УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Рената Вадерна

ПРОШИРЕЊЕ КРОКИ АЛАТА ЗА СКИЦИРАЊЕ ПОСЛОВНИХ АПЛИКАЦИЈА ГРАФИЧКИМ UML ЕДИТОРОМ

МАСТЕР РАД

Нови Сад, 2013

	УНИВЕРЗИТЕТ У НОВОМ САДУ • **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	**Рачунарство и аутоматика**
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	**Проф. др Никола Јорговановић**

Студент:	Рената Вадерна	Број индекса:	E2 50/2012
Област:	Методе брзог развоја софтвера		
Ментор:	др Гордана Милосављевић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Проширење Кроки алата за скицирање пословних апликација графичким UML едитором

ТЕКСТ ЗАДАТКА:

Проширити алат Кроки намењен за скицирање форми пословних апликација графичким UML едитором, у циљу добијања још једног погледа на скицирану апликацију. Посебан акценат ставити на спецификацију корисничког интерфејса уз ослонац на UML дијаграм класа и EUIS UML профил, док спецификацију перзистентног слоја апликације оставити за касније фазе развоја. Документовати решење.

Руководилац студијског програма:	Ментор рада:

Примерак за: O - Студента; O - Ментора

Sadržaj

1. UVOD

Cilj ovog rada je proširenje alata Kroki, namenjenog za interaktivni razvoj poslovnih aplikacija baziranih na skicama, dodavanjem mogućnosti skiciranja korisničkog interfejsa aplikacija i pomoću UML dijagrama klasa. Takođe, cilj je i postavljanja osnove za kasniju implementaciju modelovanja perzistentnog sloja.

Kroki (fr. kroki – skica) je alat namenjan za interaktivni razvoj poslovnih aplikacija baziranih na skicama. Za razliku od uobičajene prakse, ovde se skica koristi u toku celog procesa razvoja softverskog proizvoda kao podloga za automatsko izvršavanje ili generisanje koda poslovnih aplikacija.[8]

Grafički editor dijagrama klasa predstavlja proširenje rešenja [1]. Osnovna ideja je da se projektantu ponudi još jedan način skiciranja korišćenjem UML klasa, njihovih atributa i metoda, kreiranje strukture projekta direktnim dodavanjem paketa na dijagram i pregleda trenutnog stanja u vidu dijagrama klasa koji se automatski kreira u slučaju skiciranja ostalim raspoloživim načinima (korišćenjem dizajnera formi ili komandne konzole)[8]. Projektant koji je više navikao na pomenuti način rada, može brže da vrši skiciranje tokom razgovora sa korisnikom i pregleda urađeno na njemu pregledniji način. Dijagrami automatski kreirani na osnovu postojećih skica formi se mogu dopunjavati i menjati na proizvoljan način, pri čemu se sve promene direktno reflektuju na skicama.

U nastavku teksta biće više rači o samom grafičkom editoru, Kroki alatu i njihovoj integraciji. Pregled Kroki alata dat je drugom poglavlju, dok treće poglavlje sadrži opis implementacije editora dijagrama klasa i njegove implementacije u okviru Kroki alata. U četvrtom poglavlju se može naći nekoliko primera upotrebe editora radi kreiranja novih modela i pregleda modela nastalih na osnovu postojećih skica, a u petom, finalnom, zaključak rada.

2. PREGLED KROKI ALATA

Kroki (fr. croquis – skica) je alat namenjan za interaktivni razvoj poslovnih aplikacija baziranih na skicama. Za razliku od uobičajene prakse, ovde se skica koristi u toku celog procesa razvoja softverskog proizvoda kao podloga za automatsko izvršavanje ili generisanje koda poslovnih aplikacija. Kroki implementira konkretnu sintaksu datog EUIS (*Enterprise User Interface Specification)* DSL-a (*Domain Specific Language*), koja je projektovana tako da se omogući modelovanje korisničkog interfejsa na „prirodan“ način. Pokretanje skica obezbeđuju dve generičke aspekt-orijentisane (AOP) web odnosno desktop aplikacije kreirane na bazi EUIS DSL-a. Kroki podržava skiciranje više vrsta formi, poštujući standard koji definiše njihov izgled i funkcionalnost.[8]

U nastavku će biti dat kratak pregled EUIS DSL-a, standarda korisničkog interfejsa na kojem je baziran i Kroki alata uz stavljanje akcenta na elemente bitne za razumevanje načina integracije sa editorom dijagrama klasa, bez ulaženja u detalje implementacije.

2.1 Standard korisničkog interfejsa poslovnih aplikacija

U ovom poglavlju biće dat prikaz samo onih elemenata standarda bitnih za formulaciju EUIS DSL-a, ali i podržanih od strane Kroki alata, dok se ceo opis može naći u [9]. U pitanju su standardni panel, standardna forma, *parent-child* i *many-to many* forma.

Standardna forma je projektovana sa ciljem da podaci i sve operacije nad njima budu vidljivi na ekranu tako da korisnik može da izabere podatak i pokrene operaciju nad njim. Standardne operacije (operacije zajedničke za sve entitete) su prikazane ikonama koje se nalaze u gornjem delu forme (*toolbar*), dok su specifične operacije prikazane dugmadima sa natpisima, i ukoliko postoje, nalaze se sa desne strane forme. Operacije nad podacima zajedničke za sve entitete, podržane od strane standardne forme su: pretraga po svim poljima (*query by form*), pregled (listanje), unos, izmena, brisanje, kopiranje, navigacija i promena pogleda (tabelarni prikaz ili "jedan ekran – jedan slog"). Specifične operacije uključuju kompleksne obrade podataka koji su vezani za dati entitet (transakcije), pozivanje izveštaja kao i aktiviranje vezanih (*Next*) formi.

Standardni panel poseduje izgled i ponašanje standardne forme ali umesto da je prikazan u svom prozoru, koristi se kao sastavni element

složenijih formi. Standardni paneli se najčešće koriste u *parent-child* i *many-to-many* formama. Panel standardne forme sadrži polja pridružena obeležjima entiteta i izvedena polja. Izvedena polja mogu biti: agregirana, kalkulisana i *lookup* polja.

Navigacija između ekranskih formi se odvija pomoću tri mehanizma: *Zoom, Next* i *activate*. Pod *zoom*-om se podrazumeva aktiviranje forme pridružene matičnom entitetu, iz koje korisnik može izabrati željeni red i preuzeti njegove vrednosti u okviru prethodne forme. *Combozoom* mehanizam nudi mogućnost izbora reda iz liste ponuđenih preko *comboBox* komponente, kao i aktiviranje pomenute forme putem pridrženog komandnog dugmeta. *Next* mehanizam predstavlja mehanizam za aktiviranje bliskih formi. Omogućava direktan prelazak sa forme pridružene parent entitetu na formu pridruženu child entitetu, tako da se u okviru poslednje nalaze samo podaci vezani za vrednost ključa tekućeg reda iz parent forme. Next se u okviru web aplikacija obično implementira u obliku linka koji aktivira drugu stranicu.

Activate mehanizam omogućava aktiviranje jedne forme od strane druge forme bez ikakvih ograničenja. Aktivirana forma ne mora biti iste vrste kao forma aktivator. Mehanizam se sa formama može implementirati kao stavka padajućeg menija ili kao link na web formi.

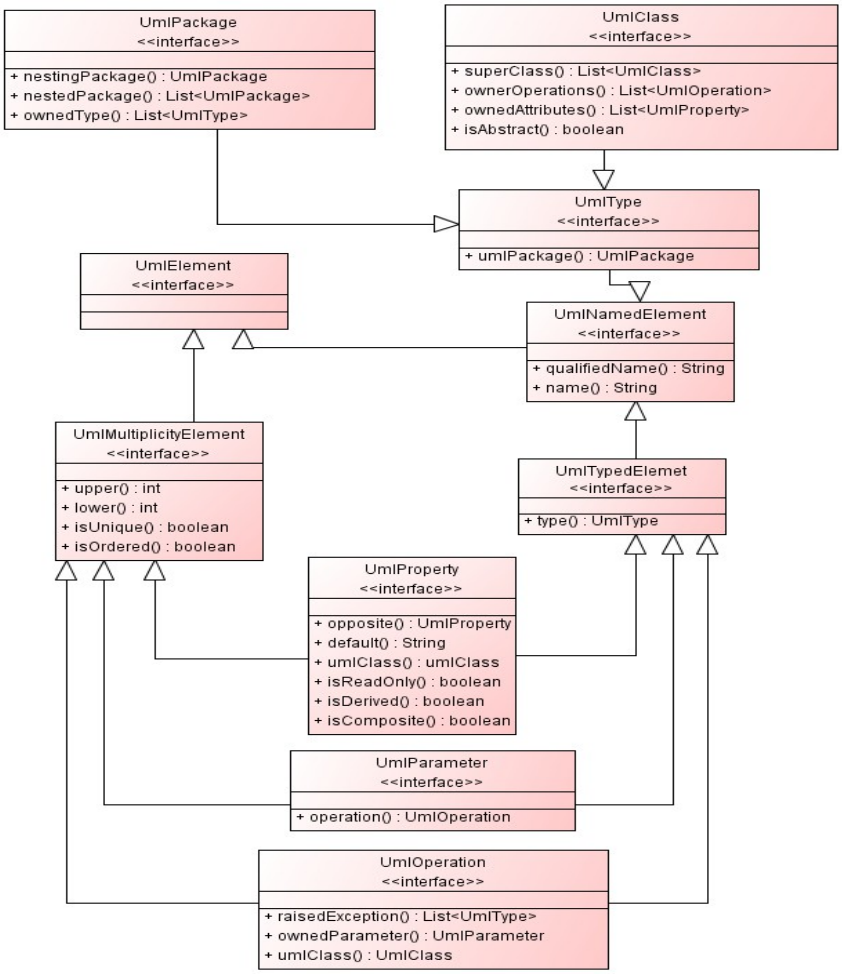
Parent-child forma se koristi za prikaz podataka koji imaju hijerarhijsku strukturu, gde je svaki element hijerarhije modelovan kao entitet u bazi podataka. Svaki element hijerarhije prikazan je u okviru standardnog panela, gde panel na n-tom nivou hijerarhije filtrira svoj sadržaj u zavisnosti od označenog podatka na nivou n-1.

Many-to-many forma se koristi za brz unos podataka u okviru entiteta nastalih kao posledica veza sa kardinalitetom "više na više", sa ili bez asocijativnih klasa.

2.2 UML Profil

EUIS DSL je implemetiran u vidu UML profila [10], koji je dalje korišćen pri izradi Kroki alata. Profil poseduje brojne stereotipove koji predstavljaju proširenje metaklasa *Element*, *Class*, *Property*, *Operation*, *Parameter*, *Constraint* i *Package* iz paketa UML::Kernel [8] i realizovani su u vidu interfejsa koji deklarišu metode za pribavljanje i podešavanje vrednosti obeležja koja su definisana unutar klasa elemenata UML profila. [11] Elementi *UML Core::Basic* biblioteke prikazani su na slici 2.1. (Dijagram prikazan na slici je napravljen i eksportovan upravo pomoću editora dijagrama klasa koji će biti opisan u ovom radu. Ista konstatacija važi i za sve ostale dijagrame koji će biti prikazani u ovom i narednim poglavljima).

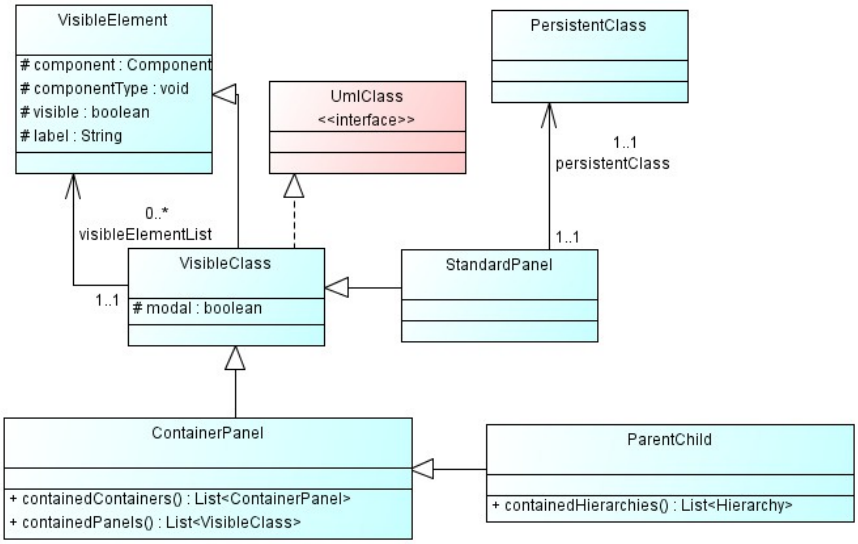
Osnovni element korisničkog interfejsa predstavlja stereotip *VisibleElement*, koji proširuje metaklasu *Element* i implementiran je istoimenom klasom. Vidljivim elementima pridružuje se komponenta korisničkog interfejsa (tekstualno polje, dugme, *comboBox* itd.) i natpis.



Slika 2.1 Elementi *UML Core::Basic* biblioteke implementirani korišćenjem programskog jezika Java u okviru Kroki alata

Vidljive klase proširuju metaklasu *Class*, reprezentujući klase koje se preslikavaju na panele različite vrste, koji će biti prikazani u narednom poglavlju. Standardni panel uvek se pridružuje nekoj perzistentnoj klasi, dok *parent-child* panel omogućuje hijerarhijsku organizaciju drugih

standarnih i *parent-child* panela. Vidljive klase implementirae su klasom *VisibleClass,* koju nasleđuju *StandardPanel i ContainerPanel,* koji predstavlja složeni panel koji može da sadrži druge panele. Klasa *ParentChild* je konkretni naslednik pomenute apstraktne klase. Dijagram klasa koji sadži pomenute klase prikazan je na slici 2.2. Opis značenja atributa i detaljnije objašnjenje može se naći u [11].



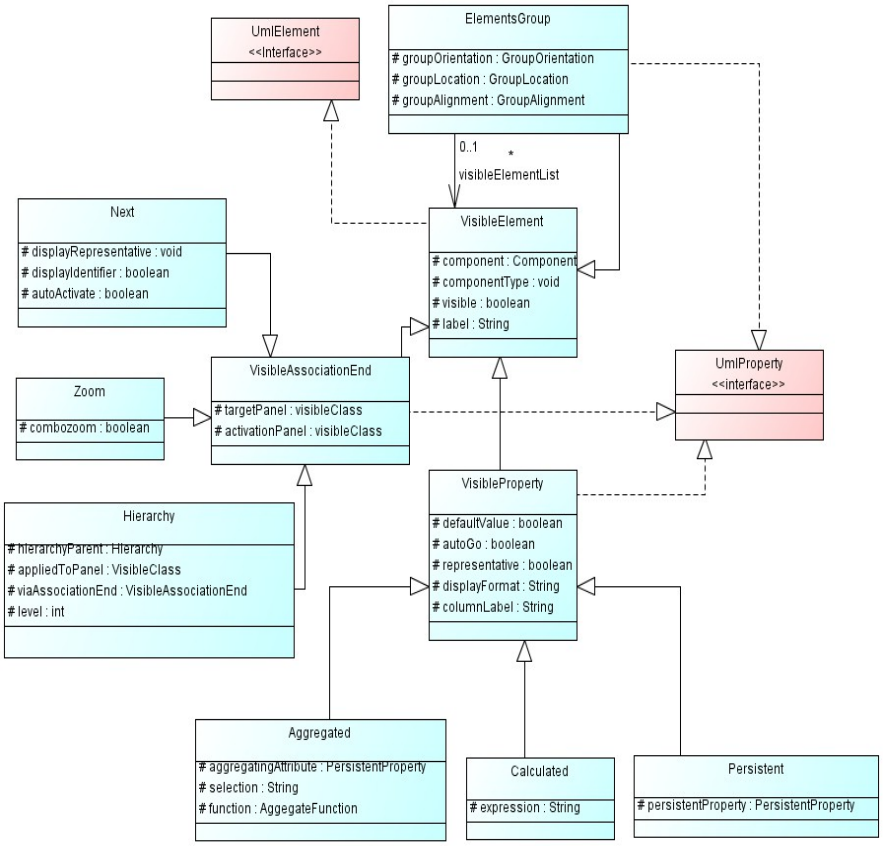
Slika 2.2 Dijagram klasa koje reprezentuju panele u okviru Kroki alata

Vidljiva obeležja proširuju metaklasu *Property*, omogućavajući preslikavanje atributa vidljivih klasa na komponente korisničkog interfejsa, čime oni postaju vidljivi u okviru odgovarajućeg panela, a implementirana su klasom *VisibleProperty.* Vidljiva obeležja mogu biti agregirana, kalkulisana, *lookup* itd. Na slici 2.3 prikazan je dijagram klasa koje implementiraju interfejs *UmlPropery*, kojim je realizovana metaklasa *Property.*

Grupa elemenata predstavlja još jedno proširenje metaklase *Property* i koristi se za grupisanje elemenata vidljivih klasa (obeležja, metoda i veza), formirajući na taj način semantičke celine koje se preslikavaju na grupe komponenti korisničkog interfejsa u okviru panela. Grupa može biti reprezentovana okvirom, panelom, tabulatorskim panelom ili podmenijem (u slučaju grupisanja stavki menija). Stereotip je implemetiran klasom *ElementsGroup.*

Vidljivi krajevi asocijacije ponovo proširuju metaklasu *Property* i mogu se primeniti na vidljivo navigabilno obeležje koje pripada binarnoj asocijaciji uspostavljenoj između dve vidljive klase. Definiše odnos između

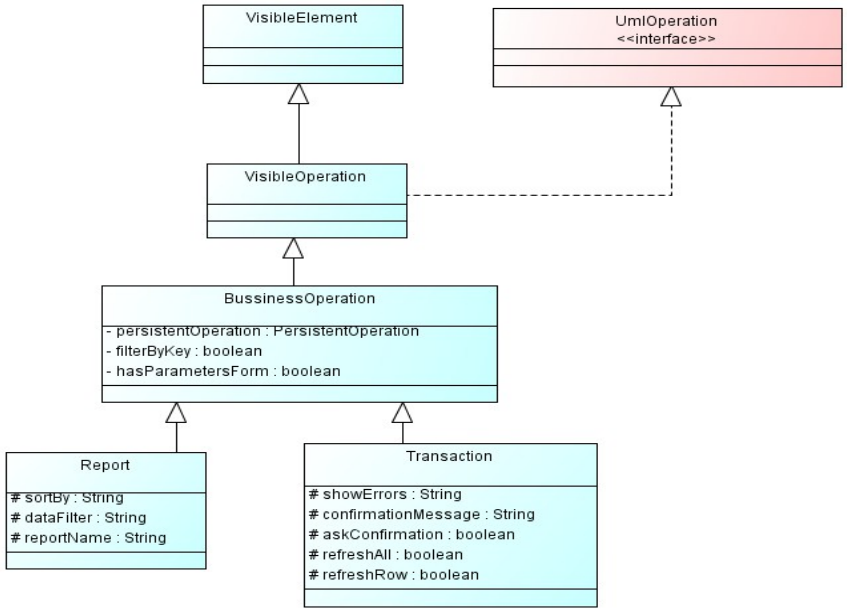
panela pridruženog klasi koja je vlasnik obeležja i panela koji pripada klasi na drugom kraju asocijacije. Pri implementaciji alata, kako bi se uvela ova funkcionalnost, kreirana je klasa *VisibleAssociationEnd*. Konkretna priroda odnosa između panela definisana je vrstom primenjenog stereotipa, te se uočavaju klase *Next*, *Zoom* i *Hierarchy.*



Slika 2.3 Dijagram klasa koje implementiraju interfejs UmlPropery u okviru Kroki alata

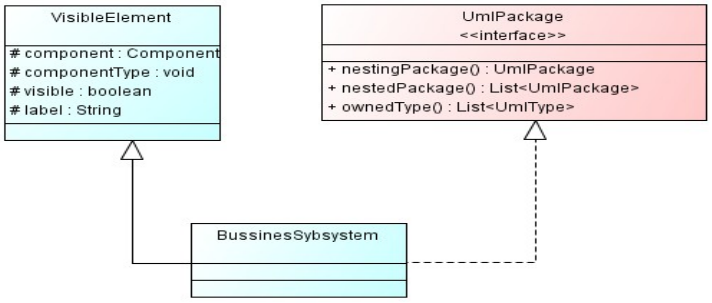
Vidljive metode jesu proširenje metaklase *Operation* i označavaju metode kojima se pridružuje komponenta korisničkog interfejsa, u vidu dugmeta ili stavke menija, i mogu se aktivirati sa forme od strane korisnika. Realizuju se klasom *VisibleOperation,* koju nasleđuje klasa *BussinessOperation* označavajući metodu koja omogućava aktiviranje pridružene perzistentne metode. Perzistentnim metodama pridružena je semantika iz poslovnog domena, poput knjiženja i drugih složenih poslovnih transakcija, označenih klasom *Transaction,* i prikaza izveštaja, za

šta se brine klasa *Report.* Na slici 2.4 prikazan je dijagram klasa naslednica klase *VisibleOperation.*



Slika 2.4 Dijagram klasa naslednica klase VisibleOperation u okviru Kroki alata

Poslovni podsistem proširuje metaklasu *Package* i koristi se za definisanje poslovnih sistema i referata. Implementacija je data u vidu klase *BussinesSubsystem*. Na slici 2.5 prikazan je dijagram klasa opisanog dela implementacije profila.



Slika 2.5 Dijagram klasa poslovnog podsistema u okviru Kroki alata

Poput vidljivih klasa, obeležja i metoda, radi modelovanja perzistentnih podataka uvode se perzistentne klase, obeležja i metode. Odnosno, klase *PersistentClass, PersistentProperty i PersistentOperation.*

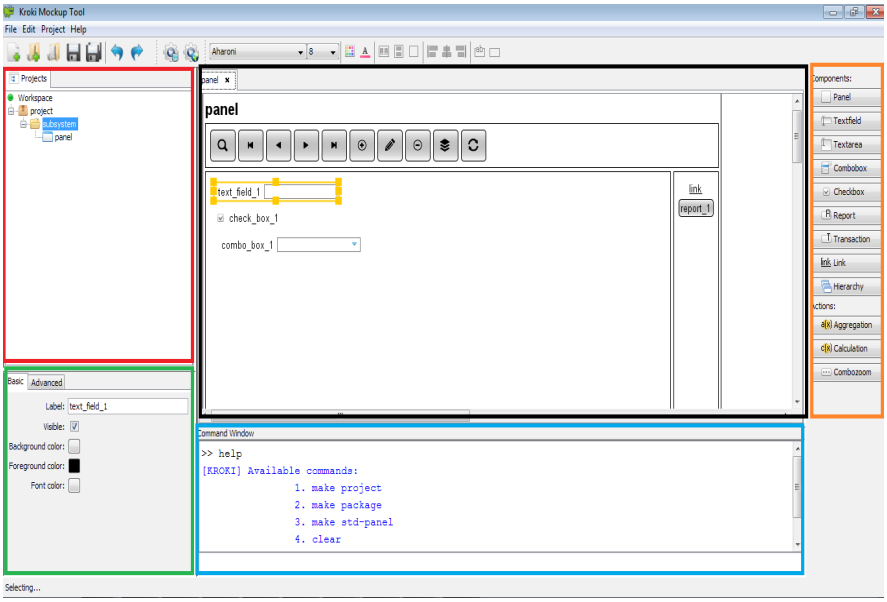
2.3 Prikaz Kroki alata za skiciranje formi

Alat omogućava korisnicima da učestvuju u procesu specificiranja izgleda i funkcionalnosti aplikacije na „prirodan“ način, korišćenjem dizajnera formi, nudeći pri tome projektantima i alternativni, brži, način modelovanja. Alat je tako organizovan da se skiciranjem dobijaju i informacije potrebne za kreiranje nižih slojeva aplikacije, što omogućava izvršivost skice u svakom trenutku.

2.3.1 Glavni prozor Kroki alata i kratak pregled osnovnih funkcionalnosti

Glavni prozor Kroki alata je prikazan na slici 2.6. Sastoji se iz:

- Palete alatki locirane na vrhu prozora
- Stabla, zaokruženog crvenom bojom na slici 2.6
- Prostora za skiciranje ekranskih formi, zaokruženog crnom “bojom”
- Palete komponenti, zaokruže narandžastom bojom
- Panela za podešavanja osobina selektovane forme ili neke komponente koja pripada datoj formi, zaokruženog zelenom bojom na slici 2.6
- Konzole za kreiranje ekranskih formi korišćenjem komandnog jezika, zaokružene plavom bojom

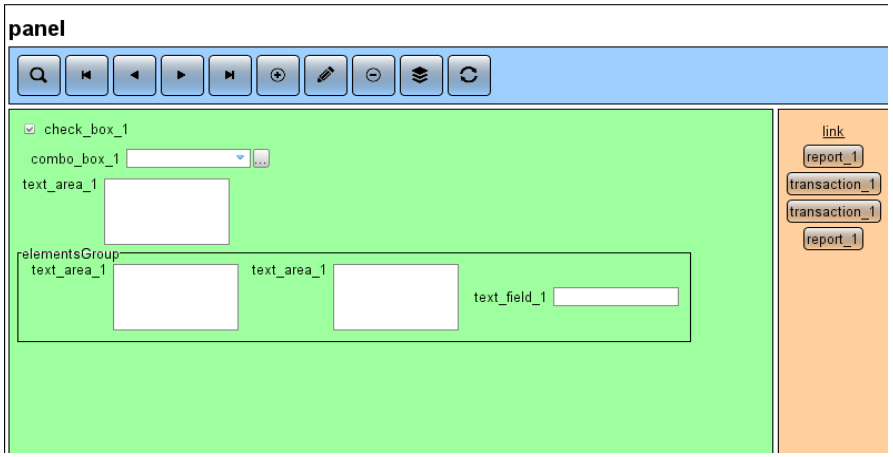


Slika 2.6 Glavni prozor alata za skiciranje formi

Stablo, koje se nalazi u gornjem levom uglu prozora, omogućuje prikaz hijerarhijske strukture poslovnih sistema, ali i njihovo kreiranje, brisanje i preimenovanje odabirom odgovarajuće stavke kontekstnog menija. Pored njega, u centru prozora, nalazi se prostor za skiciranje ekranskih formi. Kao što se vidi na slici 2.6, prostor je podeljen u jezičke (*tab-ove*). Sistem podržava MDI (*Multiple Document Interface*) koncept, koji pruža mogućnost rada sa više dokumenata istovremeno. U konkretnom slučaju, dokument se mapira na jednu formu.

U desnom delu prozora nalazi se paleta komponenti, koje se mogu smeštati na formu, i akcija koje se eventualno mogu primeniti nad njima. Akcije podrazumevaju proglašavanje polja agregiranim, kalkulisanim ili *combozoom*-om i mogu se primeniti samo ako je odabrana komponenta odgovarajućeg tipa. Naime, samo tekstualno polje (*Textfield*) može biti transformisano u agregirano ili kalkulisano, dok se ni nad jednim elementom osim *comboBox*-a ne može primeniti *combozoom* akcija. Agregirana i kalkulisana polja ne mogu se ručno popunjavati, pa osim više mogućnosti dodatnih podešavanja, o kojima će uskoro biti više reči, tekstualnim poljima nad kojima su izvršene pomenute dve transformacije se automatski postavlja indikator ove osobine.

U primeru sa slike 2.6, kreirana je jedna jednostavna standardna foma i može se primetiti da je ispoštovan standard opisan u poglavlju 2.1, podelom forme na tri dela čije se osobine mogu nezavisno podešavati. Imajući to u vidu, komponente *Transaction*, *Report* i *Link* mogu se staviti samo u desni deo forme, rezervisan za specifične operacije, dok se *Textfield*, *Textarea*, *Checkbox*, *Panel* i *comboBox* smeštaju u centralni deo. Na slici 2.7 prikazan je primer standardne forme gde je prethodno napomenuto naglašeno.



Slika 2.7 Primer standardne forme sa naznačene tri celine

Dodate komponente se mogu postavljati redom horizontalno, vertikalno, ili mogu biti slobodno uređene, po želji korisnika sistema. Prve dve opcije omogućuju i dodatno specificiranje poravnanja – levo, desno ili po sredini. Navedeno se u aplikaciji zadaje preko gornje palete alatki.

Osobine panela i komponenti koje se na njima nalaze mogu se podešavati preko panela smeštenog u donjem levom uglu prozora. Osim podešavanje osnovnih obeležja, poput labele i boje pozadine, moguće je i specificiranje naprednijih svojstava, uključujući i već spomenute informacije potrebne za kreiranje nižih slojeva aplikacija. Treba naglasiti da se u okviru dodatnih podešavanja može podesiti formula agregiranog polja, zadavanjem atributa i izborom jedne od ponuđenih operacija, formula kalkulisanog polja, naziv izveštaja i transakcije itd. Kada je reč o transakcijama i izveštajima, izdvaja se i mogućnost definisanja postojanja ili nepostojanje forme za unos parametara.

Alternativni način za kreiranje hijerarhijske strukture (projekata, paketa i standardnih panela) jeste upotreba komandne konzole koja se nalazi ispod prostora za crtanje ekranskih formi. EUS DSL poseduje i tekstualnu sintaksu, koja upravo omogućava rad u okviru konzole. Osnovni cilj jeste da se ekspertima omogući brži rad, pogotovo u situacijama kada su u direktnoj interakciji sa korisnikom. Prilikom kreiranja ekranskih formi na ovaj način odmah se zadaje kojem paketu pripadaju, njihov naziv i, eventualno, lista komponenti koje sadrže u vidu parova tip – labela. Prepoznavanje tipa komponente nije sintaksno osetljivo. Dakle, umesto kreiranja novog panela preko stabla, ručnog dodavanja komponenti i podešavanja labela, sve je moguće uraditi samo jednom komandom u okviru konzole.

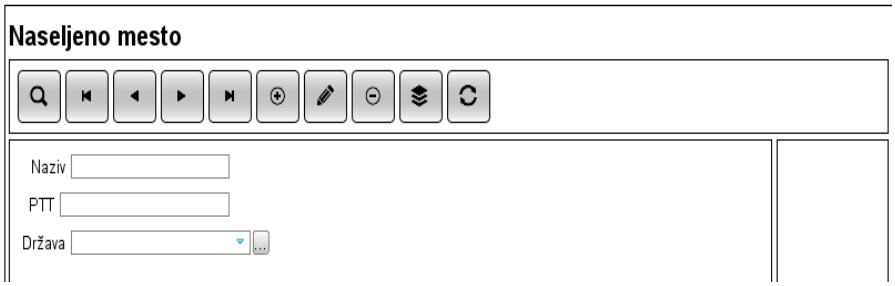
Aplikacija koja se skicira tokom rada se može pokrenuti radi pregleda i isprobavanja, i to bez ikakvih dodatnih podešavanja, nad test bazom, ili uz prethodno podešavanje parametara za konekciju na MySQL, PostgreSQL, SQL Server ili H2 bazu. Pokretanje se postiže klikom na jedno od dva zaokružena dugmeta sa slike 2.8.

Slika 2.8 Paleta alatki sa zaokruženim ikonama za pokretanje aplikacije

Na slici 2.8 je plavom bojom zaokruženo dugme koje se aktivira radi pokretanja skice kao *desktop* aplikacije, dok je susedno, zaokruženo zelenom bojom, zaduženo za startovanje *web* aplikacije.

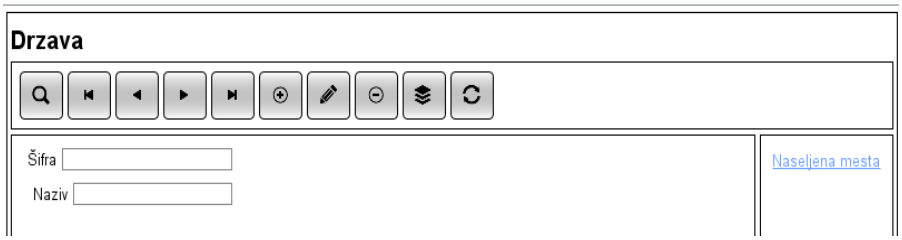
2.3.2 *Zoom* i *Next* mehanizmi

Kao što je već rečeno, i primetno na slici 2.7, na desni deo panela može se dodati posebna komponenta dostupna preko palete komponenti, koja predstavlja vezu ka drugoj formi. Sa druge strane, proizvoljni *comboBox* dodat na panel rezervisan za podatke može se transformisati u *combozoom*. Osnovnoj komponenti se u tom slučaju dodaje dugme, čija je uloga aktiviranje povezane forme, što se može uočiti na primeru sa slike 2.9, gde je prikazana standardna forma "Naseljeno mesto", sa komponentom koja omogućuje traženje željene države. Nakon transformacije *comboBox*-a u *combozoom*, dodatna podešavanja date komponente su proširene mogućnosti izbora ciljnog (*target*) panela.



Slika 2.9 Standardnia forma sa *combozoom-om*

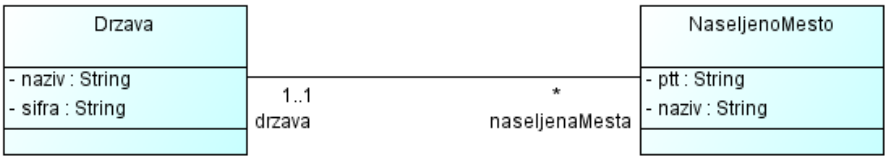
Nakon što je na formi "Naseljeno mesto" izabran ciljni panel *combozoom*-a, dobija se mogućnost odabira upravo panela "Naseljeno mesto" kao odredišnog pri aktivaciji *link* komponente dodate na formu "Država", koja je prikazana na slici 2.10.



Slika 2.10 Standardna forma sa *link*-om

Pri samom dodavanju link komponente, odnosno, transformisanju *comboBox*-a u *combozoom*, automatski se postavlja obeležje koje specificira aktivacioni panel i ne može se naknadno menjati, što ne bi ni imalo smisla.

Na slici 2.11 prikazan je dijagram klasa koji nastaje kao posledica skiciranja.

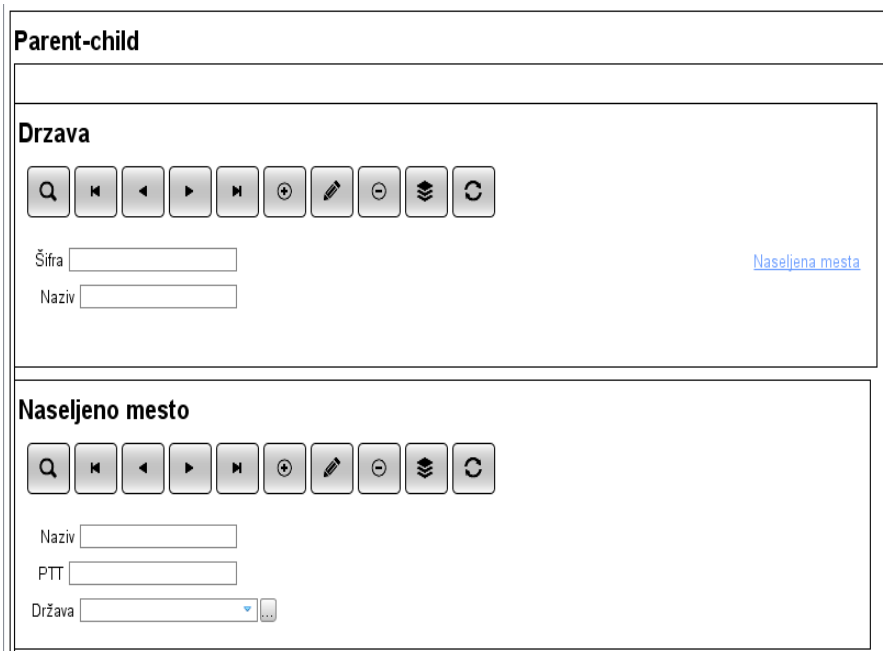


Slika 2.11 Dijagram klasa dobijen na osnovu skica

Dakle, klase imaju atribute formirane na osnovu polja formi, dok se između njih uspostavlja *one-to-many* veza. Jedno naseljeno mesto može se nalaziti u samo jednoj državi, koje se bira *Zoom* mehanizmom. Sa druge strane, u jednoj državi se može nalaziti više naseljenih mesta, te se iz forme "Država" može aktivirati "Naseljeno mesto" gde će se izvršiti filtriranje po pripadajućoj državi

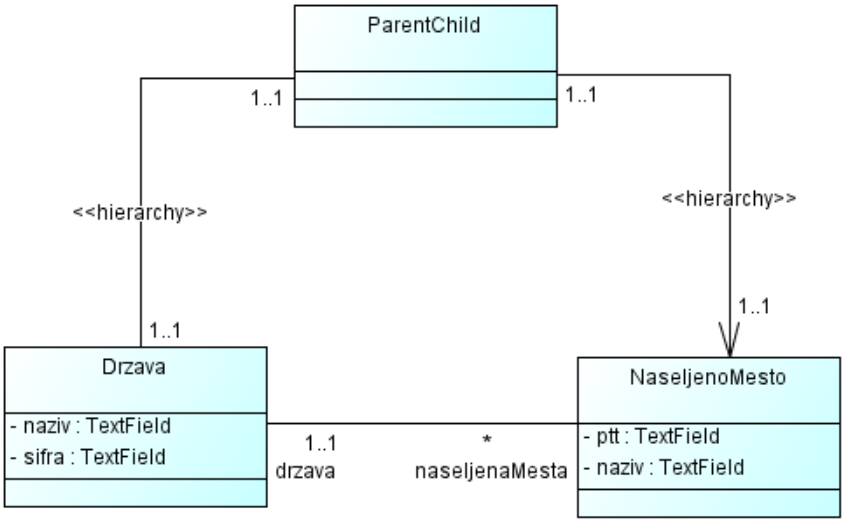
2.3.3 *Parent-child* panel

Pored standardnih panela, prikazanih na prethodnim slikama, Kroki alat omogućuje i kreiranje *parent-child* panela, uvezivanjem prethodno kreiranih standardnih u hijerarhijsku strukturu. Primera radi, *parent-child* panel sa prethodno kreiranim panelima "Država" i "Naseljeno mesto" prikazan je na slici 2.12



Slika 2.12 Primer *parent-child* panela

U konrektnom primeru, radi se o dvonivovskoj hijerarhiji, gde se na prvom nivou nalazi panel “Država”, a “Naseljeno mesto” na drugom. Kako se prvopomenuti panel nalazi na vrhu hijerarhije, on nema roditelja, dok drugi panel, “Naseljeno mesto” ima. U pitanju je, naravno, panel "Država", što se automatski postavlja pri dodavanju panela na nivo u hijerarhiji nakon korenskog. Dijagram klasa korisničkog interfejsa nastao na osnovu ove skice prikazan je na slici 2.13. Vidi se da se *parent-child* panel ima dve *one-to-one* veze, prema oba sadržavajuća panela.



Slika 2.13 Dijagram klasa korisničkog interfejsa dobijen na osnovu skice koji uključuje i *parent-child* panel

3. GRAFIČKI EDITOR DIJAGRAMA KLASA I INTEGRACIJA SA KROKI ALATOM

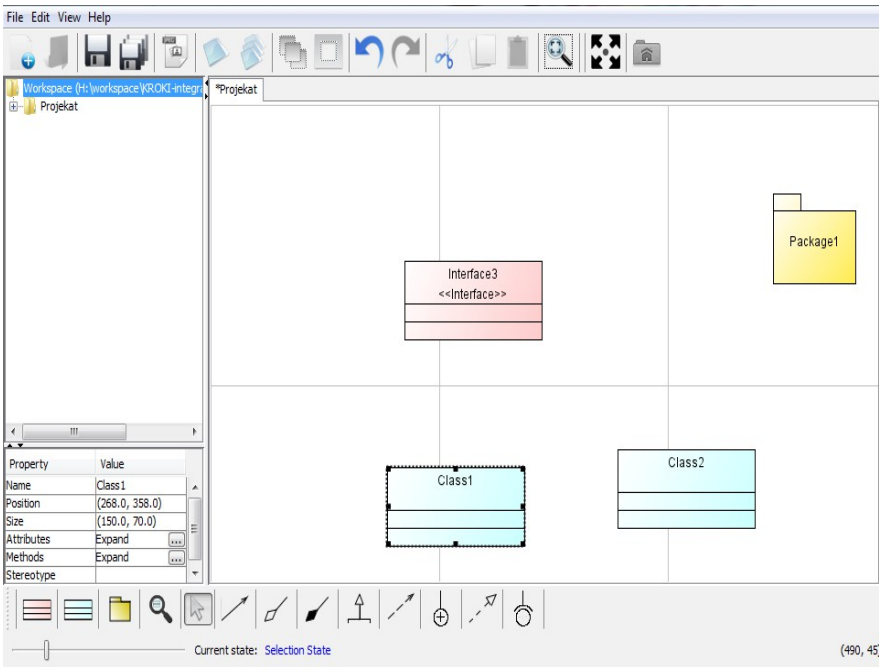
U predstojećem poglavlju će biti detaljnije opisan grafički editor dijagrama klasa, njegove mogućnosti i implementacija, kao i integracija sa Kroki alatom radi skiciranja i pregleda dijagrama klasa već kreiranih skica, uz postavljanje podloge za kasniju realizaciju modelovanja perzistentnog sloja.

3.1 Pregled editora dijagrama klasa

Pre nego što se više pažnje posveti korišćenju editora u okviru Kroki alata, biće dat pregled samog editora i njegove implementacije.

3.1.1 Glavni prozor editora

Glavni prozor implementiranog rešenja [1], prikazan na slici 3.1, sastoji se iz glavnog menija, gornje palete alatki, stabla, prostora za podešavanja osobina elemenata, prostora za prikaz dijagrama (*canvas*), donje palete alatki, *zoom* klizača i statusne linije.



Slika 3.1 Glavni prozor grafičkog editora

Najveći deo glavnog prozora zauzima prostor za prikaz dijagrama (*canvas*). Svakom paketu i projektu pridružen je jedan dijagram, što dalje znači da im se dodeljuje po jedan *canvas* u okviru posebnog jezička (*tab*-a), čiji se naslov poklapa sa nazivom paketa, odnosno, projekta.

Glavni meni, koji se nalazi u gornjem delu prozora sadrži sve akcije dostupne i preko gornje palete alatki radi lakšeg korišćenja, uz dodatak pomoći kojoj se pristupa preko *Help* menija. Na suprotnom kraju prozora, pri samom dnu, nalazi se statusna linija koja pokazuje u kom se stanju editor trenutno nalazi, te *zoom* klizač i koordinate kursora, što će kasnije biti detaljnije objašnjeno.

3.1.1.1 Gornja paleta alatki

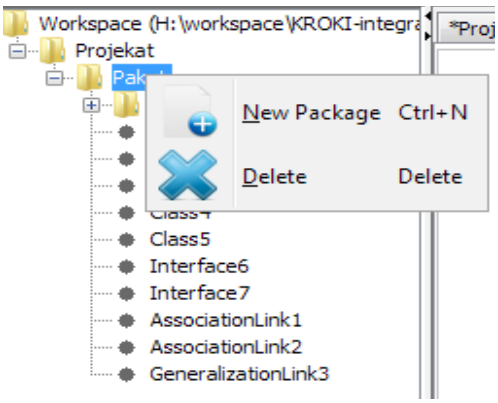
Paleta alatki, locirana odmah ispod glavnog menija, prikazana je na slici 3.2.

Slika 3.2 Gornja paleta alatki

Brojevima od 1 do 17 označene su sve akcije dostupne preko prikazane palete. Brojem 1 numerisano je kreiranje novog projekta, brojem 2 otvaranje dijagrama pridruženog paketu ili projektu selektovanog na stablu, čime se završava prva grupa akcija separatorom odvojena od naredne, koju čine akcije za snimanje tekućeg, odnosno, svih otvorenih projekata. Broj 5 označava akciju za eksportovanje dijagrama, odnosno, željenog dela u neki od ponuđenih formata, nakon čega slede akcije za zatvaranje dijagrama čiji je sadržaj trenutno prikazan i za zatvaranje svih otvorenih. Naredna grupa odnosi se na selekciju elemenata. Naime, broj 8 pridružen je selekciji svih elemenata tekućeg dijagrama, dok je sledeći broj dodeljen inverznoj selekciji. Lako se može zaključiti da 10 i 11 odgovaraju poništavanju, odnosno, ponovnom izvršenju akcije (*undo i redo)* redom. Lista akcija nastavlja se sečenjem (*cut*), te kopiranjem i lepljenjem (*copy i paste*), da bi se došlo do optimalnog *zoom*-a (*best fit zoom*), numerisanog brojem 15, prikaza preko celog ekrana i promene radnog okruženja (*workspace*-a). Posebno interesante akcije, eksportovanje, zatvaranje dijagrama i različite selekcije zaokružene su zelenom, plavom i crvenom bojom respektivno.Primećuje se da su neke akcije onemogućene, što je posledica zabrane njihovog izvršenja u situacijama kada to nema smisla, poput lepljenja pre kopiranja ili poništavanja akcije pre sprovođenja prve i sl..

3.1.1.2 Stablo

Hijerarhijska struktura projekata i paketa predstavljena je stablom koje se nalazi u gornjem, levom delu glavnog prozora i uočava se na slici 3.3. U okviru stabla je moguće kreiranje novih projekata (ukoliko je selektovan sam vrh hijerarhije), paketa (ukoliko je selektovan projekat ili već postojeći paket), kao i njihovo uklanjanje. Sve klase, interfejsi i uspostavljene veze se takođe prikazuju u okviru stabla na specificiranim pozicijama. Poput pomenutog brisanja projekata i paketa i elementi dijagrama se mogu izbrisati direktno iz stabla. Pritiskom desnog tastera miša na određeni čvor stabla se prikazuje konteksni meni sa dozvoljenim akcijama, koje zavise od tipa samog čvora. Dok se elemeti i veze mogu samo brisati, desnim klikom na projekat ili paket otvara se meni koji dozvoljava i kreiranje novog paketa unutar njih.

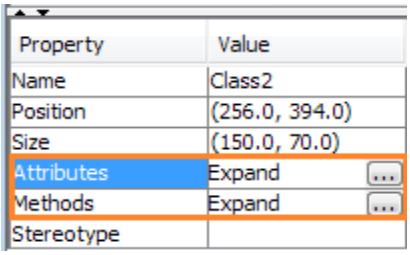


Slika 3.3 Stablo i konteksni meni

Na slici 3.3 vidi se da je uvedeno uređenje u prikaz elemenata i veza unutar paketa. Prvo se navode svi paketi, potom klase, interfejsi, da bi na kraju bile izlistane sve veze.

3.1.1.3 Panel za podešavanje osobina selektovanih elemenata i njihovih veza

Neposredno ispod stabla, kao što se vidi na slici 3.1, nalazi se deo predviđen za podešavanje osobina elemenata i veza, prikazan na slici 3.4. Organizovan je kao tabela sa dve kolone – naziv obeležja i njegova vrednost. Broj redova, tj. konfigurabilnih obeležja zavisi od toga šta je selektovano. Logično, ne može se ustanoviti isti skup obeležja koji bi karakterisao kako klase i interfejse, tako i pakete i veze.



Slika 3.4 Panel za podešavanje osobina

Kako je specificiranje atrbuta i metoda složenije i ne postoji gornje ograničenje njihovog broja, odgovarajući redovi sadrže samo dugme koje otvara odgovarajuće dijaloge i uokvireni su narandžastom bojom na slici 3.4.

3.1.1.4 Paleta alatki za dodavanje elemenata, povezivanje i uveličavanje

Donja paleta alatki, pozicionirana ispod radne površine prikazana je na slici 3.5, dok su značenja pojedinačnih ikona pojašnjena u tabeli 3.1.

Slika 3.5 Paleta alatki grafičkog editora

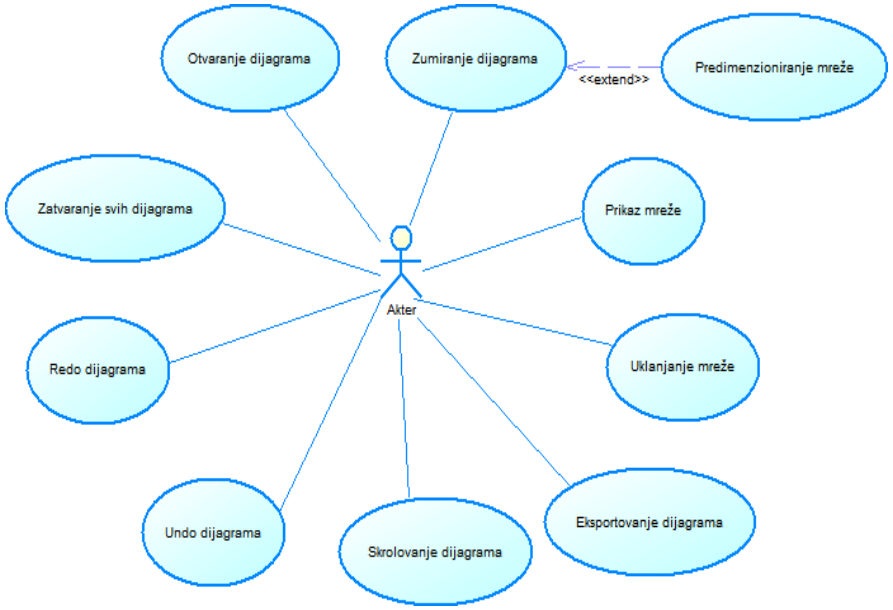
Ikona	Opis akcije
	Interfejs
	Klasa
	Paket
	Uveličanje selektovanog dela dijagrama (*Lasso Zoom*)
	Selekcija
	Veza asocijacije
	Veza agregacije
	Veza kompozicije
	Veza generalizacije

	Veza zavisnosti
	Unutrašnja klasa (*Inner link*)
	Veza realizacije
	*Require link*

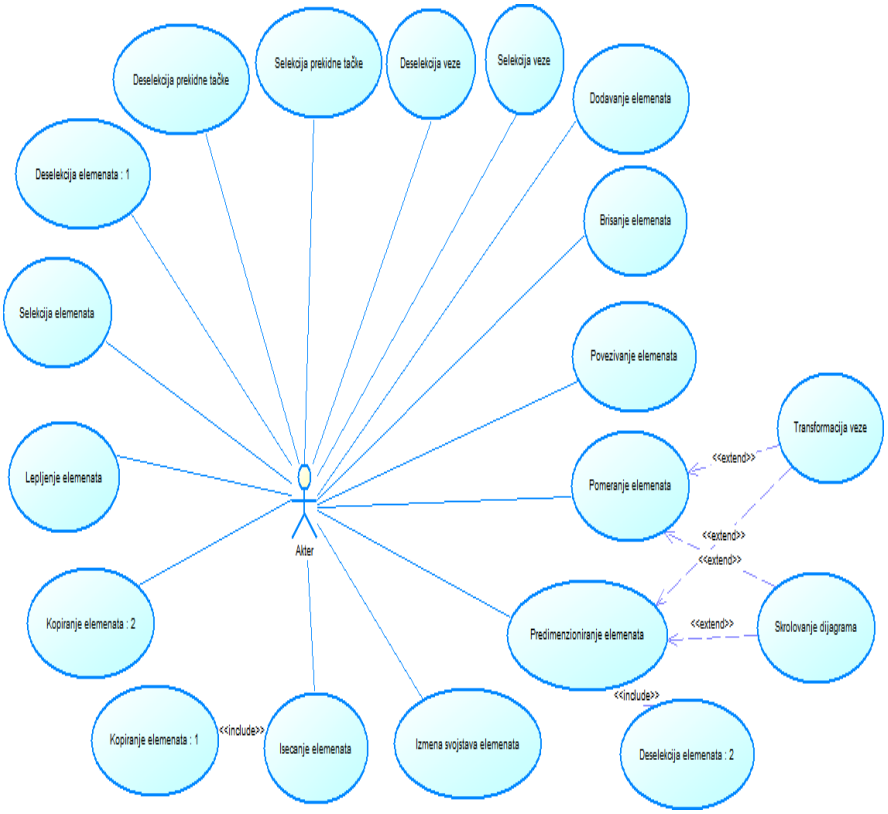
Tabela 3.1 Značenje ikona na paleti alatki

Realizovano rešenje omogućuje rukovanje elementima dijagrama klasa (paketima, klasama i interfejsima i njihovim atributima i metodama), uspostavljanje relacija između njih, kao i podešavanje određenih svojstava dijagrama, poput faktora uveličanja, prikaz i sklanjanje mreže i sl.

Na slici 3.6 prikazan je dijagram slučajeva korišćenja koji opisuje rad sa dijagrama, dok dijagram slučajeva korišćenja na slici 3.7 prikazuje rad sa njihovim elementima. Uvedene funkcije će u nastavku poglavlja biti detaljnije objašnjene.



Slika 3.6 Dijagram slučajeva korišćenja rada sa dijagramima

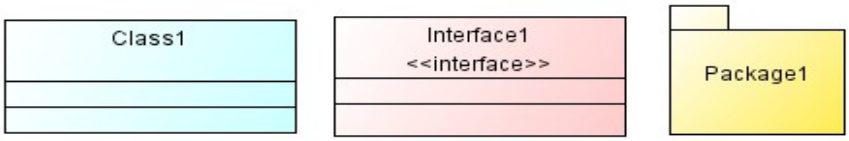


Slika 3.7 Dijagram slučajeva korišćenja rada sa elementima dijagramima

3.1.2.1 Rad sa elementima dijagrama

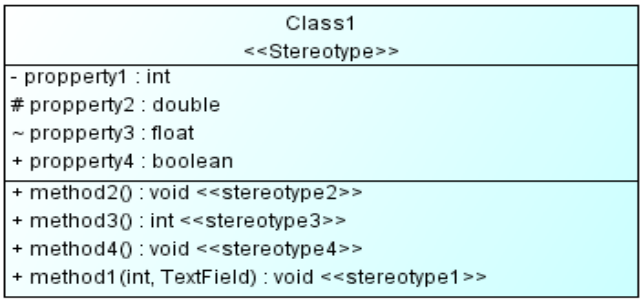
Rad sa elemetima dijagrama podrazumeva njihovo dodavanje, izborom željenog elementa na paleti alatki prikazanoj na slici 3.5, izmenu njihovih svojstava, poput naziva, veličine i pozicije, povezivanje uz ograničenja nametnuta UML standardom, kopiranje, lepljenje, isecanje i brisanje selektovanih elementa, pri čemu je predviđeno više načina za postizanje željenog cilja, kao i poništavanje sprovedenih akcija.

Klase, interfejsi i paketi se dodaju izborom odgovarajućeg elementa na paleti alatki, nakon čega će kursor poprimiti izgled datog elementa, praćenog klikom na željeno mesto na dijagramu. Po dodavanju elementi imaju automatski dodeljena imena i stereotipove, ali je predviđena mogućnost njihove naknadne izmene, kao i dodavanje i uklanjanje metoda i atributa, za elemente za koje to ima smisla. Na slici 3.8 su prikazana ova tri tipa elemenata.



Slika 3.8 Grafički prikaz klase, interfejsa i paketa redom

Selekcijom jednog elementa u donjem levom uglu pojavljuje se panel za podešavanje njegovih svojstava. Kao što je već napomenuto, njegov sadržaj zavisi od toga šta je selektovano. Dok klase i interfejse karakterišu njihovi atributi i metode, stereotip, naziv, pozicija i veličina, za pakete su prva dva obeležja nerelevantna. Još je veća razlika kada su u pitanju veze, kod kojih su bitni kardinaliteti, uloge i navigabilnost oba kraja. Zadavanjem nove vrednosti obeležja u tabeli se automatski menjaju i sami elementi i veze. Na slici 3.9 je prikazana klasa sa nekoliko dodatih atributa, različitih vidljivosti, i nekoliko metoda, sa i bez parametara.



Slika 3.9 Klasa sa atributima i metodama

Grafički simbol podeljen je na tri segmenta, gde se na samom vrhu nalazi naziv klase i stereotip, dok slede liste atributa i metoda. Vidljivost, *private, protected, default i public,* označeni su redom sa - , #, ~ i +. Simbol interfejsa se, izuzev boje i automatski dodeljenih početnih naziva ne razlikuje od prikazanog.

Elementi se, pored klasičnih načina selekcije (selektovanje svih, dodavanje i izuzimanje iz selekcije, laso selekcija), mogu selektovati i putem inverzne selekcije. Dakle, omogućeno je selektovanje svih elemenata koji u datom trenutku nisu bili selektovani, što je izuzetno zgodno ukoliko se žele selektovati svi osim malog broja elemenata.

Selektovani elementi se dalje mogu pomerati. Omogućeno je pomeranje više elemenata istovremeno, pri čemu veze između njih zadržavaju početni oblik ukoliko su selektovani svi povezani elementi.

Ukoliko, pak, neki od elemenata na kraju veze nije selektovan, dok drugi jeste, i veza će biti promenjena u skladu sa načinjenim izmenama.

Selektovanjem samo jednog elementa dobija se i mogućnost menjanja njegove veličine. Elementi se mogu smanjiti samo do određene granice, koja je uslovljena njihovim obeležjima. Naime, nije moguće toliko umanjiti klasu, interfejs ili paket, da im ne bude vidljiv ceo naziv ili stereotip, kao i svi atributi i metode u slučaju poslednja dva navedena elementa.

Elementi se povezuju odabirom jedne od dostupnih veza na paleti alatki, i klikom na dva elementa koji se žele povezati. Pritom je moguće i uspostaviti više prekidnih tačaka, ukoliko se klikne na površinu dijagrama na kojoj se ne nalazi nijedan element, kao i njihovo naknadno pomeranje. Međutim, postoje ograničenja pri uspostavljanju veza, nametnuta poštovanjem UML standarda, te se ne mogu proizvoljno povezati dva nasumična elementa. Ukoliko nije dozvoljeno da dotični element bude polazni za odabranu vezu pri datim uslovima, povezivanje neće biti ni započeto, dok će u slučaju nevalidnog odredišnog elementa povezivanje biti poništeno. Recimo, veza realizacije ne sme krenuti iz interfejsa, ali se mora na njemu završiti, klasa A ne može da nasledi svog potomka niti interfejs i slično. Kod veza asocijacije moguća je i promena njihovog tipa (kompozicija, agregacija i obična asocijacija). Preko *Preferences* dijaloga dostupnog u okviru *View* menija može se podesiti i iscrtavanje veza sa pravim uglovima, kada će prekidne tačke biti automatski preraspoređene tako da uglovi između susednih tačaka budu pravi ili opruženi.

Elementi se mogu kopirati, isecati i lepiti, pri čemu je dozvoljeno i lepljenje na drugi dijagram. Ukoliko se među selektovanim elementima nalazi i paket, biće prekopiran i njegov unutrašnji dijagram. U svakom slučaju, veze između elemenata bivaju očuvane nakon lepljenja.

3.1.2.2 Rad sa dijagramima

Rad sa dijagramima podrazumeva njihovo otvaranje i zatvaranje, transformacije prikaza u vidu zumiranja, skrolovanja, prikaza i sklanjanja mreže (*grid-a*), eksportovanje dijagrama u neki od raspoloživih formata (.png i .jpg) kao i poništavanje i ponovno sprovođenje akcije (undo/redo).

Na raspolaganju je više varijanti zumiranja. Prva opcija je zumiranje držanjem tastera *ctrl* i pomeranjem točkića miša, kada se zumiranje vrši u odnosu na tekuću poziciju kursora na dijagramu. Druga raspoloživa opcija jeste korišćenje klizača smeštenog u donjem levom uglu. Povlačenje udesno dovodi do uveličavanja, dok kretanje ka levoj strani ima suprotan efekat. Zumiranje se u ovom slučaju vrši u odnosu na centar vidljivog dela dijagrama. Treći način jeste *best fit zoom*, kad se faktor uveličavanja,

odnosno, smanjivanja, automatski računa tako da svi elementi dijagrama budu vidljivi bez potrebe za skrolovanjem. Poslednji način za postizanje ovog cilja jeste laso zumiranje, kada će se prikazati samo naznačeni deo dijagrama, a faktor automatski izračunati.

Veličina dijagrama nije oganičena, te je često neophodno pomeranje vidnog polja, što se postiže okretanjem točkića miša, sa ili bez držanja tastera *shift*, za horizontalno, odnosno, vertikalno kretanje. Ukoliko se, pri pomeranju, predimenzioniranju ili povezivanju elemenata počne napuštati vidno polje, ono će automatski biti pomereno.

3.1.3 Implementacija rešenja

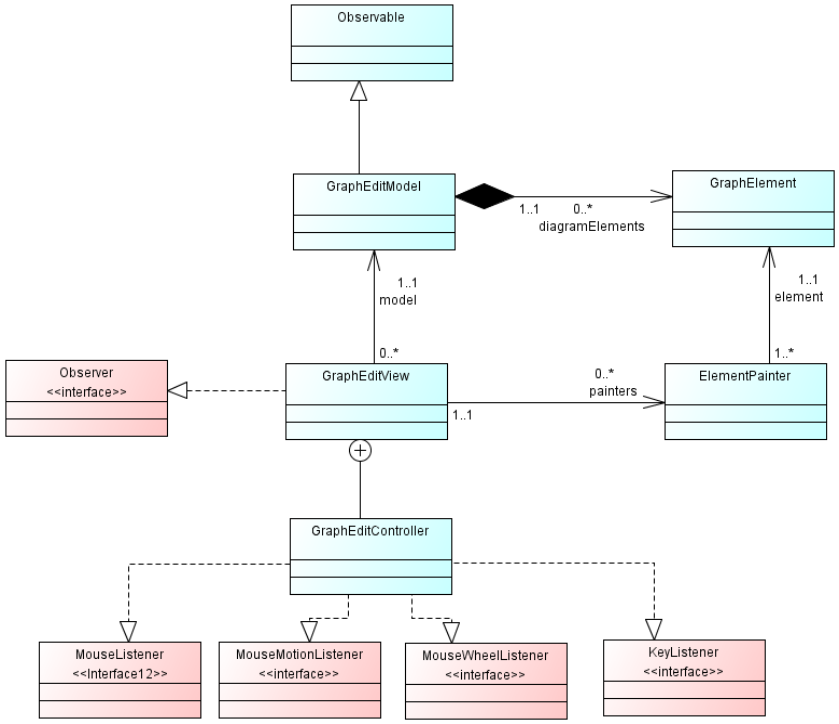
U nastavku će ukratko biti prikazana implementacija rešenja, uz napomenu da će detalji vezani za integraciju sa Kroki alatom biti razrađeni u kasnijim poglavljima.

3.1.3.1 Arhitektura rešenja

Rešenje je implementirano uz oslonac na MVC (*Model-view-controller*) šablon [2], koji razdvaja prezantaciju informacija od korisničke interakcije sa njima. Sami podaci, poslovna logika i pravila se smeštaju u model, prezentacioni sloj zadužen je za predstavu podataka korisniku, dok kontroler preuzima korisnikov unos i pretvara ih u komande koje se dalje prosleđuju jednom od druga dva sloja. Šablon, međutim, nije ispoštovan u izvornom obliku, nego se primenjuje njegova varijacija, *Model-delegate* [3], gde su *view* i *controller* spojeni čineći takozvani *User Interface Delegate*. Dakle, mogu se izdvojiti klase modela, koje sadrže podatke specifične za aplikaciju, i *view/controller* klase, zadužene za prikaz informacija i interakciju sa korisnikom.

Na slici 3.10 prikazan je deo modela koji pokriva prethodno izloženo i opisuje čuvanje elemenata, njihovo iscrtavanje i interakciju sa korisnikom. Klasa *GraphEditModel* odgovara dijagramu, sadržavajući listu njemu pripadajućih elemenata. *GraphElement* je apstraktna klasa koja predstavlja zajednički predak svih grafičkih elemenata, koji se prikazuju na dijagramu zahvaljujući odgovarajućoj instanci jednog od naslednika *ElementPainter* klase, čiji je osnovni zadatak iscrtavanje elemenata na radnoj površini. O ovim dvema klasama i njihovim potomcima će u nastavku biti više reči.

Kao što je *GraphEditModel* klasa zadužena za manipulaciju listom grafičkih elemenata, tako je *GraphEditView* implementirana sa ciljem upravljanja listom klasa za njihovo iscrtavanje. Takođe, klasa *GraphEditView* zadužena je i za iscrtavanje mreže, lasa pri laso selekciji ili zumu, vodeći računa o trenutnom faktoru uveličanja i sl.



Slika 3.10 Implementacija MVC šablona

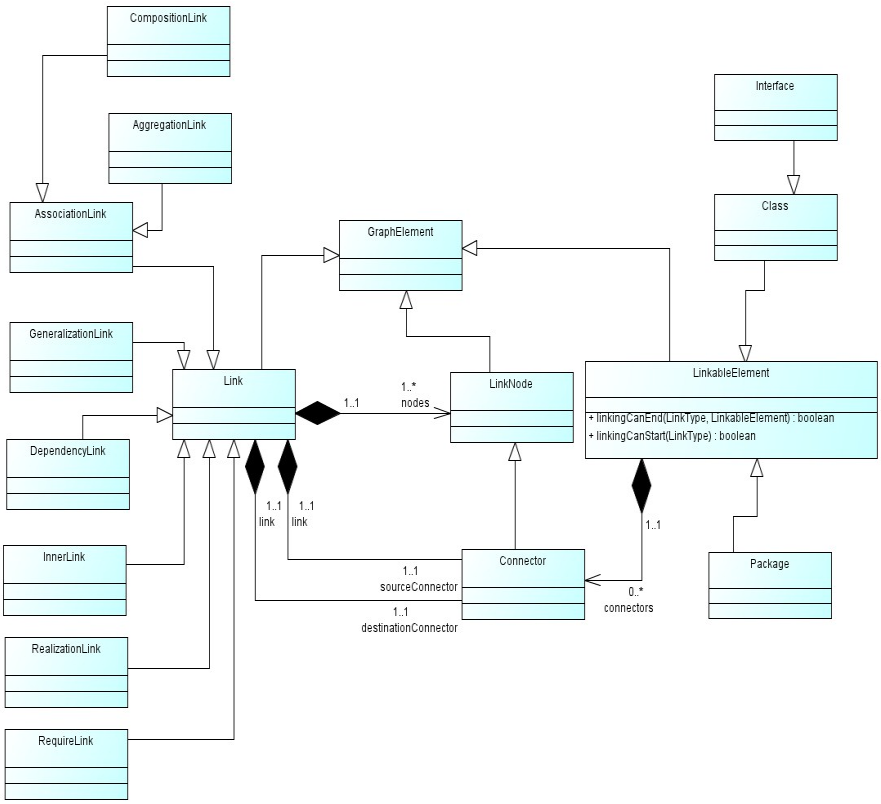
Klasa *GraphEditController* je unutrašnja klasa u okviru više puta spominjane *GraphEditView*, pa se iz tog razloga i govori o odstupanju od orignalnog MVC šablona. Klasa je zadužena za obradu korisnčkog unosa implementirajući interfejse koji omogućuju reakciju na događaje poput pritiska tastera miša ili tastature, pomeranje miša i njegovog točkića. Po registrovanju događaja, odgovarajuće komande se dalje prosleđuju modelu, ako je u pitanju zahtev za promenom elementa (brisanje, dodavanje, promena njihovih svojstava i slično) ili se obrađuju ako se, pak, radi o promeni koja se tiče samo prezentacionog sloja (skrolovanje, uveličavanje itd.).

Na slici 3.10 može se zapaziti i da *GraphEditMode*l nasleđuje javinu klasu *ObservablI*, dok *GraphEditView* implementira interfejs *Observer*, što odgovara implementaciji *Observer* dizajn šablona [4]. Osnovna ideja ovog šablona ponašanja je da svi objekti koji se registruju kao "posmatrači" budu obavešteni kada objekat koji sadrži podatke ili poslovnu logiku promeni stanje. U konkretnom primeru, *GraphEditView* objekti bivaju obavešteni o promeni stanja odgovarajućeg *GraphEditModel* objekta, te se tako

omogućuje da prikaz bude osvežen po dodavanju ili brisanju elementa, uspostavljaju i uklanjanju veza, kao i promeni svojstava nekog elementa.

3.1.3.2 Model grafičkih elemenata

Model grafičkih elemenata prikazan je na slici 3.11.



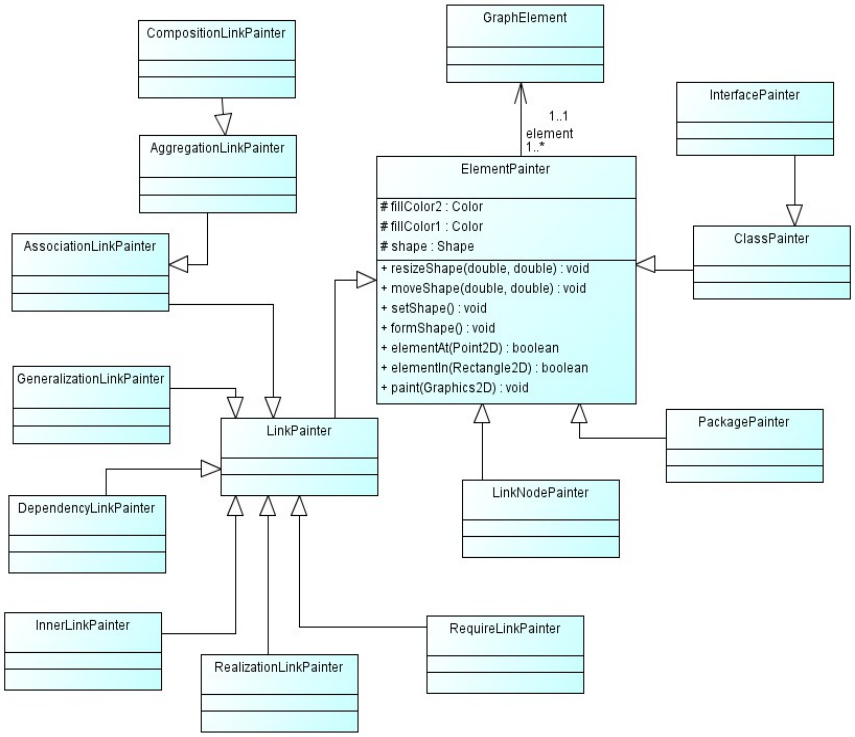
Slika 3.11 Model grafičkih elemenata

Na vrhu hijerarhije nalazi se apstraktna klasa *GraphElement* koja apstrahuje sve zajedničke osobine grafičkih elemenata, koji se iscrtavaju na dijagramu. Klasa *Link* je predak svih veza i enkapsulira njihova opšta svojstva, poput posedovanja liste prekidnih tačaka. Klasa *LinkNode*, koja predstavlja prekidne tačke, takođe nasleđuje *GraphElement* s obzirom na činjenicu da se i one iscrtavaju na dijagramu, iako samo kada je selektovana veza kojoj pripadaju. Klasa *Connector* reprezentuje posebnu vrstu prekidnih tačaka, nasleđujući klasu *LinkNode*. Radi se o prekidnim tačkama koje pripadaju i povezanim elementima i označavaju početak i kraj veze. Veze, dakle, osim liste prekidnih tačaka sadrže i polazni i odredišni konektor.

Klasa *LinkableElement* ima dodatno svojstvo u odnosu na roditeljsku klasu u vidu mogućnosti povezivanja sa drugim elementima. Dve bitne metode za implementaciju provere ograničenja pri povezivanju su *linkingCanStart* i *linkingCanEnd,* koje vraćaju indikator mogućnosti započinjanja povezivanja ako je dati tip elementa odabran kao početni, odnosno, završavanja poevzivanja ako je element u ulozi odredišnog. Klase *Class*, *Interface* i *Package*, direktno ili indirektno nasleđuju *LinkableElement*, redefišući pomenute metode. Dakle, samo se navedena tri tipa elemenata mogu povezivati, ali, kao što je već naznačeno, ne na proizvoljan način. Klasa *LinkableElement* poseduje i listu konektora. Imajući u vidu bidirekcionu prirodu veze između klasa *Connetor* i *Link*, tačno se može odrediti koje sve veze počinju ili se završavaju u datom elementu, te se tako i one mogu izbrisati pri brisanju elementa ili transformisati pri njegovom pomeranju ili predimenzioniranju.

3.1.3.3 Model klasa zaduženih za iscrtavanje

Model klasa zaduženih za iscrtavanje grafičkih elemenata prikazan je na slici 3.12.



Slika 3.12 Model klasa zaduženih za iscrtavanje

Na vrhu hijerarhije se nalazi apstraktna klasa *ElementPainter*, koja je povezana sa grafičkim elementom i poseduje metode za njegovo iscrtavanje, kao i one koje se koriste za implementaciju selekcije, pomeranje i predimenzioniranje elemenata i metode za pomeranje i predimenzioniranje elemenata upotrebom afinih transformacija.

Konektori se po svom izgledu ne razlikuju od običnih prekidnih tačaka, pa nema potrebe za uvođenjem posebne klase zadužene za njihovo iscrtavanje, dok za svaki od preostalih konkretnih grafičkih elemenata postoji i po jedna klasa koja omogućava njihov prikaz na dijagramu. Dakle, za veze, klase, interfejse, prekidne tačke i pakete.

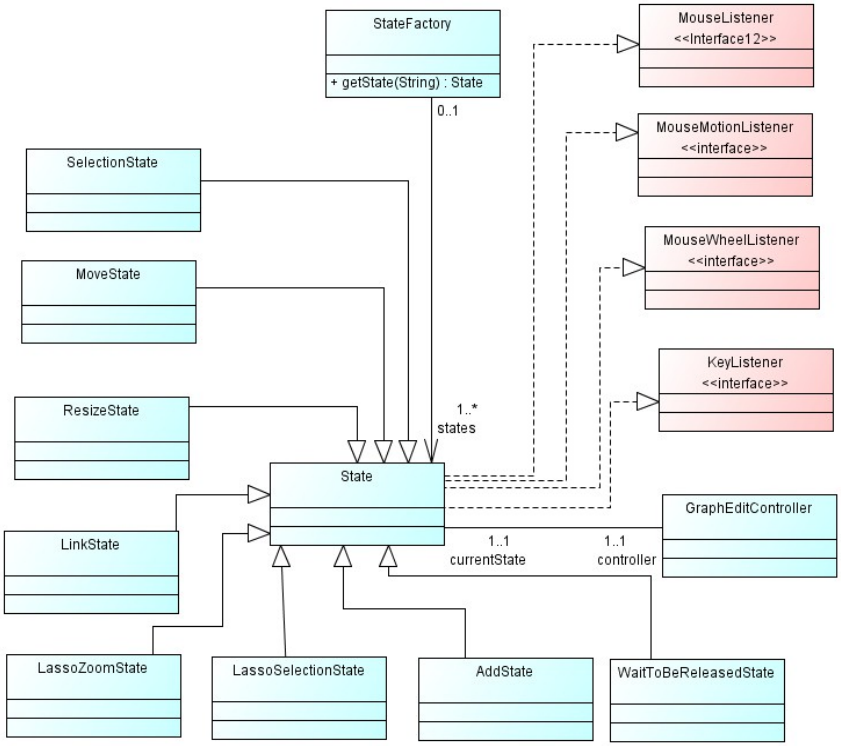
3.1.3.4 Implementacija stanja sistema

Sistem se mora prilagođavati korisničkim akcijama, što podrazumeva promenu stanja u toku izvršavanja. Može se nalaziti u ukupno 8 stanja – dodavanja elemenata, laso selekcije, laso zumiranja, povezivanja, pomeranja elemenata, promene dimenzije elemenata, čekanja da korisnik otpusti tastere miša, koje sprečava neprirodno ponašanje u nekim specifičnim situacijama (odustanak od akcije pri držanju levog tastera miša i sl.) i selekcije, koje je ujedno i početno stanje. Automatsko skrolovanje nije realizovano kao posebno stanje, iako bi i to mogao biti sasvim validan pristup, već kao konkurentni proces. Za opisana ponašanja je korišćen *State* dizajn šablon [5], gde sve klase koje reprezentuju stanja nasleđuju jednu, apstraktnu, koju referencira kontekst sistema.

Pošto se može očekivati čest prelazak iz jednog u drugo stanje, *State* šablon se koristi u kombinaciji sa *Flyweight* dizajn šablonom [6] koji objašnjava deljenje objekata. *Flyweight* objekti (stanja u ovom slučaju) se čuvaju u *Factory* objektu, gde se ujedno vrši i njihovo instanciranje. Dakle, postoji maksimalno jedna instanca svakog stanja, čime se izbegava njihovo često instanciranje i štede resursi. Na slici 3.13 je prikazan kompletni model koji opisuje rukovanje stanjima sistema.

Sa slike se vidi da je kontekst u ovom slučaju klasa *GraphEditController*, dok klasa *StateFactory*, kao *Factory* objekat, vrši instanciranje stanja i čuva ih radi kasnijeg jednostavnog pribavljanja bez potrebe ponovnog instanciranja.

Klasa State implementira interfejse koji poseduju metode za reakciju na određenu akciju korisnika, poput pritiska dugmeta miša, pomeranje miša, pritiska tastera tastature itd. Ishod akcija je drugačiji u različitim stanjima, pa se metode redefinišu u naslednicima.

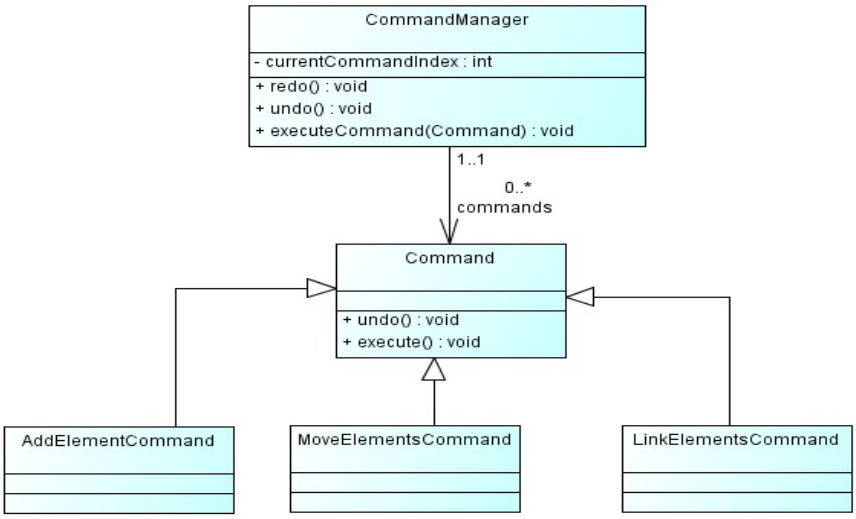


Slika 3.13 Model stanja

3.1.3.5 Komande

Kako bi se dozvolilo korisniku eksperimentisanje i olakšao rad, implementirano je poništavanje i ponovno izvršavanje akcija (*undo i redo*), oslanjanjem na *Command* dizajn šablon [7]. Ovaj šablon razdvaja objekat koji poziva operaciju od onog koji zna da je izvrši uvođenjem apstraktne klase sa *execute* i *undo* metodama koje su zadužene za izvršavanje i poništavanje komande i bivaju redefinisane u naslednicima. Posebna klasa, *CommandManager,* sadrži listu svih komandi i očuvava hronološki redosled njihovog izvršavanja, tako da se komande mogu redom poništiti, ili ponovo izvršiti. Na slici 3.14 prikazan je dijagram klasa koji opisuje pomenuti deo funkcionalnosti.

Svaka manipulacija nad elementima dijagrama je implementirana komandom (dodavanje elemenata, promena njihovih svojstava, brisanje itd.), te mogu biti poništene i ponovo izvršene prema korisnikovim željama. Treba napomenuti da nisu prikazane sve komande na slici 3.14, već samo mali podskup, kako se dijagram ne bi previše opteretio.



Slika 3.14 Dijagram klasa komandi

3.2 Integracija Kroki alata i editora dijagrama klasa

U nastavku će biti više reči o korišćenju opisanog editora dijagrama klasa u okviru Kroki alata u cilju skiciranja, uz postavaljanje podloge za kasnije proširivanje funkcionalnosti radi podrške projektovanja perzistentnog sloja.

3.2.1 Režimi rada editora

Editor omogućuje skiciranje kreiranjem dijagrama klasa, pružajući tako još jednu alternativu projektantima. Međutim, pri pravljenju dijagrama sa tim ciljem, moraju se ispoštovati određena ograničenja, poput zadavanja tipa atributa ili stereotipa klasa i metoda iz limitarog skupa mogućih vrednosti. Kako o tome ne bi morao da vodi računa korisnik sistema, uvedeni su režimi rada editora. Naime, editor se može pokrenuti u jednom od dva režima, gde je za svaki specificirano koje od funkcionalnosti prikazanih u poglavlju 3.1 su dostupne, kao i početni nazivi klasa, vrednosti njihovih stereotipa i sl. Režimi odgovaraju dvema primenama editora – modelovanju korisničkog interfejsa i perzistentnog sloja. Perzistentni režim rada nije ograničen u odnosu na puni režim, ali je pred rešenje postavljen zahtev da bilo kakva promena ovog segmenta ne predstavlja nikakav problem. UI (*User Interface*) režim, sa druge strane, podrazumeva poštovanje određenih pravila koja će biti detaljnije objašnjena.

3.2.1.1 UI režim rada editora

U UI režimu paketi, klase i veze između njih imaju posebna značenja, navedena u tabeli 3.2.

Element dijagrama klasa	Element skice
Klasa sa stereotipom *StandardPanel*	Standardni panel
Klasa sa stereotipom *ParentChild*	*Parent-child* panel
Paket	Poslovni podsistem
Atribut	Vidljivo obeležje
Metoda sa stereotipom *Transaction*	Transakcija
Metoda sa stereotipom *Report*	Izveštaj
Veza asocijacije između dve klase sa *StandardPanel* stereotipom	Povezivanje panela *Next* i *Zoom* mehanizmima
Uloga na krajevima asocijacije	Labele odgovarajućih komponenti
Veza asocijacije između klase sa *ParentChild* stereotipom i druge klase	Uspostavljanje hijerarhijske strukture između panela

Tabela 3.2 Mapiranje elemenata dijagrama klasa na elemente skice

Dijagram klasa korisničkog interfejsa se, dakle, može sastojati samo od određenih tipova elemenata i veza, te je onemogućeno dodavanje ostalih na radnu površinu. Naime, paleta alatki je "osiromašena" u odnosu na sliku 3.5, što se jasno zapaža na slici 3.15.

Slika 3.15 Paleta alatki u UI režimu rada

Rezultujući dijagram, dakle, ne može posedovati interfejse, niti bilo koje vrste veza između klasa osim asocijacije. I dalje je dopušteno povezivanje formiranjem veza kompozicije i agregacije, ali se ne pravi nikakva razlika u odnosu na običnu asocijaciju, izuzev, naravno, samog prikaza na dijagramu. Formiranje skica na uvedeni način podrazumeva dodavanje i povezivanje panela (standardnih ili *parent-child*), predstavljenih klasama sa odgovarajućim stereotipovima, što objašnjava razlog skrivanja ostalih dugmadi sa palete.

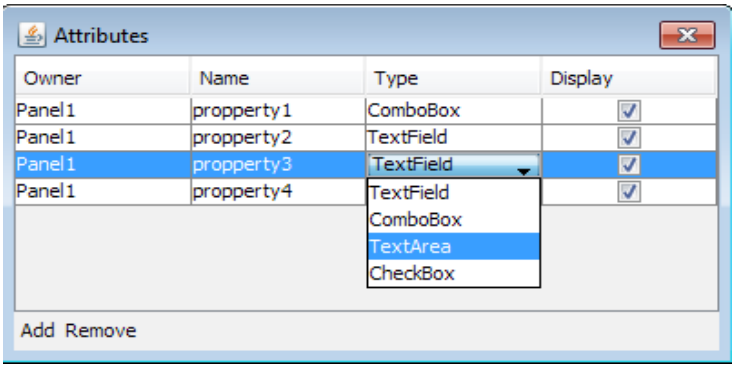
Klase pri kreiranju dobijaju nazive koje odgovaraju njihovoj semantici. U pitanju je prefiks "Panel" praćen brojem koji odgovara rednom

broju dodate klase. Naknadne promene se mogu vršiti na proizvoljan način. Stereotip se, sa druge strane, može birati samo putem *comboBox*-a sa vrednostima "*StandardPanel*" i "*ParentChild*", koji odgovaraju tipovima panela koji se mogu kreirati skiciranjem. Klasa inicijalno ima prvospomenuti stereotip, pošto se može očekivati da će standardni paneli biti zastupljeni u većem broju.

Osnovna ideja u UI režimu rada jeste kreiranje panela sa komponentama na alternativni način, pa je razumljivo da klase, atributi i metode, koji ih reprezentuju, ne mogu biti potpuno proizvoljni. Atributi odgovaraju komponentama za unos i prikaz podataka, te mogu imati samo tipove koji predstavljaju jednu od podržanih komponenti. Osim specificiranja tipa, u ovom režimu može se zadati još i naziv, na osnovu kog se dodeljuje vrednost odgovarajućoj labeli. Prema izloženom, svi atributi su sledećeg oblika:

- naziv : tip, tip $\in$ {*TextField*, *ComboBox*, *TextArea*, *CheckBox*}

Privatni su, nisu *static* ili *final,* proizvoljnog imena i tipa iz ograničenog skupa. Dijalog za unos atributa koji omogućava zadavanje pomenuta dva obeležja, prikazan je na slici 3.16.



Slika 3.16 Dijalog za unos atributa

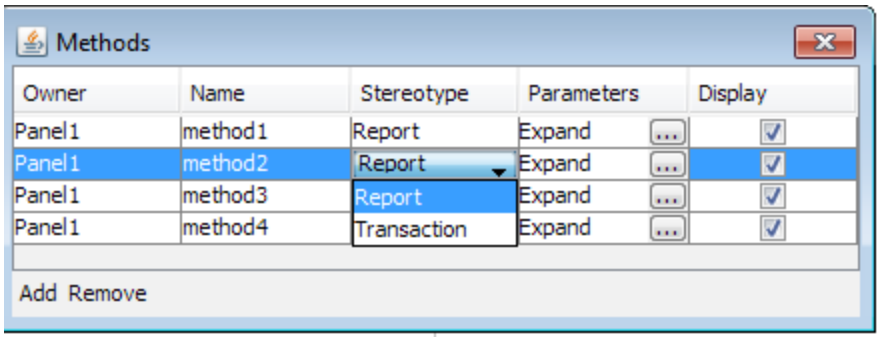
Na slici 3.16 vidi se takođe da se može promeniti klasa kojoj atribut pripada (*Owner*) kao i vidljivost na dijagramu, ali to nema uticaja na komponentu korisničkog interfejsa. Takođe, može se primetiti i da je naziv klase kojoj pripadaju atributi iz primera "Panel1", što odgovara konstataciji o inicijalnim vrednostima naziva klasa u razmatranom režimu. Na slici se vidi i da se tip može zadati samo izborom jedne od vrednosti iz *comboBox*-a, čime se nameće poštovanje ograničenja i ubrzava skiciranje. Upravo

spomenuto važi za svako obeležje čije vrednosti pripadaju nekom ograničenom skupu.

Metode, sa druge strane, odgovaraju pozivanju izveštaja i transakcija, gde se distinkcija između dva tipa zadaje njihovim stereotipom. I u ovom slučaju može se zadati naziv, ali i parametri. Validne metode imaju sledeći zapis:

+ naziv(parametri) : void <<stereotip>>, stereotip ∈{*Report, Transaction}*

Dakle, javne su, sa proizovljnim parametrima i nazivom, ne vraćaju nikakvu vrednost, dok se stereotip bira između dve vrednosti. Odgovarajući dijalog prikazan je na slici 3.17.



Slika 3.17 Dijalog za unos metoda

U UI režimu i veze imaju posebno značenje. Naime, njima se zadaju odnosi između panela. Između dve klase sa stereotipom "*StandardPanel"*, veza asocijacije se prevodi u *Zoom* i *Next* mehanizme. Kardinaliteti takođe nisu podložni potpuno slobodnom izboru, pošto se radi o vezi *one-to-many*. Veza u kojoj učestvuje *parent-child* panel kreira hijerarhiju, pa ni nju projektanti ne mogu menjati na bilo koji način.

3.2.1.2 Perzistentni režim rada

Kao što je već napomenuto, u perzistentnom režimu se ne nameće poštovanje prehodno izloženih pravila. Na raspologanju je puna funkcionalnost, što podrazumeva, između ostalog, dodavanje interfejsa i uspostavljanje proizvoljnih veza između elemenata. Inicijalni nazivi klasa počinju prefiksom "*Class*", a ne "*Panel*" kao u prethodnom slučaju. Pri kreiranju dijagrama perzistentnog sloja tipovi atributa, kao i stereotipovi klasa i metoda se zadaju kucanjem željene vrednosti, a moguće je

podešavanje i svih obeležja koja su u slučaju UI režima dobijala unapred predefinisane vrednosti.

U narednim poglavljima biće više reči o skiciranju upotrebom editora, što podrazumeva UI režim rada, dok je perzistentni uveden radi kasnijeg proširivanja funkcionalnosti čija implementacija nije primarni cilj ovog rada.

3.2.2 Skiciranje upotrebom editora klasa

Editor se, u UI režimu rada, koristi na identičan način kao što je objašnjeno u drugom poglavlju, ali paketi i klase koje se dodaju, kao i veze između njih imaju dodatnu semantiku. Kratak pregled značenja elemenata dijagrama dat je u tabeli 3.2, dok će u nastavku biti detaljnije izložene mogućnosti editora u kontekstu kreiranja skica.

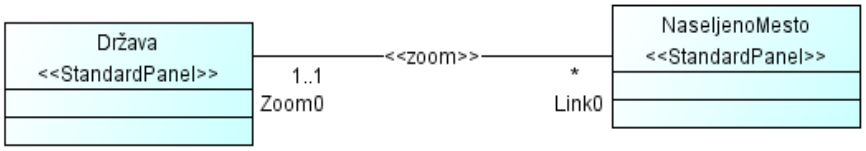
Kreiranje projekta i paketa u okviru njih ekvivalentno je istim akcijama dostupnim već preko glavnog prozora Kroki alata. Međutim, kreiranje paketa može se vršiti ne samo odgovarajućim akcijama nad stablom, nego i dodavanjem na dijagram. Kreiranje novog projekta automatski kreira i otvara njemu pridružen dijagram, koji se dalje može proizvoljno popunjavati. Popunjavanje projekta paketima se, dakle, može vršiti preko stabla, putem pridruženog kontekstnog menija, kada se paketi automatski dodaju na predefinisanu poziciju na dijagramu, ili dodavanjem odgovarajućeg grafičkog elementa na željeno mesto. Na ovaj način se struktura poslovnog sistema može definisati znatno brže. Naime, svaki paket i sam dobija pridruženi dijagram, čijim punjenjem se ažurira sadržaj dotičnog paketa. Već je napomenuto da svi elementi dobijaju inicijalne nazive koji zavise od njihovog tipa, pa se tako radi o nazivu "*Package"* praćenim radnim brojem za paket. Ova osobina se, međutim, lako i brzo može promeniti putem panela za podešavanja prikazanog na slici 3.4, a smeštenog u donji desni ugao prozora.

Klase odgovaraju panelima, dok je stereotipovima definisano o kojoj se tačno vrsti radi. Atributi klasa mogu se posmatrati kao elementi korisničkog interfejsa, gde tip definiše tip komponente, a naziv labelu. Metode sa druge strane, u zavisnoti od svog stereotipa, odgovaraju pozivima izveštaja i transakcija, a detekovanje postojanja parametara postavlja indikator potrebe prikaza forme za unos parametara.

Klase se mogu međusobno povezivati, uz, naravno, ograničenja izložena u 3.2.1. Povezivanjem dva standardna panela inicijalizuju se *Zoom* i *Next* mehanizmi, a veza u kojoj učestvuje p*arent-child* panel uspostavlja hijerarhiju, što će biti detaljnije izloženo u nastavku.

3.2.2.1 Veze između standardnih panela

Dve standardna panela mogu biti povezana samo *one-to-many* vezama, što se poklapa sa početnim kardinalitetima, gde se odredišni panel nalazi na "više" strani veze. Inicijalno stanje veze prikazano je na slici 3.18.



Slika 3.18 Veza uspostavljena između dva standardna panela

Vidi se da su početne uloge "*Zoom0*" i "*Link0*", što ukazuje na semantiku ove veze. Stereotip "*zoom*" takođe je postavljen da bi značenje veze bilo uočljivije, ali nema posebnu ulogu i može se menjati prema sopstvenim žaljama. Naseljeno mesto nalazi se u jednoj državi, koja se bira uz pomoć *Zoom* mehanizma. Sa druge strane, prelazak sa panela "Država" na povezani, "Naseljeno mesto", odgovara *Next* mehanizmu. Iz primera se vidi da tip mehanizma na određenoj strani veze zavisi isključivo od kardinaliteta. Promena kardinaliteta se može vršiti, ali samo pod uslovom da se izmenom ne promeni tip veze. Naime, dozvoljena je promena donjeg ograničenja, pa tako početni kardinalitet 1..1 može postati i 0..1 ili 1, dok je korektno promeniti * u 0..*, 1..* i slično. Pre svake promene se proverava validnost unosa, a korisnik se obaveštava o eventualnim greškama. Takođe, kardinalitet * se može promeniti u 1, ali se tada forsira i promena 1 u * na drugoj strani, kako bi veza ostala *one-to-many*.

Uloge u okviru veze definišu labelu odgovarajućih komponenata korisničkog interfejsa. Inicijalno dobijaju vrednosti u vidu prefiksa "*Zoom*" ili "*Link*" praćenih rednim brojem, ali se mogu naknadno menjati, što će se reflektovati u vidu promena odgovarajućih labela. U 2.3.2, gde je bilo više reči o pomenutim mehanizmima, prikazano je da se radi o *link* i *comboBox* komponentama, gde je nad drugom primenjena combozoom transformacija. Tada je objašenjeno da se i u okviru naprednih podešavanja mora birati ciljni panel za svaki od mehanizama i redosled u kome se to mora činiti. Uspostavaljem veze ceo postupak je već u celini izvršen – obe komponente su dodate u odgovarajuće grupe elemenata, a ciljni paneli se postavljaju na osnovu povezanih klasa. Promenom kardinaliteta se uklanjaju dodate komponente i postavljaju druge, u skladu sa zadatim vrednostima. Raskidanjem veze uklanja se i svaki trag njihovog postojanja na panelima, pa se tako ne mora voditi računa da se otvore oba povezana panela i

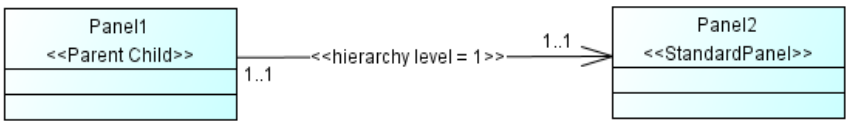
pronađu i uklone komponente preko kojih su veze uspostavljene. Na ovaj način se dobija kako na brzini rada, tako i smanjenju rizika od pravljenja greške. Jasno se vidi koje su tačno veze uspostavljene između kojih panela, a manipulacija sa njima ažurira oba panela. Brisanje klase dovodi do brisanja odgovarajućeg panela, ali i do raskidanja svih postojećih veza. Dakle, ne može se desiti ni da zaostane komponenta koja bi aktivirala nepostojeći panel i slično, oslobađajući korisnika i od ove brige.

Editor podržava poništavanje i ponovno izvršenje akcija, što važi i za povezivanje. Treba napomenuti da je vođeno računa da se uklanjanjem veza i klasa i poništavanjem ove akcije zadržavaju sve njihove osobine, pa je tako omogućeno slobodno eksperimentisanje. Dakle, ako je komponentama podešena boja fonta ili pozadine, ili bilo koje drugo svojstvo, ono neće biti promenjeno.

Uprkos postojanju veza u kojima učestvuje dati standardni panel, njegov stereotip se može promeniti, čime on menja tip. Međutim, to dovodi do raskidanja upostavljenih veza, ali bi eventualnim poništavanjem ove akcije bile ponovo formirane.

3.2.2.2 Veze parent-child panela sa drugim panelima

Parent-child paneli, prepozantljivi po stereotipu "*ParentChild",* mogu imati učešće u vazama sa drugim *parent-child* ili standardnim panelima Na slici 3.19 prikazan je primer jedne ovakve veze.



Slika 3.19 Veza između *parent-child* i standardnog panela

Na ovaj način se može odrediti sadržaj *parent-child* panela, pri čemu se formira hijerarhijska struktura, što se može naslutiti na osnovu stereotipa koji se postavlja pri kreiranju veze. Polazni element, u ovom slučaju "Panel1", od koga počinje povezivanje, na prvom nivou hijerarhije (što se takođe vidi iz stereotipa), sadrži panel "Panel2". Sadržavajući panel je uvek onaj polazni, čak i u situacijama kada je odredišni istog tipa, o čemu se mora voditi računa pri povezivanju dva *parent-child* panela.

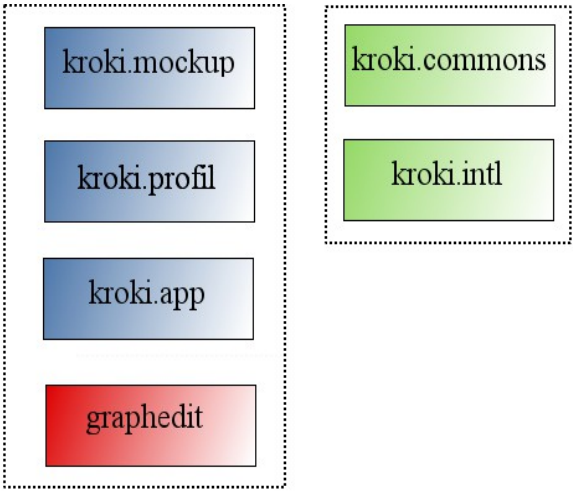
Kao i kada je reč o vezama između dva standardna panela, vrednosti dodatnih obeležja, u ovom slučaju hijerarhije, se automatski postavljaju i korisnik je rasterećen te brige. Radi se o ciljnom panelu, koji zavisi od panela na drugoj strani veze. Dakle, umesto dodavanja hijerarhijske komponente i postavljanja pomenutog obeležja, potrebno je

samo uspostaviti vezu između željenih panela. Uključivanjem još jednog panela u hijerarhiju, uspostavljanjem nove veze gde je razmatrani *parent-child* panel u polaznoj ulozi, uvedeni panel će se nalaziti na drugom nivou, pri čemu će se automatski postaviti i dodatno obeležje hijerarhijskog roditelja. Naime, pomenuto obeležje, na primeru sa slike 3.19, ukazivalo bi na "Panel2". Stereotip bi takođe bio modifikovan tako da ukazuje da se radi o drugom hijerarhijckom nivou. Naravno, i sam panel "Panel1" može biti na drugom kraju veze za neki *parent-child* panel.

Kardinalitet se ni u ovom slučaju ne može proizvoljno menjati. U pitanju je *one-to-one* veza, i bilo kakve promene ne smeju ugroziti ovu činjenicu. Dakle, promena u 0..1, 1 i sl. je dopustiva, sve dok gornje ograničenje ostaje isto. Provera validnosti se vrši pri svakom pokušaju izmene, a korisnik obaveštava o eventualnom postojanju problema.

3.2.3 Struktura Kroki alata integrisanog sa editorom

Struktura celokupnog Kroki sistema, uključujući i dodati grafički editor, prikazana je na slici 3.20.



Slika 3.20 Struktura celokupne Kroki aplikacije

Vidi se da se rešenje sastoji iz tri veće celine, na slici prikazanih unutar plavih simbola, kojima se dodaje i četvrta – grafički editor dijagrama klasa. Prva celina (*kroki.mockup*) zadužena je za iscrtavanje komponenti koje su sastavni deo UML profila, koji je definisan u drugoj celini (*kroki.profil*). Treća celina (*kroki.app*) predstavlja grafički editor za skiciranje koji omogućuje pozivanje editora dijagrama klasa, koji direktno operiše nad elementima UML profila.

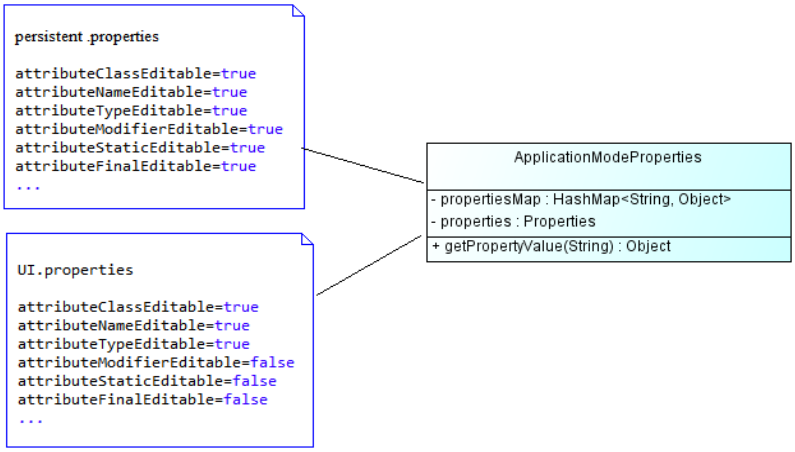
Iako modelovanje perzistentnog sloja nije implementirano, postavljena je podloga za podršku tog aspekta. Naime, uvedeni su režimi rada, opisani u 3.2.1, na osnovu kojih se prilagođava ponašanje editora. Međutim, vođeno je računa da se ovo postigne na što fleksibilniji način, kako bi naknadne promene mogle biti sprovedene brzo i jednostavno.

3.2.4 Implementacija skiciranja pomoću editora

Kao što je već napomenuto, uvedeni su režimi rada editora kako bi se povećala produktivnost i lakoća korišćenja, pa će, pre pregleda realizacije skiciranja, biti više reči o prilagođavanja editora tekućem režimu.

3.2.4.1 Podešavanja editora

Podešavanja editora u zavisnosti od aktivnog režima rada se vrši čitanjem specificiranih ograničenja i dozvola iz eksternih fajlova, čime se postiže jednostavnost pri naknadnim promenama. Kreirana su dva ovakva fajla, po jedan za svaki režim, pri čemu se inicijalnim zadavanjem režima automatski specificira i fajl koji će biti korišćen. Grafička predstava izloženog prikazana je na slici 3.21.



Slika 3.21 Implementacija specificiranja ponašanja aplikacije

Na slici 3.21 se vidi da su kreirana dva *.properties* faja – *UI.properties* i *persistent.properties.* Svaki od njih propisuje koji će delovi korisničkog interfejsa biti vidljivi, kao i početne nazive klasa, paketa, atributa itd. Na slici 3.21 prikazan je samo jedan deo, koji se odnosi na vidljive kolone tabele za unos atributa. *ApplicationModeProperties* je

singleton klasa koja učitava jedan od dva fajla i poseduje metodu za pribavljanje vrednosti zadate preko *.properties* fajla.

Na slici 3.21 zapaža se da su i ključevi identični, a razlika između dva fajla uočljiva jedino posmatranjem njima pridruženih vrednosti. Na taj način je smanjena zavisnost od poznavanja aktivnog režima. Recimo, prilikom prikaza tabele za unos atributa, potrebno je proveriti da li je određena kolona vidljiva. To se postiže preuzimanjem vrednosti iz *.properties* fajla inicijalno učitanog u *ApplicationModeProperties* klasi, a pošto su ključevi isti u oba slučaja, nakon instanciranja pomenute klase, nema dodatnih potreba za proverom aktivnog režima radi postizanja željenog cilja. Klasa *ApplicationModeProperties* stara se i o tome da tip podatka koji se vraća bude prilagođen potrebi, obezbeđujući i da se samo jednom vrši čitanje iz *.properties* fajla i konverzija, a da nakon toga podaci budu sačuvani u brzoj *hash* strukturi.

3.2.4.2 Modelovanje korisničkog interfejsa poslovnih aplikacija

Osnovni cilj proširenja editora jeste omogućavanje modelovanja korisničkog interfejsa kreiranjem dijagrama klasa, što podrazumeva direktnu manipulaciju nad strukturom UML profila u okviru Kroki alata, kao i nad elementima koji su neophodni za iscrtavanje dijagrama.

Kao što je već više puta spomenuto, jednom akcijom na dijagramu može se ostvariti promena više elemenata profila, pogotovo ako se manipuliše elementom koji učestvuje u jednoj ili više veza. Samo povezivanje podrazumeva ustanovljavanje tipa veze (hijerarhija ili *Zoom/Next*) na osnovu učesnika, dodavanje vidljivih obeležja na panele smeštanjem u odgovarajuće grupe uz postavljanje pojedinih atributa, poput aktivacionog i odredišnog panela, roditelja u hijerarhiji i sl. Raskidanje veze, sa druge strane, zahteva poništavanje spomenutih akcija, uz vođenje računa da se, ako se radilo o vezi između *parent-child* i nekog drugog panela, hijerarhijska struktura ne dovede u nevalidno stanje. Brisanje elementa povezanog sa nekim drugim se ogleda u uklanjanju panela iz paketa kome pripada, ali i uklanjanju vidljivih obeležja koja su ga referencirala sa svih drugih panela tj. kojima je on odredišni (*target*) panel. Promena stereotipa opet zahteva više od čiste izmene prikaza na dijagramu. Standardni paneli, primera radi, ne mogu sadržati hijerarhije, pa se ova obeležja ne smeju preneti na novi panel, što ne važi za izveštaje i transakcije. Promena kardinaliteta uspostavljenih veza između standardnih panela, odnosno, klasa kojima su predstavljeni, na način koji zahteva zamenu mehanizama, podrazumeva uklanjanje vidljivih obeležja nastalih

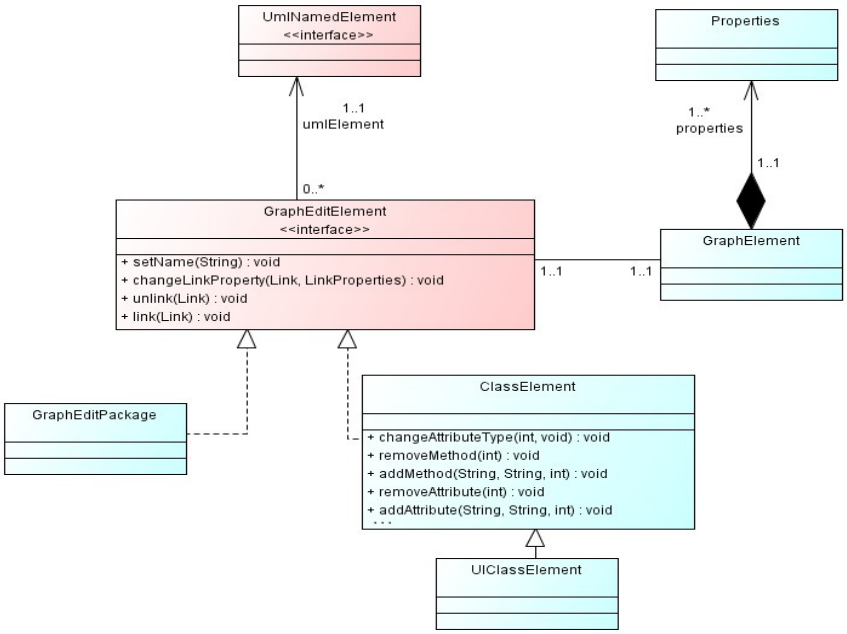
nakon povezivanja, i dodavanje onih kojima se realizuje drugi mehanizam. Dakle, izbacivanje *combozoom-a* i ubacivanje *link*-a i obrnuto.

Potrebno je obratiti pažnju i na poništavanje akcija. Naime, ukidanjem veze i njenim ponovnim uospostavljanjem ne bi se trebala izgubiti podešavanja načinjena nad komponentama korisničkog interfejsa. Takođe, raspored komponenti u okviru odgovarajuće grupe panela se ne bi smeo promeniti. Dakle, ponovno dodavanje nakon brisanja ne sme podrazumevati postavljanje na sam kraj, ukoliko se komponenta već nije tamo nalazila. Promena tipa atributa takođe nije trivijalan problem. Drugi tip uslovljava promenu komponente pridružene vidljivom obeležju, pri čemu ponovo mora da se vodi računa da se pozicije na panelu pre i nakon promene ne razlikuju. Takođe, određene osobine same veze se postavljaju u zavisnosti od tipa i svojstava elemenata UML profila koji se povezuju. Radi se, pre svega, o stereotipu i nazivima uloga na krajevima asocijacije. Naime, stereotip treba da pokazuje nivo hijerarhije, koji se može promeniti raskidanjem neke druge veze u kojoj učestvuje isti *parent-child* panel. Kako bi se grupisao kod koji obezbeđuje poštovanje svega navedenog, uvedene su posebne klase, prikazane na slici 3.22, koje sadrže metode za izvršavanje svih povezanih akcija nad instancama elemenata UML profila uz eventualno ažuriranje grafičkih elemenata.

Sve klase ovog tipa implentiraju interfejs *GraphEditElement* koji sadrži element UML profila koji se menja. Dakle, klasa *GraphEditPackage* će sadržati element koji implementira *UmlPackage* interfejs, a *UIClassElement* instancu *VisibleClass* klase. Interfejs *GraphEditElement* obezbeđuje i vezu sa odgovarajućim grafičkim reprezentom i sadrži metode zajedničke za sve klase koje ga realizuju. Radi se, o metodama koje, pre svega, obezbeđuju izvršavanje svih potrebnih akcija pri povezivanju datog elementa sa nekim drugim, raskidanju postojeće veze ili promeni njenog svojstva. Paketi, u principu, mogu biti povezani sa drugima, te je zato i ostavljena mogućnost implementacije ovih metoda. *ClassElement* je apstraktna klasa koja proširuje skup metoda sa onima koje pružaju reakciju na dodavanje, brisanje i promenu atributa i metoda. *UICLassElement* je konkretna klasa, naslednica *ClassElement* klase, implementira sve metode i tako obezbeđuje poštovanje svega izloženog na početku poglavlja. Apstraktna klasa je uvedena kako bi se eventualno nešto slično moglo kasnije implemetirati i pri modelovanju perzistentnog sloja.

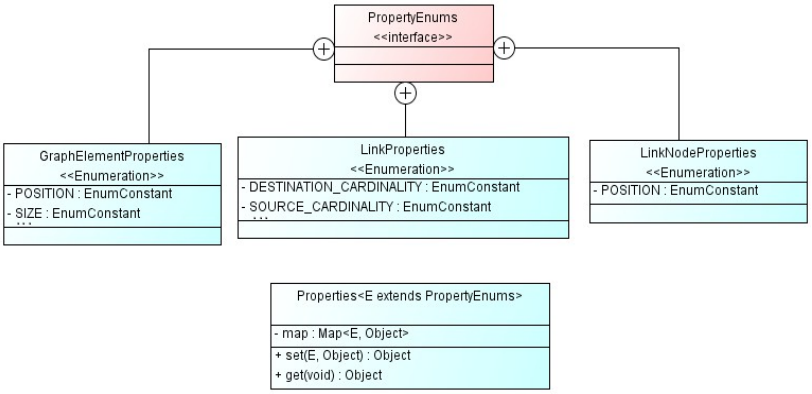
Klasa *GraphElement* je predak svim grafičkim elementima, što je detaljnije objašnjeno u 3.1.3.2, i poseduje obeležja potrebna za iscrtavanje na dijagramu. Naime, grafički element je povezan sa parametrizovanom klasom *Properties*, koja sadrži mapu obeležja čiji ključevi zavise od parametra. U pitanju je neka od enumeracija koja sadrži sva obeležja koja

imaju smisla za dati element. Veze, recimo, imaju kardinalitete, uloge, stereotip itd. Sa druge strane, klase, paketi i interfejsi, odnosno simboli sa kojima su reprezentovani na dijagramu, imaju poziciju, veličinu i sl.



Slika 3.22 Deo modela zadužen za ažuriranje svih elemenata UML profila nakon akcija nad dijagramom

Na ovaj način je omogućeno istovetno postavljanje svih obeležja. Model ovog dela editora prikazan je na slici 3.23. Na slici nisu navedena sva obeležja, kako se dijagram ne bi previše opteretio.



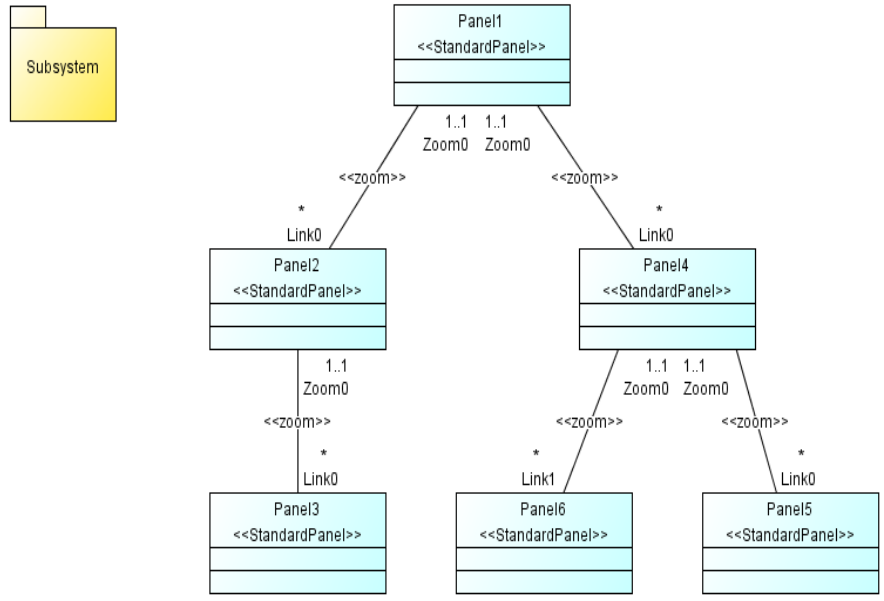
Slika 3.23 Model dela editora klasa koji snadbeva grafičke elemente obeležjima

3.3 Prikaz dijagrama skiciranih poslovnih sistema

Svaka skica, bez obzira na način kreiranja, se može prikazati u okviru editora dijagrama klasa. Tom prilikom se vrši automatsko raspoređivanje elemenata, kako bi rezultujući dijagram bio što pregledniji. Nakon otvaranja, dijagram je moguće menjati na istovetan način kao i pri kreiranju, uključujući i kreiranje novih paketa i klasa (panela) i modifikaciju postojećih.

3.3.1 Otvaranje dijagrama klasa korisničkog interfejsa

Omogućen je prikaz čitavog sadržaja *workspace*-a, kao i pojedinačnih projekata. Prvospomenuto se postiže preko ikone na paleti alatki, dok je druga akcija dostupna preko kontekstnog menija vezanog za projekat. U oba slučaja, nakom pokretanja editora, mogu se kreirati i popunjavati novi projekti. Prilikom automatskog raspoređivanja, pozicija elemenata se određuje na osnovu njihovog tipa i veza sa drugim elementima. Na krajnju levu poziciju kreiranog dijagrama smeštaju se paketi, prethodeći tako povezanim, te nepovezanim panelima, odnosno, klasama koje ih reprezentuju. Naime, povezani paneli se organizuju u stabla, pri čemu se korenski element postavlja na vrh. Na slici 3.24 prikazan je jedan primer automatski kreiranog dijagrama.



Slika 3.24 Dijagram klasa automatski kreiran na osnovu postojeće skice

Vidi se da je paket odvojen, i gledajući sa leva prema desno, pozicioniran pre strukture stabla, u koju su uvezani paneli koji se međusobno referenciraju.

3.3.2 Realizacija kreiranja dijagrama na osnovu skica

Osnovni problem koji se opisuje u nastavku jeste kreiranje i automatsko raspoređivanje klasa u okviru dijagrama na osnovu postojećih skica aplikacije u okviru Kroki editora formi. Cilj je dobijanje preglednih rezultujućih dijagrama.

3.3.2.1 Kreiranje grafičkih elemenata

Samo kreiranje grafičkih predstavnika paketa i panela na dijagramu klasa nije problem, kako se veličina inicijalno ne zadaje, nego računa na osnovu naziva, stereotipova, sadržanih metoda i atributa, što je sve dostupno preko instance elementa UML profila koji se iscrtava. Postavljanje pozicija se nakdnadno vrši, pošto bez posedovanja čitave slike o projektu nije moguće izvršiti pomenuto na način koji ispunjava gorenavedene kriterijume.

Ne tako trivijalan problem jeste kreiranje veza, što podrazumeva analiziranje vrednosti atributa *targetPanel* i *activationPanel* klase *VisibleAssociationEnd*, koju nasleđuju klase *Zoom*, *Next* i *Hierarchy*, koji specificiraju koji je panel odredišni, a koji aktivacioni za dati kraj asocijacije. Na osnovu prvospemenutog atributa se može zaključiti da je njegov vlasnik povezan sa drugim panelom. Međutim, trebalo bi ustanoviti i da li se radi o dvosmernoj vezi, tj. da li panel koji ima ulogu *Zoom* forme poseduje link ka aktivacionom panelu. Dakle, da bi se uspostavila bidirekciona veza, potrebno da za dva panela važi da jedan poseduje *Zoom* čiji je aktivacioni panel jednak odredišnom za *Next* drugog panela, ili obrnuto.

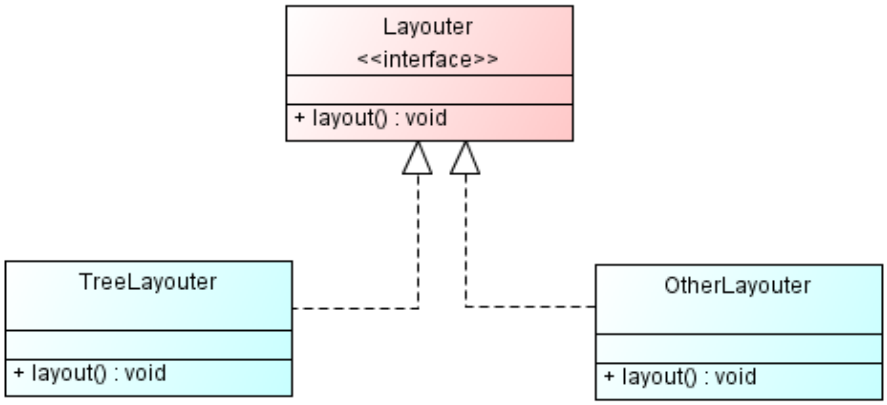
Kreiranje veza u grafičkom editoru zahteva prosleđivanje liste prekidnih tačaka, gde se pretpostavlja da su prva i poslednja konektori. Bitno je da se pozicija konektora instancira tako da se on nalazi unutar elementa. Kako se raspoređivanje ne bi previše zakomplikovalo, sve veze inicijalno imaju samo dve prekidne tačke (odredišni i polazni konektor), koji se postavljaju u sredinu elementa kome pripadaju. Naravno, i njihove pozicije će biti modifikovane pri kasnijem raspoređivanju.

Nakon kreiranja svih grafičkih elemenata i njihovog prvobitnog iscrtavanja, kada im se računaju i postavljaju dimenzije, na raspolaganju su sve informacije potrebne za njihovo raspoređivanje. Naime, pošto se u obzir uzimaju veličine elemenata i sve veze u kojima učestvuju, nije moguće

izvršiti raspoređivanje pri kreiranju elemenata. Oni se inicijalno postavljaju u centar dijagrama (tačku sa koordinatama (0,0)), kako njihova pozicija nije ni od kakvog značaja, te činjenica da se preklapaju ne predstavlja nikakav problem.

3.3.2.2 Podrška za više algoritama

U cilju ostavljanja mogućnosti za kasnija poboljšanja i implementaciju više različitih algoritama za raspoređivanje elemenata, uz eventualno nuđenje korisniku više alternativa, korišćen je *Strategy* dizajn šablon [12]. Šablon pruža nezavisnost od konkretnog algoritma, izdvajajući apstrakciju u interfejs koga implementira više klasa. Na slici 3.25 prikazana je primena šablona na problem raspoređivanja.



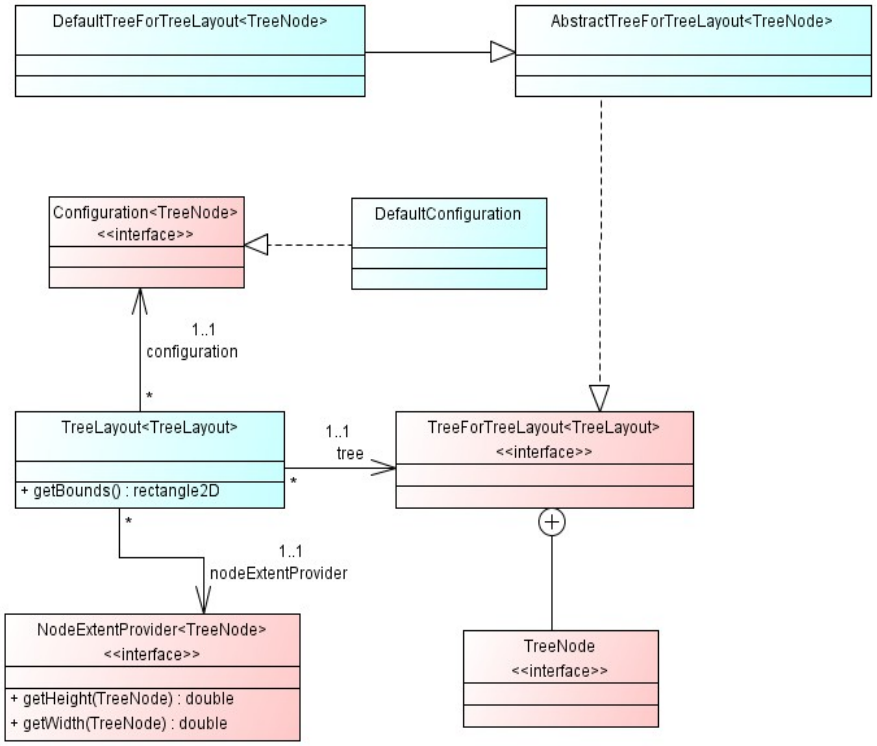
Slika 3.25 Upotreba s*trategy* šablona radi podrške raspoređivanju na više načina

Interfejs *Layouter* uvodi metodu *layout* koja vrši raspoređivanje elemenata aktivnog dijagrama. Njenu konkretnu implementaciju pruža klasa *TreeLayouter*, raspoređivanjem u strukturu tipa stabla, dok je druga klasa uvedena kako bi se lakše uvidela svrha šablona. Naime, može se dodati još jedna ili više klasa koje implementiraju *layout* metodu na različite načine, da bi se instancirala ona čija metoda dovodi do postavljanja pozicija na žaljeni način. Upotreba šablona omogućava lakoću promene algoritma, te bi se pri otvaranju više dijagama svaki put mogla izabrati drugačija opcija.

3.3.2.3 TreeLayouter

Klasa *TreeLayouter* raspoređuje elemente u stabla, oslanjajući se na projekat *abego TreeLayout* [13]. Algoritam koji se koristi jeste proširenje *Walker*-ovog algoritma za crtanje stabala proizvoljne veličine o čemu se

može više naći u [14]. Projekat omogućuje izračunavanje optimalne pozicije čvorova stabla na osnovu njihovih veličina i veza u kojima učestvuju. Međutim, kako bi se mogla iskoristiti ova osobina, prvo je potrebno organizovati elemente dijagrama u strukturu koja se može proslediti klasi koja implementira algoritam. Model *abego TreeLayout* projekta prikazan je na slici 3.26.



Slika 3.26 Model *abego TreeLayout* projekta

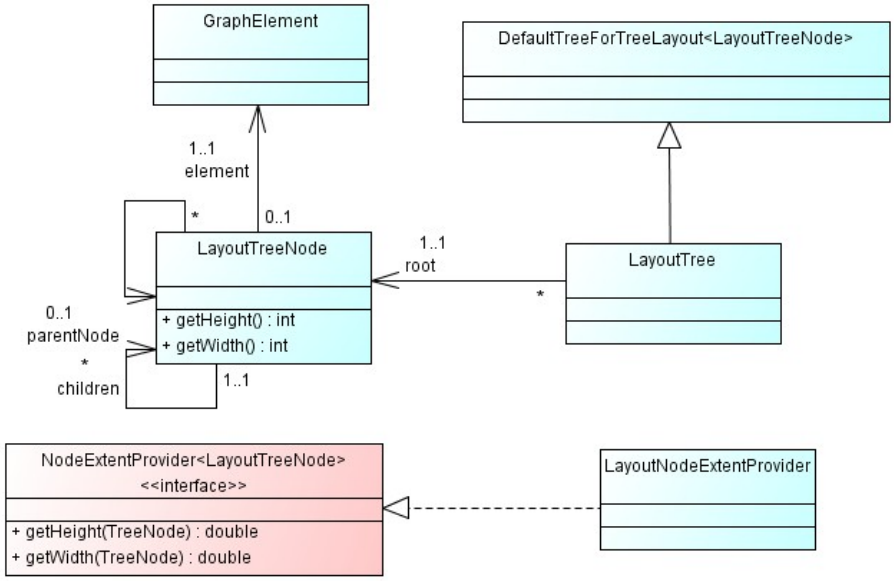
Klasa *TreeLayout* implementira algoritam i izračunava pozicije čvorova prosleđenog stabla, pri čemu se garantuje sledeće [14]:

- Čvorovi čine hijerarhijsku strukturu stabla, pri čemu se y koordinata određuje na osnovu nivoa u hijerarhiji
- Veze se ne presecaju, a čvorovi na istom nivou su na minimalnom horizontalnom rastojanju
- Prikaz podstabla ne zavisi od pozicije na stablu
- Zadati raspored direktnih potomaka čvora se prikazuje na dijagramu
- Algoritam omogućuje crtanje simetričnih stabala

Klasi je neophodno proslediti stablo, odnosno, instancu klase koja realizuje interfejs *TreeForTreeLayout,* pri čemu se može naslediti apstraktna klasa *AbstractTreeForTreeLayout* ili njena implementacija *DefaultTreeForTreeLayout,* koja poseduje gotove metode za dodavaje potomačkog čvora u stablo, proveru da li je čvor dodat i sl. *TreeLayout* takođe mora da zna i dimenzije svakog od čvorova, što je dovelo do uvođenja interfejsa *NodeExtentProvider.* Na kraju, pojavljuje se i interfejs *Configuration*, koji dozvoljava podešavanje različitih aspekata, poput:

- Razmaka između redova
- Minimalni razmak između elemenata
- Poziciju korenskog čvora
- Poravnanje manjih čvorova u okviru nivoa (na sredini, vrhu ili pri dnu)

Dakle, kako bi se iskoristile sve pogodnosti koje opisani projekat uvodi, potrebno je kreirati implementacije interfejsa *NodeExtentProvider, TreeNode* i *TreeForTreeLayout*. Na slici 3.27 prikazan je model dela editora zadužen za snadbevanje klase *TreeLayout* potrebnim podacima.



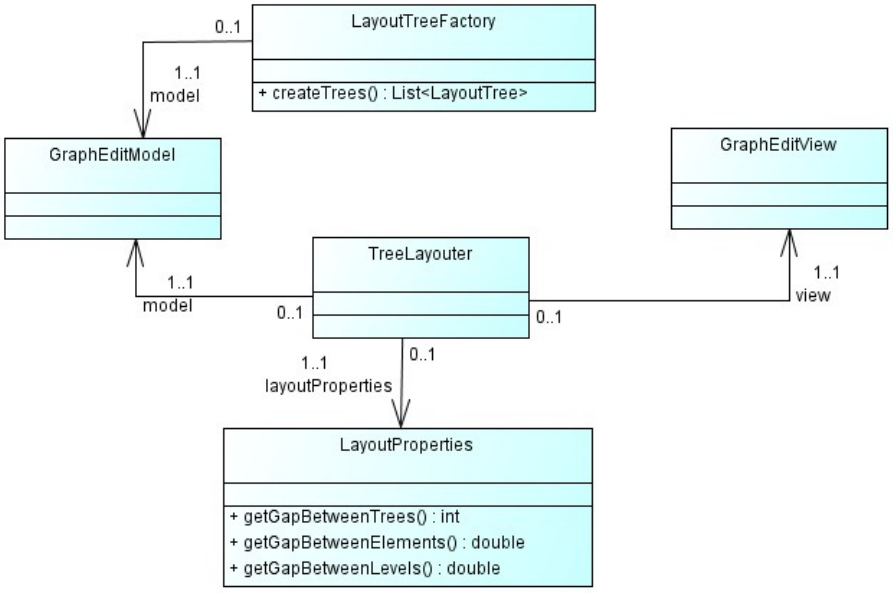
Slika 3.27 Model dela editora zadužen za snadbevanje klase *TreeLayout* potrebnim podacima

Klasa *LayoutTreeNode,* čije instance predstavljaju čvorove stabla, povezan je za grafičkim elementom, čime se omogućuje metodama klase *LayoutNodeExtentProvider* vraćanje dimenzija datog elementa kao

dimenzija čvora stabla. Vidi se da se direktno nasleđuje *DefaultTreeForTreeLayout,* pošto su implementacije metoda interfejsa *TreeForTreeLayout* koja ova klasa pruža, sasvim dovoljne za potrebe raspoređivanja. Takođe, nije napravljena posebna klasa čija bi uloga bila snadbevanje konfiguracijom, jer je već postojeća potpuno dovoljna.

LayoutTree ima samo jedan atribut i radi se o korenskom čvoru. Svaki čvor ima roditeljskog i listu potomačkih čvorova, te je prosleđivanje čvora na vrhu hijerarhije dovoljno da se pristupi i svim drugim. Međutim, potrebno je prvo kreirati takvu strukturu čvorova. Svaki grafički element sadrži listu konektora, gde se za svaki može proveriti da li je polazni ili odredišni, pa je upravo ova osobina iskorišćena za prevazilaženje navedenog problema i kreiranje stabla tj. instance *LayoutTree* klase.

Na jednom dijagramu može se nalaziti više stabala, ukoliko nisu između svih panela uspostavljene veze. Dakle, kreira se više stabala, a zatim se i ona sama raspoređuju na dijagramu tako da ne dođe do preklapanja i sličnih problema. Deo modela koji je zadužen za samo raspoređivanje prikazan je na sa slici 3.28.



Slika 3.28 Deo modela zadužen za raspoređivanje

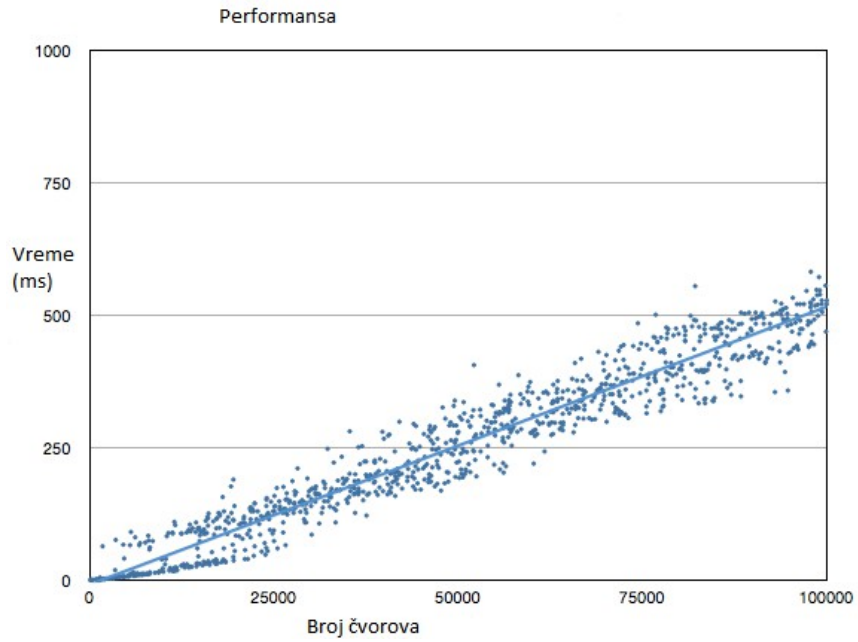
Klasa *LayoutTreeFactory* ima zadatak da formira listu stabala na osnovu elemenata modela, koje će dalje koristiti *TreeLayouter* za instanciranje *TreeLayout* klase koja će izračunati njihove optimalne pozicije. Međutim, tom prilikom se podešavaju i neki od parametara

rasporeda – razmak između elemenata i nivoa. Ovo se takođe radi preko *.properties* fajla, dok je za njegovo čitanje i vraćenje podataka u pogodnom obliku zadužena *singleton* klasa *LayoutProperties.* Raspoređivanje više stabala vrši se tako da se na samom početku prikažu paketi, pa veća i na kraju manja stabla, gde je razmak između njih takođe naveden u *.properties* fajlu.

Instanciranjem klase *TreeLayout* odmah se računaju pozicije svakog čvora stabla. One se mogu preuzeti preko metode *getBounds* ove klase i zatim iskoristiti za raspoređivanje grafičkih elemenata na dijagramu, što je zadatak *layout* metode. Prolazi se, dakle, kroz sve elemente, postavljaju njihov nove pozicije, ali i nove pozicije konektora koji im pripadaju, te pozivaju metode njihovih *painter*-a, koje će omogućiti da sve promene budu odmah vidljive i na samom dijagramu.

3.3.2.4 Analiza performansi

Algoritam koji se implementira, baziran na *Walker*-ovom uz poboljšanja o kojima se može više pročitati u [14], ima izuzetno dobre performanse, zahtevajući linearo vreme, čak i kada se radi sa velikim stablima, sa puno čvorova. Na slici 3.29 su prikazani rezultati testiranja ovog algoritma na računaru MacBook Pro 2.4 GHz Intel Core 2 Duo.



Slika 3.29 Rezultati testiranja algoritma na računaru [13]

Međutim, izbegavanje presecanja se komplikuje ako postoje veze između elemenata koji se ne nalaze na susednim nivoima u hijerarhiji. Dakle, tada bi se trebalo javiti više čvorova koji se odnose na isti grafički element. Dobijena preglednost je, ipak, u većini situacija sasvim solidna, ali se nameće zaključak da je postupak potrebno unaprediti u nastavku razvoja alata, poboljšavanjem uvedenog algoritma ili implementacijom drugih. Uvođenje prečica (*shortcut*-a), omogućile bi pojavu reprezenata panela na više mesta na dijagramu, te bi se ovakav pristup mogao iskoristiti za poboljšanje preglednosti dijagrama u spomenutim problematičnim situacijama. Prečice bi se tada mogle pojaviti na više nivoa na dijagramu, te ne bi postojala potreba za postojanjem veza između elemenata na udaljenijim nivoima, ili, pak, na istom nivou.

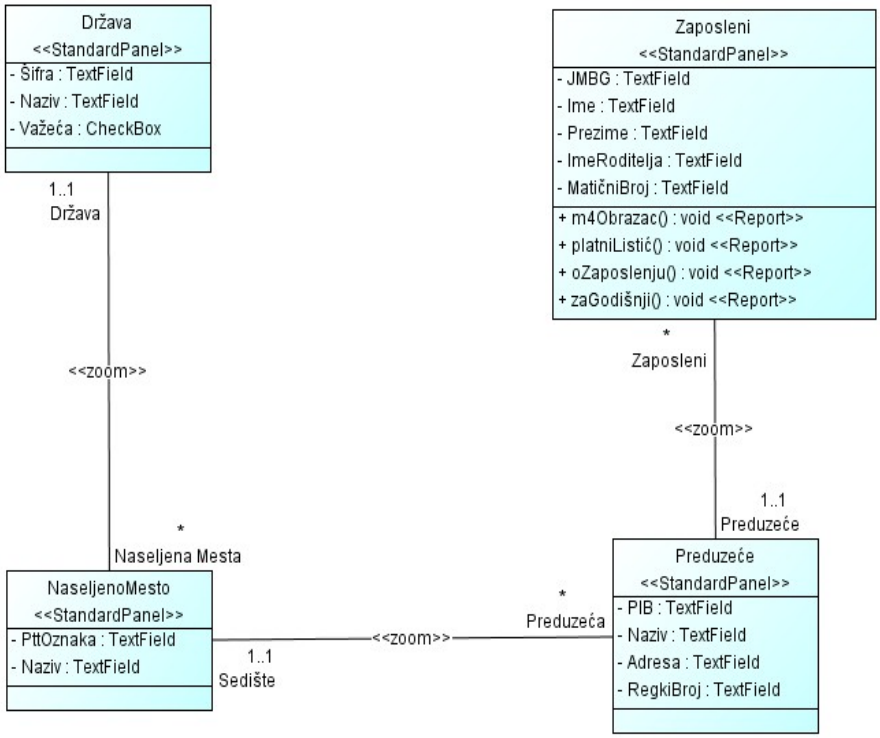
Moguće je i da se izložena opcija pokaže boljom pod nekim okolnostima, pa je zato i vođeno računa da se bez većih problema može vršiti biranje jedne od više raspoloživih pri prikazu svakog dijagrama, ili čak pozivanju automatskog raspoređivanja u toku rada, čime bi korisnik mogao da isproba više ponuđenih alternativa i izabere onu koja najviše odgovara datoj situaciji. Naravno, naknadne promene nad dijagramom su moguće, pa je korisnik svakako i u mogućnosti da dijagram organizuje prema svojim željama.

4. PRIMERI

U ovom poglavlju biće demonstrirano kreiranje panela preko dijagrama klasa, kako standardnih, tako i *parent-child.*

4.1 Kreiranje standardnih panela

Na slici 4.1 prikazan je dijagram klasa korisničkog interfejsa organizacione strukture jednog preduzeća.

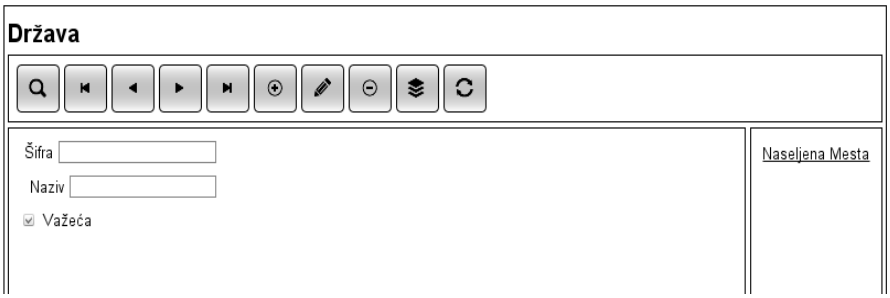


Slika 4.1 Dijagram klasa korisničkog interfejsa organizacione strukture jednog preduzeća.

Primećuje se da sve klase imaju stereotip "*StandardPanel",* čime je označeno da reprezentuju standardne panele. Vidi se i da su uspostavljene veze između klasa "Država" i "NaseljenoMesto", gde se na 1..1 strani nalazi "Država". Na slici se uočava i da su krajevi asocijacije imenovani. Radi se o nazivima "Država" i "NaseljenaMesta". Slično se zapaža i kod parova klasa "NaseljenoMesto" i "Preduzeće", te "Preduzeće" i

"Zaposleni". Sve klase imaju više atributa, dok "Zaposleni" ima i četiri metode sa stereotipom "Report".

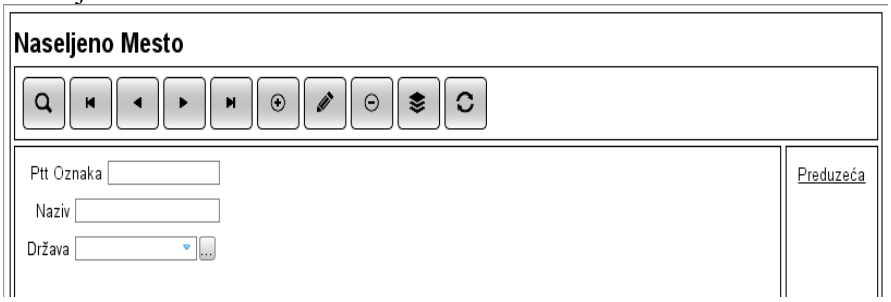
Na slici 4.2 prikazan je standardni panel koji je kreiran na osnovu klase "Država".



Slika 4.2 Standardni panel "Država"

Klasa država ima dva atributa čiji je tip *TextField,* koji označava tekstualno polje. Na slici 6.2 primećuju se upravo dve komponente odgovarajućeg tipa, čiji se labele poklapaju sa nazivima atributa. Treći atribut je tipa *CheckBox,* koji označava polje za čekiranje, koje se takođe nalazi na panelu. Vidi se da je i u ovom slučaju labela određena nazivom atributa. Posmatrana klasa se nalazi na 1..1 strani veze, te standardni panel ima i *link* komponentu, preko koje je povezan sa panelom "Neseljeno Mesto". Labela komponente je dobijena transformacijom naziva kraja asocijacije. Naime, ukoliko se nazivi pišu u *camel case* notaciji, transformišu se dodavanjem razmaka između reči.

Na slici 4.3 prikazan je standardni panel kreiran na osnovu klase "NaseljenoMesto".

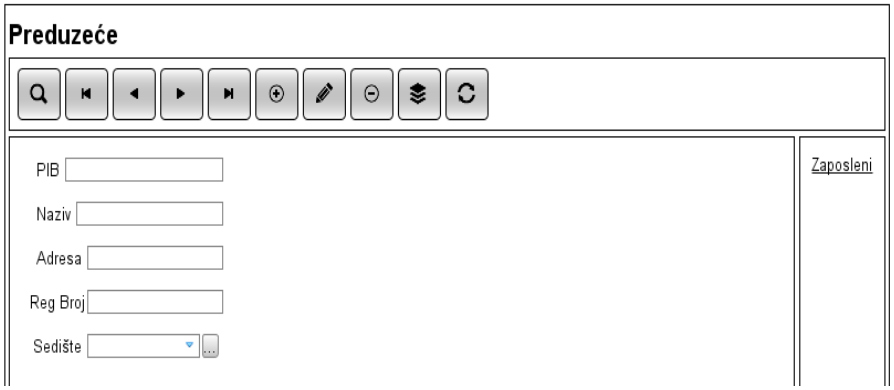


Slika 4.3 Standardni panel "Naseljeno Mesto"

Posmatrani panel liči na panel "Država", sadržavajući dva tekstualna polja i *link* ka drugom panelu, u ovom slučaju "Preduzeće", ali ima i *combozoom* komponentu koja je nastala kao posledica nalaženja odgovarajuće klase na

"više" strani asocijacije, gde se na drugoj strani nalazi klasa "Država". Labela komponente je i u ovom slučaju dobijena na osnovu naziva kraja asocijacije.

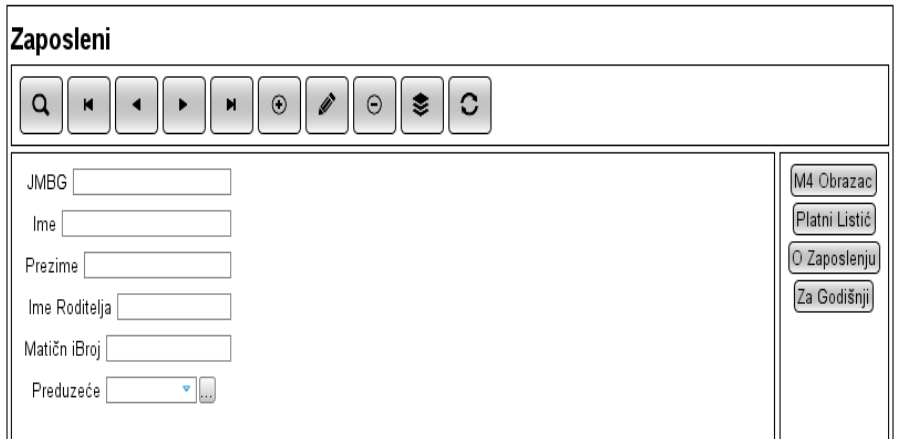
Na slici 4.4 prikazan je standardni panel kreiran na osnovu klase "Preduzeće".



Slika 4.4 Standardni panel "Preduzeće"

Na slici se vidi sve što je konstantovano u prethodnim slučajevima. Dakle, po jedna *link* i *combozoom* komponenta, kao posledice dve veze u kojima klasa učestvuje, uz još četiri tekstualna polja nastala na osnovu četiri atributa.

Na slici 4.5 prikazan je standardni panel kreiran na osnovu klase "Zaposleni".

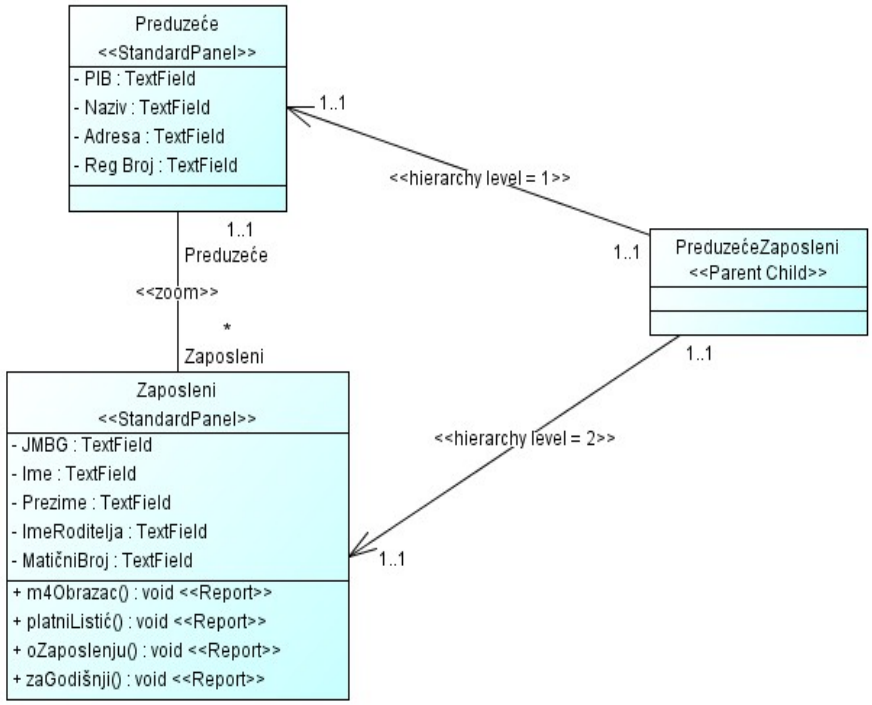


Slika 4.5 Standardni panel "Zaposleni"

Klasa “Zaposleni” učestvuje u samo jednoj vezi, nalazeći se na * strani asocijacije, što znači da panel ima *combozoom* komponentu, koja se i može uočiti na slici 6.5. Kao i u prethodnim slučajevima, može se reći isto vezano za kreiranje tekstualnih polja na osnovu atributa, ali klasa ima i četiri metode, sa stereotipom “Report”. Posledica njihovog kreiranja jeste dodavanje komponenata tipa *Report* (izveštaj) na panel, čije su labele izvedene na osnovu naziva odgovarajućih metoda.

4.2 Kreiranje *parent-child* panela

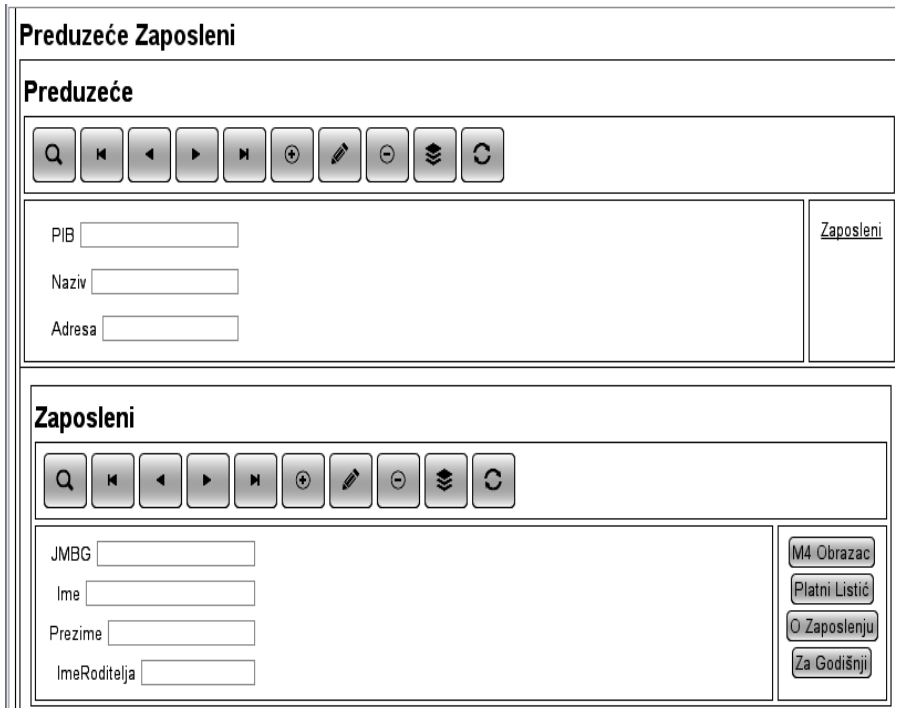
Na slici 4.6 je prikazan dijagram korisničkog interfejsa u kome se javlja jedan *parent-child* panel, povezan sa dva prethodno definisana standardna.



Slika 4.6 Dijagram korisničkog interfejsa u kome se javlja jedna *parent-child* panel

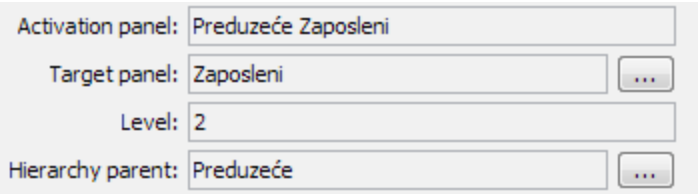
Klasa “PreduzećeZaposleni” predstavlja *parent-child* panel, što je specificirano preko stereotipa “*ParentChild”*. Klasa je povezana sa dva standardna panela, koja su već bile analizirana. Na osnovu stereotipa veze se može videti na kom se nivou hijerarhije nalazi koji od dva panela. Dakle, “Preduzeće” se nalazi na prvom, a “Zaposleni” na drugom.

Na slici 4.7 prikazan je *parent-child* panel kreiran na osnovu klase "PreduzećeZaposleni".



Slika 4.7 *Parent-child* panel "Preduzeće Zaposleni"

Standarni paneli su nešto smanjeni, kako bi stali celi na sliku, pa treba napomenuti da se zaista radi o istim panelima koji su prikazani na slikama 4.4 i 4.5. Kako bi se pokazalo da su nivoi hijerarhije i roditeljski panel ispravno podešeni, na slici 4.8 je prikazan deo naprednih podešavanja za hijerarhiju čiji je ciljni (*target*) panel "Zaposleni".

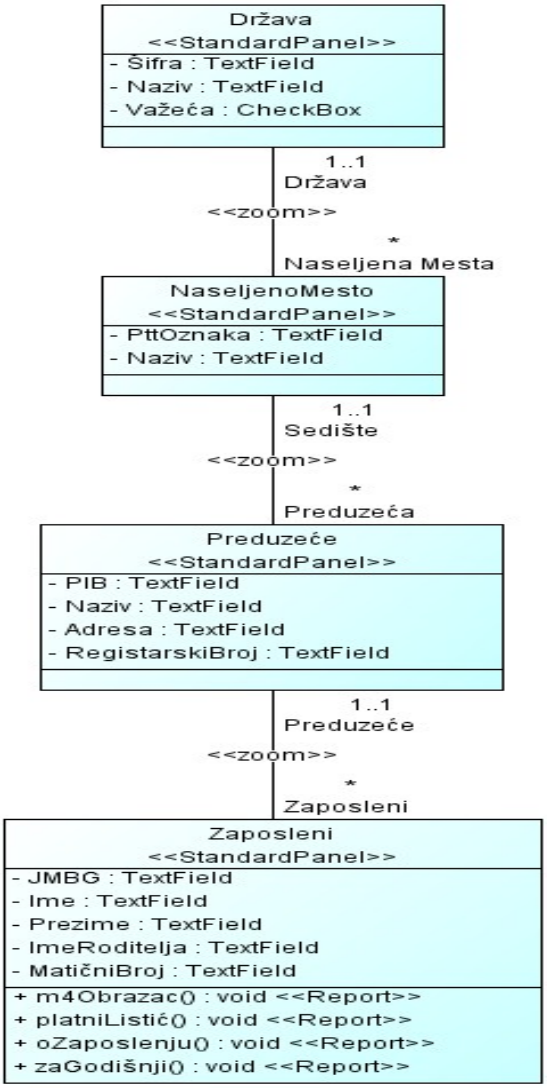


Slika 4.8 Napredna podešavanja hijerarhije "Zaposleni"

Na slici se vidi da je aktivacioni panel "Preduzeće Zaposleni", prikazan na slici 4.7, a ciljni "Zaposleni". Takođe, vidi se da se radi o drugom nivou hijerarhije, kao što bi i trebalo biti na osnovu dijagrama klasa. Roditeljski panel je "Preduzeće", koji se, samim tim, nalazi na prvom nivou hijerarhije.

4.3 Kreiranje dijagrama klasa na osnovu panela

Na slici 4.9 prikazan je dijagram klasa nastao na osnovu prethodno kreirane skice. Radi se o istim standardnim panelima koji su ranije opisivani, kreiranim na drugi način.

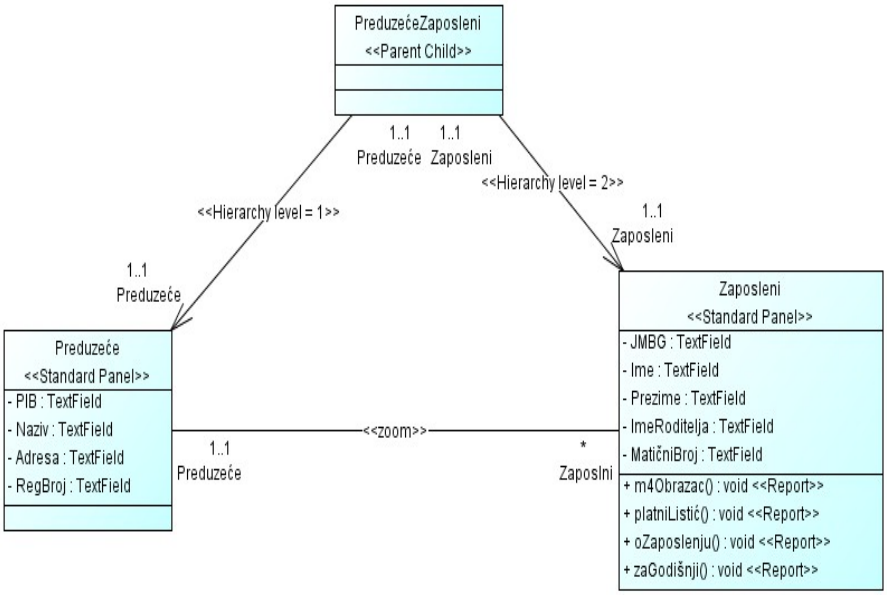


Slika 4.9 Dijagram klasa nastao na osnovu skice sa standardnim panelima

Na slici se može zapaziti da su uspostavljene sve veze, pri čemu nazivi krajeva asocijacija odgovaraju labelama odgovarajućih komponenata.

Kardinaliteti se postavljaju na osnovu uloge u asocijaciji, tj. da li panel ima *Zoom* ili *Next* komponentu. Takođe, može se videti i da sve klase imaju upravo onoliko metoda i atributa koliko komponenata imaju odgovarajući paneli, dok su njihovi nazivi izvedeni na osnovu labele, a tipovi na osnovu tipa komponenata. Sve klase imaju stereotip "*StandardPanel*", što se poklapa sa tipom panela. Klase su raspoređene tako da nema preklapanja i da se nalaze na malom međusobnom rastojanju, ali ipak dovoljno razmaknute da mogu lako da se uoče kardinaliteti i nazivi krajeva asocijacije, kao i stereotip veze. Prikazani dijagram se može dalje menjati, a sve promene će biti vidljive i na samoj skici po zatvaranju editora.

Na slici 4.10 prikazan je dijagram klasa nastao na osnovu skice kreirane preko Kroki editora koja obuhvata *parent-child* panel "Preduzeće Zaposleni" sa dvonivovskom hijerarhijom.



Slika 4.10 Dijagram klasa nastao na osnovu skice koja obuhvata *parent-child* panel

Na slici se vidi da su upostavljene veze i između *parent-child* panela i standardnih uvezanih u hijerarhiju. Stereotipovi ovih veza su inicijalizovani tako da ukazuju na nivo u hijerarhiji na kom se nalazi dati standardni panel, dok je *parent-child* panel ima stereotip *"ParentChild",* čime je označen njegov tip.

5. ZAKLJUČAK

U ovom radu je opisano proširenje Kroki alata u vidu editora dijagrama klasa za specifikaciju korisničkog interfejsa uz oslonac na EUIS UML profil, koji omogućuje kreiranje i povezivanje panela, kao i dodavanje projekata i poslovnih podsistema na alternativni i potencijalno brži način. Uvedeni su režimi rada editora, čime je olakšano modelovanje korisničkog interfejsa i smanjena verovatnoća činjenja grešaka. Takođe, dijagrami skica kreiranih na druge načine se mogu prikazati, automatski se kreiraju na zahtev korisnika i potpuno su editabilni. Naime, njihovom izmenom se istovremeno menjaju i odgovarajući paneli. Podržano je i eksperimentisanje, poništavanjem i ponovnim izvršenjem akcija, pri čemu sve osobine odgovarajućih komponenata ostaju očuvane.

Vođeno je računa o postavljanju podloge za jednostavna naknadna proširivanja:

- dodavanjem mogućnosti modelovanja perzistentnog sloja,
- uvođenja drugih načina automatskog raspoređivanja elemenata dijagrama.

Dalji razvoj obuhvatao bi i proširivanje mogućnosti samog editora, poput:

- kreiranja prečica
- implementacije naprednijih načina navigacije kroz dijagram
- više ponuđenih načina za menjenja osobina elemenata.

LITERATURA

[1] I. Cverdelj – Fogaraši, R. Vaderna, M. Jokić, R. Molnar, Projekat iz predmeta Projektovanje softvera, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2012.
[2] *Model–view–controller.* http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[3] *Model Delegate* http://c2.com/cgi/wiki?ModelDelegate
[4] *Observer Design Pattern.* http://sourcemaking.com/design_patterns/observer
[5] *State Design Pattern* http://sourcemaking.com/design_patterns/state
[6] *Flyweight Design Pattern.* http://sourcemaking.com/design_patterns/flyweight
[7] *Command Design Pattern.* http://sourcemaking.com/design_patterns/command
[8] M. Filipović, V. Marsenić, G.Milosavljević, I. Dejanović. Kroki: alat za interaktivni razvoj poslovnih aplikacija baziranih na skicama.
[9] G. Milosavljević, Priulog metodama brzog razvoja adaptivnih poslovnih informacionih sistema, doktorska disertacija, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2010.
[10] B. Perišić, G. Milosavljević, I. Dejanović, B. Milosavljević, UML Profile for Specifying User Interfaces of Business Applications. Computer Science and Information Systems, Vol. 8, No. 2, 405-426. (2011)
[11] V. Marsenić. Interaktivno zarvojno okruženje za specifikaciju poslovnih aplikacija, master rad, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2012
[12] *Strategy Design Pattern* *http://sourcemaking.com/design_patterns/strategy*
[13] Udo Borkowski *abego TreeLayout - Efficient and Customizable Tree Layout Algorithm in Java* *http://code.google.com/p/treelayout/*
[14] Buchheim C, Jünger M, Leipert S. Drawing rooted trees in linear time. *Software—Practice and Experience* 2006; 36(6):651–665

BIOGRAFIJA

Renata Vadetna je rođena 26.09.1989. godine u Novom Sadu. Osnovnu školu „Đorđe Natošević“ završila je 2004. godine, a matematički smer gimnazije „Jovan Jovanović Zmaj“ u Novom Sadu 2008. godine. Iste godine upisala se na Fakultet tehničkih nauka, odsek Računarstvo i automatika. Osnovne akademske studije završila je 2012. godine sa prosekom ocena 9,94. Iste godine upisala je diplomske akademske studije na istom odseku. Položila je sve ispite predviđene planom i programom sa prosekom ocena 10,00.

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj, **RBR**:	
Identifikacioni broj, **IBR**:	
Tip dokumentacije, **TD**:	monografska publikacija
Tip zapisa, **TZ**:	tekstualni štampani dokument
Vrsta rada, **VR**:	diplomski-master rad
Autor, **AU**:	Renata Vaderna
Mentor, **MN**:	dr Gordana Milosavljević
Naslov rada, **NR**:	Proširenje Kroki alata za skiciranje poslovnih aplikacija grafičkim UML editorom
Jezik publikacije, **JP**:	srpski
Jezik izvoda, **JI**:	srpski / engleski
Zemlja publikovanja, **ZP**:	Srbija
Uže geografsko područje, **UGP**:	Vojvodina
Godina, **GO**:	2013
Izdavač, **IZ**:	autorski reprint
Mesto i adresa, **MA**:	Novi Sad, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6
Fizički opis rada, **FO**:	5/ 57/ 2 / 2 /52/ 0/ 0
Naučna oblast, **NO**:	Informatika
Naučna disciplina, **ND**:	Metodologije brzog razvoja softvera
Predmetna odrednica / ključne reči, **PO**:	Mokapi, skiciranje, UML editor, poslovne aplikacije, dijagram klasa korisničkog interfejsa
UDK	
Čuva se, **ČU**:	Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6, Novi Sad
Važna napomena, **VN**:	
Izvod, **IZ**:	U radu je opisano proširenje alata Kroki, namenjenog za interaktivni razvoj poslovnih aplikacija baziranih na skicama, dodavanjem mogućnosti skiciranja pomoću dijagrama klasa i prikaza dijagrama postojećih skica i postavljanje podloge za kasniju implementaciju dodatnih funkcionalnosti.
Datum prihvatanja teme, **DP**:	
Datum odbrane, **DO**:	26.09.2013.
Članovi komisije, **KO**:	
predsednik	prof. dr Sonja Ristić, vanr. prof., FTN Novi

	Sad
član	dr Igor Dejanović, docent, FTN Novi Sad
mentor	dr Gordana Milosavljević, docent, FTN Novi Sad
	Potpis mentora

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	monographic publication
Type of record, **TR**:	textual material
Contents code, **CC**:	MSc thesis
Author, **AU**:	Renata Vaderna
Mentor, **MN**:	PhD Goradana Milosavljević
Title, **TI**:	Enhancement of Kroki, a tool for development of business applications based on mockups, by providing a graphical UML editor
Language of text, **LT**:	serbian
Language of abstract, **LA**:	serbian / english
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2013
Publisher, **PB**:	author's reprint
Publication place, **PP**:	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, **PD**:	5/ 57/ 2 / 2 /52/ 0/ 0
Scientific field, **SF**:	Computer science
Scientific discipline, **ND**:	Agile software development
Subject / Keywords, **S/KW**:	Mockups, graphical UML editor, business applications, user interface class diagrams
UDC	
Holding data, **HD**:	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, **N**:	
Abstract, **AB**:	This paper describes an enhancement of

	Kroki, a tool for participatory development of business applications based on mockups, which offers the posibility of sketching by creating class diagrams and viewing diagrams of existing sketches and lays the ground for future development of additional features.
Accepted by sci. board on, **ASB**:	
Defended on, **DE**:	26th of September, 2013.
Defense board, **DB**:	
president	Sonja Ristić, PhD, assoc. prof., FTN Novi Sad
member	Igor Dejanović, PhD, assist. prof., FTN Novi Sad
mentor	Gordana Milosavljević, PhD, assist. prof., FTN Novi Sad
Mentor's signature	