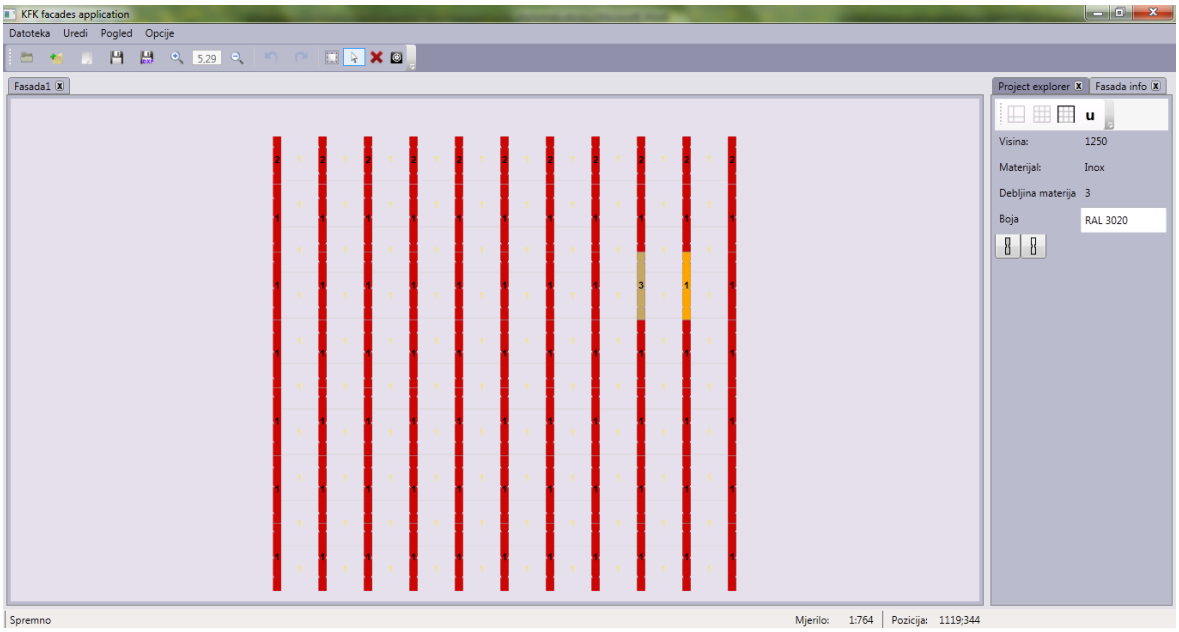
6.5 Upravljanje podkonstrukcijom

Podkonstrukciju se generira na temelju stanja pročelja odabirom opcije „Generiraj podkonstrukciju“ iz glavne alatne trake. Podkonstrukcija se generira prema definiciji iz čarobnjaka za stvaranje projekta.



Slika 6.28 Opcija za generiranje podkonstrukcije

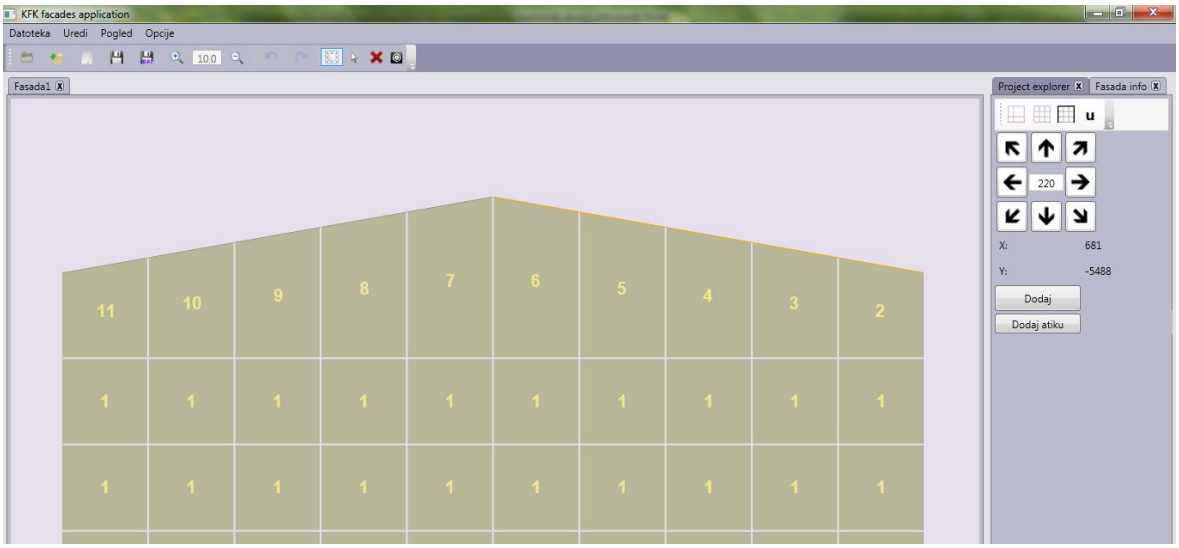
Odabirom pojedinog nosača moguće je vidjeti njegova svojstva te promijeniti mu boju, materijal ili parametre.



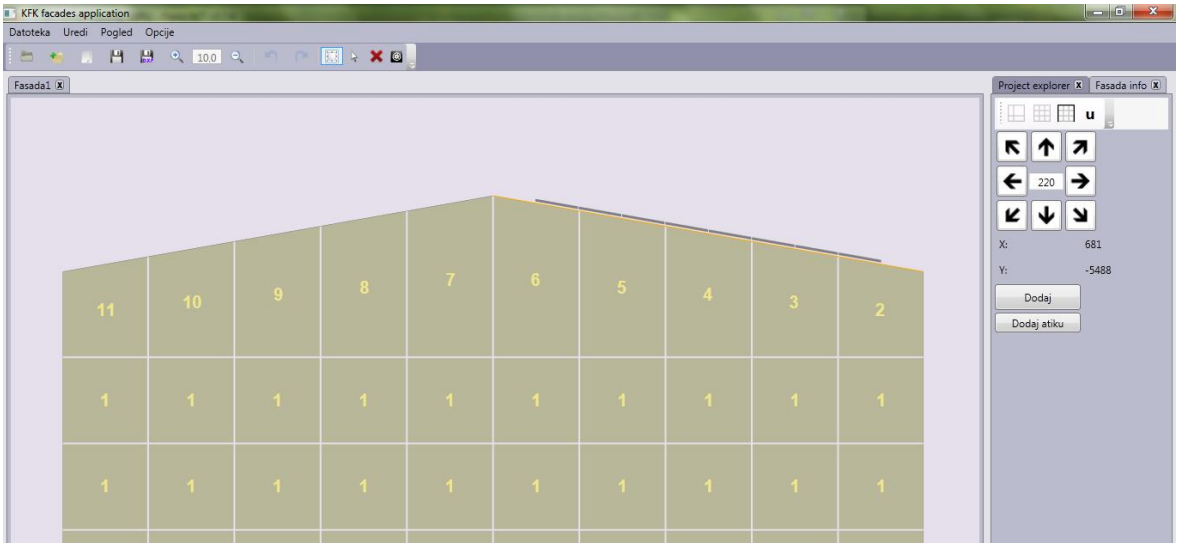
Slika 6.29 Primjer rezultata generiranja podkonstrukcije uz naknadno promjenjenu boju nosača

6.6 Upravljanje krovnim elementima

Za dodavanje krovnih elemenata potrebno je prethodno postaviti granice odabirom opcije „Postavi granicu fasade“. Potom je potrebno označiti jednu od gornjih linija pročelja za koje se žele generirati krovni elementi. Pojaviti će se opcija „Dodaj atiku“ čijim se odabirom generiraju krovni elementi prema definiciji iz čarobnjaka za stvaranje projekta.



Slika 6.31 Označena linija granice pročelja i ponuđena opcija za dodavanje završnih krovnih elemenata

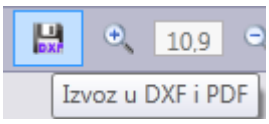


Slika 6.30 Rezultat dodavanja završnih krovnih elemenata

Na uobičajeni način je odabirom pojedinog elementa moguće doći do informacija o svojstvima te opcija o izmjeni materijala i parametara.

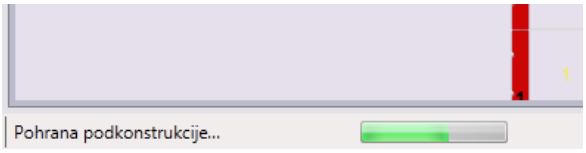
6.7 Izvoz razvijenih oblika i radioničke dokumentacije

Izvoz razvijenih oblika u DXF te radioničke dokumentacije u PDF format se pokreće odabirom opcije Izvoz u DXF i PDF iz glavne alatne trake.



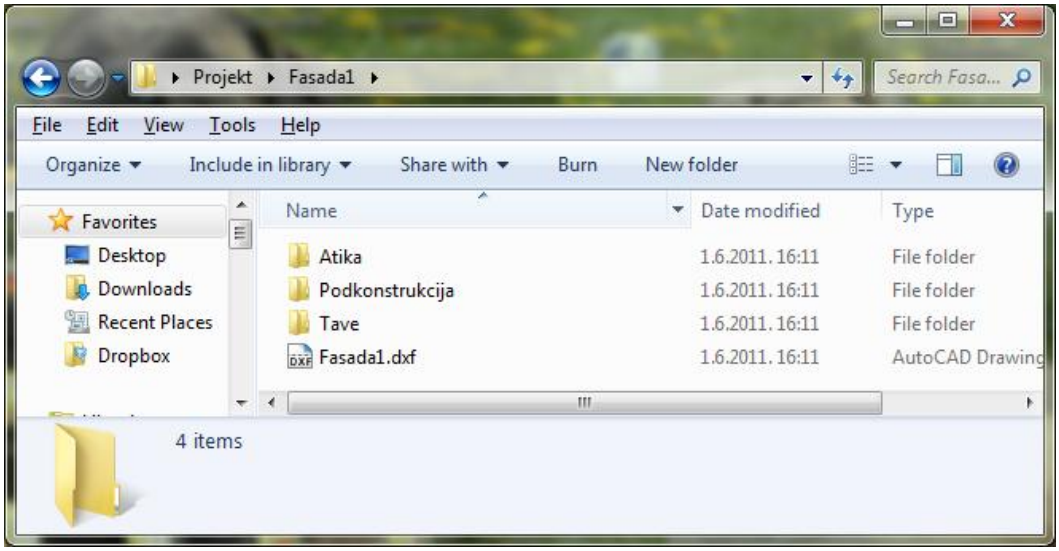
Slika 6.32 Opcija za izvoz u DXF i PDF format

Prilikom procesa izvoza, korisnik može pratiti napredak putem ProgressBar kontrole te poruka koje se pojavljuju na StatusBar kontroli.

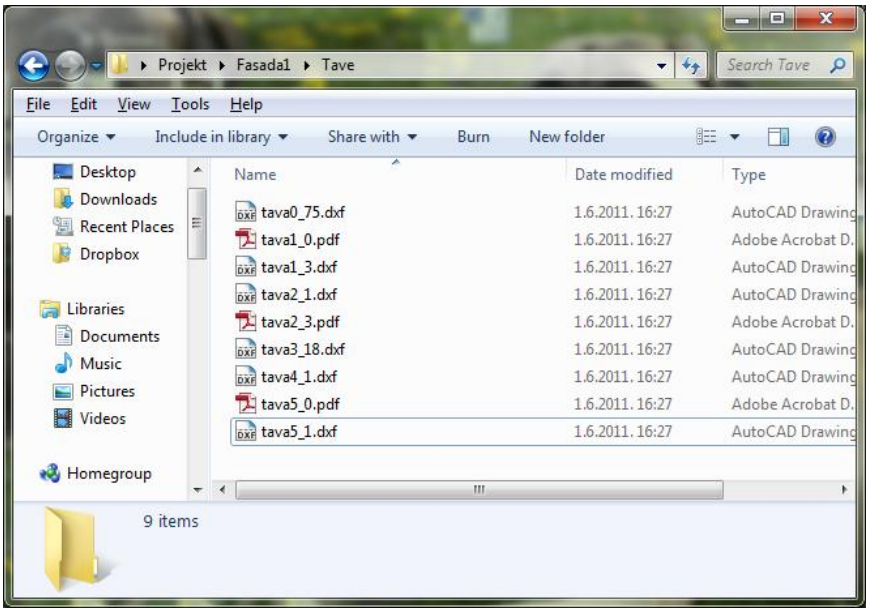


Slika 6.33 Informacije tokom pohrane

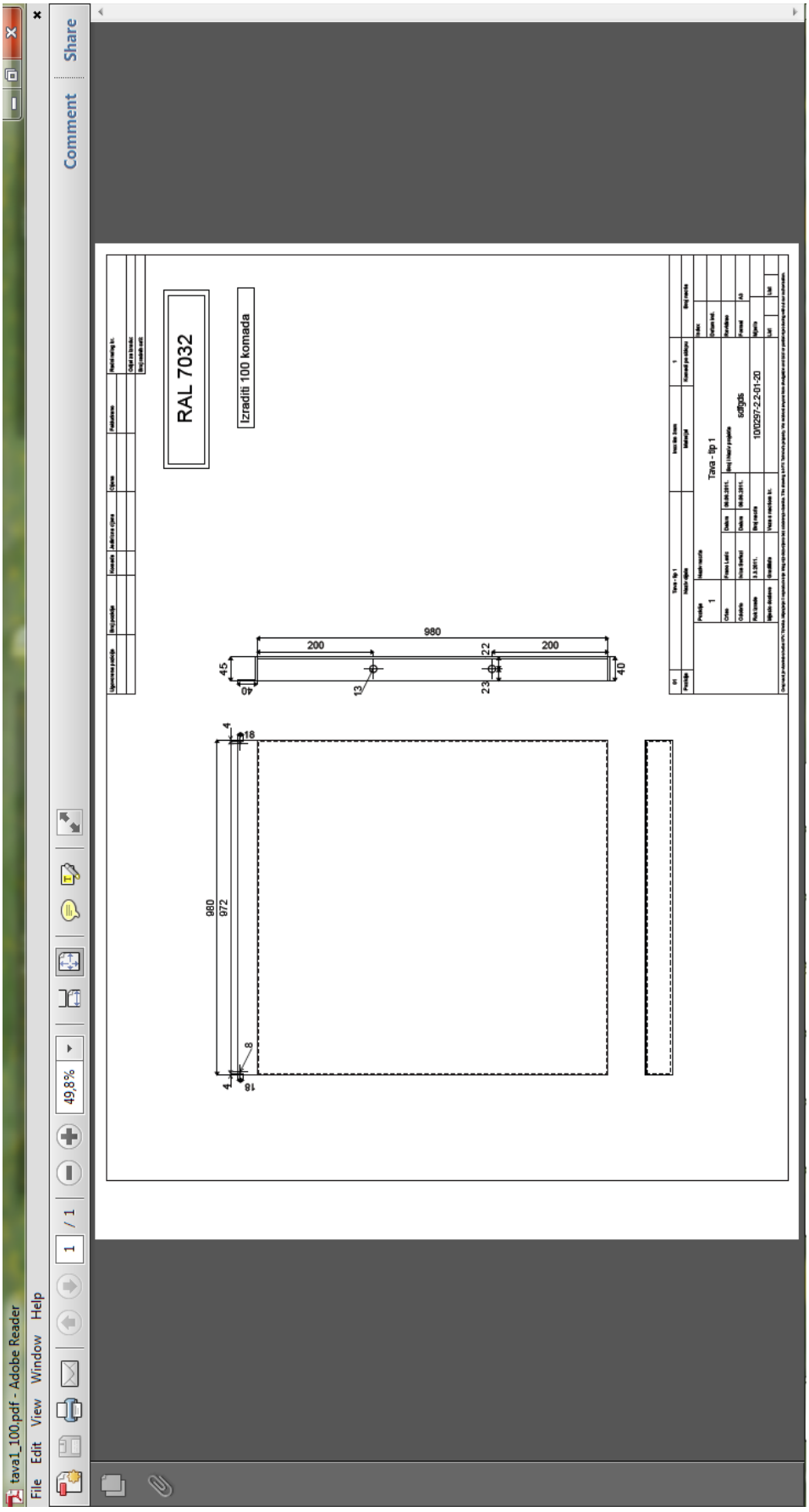
Datoteke se pohranjuju u mapama prema specifikaciji.



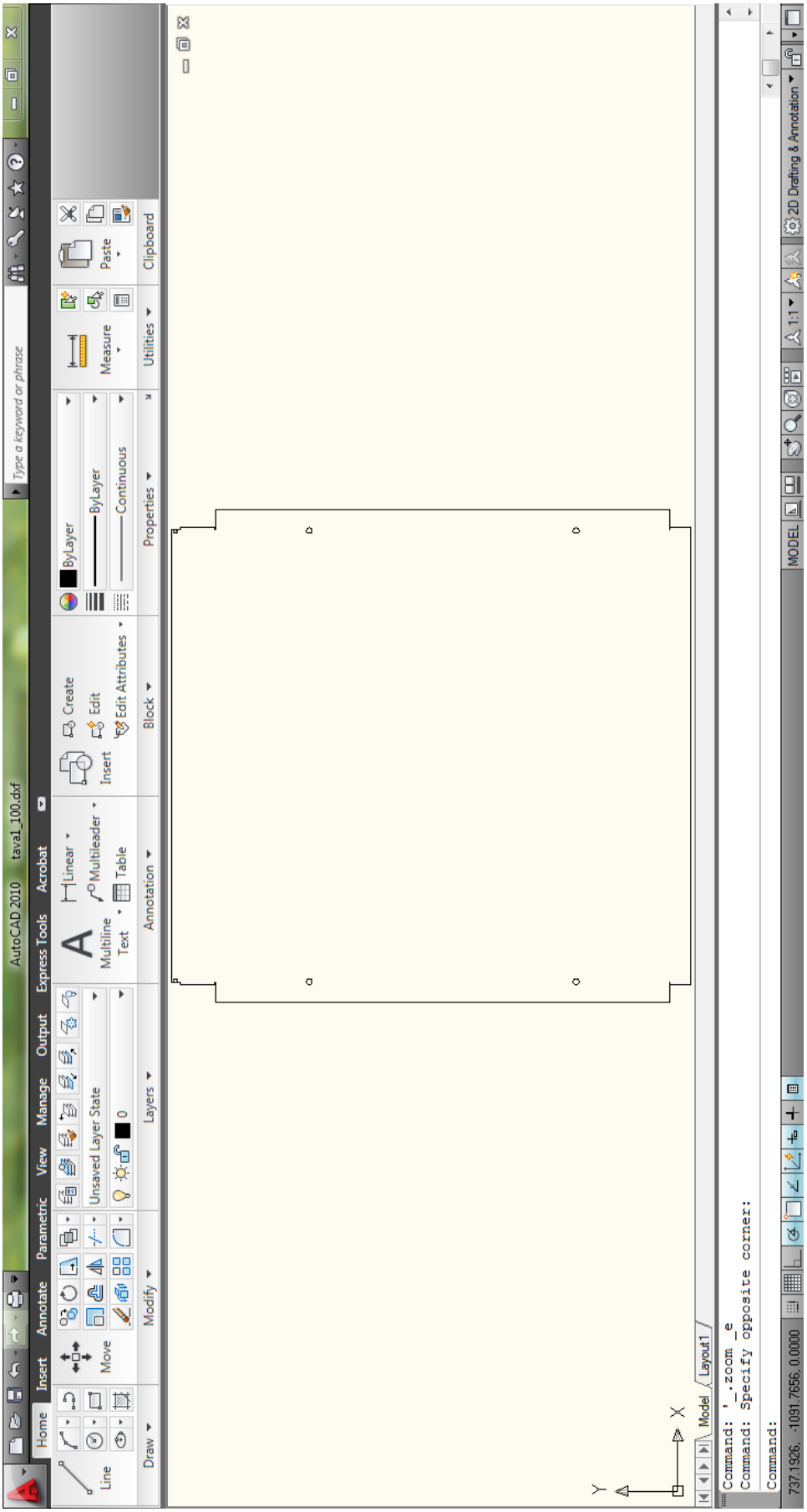
Slika 6.34 Primjer mape pročelja u kojoj su pohranjeni pripadni elementi



Slika 6.35 Primjer mape plošnih elemenata



Slika 6.36 Primjer generirane radioničke dokumentacije plošnog elementa.



Slika 6.37 Primjer generiranog nacrta plošnog elementa u DXF formatu

7 Optimizacijski algoritam

Unutar ovog poglavlja se nalazi opis algoritma implementiranog sa svrhom izračuna optimalne količine potrebnog materijala za izradu pročelja. Prvo poglavlje opisuje problemsko područje. Potom slijedi kratki osvrt na teorijsku pozadinu genetskih algoritama u poglavlju 7.2 te opis implementiranog genetskog algoritma u poglavlju 7.3. Na kraju je provedena analiza ostvarenih rezultata.

7.1 Problemsko područje

Jedan od funkcionalnih zahtjeva sustava je omogućavanje izračuna optimalne količine materijala potrebnog za izradu elemenata pročelja. Ovakav problem je već poznat te se za njega u literaturi koristi naziv „problem 2D krojenja“. Osim u građevinskoj, problem je učestao i u tekstilnoj industriji.

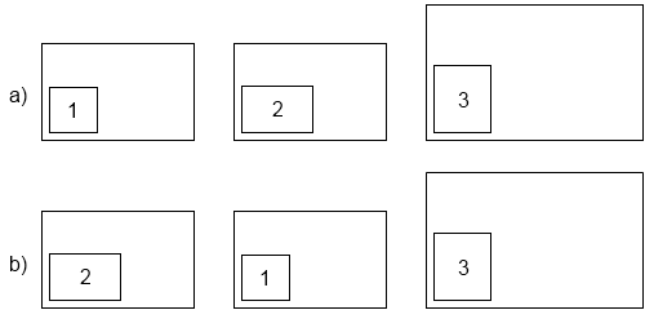
Potrebno je uzeti u obzir 3 moguće vrste dimenzija ploča iz kojeg se mogu izrezivati elementi pročelja: 1000mm sa 2000mm, 1250mm sa 2500mm te 1500mm sa 3000mm. Optimalno rješenje se sastoji od kombinacije takvih ploča koja će dati najmanje otpada nakon što se izrežu elementi pročelja.

Ako se uzme u obzir broj ploča, najveći broj mogućih rješenja uvjetuje da su svi elementi dovoljno mali da svi stanu na najmanju ploču. U tom slučaju vrijedi:

$$N_{mogućih\ rješenja} = N_{ploča}{}^{N_{elemenata}}$$

Formula 7.1 Izraz za izračun ukupnog broja mogućih rješenja

Međutim, u slučajevima kada se nalazi više ploča istih dimenzija, prema prethodnom izrazu će se uračunati i duplicirane raspodjele kao što je prikazano sljedećom slikom.



Slika 7.1 Primjeri raspodjela unutar istih ploča. Raspodjele u slučajevima a i b su redundantne.

Zbog navedenog razloga je potrebno uračunati sve moguće raspodjele između različitih ploča ($N_{različitih\ ploča} \in [1..3]$) unutar rješenja prema sljedećem izrazu:

$$N_{mogućih\ rješenja} = N_{različitih\ ploča}{}^{N_{elemenata}}$$

Formula 7.2 Izraz za izračun broja mogućih rješenja među različitim pločama

Zatim je potrebno dodati kombinacije moguće unutar ploča ali bez ponavljanja duplikata. U tu svrhu možemo promatrati samo dvije ploče istog tipa. Moguće kombinacije unutar dvije ploče iznose $2^{broj\ elemenata}$. Unutar tog broja se nalazi točno dvostruko duplikata. Dodatno, unutar tog broja se nalazi jedna raspodjela koja je uzeta u obzir prilikom prebrojavanja kombinacija između različitih ploča (ona sa svim elementima unutar jedne ploče). Izraz sada izgleda ovako:

$$N_{mogućih\ rješenja} = N_{različitih\ ploča}{}^{N_{elemenata}} + x + y + z$$

$$x = \begin{cases} 0;\ i = 0{,}1 \\ \frac{2^{N_{elemenata}}}{2} - 1;\ i > 1 \end{cases}, i - broj\ jednakih\ ploča\ najmanjih\ dimenzija$$

$$y = \begin{cases} 0;\ j = 0{,}1 \\ \frac{2^{N_{elemenata}}}{2} - 1;\ j > 1 \end{cases}, j - broj\ jednakih\ ploča\ srednjih\ dimenzija$$

$$z = \begin{cases} 0;\ k = 0{,}1 \\ \frac{2^{N_{elemenata}}}{2} - 1;\ k > 1 \end{cases}, k - broj\ jednakih\ ploča\ najvećih\ dimenzija$$

Formula 7.3 Izraz za izračun broja mogućih rješenja sa izuzetim duplikatima

Ovdje treba uzeti u obzir kako je teško unaprijed pretpostaviti točan broj i tipove ploča optimalnog rješenja. Također, još nije uzeta u obzir rotacija elemenata. Ako se uzme da se element može zarotirati 360 puta po jedan stupanj, to znači da je moguće $360^{broj\ elemenata}$ rotacijskih kombinacija. Konačno, najveći broj mogućih rješenja ovisno o broju elemenata te broju i tipu ploča iznosi:

$$N_{mogućih\ rješenja} = (N_{različitih\ ploča}{}^{N_{elemenata}} + x + y + z) \cdot 360^{N_{elemenata}},$$

$$\text{uz } N_{različitih\ ploča} \in [1..3] \text{ te}$$

$$\text{prethodno definirane } x, y, z$$

Formula 7.4 Konačni izraz za izračun ukupnog broja rješenja u najgorem slučaju

Pretraživanje cijelog prostora rješenja je vremenski prezahtjevno zbog dva faktora: broj elemenata koji kod pročelja doseže više stotina te nemogućnost predviđanja optimalnog broja i tipova ploča. Zbog toga je, uz određene kompromisne pretpostavke, implementiran genetski algoritam koji bi se sa svakom iteracijom izvođenja približavao optimalnom rješenju.

7.2 Genetski algoritam

Evolucijski algoritmi pri rješavanju problema preslikavaju pristup prirodne evolucije. Oponašajući prirodnu evoluciju nad jedinkama se primjenjuju operatori selekcije, križanja i mutacije.

Općenito za svaki evolucijski algoritam je potrebno pretpostaviti rješenja nad kojima će algoritam djelovati. Takva rješenja su slučajna i nazivaju se jedinke. Jedinke se sastoje od gena, a skup jedinki predstavlja populaciju. Generiranje populacije predstavlja početni korak u radu genetskog algoritma. Na veličinu populacije utječu:

- zahtjevnost problema i veličina prostora rješenja
- dostupna količina memorije
- vremensko ograničenje unutar kojega problem mora biti riješen

Prvi čimbenik nije uvijek poznat na početku stvaranja populacije, tako da je najbolje mijenjati veličinu populacije i na temelju dobivenog rješenja, koje se ne mijenja znakovito sa daljnjim povećanjem, zaključiti koja veličina je najprikladnija. Drugi čimbenik je važan s obzirom da generirana populacija i svi potomci moraju biti pohranjeni u memoriji računala. To postaje problem ukoliko se populacija sastoji od vrlo velikog broja kompleksnih rješenja. Na treći čimbenik utječu želje korisnika gdje se pri zahtjevnijim zadacima mora raditi kompromis između sporog i kvalitetnog te brzog i ne toliko kvalitetnog rješenja.

Populacija će tokom rada algoritma doživljavati brojne preinake. Početna populacija će se dodati posebnoj populaciji nastaloj pod utjecajem mutacije i križanja jedinki. Potom će se ukupna populacija podvrgnuti selekciji te će preživjele jedinke oformiti trenutnu populaciju koja će dalje ispočetka sudjelovati u iteraciji algoritma.

Genetskim operatorom križanja novonastala jedinka se stvara od genetskog materijala dvaju roditeljskih jedinki. Kako će algoritam ostavljati na životu sve bolje jedinke, očekuje se da će križanjem nastajati i sve bolji potomci.

Mutacija mijenja pojedine slučajno odabrane gene unutar jedinke. Ovisno o namjeni algoritma može se mijenjati intenzitet mutacije odnosno broj gena koji će biti podvrgnuti mutaciji.

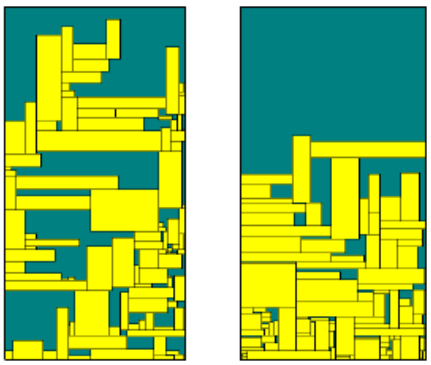
Funkcijom dobrote se definira koje su značajke poželjne kod pojedine jedinke, te o njezinom postavljanju ovisi ispravan rad algoritma. Funkcija evaluira svaku jedinku te dobiveni rezultati utječu na preživljavanje određene jedinke.

Selekcijom se iz ukupne populacije pomoću funkciju dobrote odabiru jedinke koje će činiti novu trenutnu populaciju.

7.3 Implementacija genetskog algoritma

Zbog velikog prostora mogućih rješenja, uspostavljeno je kompromisno pravilo kojim bi se smanjio prostor rotacijskih kombinacija. Svim razvijenim elementima pročelja se opisuje pravokutnik, te optimizacijski algoritam pokušava rasporediti isključivo takve oblike. Ovime su se rotacijska rješenja svela na 2 moguća po elementu (vertikalno ili horizontalno) odnosno jedno ukoliko je opisani pravokutnik ustvari kvadrat. Ovo i nije tako velika aproksimacija što je vidljivo prema opisu razvijenih elemenata u drugom poglavlju.

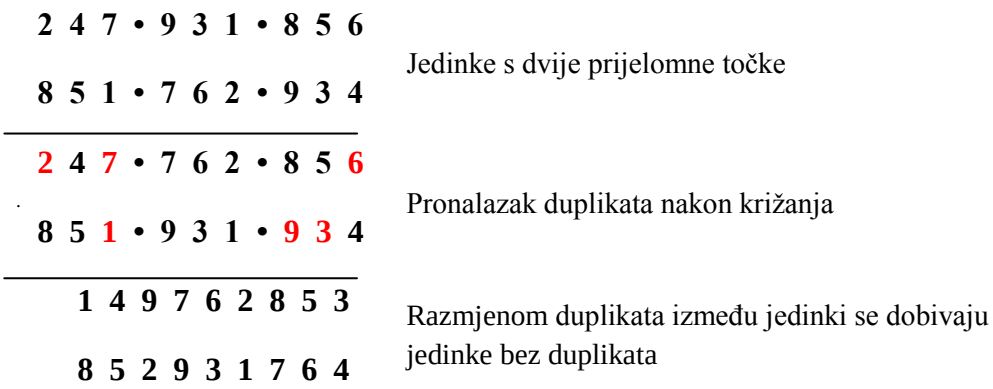
Za popunjavanje ploča materijala je korištena *bottom – left – fill* metoda. *Bottom – left* metodom je definirano da se elemente pokušava smjestiti što više prema donjem lijevom kutu. Događa se da elementi zatvore jedan dio neiskorištenog prostora. Bottom – left – fill metodom se pokušava manje elemente smjestiti u prostore koje su zatvorili drugi elementi. Usporedba ove dvije metode je dana slikom 7.2.



Slika 7.2 Primjer razmještaja elemenata bottom – left metodom i bottom – left – fill metodom [11]

Ovim načinom se generira početna populacija genetskog algoritma. Jednu jedinku čini niz ploča materijala sa raspoređenim opisanim pravokutnicima elemenata. Jedinke se potom evaluiraju te započinje prva iteracija algoritma. U svakoj iteraciji se na početku odvija 3-turnirska selekcija. Tim načinom selekcije se odabiru 3 jedinke iz rješenja. Najslabija jedinka se odbacuje, a na mjesto nje dolazi potomak nastao križanjem dvaju preživjelih jedinki. Ovime se postiže očuvanje najboljih jedinki te se nastoji ostvariti izmjena kvalitetnog genetskog materijala.

Križanje je ostvareno prema *partially matched* principu ilustriranom sljedećom slikom.



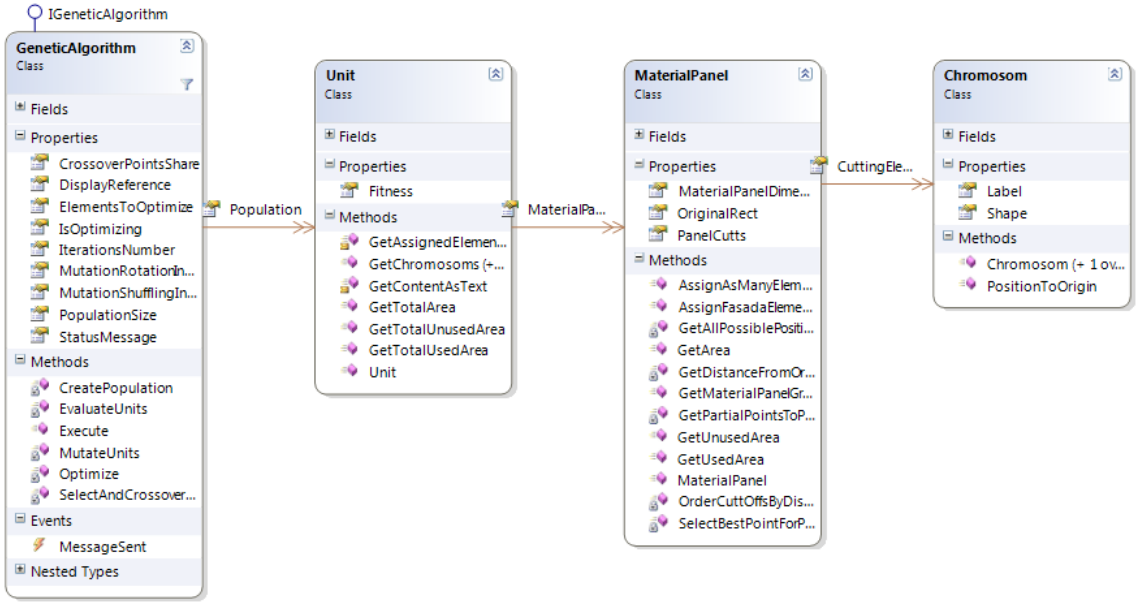
Slika 7.3 Primjer partially matched križanja

Nakon selekcije i križanja se nad novim jedinkama primjenjuje genetski operator mutacije. Mutacija se može izvesti na dva načina: translacijom i rotacijom. Translacija se obavlja na način da se oblik koji je dodijeljen jednoj ploči premjesti na drugu. Rotacijom se mijenja orijentacija oblika unutar ploče. Novonastale se jedinke evaluiraju te algoritam započinje sa novom iteracijom.

```
Genetski algoritam
{
        k = 0;
        Slučajnim postupkom generiraj početnu populaciju Pk veličine n;
        Evaluiraj Pk;
        Dok nije zadovoljen uvjet završetka evolucijskog procesa
        {
                Selektiraj3Turnirski
                {
                        Dok je broj jedinki Pk > 2
                        {
                                Uzmi 3 jedinke
                                Ukloni najlošiju iz Pk
                                Križaj preostale dvije jedinke

                                Dodaj dvije roditeljske jedinke u Pn1
                                Dodaj potomak u Pn2
                                Ukloni dvije roditeljske jedinke iz Pk
                        }
                        Ako je preostali broj jedinki > 0
                        {
                                Ukloni preostale jedinke iz Pk
                                Dodaj preostale jedinke u Pn1
                        }
                }
                Mutiraj jedinke iz Pn2
                Evaluiraj jedinke iz Pn2
                Spoji jedinke iz Pn1 i Pn2 u Pk
        }
}
```

Programski isječak 7.1 Pseudokôd genetskog algoritma

U modelu se nalazi cjelina sa implementacijom genetskog algoritma prema slici.



Slika 7.4 Implementacija genetskog algoritma. Za razred GeneticAlgorithm su prikazane samo najvažnije metode.

Evolucijski proces te genetski operatori su definirani u singleton razredu GeneticAlgorithm. Pozivom metode Optimize(), kojoj je potrebno predati pravokutne oblike za optimizaciju, algoritam započinje sa radom. Optimizacija se vrši u zasebnoj

dretvi kako bi se ispunio korisnički zahtjev funkcionalnosti sučelja tokom dugotrajnih operacije. GeneticAlgorithm također definira događaj MessageSent kojim dojavljuje pozivatelju informacije o trenutnim jedinkama te njihove grafičke reprezentacije.

Korisničko sučelje se može podijeliti na tri funkcionalne cjeline. Sa lijeve strane se nalaze promjenjivi parametri genetskog algoritma, stablasta struktura za odabir pročelja te opcije za pokretanje i zaustavljanje optimizacije. Najveći dio ekrana zauzima površina za prikaz jedinki. Za svaku jedinku su prikazane pripadajuće ploče te raspodjela oblika na njima. Oblici na sebi imaju jedinstvene oznaku kako bi se lakše pratilo promjene. Jedinke su međusobno razdvojene separatorom. Donji dio sučelja sadrži poruke GeneticAlgorithm instance o stanju jedinki kroz iteracije.

Slika 7.5 Korisničko sučelje za uporabu genetskog algoritma

7.4 Analiza rezultata

Korisničko sučelje pruža mogućnosti za definiciju veličine populacije, broja iteracija algoritma te parametara genetskih operatora. Kao parametre mutacije je moguće postaviti učestalost rotacije elementa i preraspodjele. Vrijednosti se unose u kao postotci učestalosti. Tako će za, na primjer, vrijednost 20 postavljenu kao parametar rotacije, 20% od ukupnog broja elemenata biti zarotirano tokom mutacije. Vrijednost parametra križanja se također unosi kao postotak, a odnosi se na broj prijelomnih točaka kod križanja. Tako da će se za, na primjer, unos 100 pretpostaviti prijelomna točka između svakog pojedinog elementa. Broj prijelomnih točaka će se postavljati razmjerno do vrijednosti 1 kada će bez

obzira na postotnu vrijednost biti postavljena samo jedna prijelomna točka. Za vrijednost 0 neće biti postavljena niti jedna prijelomna točka, odnosno genetski operator križanja neće imati nikakvog utjecaja.

Temeljitim ispitivanjem je utvrđeno kako su optimalni parametri za efikasnu pretragu sljedeći:

- Veličina populacije – 15
- Mutacija – rotacija – 20
- Mutacija – preraspodjela – 20
- Križanje – udio prijelomnih točaka – 1

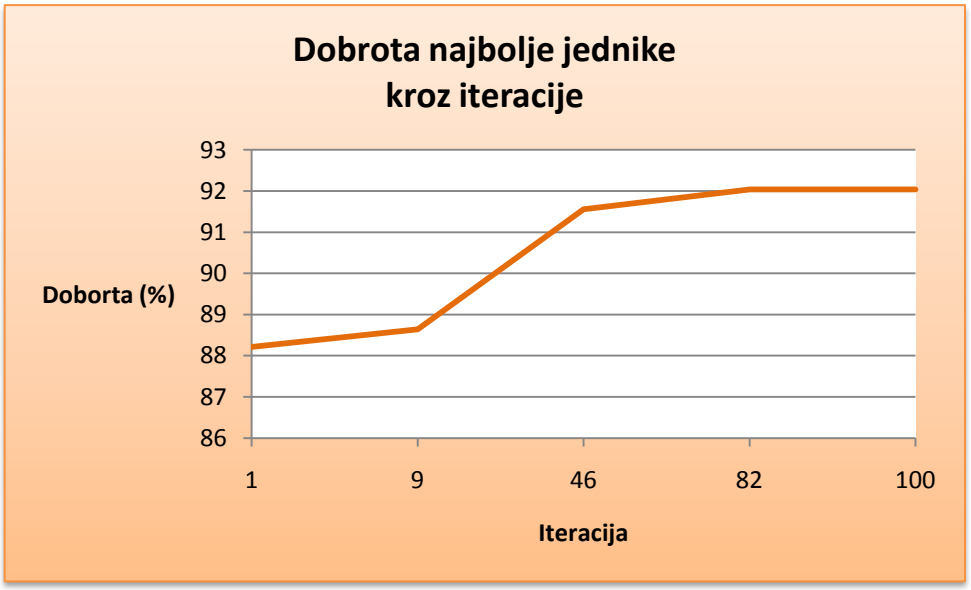
Valja napomenuti kako se kod mutacije rotacijom ne događa rotacija količine elemenata definiranih parametrom. To je iz razloga što su neki elementi predugački da bi ih se moglo iz horizontalne orijentacije postaviti u vertikalnu. Također, postoje pravokutni elementi koji su ustvari kvadrati. Tako da na njih mutacija nema nikakvog utjecaja.

Provedeno je ispitivanje nad testnim podacima. Podaci su uključivali 100 elemenata koji svojim dimenzijama odgovaraju učestalim dimenzijama oblika koji se pojavljuju na pročeljima. Dodatno, kako bi se bolje ispitala efikasnost algoritma, uvedeni su neki elementi proizvoljnih dimenzija. Rezultati su prikazani sljedećom tablicom. Dobrote jedinki su izražene u postotku iskorištenosti materijala.

Redni broj pokretanja	Najbolja jedinka (početak)	Najbolja jedinka (kraj)
1.	88,34	91,02
2.	87,27	89,56
3.	89,66	91,56
4.	88,21	92,04
5.	86,54	88,27
6.	85,32	89,13
7.	87,69	90,34
8.	88,27	90,75
9.	90,12	91,2
10.	88,45	92,00
SV	**87,99**	**90,59**

Tablica 7.1 Rezultati ispitivanja algoritma uz optimalne parametre sa srednjim vrijednostima dobrota početnih i završnih najboljih jedinki

Sljedećim prikazima je dan primjer napretka algoritma kroz iteracije te izgled konačnog rješenja (slučaj 4 iz prethodne tablice).



Slika 7.6 Napredak algoritma kroz iteracija za 4. pokretanje

Slika 7.7 Rezultat algoritma pri četvrtom pokretanju

Iz rezultata je vidljivo kako sama *bottom – left – fill* metoda prilično dobro raspoređuje elemente u vidu iskoristivosti. Algoritam za 100 iteracija uz navedene parametre pronalazi rješenja za nekoliko postotaka bolja od početnih. Iako može djelovati malo, u poslovnom procesu nekoliko postotaka može biti od velikog financijskog značaja.

Iz grafičkog prikaza se može primijetiti kako algoritam sporo konvergira prema rješenju. Sasvim sigurno postoji prostor za dodatno poboljšanje algoritma. Prijedlog za buduću nadogradnju je određivanje dobrote na razini ploča kako bi se smanjio utjecaj genetskih operatora na ploče koje već jesu dobro iskorištene. Uz to bi se trebalo osmisliti način na koji će se obavljati križanje jedinki bez da se ispočetka generiraju ploče različitih dimenzija.

8 Zaključak

Rad na ovako složenom sustavu se pokazao kao vrlo korisno iskustvo. Razvoj je uključivao opsežno razmatranje modela domene, razmjerno sa brojnošću zahtjeva na sustav te složenošću poslovnog procesa. Međutim, najviše vremena je utrošeno na izradu prikladnog korisničkog sučelja te vizualizaciju podataka. Zahvaljujući WPF-u proces je prošao bezbolnije nego što je mogao uz neku drugu tehnologiju sličnih mogućnosti. U razmatranju utrošenog vremena se ne smije zaboraviti niti vrijeme upotrijebljeno za svladavanje Model – View – ViewModel obrasca koji je uveo neke nove koncepte koji utječu na krivulju učenja (eng. *learning curve*).

Na projektu se u praksi moglo pokazati brojna znanja stečena tokom obrazovanja. Od matematičkih, trigonometrijskih i geometrijskih operacija, preko kombinatorike do fizikalnih svojstava materijala.

Svakako, najviše se moglo primijeniti znanja stečenog na kolegijima vezanima uz programiranje i razvoj programske potpore. No usprkos tome na projektu je bilo puno „grešaka u koracima“ na koje se često upozorava. Neke su proizašle iz nedostatka iskustva a neke su karakteristične za iterativno – inkrementalni razvoj kakav je primijenjen. Tako su se događale situacije u kojima su se postavljali neodgovarajući rokovi dovršetka funkcionalnosti (kao rezultat pretjeranog optimizma odnosno pesimizma), nedovoljno definirani zahtjevi, nedovoljna suradnja sa ekspertima domene, mijenjajući zahtjevi, naknadne izmjene važnijih cjelina unutar modela domene i mnoge druge. Također su se pojavljivale oprečne greške poput s jedne strane nedovoljno razrađenog modela, a s druge strane bespotrebnog kompliciranja i forsiranja neprikladnih rješenja (*analysis paralysis*). Usprkos velikom broju grešaka, one su vrlo važne kako bi se na njima moglo naučiti i obogatiti iskustvo.

Za budući razvoj je bitno kontinuirano stjecanje znanja vezanog uz struku, kao i znanja vezanog uz poslovni proces, a trenutno izvedeno rješenje pruža dobre temelje za budući razvoj te nadogradnju sustava.

Literatura

1. Vanjak, Z. Materijali za predmet Objektno oblikovanje, Fakultet elektrotehnike i računarstva, Zagreb, 2009./2010.
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J. „DesignPatterns: Elements of Reusable Object – Oriented Software“, 1. izdanje, Adisson – Wesly, SAD, 1994.
3. Josh Smith on WPF, http://joshsmithonwpf.wordpress.com/, 22.5.2011
4. Wikipedia - .NET Framework, http://en.wikipedia.org/wiki/.NET_Framework, 22.5.2011.
5. Wikipedia – MVC, http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, 22.5.2011.
6. Wikipedia – MVVM, http://en.wikipedia.org/wiki/Model_View_ViewModel, 22.5.2011.
7. Microsoft Developer Network (MSDN), http://msdn.microsoft.com/en-us/default.aspx, 22.5.2011
8. MacDonald, M. „Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4“, 3. izdanje, Apress, SAD, 2010.
9. Xu, J. „Practical WPF Charts and Graphics“, 1. izdanje, Apress, SAD, 2009.
10. Lowagie, B. „iText in Action: Creating and Manipulating PDF“, 1. izdanje, Manning, New York, 2007.
11. Hopper, E., Turton, B. Computers and Industrial Engineering: A Genetic Algorithm for a 2D Industrial Packing Problem, University of Wales, School of Engineering, Cardiff, 1999.
12. Wang, B. Modern Applied Science: An Adaptive Genetic Algorithm for 2D Packing Problem, Tianjin University of Technology and Education, School of Mechanical Engineering, Tianjin, 2010.

Sažetak

Naslov

Programski sustav za automatiziranje procesa izrade pročelja u građevinarstvu.

Ključne riječi

WPF MVVM aplikacija, desktop aplikacija, sustav za izradu pročelja, građevinarstvo, genetski algoritam, 2D krojenje

Sažetak

Cilj ovog diplomskog rada je analizirati proces izrade građevinskih pročelja i potom oblikovati potporni programski sustav. U skladu s time, rad je podijeljen na dvije glavne cjeline.

U prvom djelu se opisuje postupak izrade pročelja, sastavne elemente te fizikalna svojstva materijala koji se koriste. Identificirani su zahtjevi na sustav koji treba razviti.

Druga cjelina opisuje razvoj programskog sustava te primijenjene alate i koncepte. Na početku je opisana Windows Presentation Foundation tehnologija koja se nametnula kao najprikladniji alat za izradu ovakvog sustava. Slijedi analiza Model – View – ViewModel arhitekturnog obrasca pogodnog za implementaciju WPF aplikacija. Konačno, opisano je praktično rješenje. Dan je opis slojevite arhitekture aplikacije te opis funkcionalnosti. Na kraju se nalazi izdvojeni opis optimizacijskog algoritma za pronalazak optimalne potrošnje materijala.

Summary

Title

Application for Automating Facade Development in Civil Engineering.

Keywords

WPF MVVM application, desktop application, facade development system, civil engineering, genetic algorithm, 2D packing

Abstract

The aim of this thesis is to analyze the process of facades construction and to develope a supporting application. Correspondingly, paper is divided into two main parts.

The first part describes the process of facade construction, constituent elements and physical properties of used materials. Requirements for the system to be developed are identified.

Second section describes development of the software system, utilized tools and applied concepts. At the beginning there is a description of Windows Presentation Foundation technology which imposed itself as the most suitable tool for development of this kind of application. Description is followed by analysys of Model – View – ViewModel architectural pattern designed for implementation of WPF applications. Finally, practical work of this thesis has been described. There is a description of layerd application architecture and embeded functionality. In the end, there is a separate description of an optimization algorithm used to search for an optimal material usage.