Sveučilište u Zagrebu
Fakultet Elektrotehnike i računarstva
Zavod za primijenjeno računarstvo

Sustav za automatizirano upravljanje poslovnim procesima

Projekt primjene IT proveden uz potporu Ministarstva znanosti, obrazovanja i športa

Verzija 1.1

Autori:
Prof.dr.sc. Krešimir Fertalj
Mr. sc. Boris Milašinović
Ivan Rendulić, dipl. inž.

Zagreb, siječanj 2006.

Sadržaj:

1. Sažetak

Sustav za automatizaciju poslovanja (SUSAP) je rezultat rada na projektu primjene informacijskih tehnologija "Sustav za automatizirano upravljanje poslovnim procesima", provedenom uz potporu Ministarstva znanosti, obrazovanja i športa 2004/2005 (šifra projekta 2004-206).

Cilj projekta je razvoj univerzalnog programskog sustava za upravljanje poslovnim procesima. Sustav omogućuje opisivanje organizacije, definiranje radnih procedura te autorizirano upravljanje tokovima informacija i dokumenata putem Interneta/Intraneta. Napravljeno je osnovno, općenito rješenje koje je prilagodljivo različitim vrstama korisnika. Rješenje je primjenjivo na upravljanje standardnim administrativnim poslovima, upravljanje nastavnim aktivnostima te upravljanje projektima. Osnovno, općenito rješenje treba biti prilagodljivo različitim vrstama korisnika te primjenjivo na upravljanje standardnim administrativnim poslovima, upravljanje nastavnim aktivnostima i upravljanje projektima.

U okviru projekta istražena su trenutno dostupna rješenja i algoritmi za distribuiranje informacija, procijenjena je funkcionalnost postojećih programskih rješenja te mogućnost njihove primjene na različite tipove organizacija i poslovnih procesa. Definiran je generički model za upravljanje poslovnim procesima i informacijskim tokovima te je sagrađen općeniti sustav za distribuciju informacija i upravljanje poslovnim procesima.

Sustav se sastoji od baze podataka, riznice dokumenata, web portala te aplikacija za definiranje poslovnih procesa i upravljanje dokumentima i porukama. Za različita područja primjene strukturirane su potrebne informacije, definirani su protokoli ponašanja i radnih procedura te su predviđeni korisnički prilagodljivi predlošci. Ugrađene su funkcije za autorizirano upravljanje poslovnim aktivnostima i dokumentima uz ažurno praćenje statusa resursa. Razmjena podataka i obavješćivanje o promjenama automatizira se gdje god bude moguće. Administriranje sustava svedeno je na minimum, uz visoku razinu sigurnosti pristupa kontroliranog na individualnoj i grupnoj razini.

Ugrađene su sljedeće komponente i funkcije:
- Uređivač poslovnih procesa,
- Središnji sustav za upravljanje protokom poslova (eng. workflow engine),
- Riznica sustava,
- Korisničko sučelje,
- Administrativno sučelje.

Oblikovani su i ugrađeni sljedeći predlošci protoka poslova:
- Sekvenca,
- Paralelno grananje,
- Sinkronizacija,
- Ekskluzivni odabir,
- Jednostavno spajanje,
- Višestruki izbor,
- Sinkronizirano spajanje,
- Višestruko spajanje,
- Diskriminator,
- Čekanje N od M,

- Proizvoljni ciklusi,
- Implicitni završetak,
- Višestruke instance bez sinkronizacije,
- Višestruke instance s unaprijed poznatim brojem instanci,
- Višestruke instance s brojem instanci poznatim u trenutku izvođenja,
- Višestruke instance s nepoznatim brojem instanci,
- Neposredni izbor,
- Naizmjenično paralelno izvođenje,
- Prekretnica.

Navedeni predlošci poznati su u teoriji već dulje vrijeme ali nisu u potpunosti ugrađeni čak ni u komercijalne sustave za upravljanje protokom poslova.

Definirani su algoritmi za provjeru ispravnosti poslovnog postupka, kao i njegovu raščlambu u format prikladan za pohranu u relacijskoj bazi podataka.

Napravljen je uređivač protoka poslova, koji omogućuje definiranje procesa korištenjem standardnih predložaka za upravljanje protokom poslova, definiranje nositelja aktivnosti te razmjenu podataka sa središnjom riznicom.

Riznica sustava omogućuje pohranu i upravljanje verzijama dokumenata podržanih standardnim paketima za podršku uredskom poslovanju (npr. MS Office datoteke) ali i generiranje korisnički definiranih formulara, za koje se na temelju XML sheme dokumenta generiraju zaslonske maske za ažuriranje podataka evidentiranih dokumentom. Razvijen je uređivač XML dokumenata koji se koristi za popunjavanje generičkih predložaka. Uređivač je dinamički prilagodljiv različitim shemama dokumenata.

Zagreb, 20.siječnja 2006.

Krešimir Fertalj

Prof.dr.sc. Krešimir Fertalj
Fakultet elektrotehnike i računarstva
Zavod za primijenjeno računarstvo
Unska 3, 10000 Zagreb, Croatia
Email: kresimir.fertalj@fer.hr
URL: http://www.zpr.fer.hr/osobe/kreso
Tel: ++385 1 6129 918
Fax: ++385 1 6129 915

2. Osnovni pojmovi

Ovisno o politikama tvrtki koje razvijaju sustave za upravljanje protokom poslova te ovisno o autorima radova iz područja upravljanja poslovnim procesima terminologija koja se koristi može biti dosta različita i dovesti do zabune. Primjerice, neki autori ([5]) razlikuju potpuno automatizirane dijelove posla od onih koji uključuju ljudski rad, pa im pridjeljuju različite nazive, što može izazvati nepotrebnu inflaciju pojmova (na primjer Business Process, Workflow Process, Manual Process, Workflow). Dodatni problem koji se pojavljuje prilikom imenovanja dijelova sustava predstavljaju i poteškoće prevođenja s engleskog jezika na hrvatski jezik, posebno kod naziva koji se sastoje od nekoliko uzastopnih imenica. Uzevši u obzir te činjenice, a i vodeći se mišlju da se nazivi prilagode nekim uobičajenim, odnosno ustaljenim nazivima unutar ustanove, za potrebe realizacije i opisa vlastitog sustava za upravljanje protokom poslova opisanog u ovom dokumentu osnovni koncepti definirani su kako slijedi.

2.1. Elementi modela procesa

2.1.1. Postupak

Postupak (eng. process, workflow) je skup pravila i definicija koji opisuju neku poslovnu proceduru unutar neke tvrtke ili ustanove, kao što su primjerice postupak nabave informatičke opreme, postupak izbora u zvanje ili postupak urudžbiranja službenih dokumenata. Postupak se sastoji od slijeda manjih zadataka (procesa) i prijelaza između tih zadataka. Prijelazima je definiran način izbora zadataka (procesa) koji se trebaju izvoditi, kao i redoslijed tih zadataka.

2.1.2. Instanca postupka

Svaki put kada se pojavi zahtjev za pokretanjem nekog postupka, neovisno o tome da li se zahtjevi pojavljuju slijedno ili istovremeno, sustav treba stvoriti novu instancu postupka koja je jednoznačno određena nekom oznakom. Svi dokumenti koji budu stvoreni ili korišteni u tako pokrenutom postupku, kao i resursi pridijeljeni postupku, bit će povezani s točno određenom instancom postupka.

2.1.3. Proces

Kao što je navedeno u definiciji postupka (2.1.1), svaki postupak se sastoji od skupa zadataka koji se trebaju izvršiti određenim redoslijedom poštujući pritom definirana pravila. Svaka od tih radnji može se nazvati proces[1] (eng. process, activity, task, workflow item). U ovom radu proces je definiran kao nedjeljiva cjelina unutar postupka koja opisuje jedan dio poslovne logike postupka. Svaki proces određen je svojim imenom, izvršiteljem ili grupom kojoj izvršitelj pripada, dokumentima koji se koriste unutar procesa i resursima koji su potrebni da bi se proces izvršio.

2.1.4. Aktivnost (Instanca procesa)

Ukoliko se neki proces koji pripada određenom postupku u nekoj od instanci pokrene, ta instanca procesa bit će jednoznačno određena brojem procesa i instancom postupka što zajedno čini jednu aktivnost ili instancu procesa u sustavu.

[1] Ovisno o literaturi pod pojmom proces se ponekad misli na cijeli postupak, dok se sam proces naziva zadatkom ili poslom, dok je u nekim slučajevima proces dio cijelog postupka, ali je djeljiv na manje cjeline.

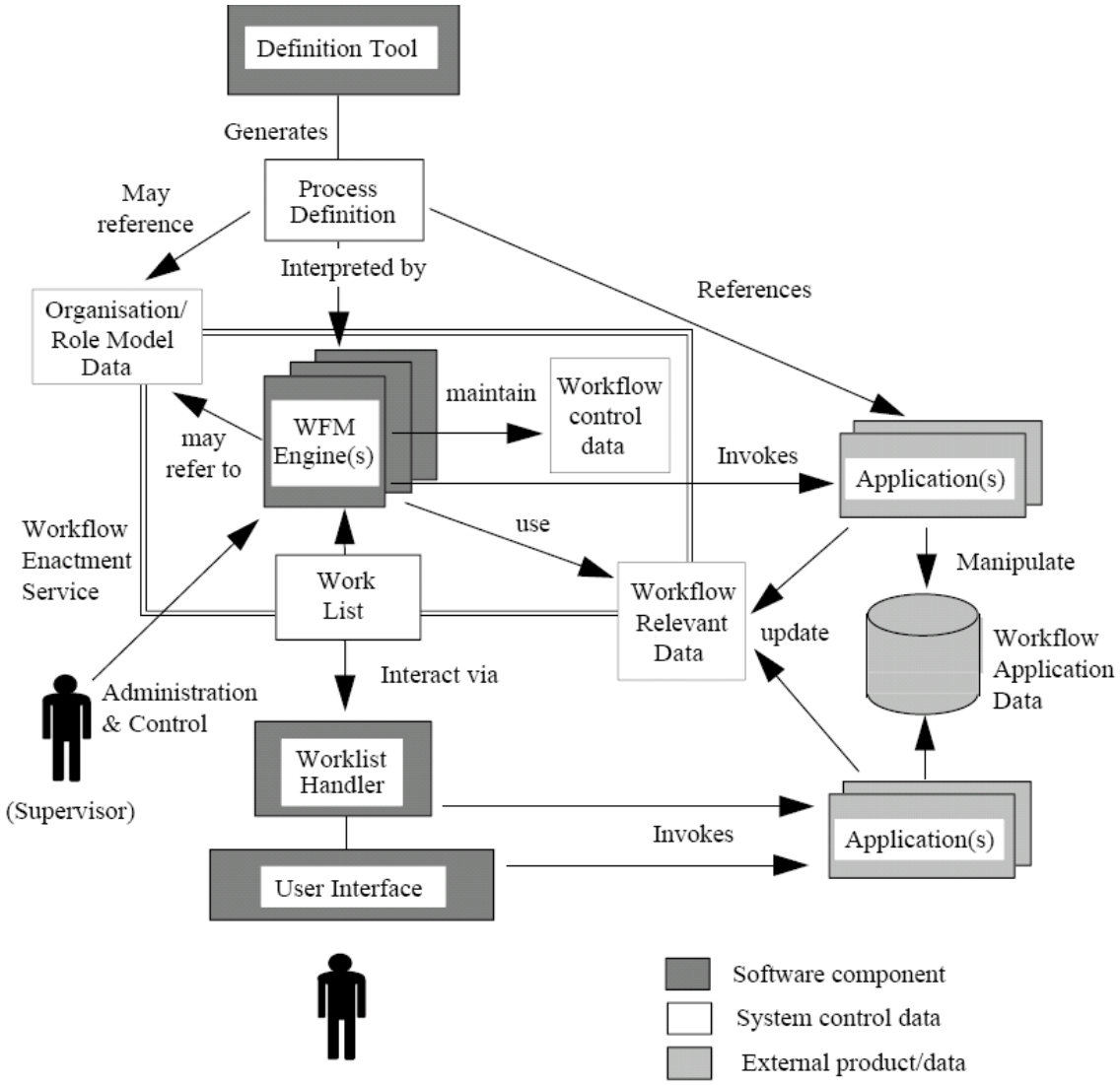
2.1.5. Prijelaz
Prijelaz (eng. transition) je nedjeljivi dio postupka koji označava promjenu aktivnog procesa, to jest završetak jedne aktivnosti i početak druge aktivnosti. Uvjet prijelaza određen je uvjetima zapisanima u svojstvima prijelaza i stanju vrijednosti specifičnih elemenata za pojedini postupak. Ti specifični elementi mogu biti dokumenti koji se koriste unutar sustava ili podaci o pojedinoj aktivnosti, kao što su trajanje aktivnosti, osoba koja obavlja danu aktivnosti i slično.

2.1.6. Instanca prijelaza
Budući da prijelaz definira odabir i redoslijed izvršavanja procesa unutar nekog postupka, u nekoj instanci postupka prijelaz između dviju aktivnosti obavlja se pomoću instance prijelaza.

2.1.7. Izvršitelj
Osim u slučajevima kada se neki proces obavlja automatski (slanje obavijesti, arhiviranje i slično) proces, odnosno aktivnost mora biti obavljena od strane neke osobe, to jest korisnika sustava. Osoba koja obavlja aktivnost zvat će se izvršitelj. U definiciji procesa može se eksplicitno odrediti izvršitelj ili se može odrediti pripadnost grupi kojoj izvršitelj mora pripadati, pri čemu se izvršitelj bira naknadno.

2.1.8. Generički obrazac
Općenito, u nekom sustavu za protok poslova mogu se koristiti različite vrste dokumenata. U većini slučajeva radi se o tipiziranim dokumentima koji predstavljaju razne obrasce definirane formalnim aktom (primjerice zakonskim rješenjem ili uredbom organa državne uprave) ili obrasce standardizirane i oblikovane unutar pojedine tvrtke (primjerice obrazac za evidenciju radnog vremena). Ukoliko se podaci pohrane odvojeno, a dizajn primijeni tek na završnu verziju dokumenta, to jest na onu koja bi se arhivirala u papirnatoj verziji, moguće je odgovarajućim shemama opisati strukturu svakog od tih obrazaca, a obrascima rukovati za tu svrhu posebno kreiranim univerzalnim uređivačem obrazaca. Definiranje shema dokumenata može se obaviti koristeći neki od općenitih jezika (primjerice XML), čime se dobivaju generički obrasci. Na jednom popunjene podatke mogu se primijeniti predlošci za stil što bi rezultiralo željenim završnim izgledom.

2.2. Općeniti implementacijski model
Workflow Management Coalition (u daljnjem tekstu WfMC) je 1995. godine u dokumentu TC00-1003 [5] definirao referentni model za poslovne procese te između ostalog i općeniti implementacijski model koji se može vidjeti na sljedećoj slici (Slika 2.1).



Slika 2.1 Generička struktura sustava za upravljanje protokom poslova [5]

Mogu se razlučiti tri tipa funkcionalnih komponenti:

- softverske komponente označene tamnom bojom koje pružaju podršku različitim funkcijama unutar sustava za upravljanje protokom poslova
- različiti tipovi sistemskih definicija i kontrolnih podataka (bijele boje) koji se koriste od strane jedne ili više softverskih komponenti
- aplikacije i baze podataka za pripadajuće aplikacije (svijetlo siva) koje nisu dio samog sustava za upravljanje protokom poslova, ali su uključene i pozivaju se kao dio cjelokupnog rješenja.

Svaki od navedenih tipova može se podijeliti u manje funkcionalne komponente, a u nastavku slijedi kratki opis onih važnijih.

2.2.1. Alat za definiciju postupka

Svrha alata za definiciju postupka (eng. Process Definition Tool) je izrada opisa poslovnog postupka i prijenos u format prilagođen sustavu za upravljanje protokom poslova. Alati za izradu mogu biti različiti i općenito ne moraju biti dio sustava, ali moraju imati odgovarajuće sučelje za razmjenu podataka sa sustavom. Unutar izrade opisa poslovnog postupka, korisnik,

to jest dizajner poslovnog postupka, definirat će redoslijed procesa i uvjete prijelaza između pojedinih procesa, dodijeliti grupe uloga i dozvola pojedinim procesima te, kao što je slučaju rješenja koje ovaj rad opisuje, definirati dokumente koji će se unutar postupka koristiti.

2.2.2. Definicija postupka

Definicija postupka (eng. Process Definition) sadrži sve informacije nužne da bi se postupak mogao pravilno započeti, korektno izvršavati i naposljetku uredno završiti od strane sustava za upravljanje protokom poslova. To su informacije o uvjetima prijelaza iz aktivnosti u neku drugu aktivnost, dokumentima i resursima unutar pojedinog procesa odnosno aktivnosti kao i pravila izvršavanja pojedinih procesa od strane grupe korisnika, što je povezano s podacima iz organizacijskog modela ustanove ili tvrtke u kojoj se poslovni postupak implementira. Format podataka koji opisuje definiciju postupaka može biti različit od sustava do sustava, što je najčešće i slučaj, ali mora biti kompatibilan s odgovarajućim alatom za izradu definicije postupka, to jest za modeliranje postupka.

2.2.3. Jezgra sustava za upravljanje protokom poslova

Element sustava nes(p)retnog prijevoda "servis za odigravanje protoka poslova" (eng. Workflow Enactment Service) je jezgra sustava za upravljanje protokom poslova koja je zadužena za pravilnu interpretaciju definicija poslovnog postupka te pokretanje niza procesa i upravljanje njihovim redoslijedom. Da bi taj posao bio obavljen korektno, brine se jedan ili više pogoniča za upravljanje protokom (eng. Workflow Management Engine).
Tijekom izvršavanja postupka vodi se interna evidencija kontrolnih vrijednosti i stanja pojedinih aktivnosti i prijelaza (eng. internal control data) te se uspostavlja veza s organizacijskom strukturom radi provjere prava izvršavanja i dodjele izvršavanja pojedinog procesa kao i s vanjskim podacima koji mogu utjecati na kontrolu toka unutar samog postupka.
Prema potrebi sustav može inicirati pokretanje vanjskih aplikacija ili, ako to tehnološki nije izvedivo, obavijestiti korisnika o potrebi pokretanja aplikacija. Okolne aplikacije nisu dio jezgre sustava, već samo komuniciraju s jezgrom u dogovorenom formatu, a najčešći primjer takvih aplikaciju su razni editori dokumenata. Jedna takva aplikacija napravljena je i tijekom provedbe ovog projekta u svrhu lakšeg ažuriranja generičkih obrazaca korištenih u postupku kao što je opisano u 6.6.

2.2.4. Relevantni podaci za protok poslova

Na osnovu relevantnih podataka (eng. Workflow Relevant Data) dolazi do odluka prilikom grananja unutar poslovnog postupka. Uvjet prijelaza temeljem kojeg dolazi do odabira slijedećeg procesa, može sadržavati podatke koji su nastali ili su bili promijenjeni tijekom rada procesa. Takvi podaci moraju biti zajednički i jezgri sustava i aplikaciji.

2.2.5. Liste poslova

Radne liste ili liste poslova (eng. Worklists) su popisi poslova, odnosno zadataka koje pojedini korisnik ili grupa korisnika moraju, odnosno mogu izvršiti. Lista poslova se puni i prazni ovisno o nastajanju novih aktivnosti, odnosno završavanjem neke od postojećih. Ovisno o implementaciji programa za upravljanje listom poslova, lista može biti različito prikazana, odnosno može biti u potpunosti skrivena od krajnjeg korisnika.

2.2.6. Korisničko sučelje i program za rad s listom poslova

Program za rad s listom poslova (eng. Worklist Handler) je spona između jezgre sustava za upravljanje protokom poslova i krajnjeg korisnika. Zadatak ove komponente je da se brine oko sučelja za prijavu i odjavu korisnika, da obavijesti korisnika o pojavi novih poslova te da,

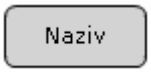
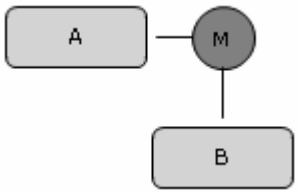
po potrebi i u ovisnosti o tehničkoj izvedbi, pokrene odgovarajuće aplikacije. Prema opisu koji iznosi WfMC ovaj program bi mogao biti proširen tako da sadrži funkcionalnost raspoređivanja poslova unutar iste grupe korisnika kako bi smanjio opterećenje pojedinog korisnika.

2.2.7. Administratorske ovlasti

Administratorske ovlasti (eng. Supervisory Operations) koriste se kada je potrebno izvršiti izmjene unutar jezgre sustava koje nisu vezane za samu definiciju posla, već za preraspodjelu dodijeljenih poslova i pravila za dodjelu, kao i za razna izvješća i pretrage za koje se ukaže potreba, poput praćenja povijesti nekog postupka, upita o statistici protoka poslova, ispisa stanja svih procesa, dodavanja obavijesti o kritičnim dijelovima (uskim grlima) postupka i slično.

3. Predlošci kontrole toka u protoku poslova

U svojoj doktorskoj disertaciji [4] Bartosz Kiepuszewski identificirao je dvadeset predložaka za kontrolu toka protoka poslova, a u nastavku njegovog istraživanja dodan je predložak *Čekanje M od N* čime je definiran skup od 21 standardnog predloška za kontrolu toka protoka poslova.[2]

	Oznaka početka, odnosno završetka postupka
	Način prikaza procesa
	Način prikaza prijelaza (boja i natpis ovise o tipu prijelaza)
	Komentari
	Spojnice koje se nalaze lijevo ili gore su ulazne spojnice u neki element, odnosno spojnice koje se nekom elementu nalaze dolje ili s njegove desne strane su izlazne spojnice iz tog elementa. Element A na slici ima jednu izlaznu spojnicu, a proces B ima jednu ulaznu spojnicu. Prijelaz ima jednu ulaznu spojnicu (ona iz procesa A) i jednu izlaznu spojnicu (prema procesu B)

Tablica 3-1 : Notacija za prikaz elemenata postupka

Preuzeta je notacija prema kojoj se proces crta kao pravokutnik a prijelaz crta kao krug. Ukoliko su nekom elementu na grafu spojnice gore ili lijevo, to znači da su te spojnice ulazne, odnosno, ukoliko su desno ili dolje, to su izlazne spojnice.
Model implementiran u ovom radu pretpostavlja da su spojnicama uvijek spojeni prijelaz i postupak, to jest ne dopušta direktna spajanja dvaju procesa odnosno dvaju prijelaza. Odstupanja su moguća u grafičkom prikazu o čemu će više biti riječi u 3.1.1 te u 6.3.1.
Određene specifičnosti, poput višestrukih instanci istog procesa ili ovisnost o nekoj prekretnici (eng. milestone) bit će označene posebnim simbolom ili bojom, odnosno posebnim znakovima što će biti navedeno u opisu pojedinog predloška[3].
Primjeri koji će biti izneseni u opisu pojedinog predloška nisu vezani uz jedan postupak, već predstavljaju neku tipičnu situaciju u kojoj se dani predložak može primijeniti.
Svi predlošci u svojoj definiciji reguliraju prijelaz među procesima, a stvarni prijelaz u nekom vremenskom trenutku odvija se među aktivnostima koje predstavljaju te procese.

3.1. Osnovni predlošci za kontrolu protoka poslova

Osnovne predloške za kontrolu protoka poslova (eng. Workflow pattern) čine jednostavna grananja i jednostavne sinkronizacije tih grananja.

[2] Neki obrasci (npr. Jednostavno spajanje) su, dijelom zbog svoje naravi, a dijelom uslijed načina implementacije sadržani u nekom drugom obrascu, ali zbog sistematičnost dan je prikaz svih

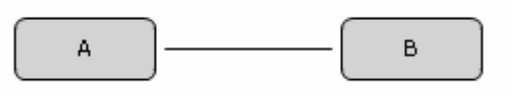
[3] Neki od obrazaca su, po svom dizajnu, stanja sustava (Implicitni završetak) ili neke dopuštene akcije (Akcije odustajanja), pa kao takvi nemaju grafički prikaz što je navedeno u opisu svakog od njih

3.1.1. Slijed

Slijed ili sekvenca (eng. Sequence) je najjednostavniji predložak i predstavlja prijelaz između dvije aktivnosti, gdje se slijedeća aktivnost izvodi odmah nakon završetka one koja joj prethodi. Ne postoje posebni uvjeti prijelaza, a početak slijedne aktivnosti ovisi samo o možebitnim vanjskim okolnostima, kao što su (ne)postojanje resursa i osoba koje mogu započeti tu aktivnost.

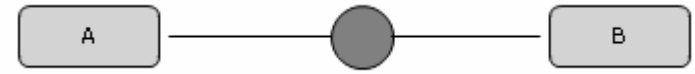
Grafički prikaz:

Prikaz sekvence može se izvesti na dva načina. U grafičkom uređivaču koji se koristi za oblikovanje u procesa u sustavu SUSAP, sekvenca se ne crta, već bilo koja izravna poveznica između dva procesa predstavlja sekvencu.



Slika 3.1 : Slijed

U svrhu očuvanja konzistentnosti modela koji predviđa da se direktno mogu spojiti isključivo prijelaz i proces te se ne dopuštaju druge kombinacije, sekvenca se može grafički crtati kao na donjoj slici:



Slika 3.2 : Interna reprezentacija slijeda

Algoritam analize i raščlambe grafa postupka koji će biti objašnjen u narednim poglavljima (6.3.1 i 6.3.2) brine se da se prikaz iz grafičkog uređivača interpretira onako kako je opisano na prethodnoj slici (Slika 3.2).

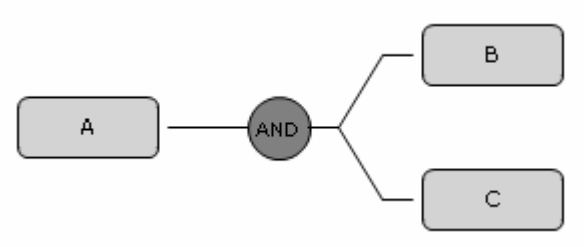
Primjer:

Nakon obavljenog ispravljanja ispita, objaviti rezultate.

3.1.2. Paralelno grananje

Paralelno grananje (eng. Parallel Split, AND-Split, fork, parallel routing) predstavlja prijelaz u kojem nakon završetka aktivnosti koja prethodi ovom prijelazu dolazi do grananja protoka poslova u dvije paralelne grane koje se obje moraju izvesti, a mogu se izvoditi istovremeno.

Grafički prikaz:
Paralelno grananje prikazuje se tamno sivim krugom u kojem je zapisana riječ AND. U prijelaz ulazi luk iz jednog procesa, a izlaznih lukova prema procesima koji se moraju izvršiti može biti dva ili više.

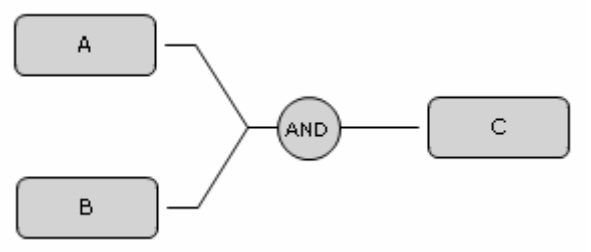


Slika 3.3 : Paralelno grananje

Primjer:
Nakon obavljenog ispravljanja ispita, objaviti rezultate na web stranicama kolegija i poslati podatke zavodskom administratoru.

3.1.3. Sinkronizacija
Sinkronizacija (eng. Synchronization, AND-Join) je prijelaz unutar postupka u kojem se više paralelnih aktivnosti spaja u jednoj točki i nastavak je omogućen tek nakon što sve prethodne aktivnosti završe.

Grafički prikaz:
Kao i ostali predlošci koji predstavljaju spajanje dvaju ili više procesa i ovaj predložak je prikazan svijetlo sivim krugom. Natpis unutar kruga za sinkronizaciju je riječ AND.

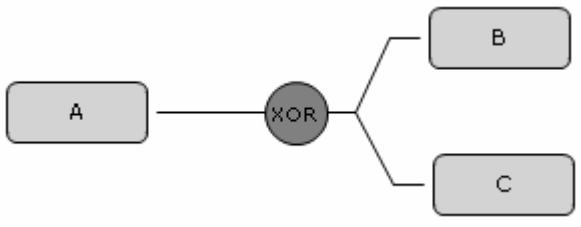


Slika 3.4 : Sinkronizacija

Primjer:
Nakon što je student predao sve domaće zadaće i pristupio svim provjerama znanja, međuispitima i završnom ispitu, slijedi izračunavanje ukupnog broja bodova i ocjenjivanje.

3.1.4. Ekskluzivni odabir
Ekskluzivni odabir (eng. Exclusive Choice, XOR-Split, switch) je prijelaz unutar postupka u kojem se na osnovu parametara sustava i izračunavanja uvjeta prijelaza odabire isključivo jedna od dvije ili više ponuđenih grana.

Grafički prikaz:
Predložak se prikazuje tamno sivim krugom s natpisom XOR. Izlaznih lukova može biti dva ili više.



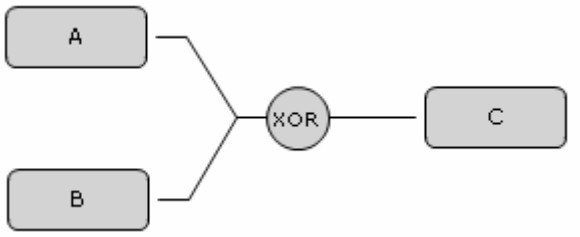
Slika 3.5 : Ekskluzivni odabir

Primjer:
Nakon što mu je izračunata ocjena, student mora odlučiti želi li prihvatiti ocjenu ili želi ići ponovno na ispit.

3.1.5. Jednostavno spajanje

Jednostavno spajanje (eng. Simple Merge, asynchronous join, OR-Join, merge) je prijelaz u kojem se dvije ili više alternativnih grana spaja u jednu točku sinkronizacije. Bitno je naglasiti da se nijedna od grana prethodnica ne izvršava u paraleli, to jest može biti obavljena samo jedna od međusobno isključivih grana. Ovakav predložak se najčešće uparuje s predloškom Ekskluzivni odabir.

Grafički prikaz:
Predložak se prikazuje svijetlo sivim krugom s natpisom XOR. Ulaznih procesa je moglo biti više, a samo je jedan izlazni proces.



Slika 3.6 : Jednostavno spajanje

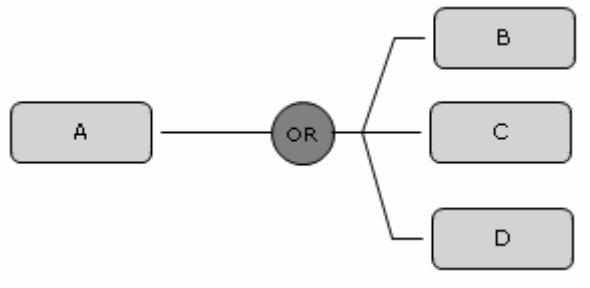
Primjer:
Formiranje konačne ocjene odvija se nakon što je izračunat broj bodova na ponovljenom ispitu, ukoliko je student išao na ponovljeni ispit, ili nakon što je student odabrao opciju u kojoj je izrazio zadovoljstvo inicijalnom ocjenom. Student je mogao odabrati samo jednu od mogućnosti, pa je stoga samo jedna grana bila aktivna.

3.2. Napredno grananje i sinkronizacija

3.2.1. Višestruki izbor

Višestruki izbor (eng. Multi-choice, OR-Split) je prijelaz u kojem se na osnovu stanja unutar postupka i evaluacije izraza za uvjet prijelaza odabire jedna ili više grana koje se mogu izvršavati istovremeno. Aktivnosti koje slijede uvjet prijelaza smiju započeti neposredno nakon što je uvjet evaluiran, a zatim se izvode nezavisno jedna od druge.

Grafički prikaz:
Višestruki izbor prikazuje se tamno sivim krugom s natpisom OR. Samo je jedan ulazni proces, dok izlaznih procesa može biti dva ili više.



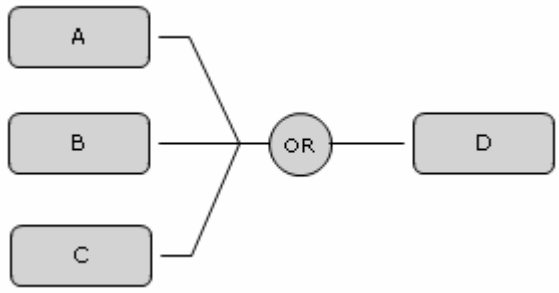
Slika 3.7 : Višestruki izbor

Primjer:
Plaćanje računa u dućanu može se obaviti u cijelosti gotovinski, u cijelosti kreditnom karticom ili djelomično gotovinski a djelomično kreditnom karticom. Uvjet za donošenje odluke o načinu plaćanja može biti iznos raspoloživog gotovog novca.

3.2.2. Sinkronizirano spajanje

Sinkronizirano spajanje (eng. Synchronizing Merge, synchronizing join) je prijelaz unutar postupka gdje se više paralelnih aktivnosti spaja te se aktivnost koja slijedi iza ovog prijelaza odvija tek nakon što se sve prethodne aktivnosti završe. Razlika u odnosu na običnu sinkronizaciju je utoliko što nisu nužno morale biti aktivirane sve grane koje prethode ovom prijelazu (na primjer ukoliko je predložak prethodio višestrukom izboru).

Grafički prikaz:
Sinkronizirano spajanje prikazuje se svijetlo sivim krugom u kojem je natpis OR. Ulaznih procesa može biti više, dok je samo jedan izlazni proces.



Slika 3.8 : Sinkronizirano spajanje

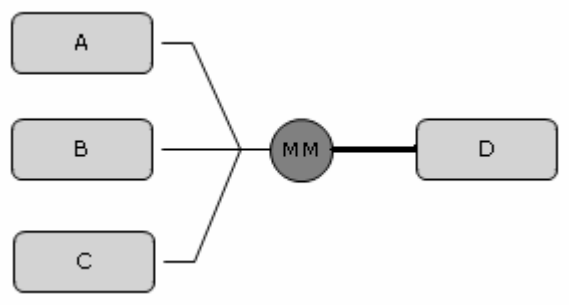
Primjer:
Nakon što su obavljene parcijalne isplate različitim načinima plaćanja, a uplaćen je dovoljan iznos, nastavlja se dalje.

3.2.3. Višestruko spajanje

Višestruko spajanje (eng. Multi-merge) je prijelaz u kojem se dvije ili više grana spaja bez sinkronizacije. Aktivnost koja slijedi iza ovog prijelaza pokrenut će se za svaku završenu aktivnost koja je prethodila ovom prijelazu.

Grafički prikaz:

Višestruko spajanje prikazuje se tamno sivim krugom u kojem se nalazi natpis MM. Ulaznih procesa moglo je biti više, a samo je jedan izlazni proces. Crta koja spoja prijelaz i izlazni proces nacrtana je deblje kako bi se naglasila mogućnost da se izvršava više instanci istog procesa (sljedbenika prijelaza).



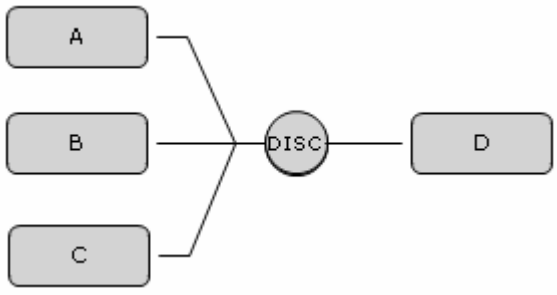
Slika 3.9 : Višestruko spajanje

Primjer:
Nakon svake ispravljene zadaće (A, B, C, …) pokreni evidentiranje ocjene u bazi podataka (D).

3.2.4. Diskriminator

Diskriminator (eng. Discriminator) je prijelaz sličan višestrukom spajanju, ali različit utoliko što ne dolazi do mogućnosti višestrukog izvršavanja aktivnosti koje slijede iza ovog prijelaza. Nakon što završi prva od aktivnosti prethodnica (u vremenskom smislu, ne u smislu poretka), dolazi do izvršavanja aktivnosti koja slijedi iza ovog prijelaza. Završetci preostalih aktivnosti koje prethode prijelazu neće imati utjecaj na daljnji tijek postupka.

Grafički prikaz:
Diskriminator se predstavlja svjetlo sivim krugom u kojem se nalazi natpis DISC. Iz prijelaza izlazi samo jedan proces dok je broj ulaznih procesa mogao biti proizvoljan.



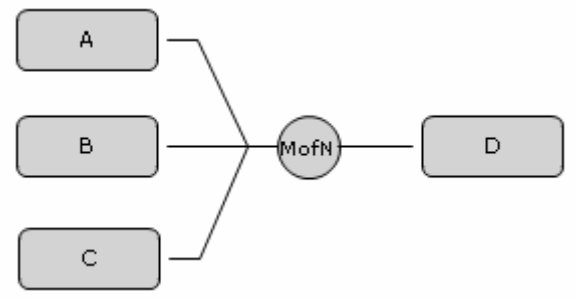
Slika 3.10 : Diskriminator

Primjer:
Nakon što se pojavi prva prijava ispita za određeni rok, započeti aktivnost sastavljanja ispitnih zadataka.

3.2.5. Čekanje M od N

Čekanje M od N mogućih grana (eng. M-out-of-N Join) predstavlja kombinaciju diskriminatora i sinkronizacije. Naime, više grana aktivnosti koje se mogu paralelno izvoditi spaja se u ovaj prijelaz koji, nakon što se završi M aktivnosti od ukupno N mogućih, pokreće aktivnost koja slijedi iza ovog prijelaza. Završetak preostalih N-M aktivnosti ne utječe na kontrolu toka u ostatku postupka.

Grafički prikaz:
Prikaz ovog prijelaza sastoji se od svijetlo sivog kruga u kojem se nalazi natpis M of N. Vrijednost M nije vidljiva na ekranu već se postavlja među svojstvima prijelaza.



Slika 3.11 : Čekanje M od N

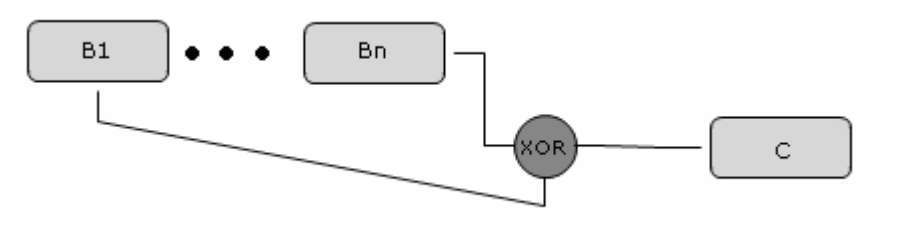
Primjer:
U knjižnicu je isporučena donacija od 5 knjiga koje potencijalno mogu zanimati sve studente. Kriterij dodjele knjige studentu je isključivo na osnovu vremena prijave. Svim studentima je upućen poziv da se prijave za preuzimanje knjiga. Nakon što 5 studenata izrazi želju za preuzimanjem knjige može se prijeći u postupak isporuke knjige, dok se sve ostale želje za knjigama mogu evidentirati, ali ne utječu na dodjelu inicijalnih 5 knjiga.

3.3. Strukturni predlošci

3.3.1. Proizvoljni ciklus

Proizvoljni ciklus je prijelaz unutar postupka u kojem, ovisno o stanju vrijednosti i postupku i uvjetu prijelaza može doći do ponovnog izvođenja niza aktivnosti (i odgovarajućih prijelaza koji ih povezuju) ili se može se prijeći na aktivnost koja bi uobičajeno slijedila iza ovakvog ciklusa.

Grafički prikaz:
Proizvoljni ciklus predstavlja varijantu ekskluzivnog odabira u kojem je jedan od izlaza iz prijelaza Ekskluzivni odabir spojen s nekim prethodnim procesom.



Slika 3.12 : Proizvoljni ciklus[4]

Primjer:
Student na nekom kolegiju tijekom semestra mora ići na međuispite i pisati domaće zadaće (B1 do Bn). Ukoliko na kraju tog postupka ne prikupi dovoljni broj bodova, smatra se da nije položio kolegij te ga mora ponovo upisati i ponoviti navedeni niz aktivnosti idući semestar. Nakon što položi predmet, može prijeći na predmet sljedbenik (C).

[4] Tri točkice na grafu označavaju da se na tom dijelu nalazi 1 ili više procesa koji zbog jednostavnosti slike nisu prikazani. Stvarni crtež sadrži i te procese.

3.3.2. Implicitni završetak

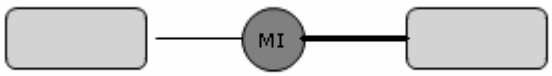
Predložak Implicitni završetak (eng. Implicit Termination) ne može se smatrati prijelazom, već stanjem sustava za protok poslova. Ukoliko u nekom postupku nema aktivnih aktivnosti ili aktivnosti koje tek mogu započeti, potrebno je završiti dani postupak. Ukoliko ovaj predložak ne bi postojao, tada bi sve aktivnosti koje su završne u nekoj grani trebale biti spojene sa završnom aktivnosti uz prethodnu sinkronizaciju. Takav način sastavljanja postupka nepotrebno bi smanjio preglednost grafičkog prikaza i uveo dodatna stanja koja bi trebalo sinkronizirati. Razumljivo, ovaj predložak nema grafički prikaz.

3.4. Predlošci s višestrukim instancama

3.4.1. Višestruke instance bez sinkronizacije

Ovaj predložak (eng. Multiple Instances Without Synchronization) predstavlja prijelaz unutar postupka u kojem se proces koji slijedi može pokrenuti u više instanci. Broj instanci može biti unaprijed poznat (prilikom dizajna postupka) ili se može odrediti u izrazu prijelaza na osnovu stanja vrijednosti dokumenata u postupku. Pokrenute instance (aktivnosti) pojedinog procesa, međusobno su neovisne i nema potrebe za njihovom sinkronizacijom.

Grafički prikaz:
Ovaj prijelaz se prikazuje tamno sivim krugom u kojem se nalazi natpis MI. Proces koji slijedi iza ovog prijelaza spojen je debelom crtom kako bi se istaknula činjenica da se izvršavanje procesa odvija u više paralelnih niti, to jest da se pokreće više instanci istog procesa.

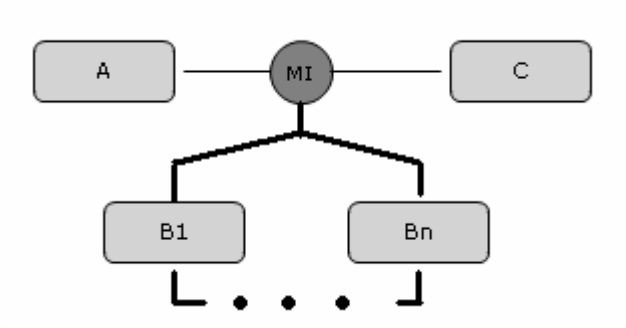


Slika 3.13 : Višestruke instance bez sinkronizacije

Primjer:
Upis izrade diplomskog rada na nekom roku ne rezultira dodatnom sinkronizacijom diplomskih radova.

3.4.2. Višestruke instance sa sinkronizacijom

Slika 3.14 prikazuje način prikaza sve tri inačice višestrukih instanci sa sinkronizacijom. Prijelaz se prikazuje tamno sivim krugom s natpisom MI, a dio grafa koji predstavlja procese koji se izvode u više instanci spojen je debelim crtama. Proces koji će se obaviti nakon sinkronizacije spojen je običnom crtom. Parametri prijelaza koji određuju o kojoj inačici se radi, postavljaju se u svojstvima prijelaza.



Slika 3.14 : Višestruke instance sa sinkronizacijom[5]

Višestruke instance s unaprijed poznatim brojem instanci

Ukoliko je za neki proces potrebno pokrenuti nekoliko instanci koje treba sinkronizirati, a njihov broj je unaprijed poznat prilikom dizajniranja poslovnog postupka, takvo rješenje dano je kroz višestruke instance gdje je broj instanci poznat prilikom dizajna (eng. Multiple Instances With a Priori Design Time Knowledge). Ovaj predložak po svom izvođenju je sličan paralelnom grananju i odgovarajućoj sinkronizaciji, ali obzirom na to da se ne radi o različitim procesima već o instancama jednog procesa, što podrazumijeva neke dijeljene resurse, i implementacijski se razlikuje od ciklusa, potrebno je definirati zasebni predložak. Nakon što sve pokrenute instance procesa završe, potrebno je pokrenuti slijedeći proces u postupku.

Primjer:
Unaprijed je poznato da će se obaviti 3 zadaće i 6 kontrolnih provjera znanja (kratkih testova). Prije formiranje konačne ocjene potrebno je pričekati završetak zadaća i provjera znanja.

Višestruke instance s brojem instanci poznatim u trenutku izvođenja

Ovaj predložak (eng. Multiple Instances With a Priori Runtime Knowledge) koristit će se onda kad je broj instanci nekog procesa poznat tek nakon što se postupak nađe u prijelazu koji prethodi procesu za kojeg se pokreće više instanci. Broj instanci će se izračunati na osnovu uvjeta prijelaza i stanja unutar postupka. Nakon što sve pokrenute instance procesa završe, potrebno je pokrenuti sljedeći proces u postupku.

Primjer:
Broj grupa (termina, nastavnika i dvorana) za održavanje pismenog ispita ili predavanja poznat je tek po obračunu broja polaznika.

Višestruke instance bez poznatog broja instanci

Ukoliko je za neki proces potrebno pokrenuti više instanci, a da broj tih instanci nije poznat niti u trenutku dizajna niti u trenutku pokretanja prve instance, tada će se koristiti ovaj predložak (eng. Multiple Instances Without a Priori Runtime Knowledge). O tome da li će se pokrenuti dodatna instanca odlučuje se u trenutku pokretanja svake od instanci i to na osnovu

[5] Tri točkice na grafu označavaju da se na tom dijelu nalazi 1 ili više procesa koji zbog jednostavnosti slike nisu prikazani. Stvarni crtež sadrži i te procese, a njihov redoslijed izvršavanja je od B1 do Bn. Sljedbenik procesa Bn je prijelaz s oznakom MI, ali radi preglednosti slike, ulazna i izlazna spojnica iz Bn nisu prikazane u skladu s dogovorenom notacijom koja predviđa da se ulazne spojnice nalaze gore ili lijevo, a izlazne spojnice desno ili dolje.

uvjeta prijelaza i stanja vrijednosti unutar postupka. Nakon što su sve instance pokrenute i završene, pokreće se proces koji slijedi iza ovog prijelaza. Napomena: u predlošku je moguće zadati minimalni, odnosno inicijalni broj instanci koje se moraju pokrenuti pri pokretanju prijelaza definiranim ovim predloškom.

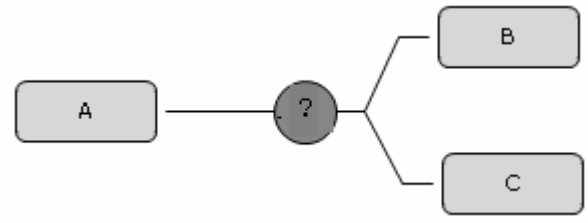
Primjer:
Ponovno skeniranje formulara s odgovorima na razredbenom ispitu obavlja se onoliko puta koliko je potrebno, ovisno o rezultatu pojedinačnog skeniranja.

3.5. Predlošci stanja

3.5.1. Neposredni izbor

Neposredni izbor (eng. Deferred Choice, implicit choice, deferred XOR-Split) je prijelaz unutar postupka u kojem korisnik mora izabrati jednu od dvije ili više alternativa. Za razliku od ekskluzivnog odabira, gdje je izbor alternative ovisio o uvjetu prijelaza i vrijednostima unutar postupka, kod neposrednog odabira ne vrši se automatski izbor, već izbor grane vrši isključivo korisnik.

Grafički prikaz:
Ovaj predložak prikazuje se tamno sivim krugom u kojem se nalazi upitnik. U prijelaz ulazi samo jedan proces dok je izlaznih proizvoljan broj.



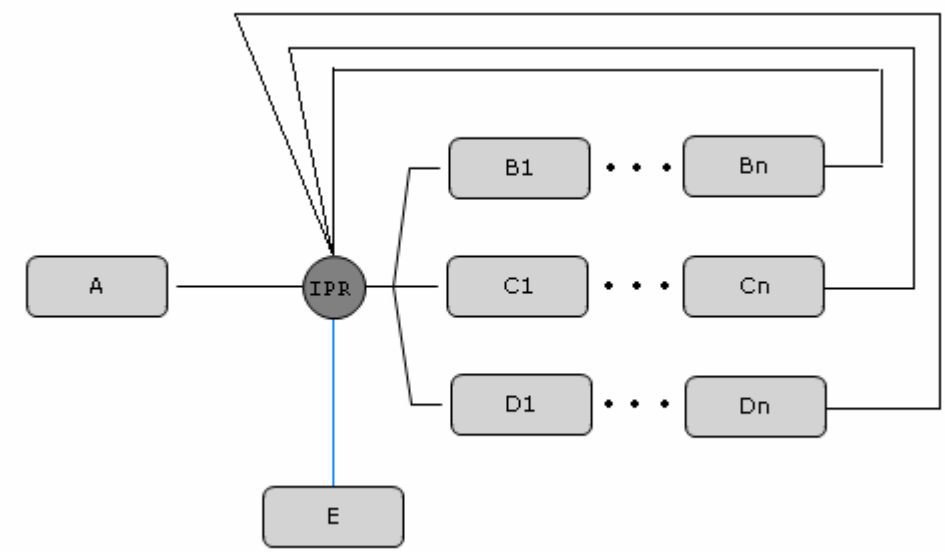
Slika 3.15 : Neposredni izbor

Primjer:
Ovisno o rezultatima nalaza, liječnik odabire jednog ili više kolega od kojih traži drugo mišljenje.

3.5.2. Naizmjenično paralelno izvođenje

Predložak Naizmjenično paralelno izvođenje (eng. Interleaved paralled routing, Interleaved sequence, Unordered sequence) se koristi ukoliko niz procesa treba izvršiti proizvoljnim redoslijedom, ali ne istovremeno. Nakon što se niz aktivnosti izvrši, može se izvršiti proces koji slijedi iza ovog prijelaza.

Grafički prikaz:
Predložak naizmjeničnog paralelnog izvođenja prikazuje se tamno sivim krugom u kojem se nalazi natpis IPR. Procesi koji se moraju naizmjenično izvesti (na slici su to nizovi procesa B1 do Bn, C1 do Cn i D1 do Dn) spojeni su s prijelazom običnom spojnicom. Proces koji se mora izvesti nakon završetka prijelaza označen je spojnicom drugačije boje (spojnica na slici je plave boje i povezuje prijelaz i proces E)



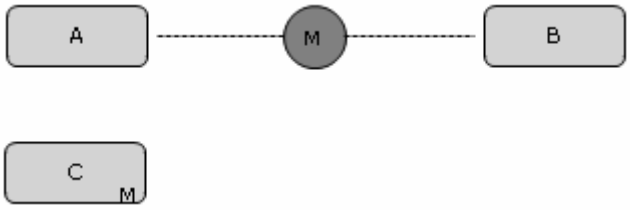
Slika 3.16 : Naizmjenično paralelno izvođenje[6]

Primjer:
Da bi upisao određene kolegij, student mora položiti sve predmete preduvjete. Student istovremeno može polagati samo jedan predmet, a redoslijed polaganja predmeta je proizvoljan.

3.5.3. Prekretnica

Prekretnica (eng. Milestone) je prijelaz unutar postupka koji aktivira mogućnost izvođenja nekog procesa koji je ovisan o stanju nekih drugih procesa.

Grafički prikaz:
Prekretnica se označava tamno sivim krugom u kojem se nalazi upisano slovo M. Obzirom da proces ovisan o nekoj prekretnici u grafičkom prikazu ne mora biti blizu prekretnice, proces i prekretnica ne povezuju se direktno, već se ovisnost označava slovom M u donjem desnom kutu procesa, a odabir prekretnice o kojoj proces ovisi obavlja se u svojstvima procesa. U primjeru sa Slika 3.17 proces C ovisan je o nekoj prekretnici)



Slika 3.17 : Prekretnica

Primjer:
U postupku udaljenog učenja, korisnik može ponavljati neku od cjelina proizvoljan broj puta, a uspješno položen ispit iz neke cjeline predstavlja prekretnicu koja omogućava pisanje sljedeće cjeline.

3.6. Predlošci odustajanja

Zbog svojih specifičnost predlošci odustajanja unutar rješenja u ovom radu ne prikazuju se kao prijelazi unutar postupka niti se crtaju u postupku dizajna, već su dani kao mogućnosti

[6] Tri točkice na grafu označavaju da se na tom dijelu nalazi 1 ili više procesa koji zbog jednostavnosti slike nisu prikazani.

pojedinom korisniku. Korisnik može otkazati izvođenje aktivnosti, nakon čega će sustav izvođenje aktivnosti nekom drugom izvršitelju. Ukoliko ima dovoljne ovlasti korisnik može otkazati čitav postupak.

4. Upravljanje resursima prilikom raspodjele poslova

Unutar svake organizacije korisnici su podijeljeni u grupe ovisno o svojim sposobnostima i radnim mjestima te nadležnostima, a pojedini korisnik može pripadati u više grupa. Prilikom izvršavanja poslovnog postupka svakoj aktivnosti koja zahtjeva korisničku interakciju, potrebno je dodijeliti korisnika koji će obaviti tu aktivnost. Definicija postupka ne određuje poimence tko će izvršiti pojedini proces, nego definira kojoj grupi korisnika će pripadati taj korisnik. Pri raspodjeli poslova dolazi do klasičnog problema kako optimalno razdijeliti poslove zaposlenicima korisnicima vodeći računa o sljedećim činjenicama:

- Iskoristiti pojedinačnu specijalnost korisnika.
- Poželjno je da korisnik obavlja slične zadatke u nizu da bi se povećala učinkovitost djelatnika i djelotvornost postupka.
- Potrebno je osigurati elastičnost u bliskoj budućnosti, što znači da ukoliko dvije ili više osoba mogu odraditi isti posao, razumno je odabrati onu koja ima manju stručnost ili kompetencije kako bi se s većom vjerojatnošću moglo obaviti više nadolazećih poslova.
- Ravnomjerno rasporediti količinu poslova na sve zaposlenike.

Obzirom na način raspodjele poslova postoji nekoliko pristupa. Sama dodjela poslova može biti inicirana od strane sustava za upravljanje (tzv. push-driven), čime je nekom korisniku dodijeljen posao. Alternativno, svaki korisnik može izabrati svoj posao iz liste mogućih poslova (pull-driven). Također, moguće su i situacije u kojima zadatak dodijeljen nekom korisniku treba delegirati drugom korisniku ili na isti posao treba dodati jednog ili više novih izvođača.

4.1. Strategije raspodjele poslova

4.1.1. Direktna dodjela posla

Tijekom kreiranja definicije dizajner postupka određuje osobu (ulogu) koja će izvršiti određenu aktivnost. Prednost ovakvog pristupa je u tome što se obavljanje potencijalno kritičnih poslova može povjeriti odabranim i provjerenim ljudima, dok su nedostatci u tome da konkretna osoba u određenom trenutku može biti zauzeta obavljanjem drugih aktivnosti ili odsutna na neko vrijeme, čime se izvršavanje zadane aktivnosti odgađa na neko vrijeme, zbog čega cijeli postupak kasni u izvršavanju. Ukoliko zadužena osoba napusti tvrtku, postupak mora biti ažuriran da bi se mogao obaviti s obzirom na novonastale okolnosti, to jest zadana se aktivnost mora povjeriti nekoj drugoj osobi ili rasporediti po nekom drugom predlošku.

4.1.2. Dodjela na osnovi pripadnosti grupi

Eksplicitno određivanje osobe koja će izvršiti posao najčešće nije dobro rješenje iz već navedenih razloga, pa se ponekad kao bolje rješenje pokazuje specificiranje potencijalne grupe korisnika koji mogu izvršiti pojedini zadatak. Sustav za upravljanje protokom će u danom trenutku odrediti jednu osobu iz zadane grupe i dodijeliti joj obavljanje zadatka. Grupe mogu biti formirane na osnovu uloga ili na osnovu organizacijske strukture unutar organizacije.

Samo određivanje osobe unutar grupe može se obaviti nekim od za to namijenjenih algoritama, primjerice RoundRobin algoritmom.

4.1.3. Odvajanje poslova

Odvajanje poslova je način dodjele posla u kojem se dvije aktivnosti moraju izvoditi od strane više različitih osoba (ne nužno iz različitih grupa). Potreba za ovakvim zahtjevom ukazuje se u slučajevima kad je unutar nekog poslovnog postupka potrebno proći dvostruku autorizaciju, provjeru, recenzije i slično, koje naravno, moraju obaviti različite osobe.

4.1.4. Kontinuirano rukovanje slučajem

Za razliku od prethodnog slučaja kada se za dvije aktivnosti trebaju izabrati dvije različite osobe, ponekad je bolje i efikasnije zadržati istu osobu koja je već upoznata s prethodnim aktivnostima što je preduvjet za brže izvođenje zadane aktivnosti. Ukoliko osoba kojoj je povjeren zadatak nije raspoloživa u trenutku kreiranja aktivnosti moguće je odabrati između dvije opcije:

- pridijeliti joj izvođenje zadane aktivnosti, pa pričekati dok ne dođe na red za izvođenje te aktivnosti, čime se uzrokuje kašnjenje cijelog postupka
- dodijeliti zadatak nekoj drugoj osobi, koja ima ovlasti izvoditi zadanu aktivnost, ukoliko je procjena da je vrijeme čekanja na prvu osobu veće od vremena prilagođavanja nove osobe na slučaj.

4.1.5. Dodjela na osnovu praćenja povijesti

Situacije kod kojih bi se kontinuirano rukovalo slučajevima bile bi idealne s aspekta vremena utrošenog na pojedinu aktivnost, ali bi vrijeme trajanja čitavog postupka moglo previše narasti zbog potencijalno velikog vremena čekanja,. Varijantu da se u tom slučaju aktivnost odmah dodijeli nekoj drugoj osobi, moguće je poboljšati praćenjem povijesti obavljenih postupaka svake od osoba i odabrati onu koja primjerice ima najveći postotak uspješno obavljenih sličnih poslova ili, s ciljem ravnomjernije razdiobe posla, odabrati onu koja je imala najmanje poslova u nekom vremenskom razdoblju.

4.1.6. Raspodjela na osnovu slučajnog odabira

Iako najmanje osjetljiva u pogledu kompetencija odabranog izvođača i njegove količine posla, raspodjela na osnovu slučajnog odabira pogodna je u situacijama kada se izvođač posla mora odrediti nasumce.

4.1.7. Dodjela Round Robin algoritmom

Round Robin algoritam omogućava cikličku raspodjelu aktivnosti po osobama unutar neke kategorije, primjerice unutar iste grupe. Prednost ovog algoritma dodjele je ravnomjerna izmjena resursa po poslovima, iako u slučaju velikih odstupanja u vremenima izvršavanja pojedinih aktivnosti može doći do većih razlika u količinama ukupno utrošenog vremena, ali statistički gledano, na velikoj količini raspodijeljenih poslova sve osobe iz iste kategorije bi trebale imati ravnomjerni utrošak vremena.

4.1.8. Dodjela na osnovu reda čekanja

Ukoliko trajanja pojedinih aktivnosti mogu znatnije varirati, a priljev poslova je takav da se stvaraju čekanja onda je razumno posao dodijeliti onom resursu koji ima najmanji red čekanja. Naravno, kod ovog algoritma razumno je za pretpostaviti da sve osobe svoje poslove obavljaju najbolje što mogu. Na ovaj način broj obavljenih aktivnosti može varirati, ali vrijeme ukupnog izvođenje je približno jednako za sve članove pojedine grupe.

4.1.9. Vremenski ovisna raspodjela poslova

Ukoliko se izbaci direktna raspodjela poslova određenoj osobi, za sve ostale vrste dodjela nameće se pitanje u kojem trenutku odabrati osobu na osnovu nekog kriterija. Mogu se razlikovati tri mogućnosti:

- ***Dodjela odmah pri početku cijelog postupka***: Na početku postupka sustav raspodjeljuje zadatke po pojedinim osobama. Ovaj način je pogodan ukoliko postupak ima duže vrijeme trajanja te se radi procjena vremenskih opterećenja pojedinih osoba. Inicijalnom dodjelom poslova na početku postupka moguće je načiniti detaljni plan rada za pojedinu osobu.
- ***Dodjela unaprijed, prije stvaranja aktivnosti***: Na osnovu statističkog predviđanja ili analize tijeka postupka, sustav može ponuditi, odnosno pridijeliti korisniku aktivnost koja će potencijalno biti stvorena nakon određenog vremena. Da li će aktivnost biti stvorena ovisi o razvoju radnog toka.
- ***Dodjela neposredno u trenutku stvaranja***: U nekom vremenskom trenutku aktivnost je stvorena te je pridružena određenom izvršitelju, koji ne mora nužno odmah započeti rad na aktivnosti.

4.2. Preuzimanje posla

Ukoliko inicijativa raspodjele posla kreće od izvršitelja, onda je riječ o tzv. preuzimanju posla (eng. Pull Pattern). Definicija procesa sadrži podatak o grupi korisnika kojoj konkretni izvršitelj procesa mora pripadati. U načinu raspodjele preuzimanjem, nakon kreiranja pojedinačne aktivnosti, tako stvorena aktivnost je vidljiva svim potencijalnim izvršiteljima. Prvi korisnik koji na sebe preuzme zadaću izvršavanja aktivnosti mora istu i završiti. Izvršitelj, međutim, ne mora odmah započeti izvršavanje aktivnosti, pa se zbog toga može reći da je dani korisnik napravio rezervaciju aktivnosti. Rezervirana aktivnost pojavit će se u listi poslova konkretnog korisnika. Ukoliko pojedina osoba definira kriterije za automatsko prihvaćanja poslova, onda se ovaj način preuzimanja posla svodi na neki od oblika direktne raspodjele.

Preuzimanje posla je situacija koja je prikladna za, primjerice, službe za korisnike, službe za obradu podataka i slično, ali odgovornost raspodjele posla leži na svakom od korisnika pojedine grupe.

4.3. Preraspodjela posla

Prilikom raspodjele posla jedno od ključnih pitanja je što napraviti s dotada učinjenim poslom, to jest s podacima koji su nastali do tog trenutka. S obzirom na to da su procesi koncipirani tako da predstavljaju male logičke cjeline posla, odnosno da se radi o uhodanim procedurama, razumno je pretpostaviti da je bolje rješenje zadržavanje postojećih podataka unutar dokumenata koji se koriste u procesu.

4.3.1. Delegiranje posla
Delegiranje posla je mogućnost da pojedina osoba proslijedi svoj posao nekoj drugoj osobi. Pri tome treba imati u vidu kompetencije osobe za raspodjelu, to jest da konkretna osoba u praksi može svoj posao proslijediti suradnicima samo u okviru svojih nadležnosti.

4.3.2. Eksplicitno oduzimanje posla
Oduzimanje posla je mogućnost da se, iz opravdanih razloga, nekoj osobi povjereni posao oduzme i izvršavanje povjeri nekoj drugoj. Može se svesti na delegiranje posla od strane djelatnika nadređenog izvršitelju.

4.3.3. Automatsko oduzimanje posla
Automatsko oduzimanje posla je pojava da sustav za upravljanje protokom poslova pojedinoj osobi oduzme izvršavanje pojedine aktivnosti ukoliko je vrijeme izvršavanja prešlo kritičnu točku i povjeri ga nekoj drugoj osobi. Problem mogu biti procjene vremena izvršavanja pojedinih zadataka ukoliko nema sličnih zadataka u zabilježenoj povijesti sustava.

4.4. Predlošci vidljivosti
Unutar svakog informacijskog sustava postavlja se pitanje javnosti pojedinih podataka, pa tako i unutar dodjele poslova. Pretpostavka ove kategorije predložaka je da postoji predložak koji propisuju tko će sve moći vidjeti koji su neraspodijeljeni poslovi (eng. Configurable Unallocated Work Item Visibility), odnosno predložak koji specificira vidljivost već dodijeljenih poslova (eng. Configurable Allocated Work Item Visibility). Potonja mogućnost je sasvim razumna ukoliko se primjerice radi o anonimnim recenzijama projekata i slično, dok u nekim drugim slučajevima kad je rad grupe javan, zbog lakše suradnje i praćenja statusa nekih slučajeva ima smisla proglasiti raspodjelu posla javnom.

4.5. Istovremeno izvršavanje zadataka i višestruki resursi
Ukoliko je riječ o ljudskim resursima, čovjek se u pojedinom diskretnom intervalu vremena uvijek bavi jednim zadatkom, odnosno jednim slučajem. Međutim, u praksi, moguće je da je pojedinoj osobi dodijeljeno više poslova koje ona mora obaviti u nekom roku. Ponekad zbog duljine tih zadataka ili zbog osobne volje osobe koja ih izvršava, svi slučajevi mogu biti započeti i osoba ih može izvršavati naizmjenično. U tu svrhu mora postojati predložak koji omogućava istovremeno izvršavanje (eng. Simultaneous Execution) dviju ili više aktivnosti.
Ukoliko pak osoba procijeni da joj je potrebna pomoć oko izvršavanja zadatka, ili je već inicijalno postojala potreba da neki posao obavlja više osoba, potrebno je implementirati predložak koji bi predviđao zahtjev za dodatnim resursima (eng. Additional Resources).

4.6. Ugrađeni predlošci raspodjele
Razmotre li se navedeni predlošci lako se može primijetiti da su neki od predložaka međusobno nekompatibilni, to jest isključiva raspodjela dodjela posla po nekom od navedenih predložaka (npr. raspodjela na osnovu slučajnog odabira) isključila bi mogućnost postojanja ostalih.

Unutar sustava za protok poslova implementiranog u sklopu ovog projekta korištena je kombinacija nekoliko navedenih predložaka. Prilikom izrade postupka dizajner postupka najčešće određuje grupu kojoj izvršitelj nekog procesa mora pripadati, a sam izbor izvršitelja prepušta se na izbor samim korisnicima sustava, to jest prakticira se preuzimanje posla, odnosno pull-pattern princip opisan u 4.2.

Omogućavanje direktnog odabira osobe (4.1.1), izazvalo bi promjenu modela baze podataka korištenog unutar sustava, pa je za takve situacije predviđeno zaobilazno rješenje koje ne narušava model i omogućava primjenu opisanog predloška, na način da se definiraju grupe korisnika koje se sastoje od samo jednog korisnika. Očigledni nedostatak ovog pristupa je inflacija broja grupa korisnika, ali uz pretpostavku da se ovakva dodjela rijetko koristi, manja odstupanja od prvotne ideje i neznatno povećanje broja grupa je prihvatljiva opcija.

Odvajanje poslova (4.1.3) i kontinuirano rukovanje slučajem (4.1.4) podržani su proširenjem modela što je opisano u 6.1.3, uz opasku da je odvajanje poslova moguće samo ukoliko je broj članova dane grupe veći od jedan. U protivnom postupak će se naći u stanju zastoja, sve dok se broj članova grupe ne poveća ili dok administratori sustava eksplicitno ne odrede nekog drugog izvršitelja.

Mogućnost promjene izvršitelja, koja je omogućena administratorima svakako nije poželjna akcija, jer se tim činom proces automatizacije poslovanja znatno umanjuje i izaziva utrošak nečijeg radnog vremena, ali ipak mora postojati zbog situacija u kojima se sustav, kao što je u prethodno navedeno, zbog lošeg dizajna ili promjene stanja ljudskih resursa dođe u stanje zastoja. Ovom mogućnosti implementiraju se predlošci Eksplicitnog oduzimanja posla i Direktne dodjele posla.

Predlošci vidljivosti definiraju se prilikom kreiranja dokumenata, gdje se specificiraju procesi i grupe osoba koje smiju čitati, odnosno mijenjati sadržaj pojedinog dokumenta u nekom određenom trenutku što može biti definirano XML (Extensible Markup Language)[29] dokumentom.

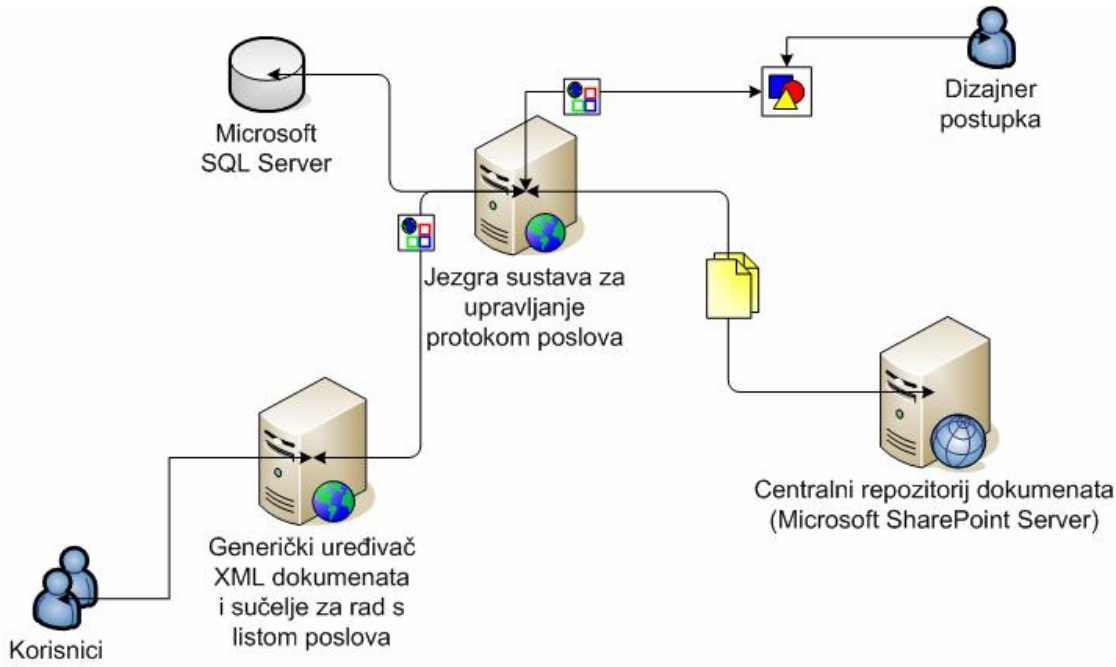
Predlošci koji rade dodjelu na osnovu praćenja povijesti, u direktnoj su suprotnosti sa principom povlačenja posla, a zbog predviđenog malog broja direktnih dodjela, nije bilo potrebe implementirati ih.

Kod predložaka s višestrukim resursima, istovremeno izvršavanje je omogućeno po samom dizajnu sustava dok je procjena da je broj slučajeva koji bi zahtijevao više izvršitelja za jedan proces vrlo mali te se specificiranjem glavnog izvršitelja može svesti na postojeće Predloške.

5. Sustav za automatizaciju poslovanja

U okviru ovog projekta napravljeno je vlastito rješenje pogonskog sklopa (engine) za upravljanje protokom poslova koje podržava predloške navedene u 3. poglavlju. U ovom poglavlju detaljnije je opisana ugradnja predložaka te je prikazan model relacijske baze podataka korištene za ugradnju predložaka.

Slika 5.1 prikazuje pojednostavljenu arhitekturu načinjenog rješenja.



Slika 5.1 : Arhitektura sustava za upravljanje protokom poslova

Osnovu sustava čini jezgra sustava za upravljanje protokom poslova. Potrebna komunikacija s jezgrom sustava od strane korisnika i dizajnera sustava obavlja se korištenjem web servisa.

Stvaranje novih postupaka dizajneru postupka je omogućeno korištenjem aplikacije za grafičko uređivanje postupaka. Aplikacija omogućava stvaranje novih te dohvat i promjenu postojećih postupaka.

Podaci o postupcima, korisnicima i dokumentima pohranjuju se u relacijskoj bazi podataka.

Sadržaj dokumenata pohranjen je u centralnom repozitoriju dokumenata, a u rješenju za centralni repozitorij je odabran SharePoint Server koji u svojoj funkcionalnosti ima mogućnost pohrane verzija dokumenata.

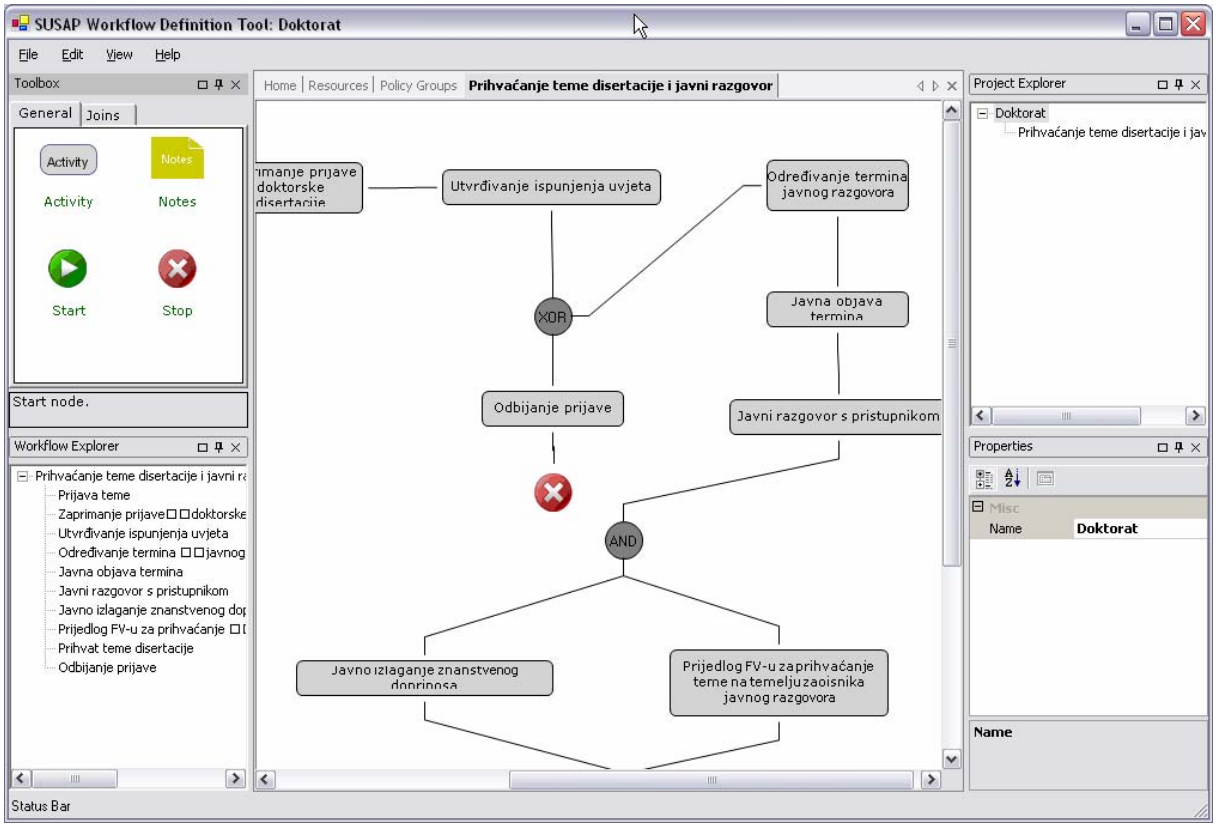
Interakcija s korisnicima odvija se putem web sučelja koje u sebi ima ugrađen grafički uređivač generičkih obrazaca pohranjenih u obliku XML dokumenata.

Navedeno korisničko sučelje komunikaciju s jezgrom sustava ostvaruje putem web servisa.

5.1. Grafički uređivač protoka poslova

Uređivanje protoka poslova pri definiranju radnih postupaka obavlja se u okviru ovog projekta posebno napravljenom aplikacijom. Program omogućuje lokalno (samostalno) definiranje protoka poslova uz naknadnu sinkronizaciju sa poslužiteljem.

Korisničko sučelje aplikacije je napravljeno po uzoru na razvojne alate i omogućuje istovremeni rad na jednom projektu koji može sadržavati više postupaka. Omogućen je paralelno uređivanje više radnih tokova istovremeno te korištenje međusobnih veza između postupaka (ugniježđeni postupci).

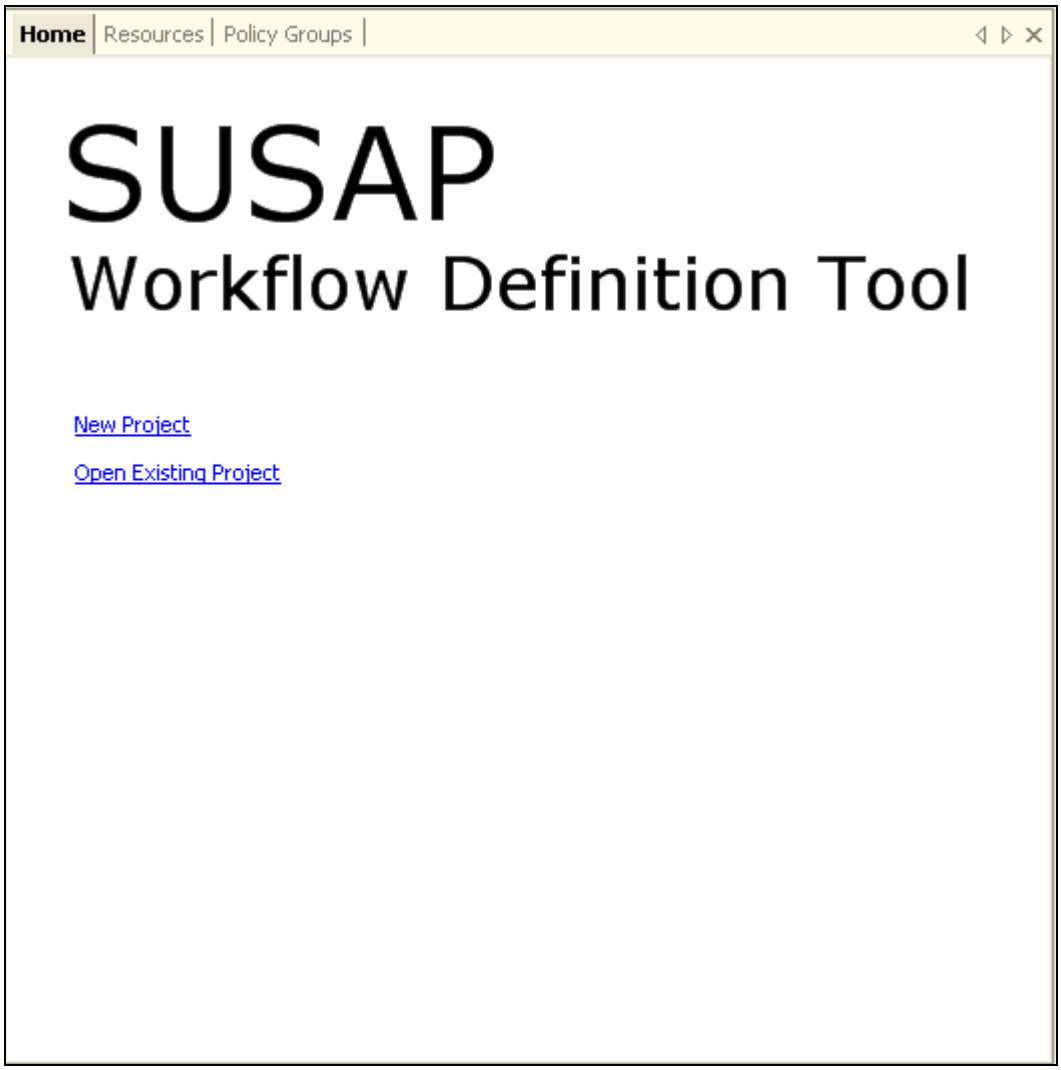


Slika 5.2 : Grafički uređivač protoka poslova

Elementi uređivača opisani su u narednim poglavljima.

5.1.1. Početna stranica

Nakon pokretanja aplikacije otvara se početni prozor koji sadrži opcije za kreiranje novog projekta i otvaranje postojećeg projekta.

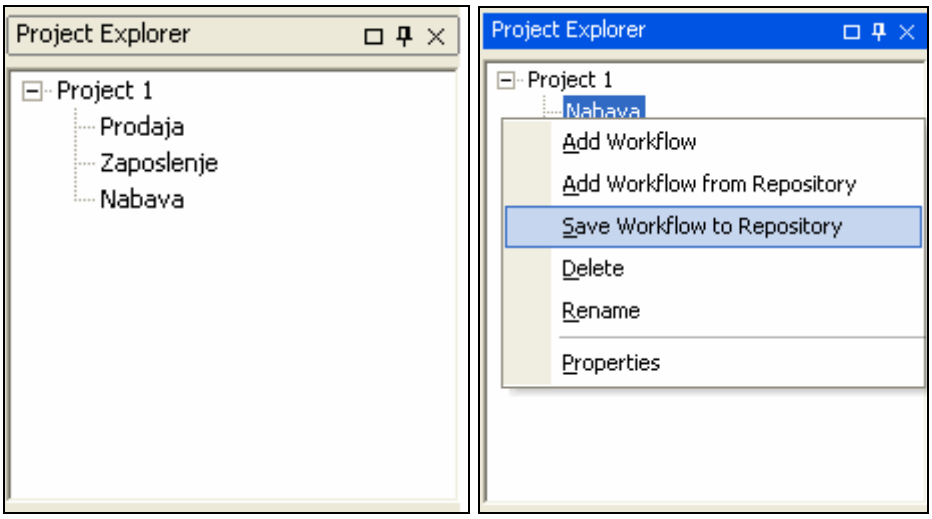


Slika 5.3 : Početna stranica

5.1.2. Prozor za navigaciju projektom (Project Explorer)

Prozor za prikaz projekta omogućuje pregled kompletnog projekta u stablastoj strukturi te je putem njega moguće kreirati nove, otvarati postojeće i brisati pojedine postupke. Za sve ove funkcije koristi se izbornik ovisan o kontekstu.

Kontekstni izbornik sadrži i opcije za pohranu postupaka na poslužitelj i učitavanje postupaka s poslužitelja.

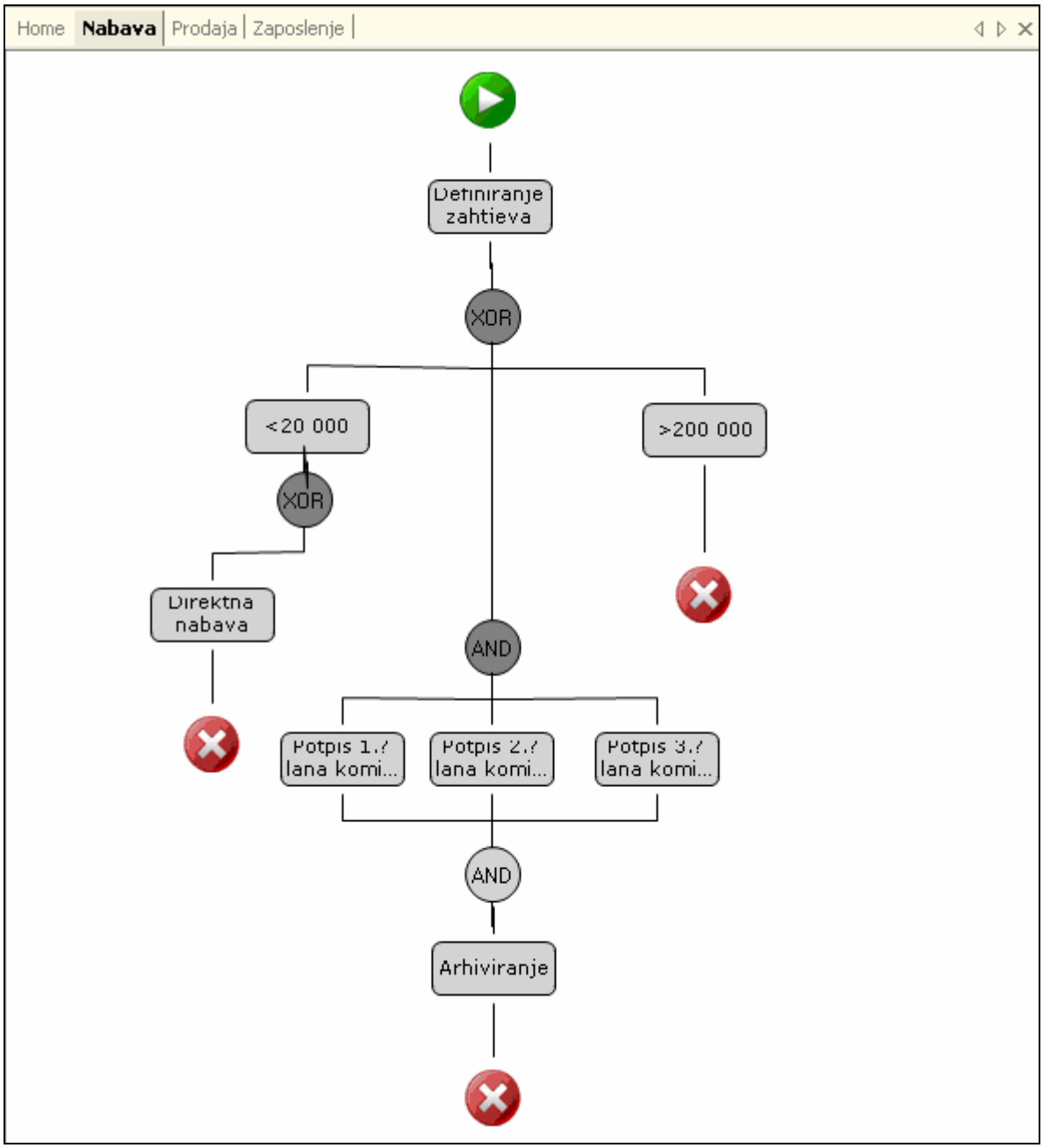


Slika 5.4 : Navigacija projektom i izbornik ovisan o kontekstu

5.1.3. Uređivanje postupka

Središnji panel za rad s postupkom je osnovni element sučelja i prikazuje protok poslova. Moguće je u istom trenutku imati otvoreno više ovakvih elemenata.

Definicija postupka sastoji se od elemenata koji se dovlačenjem (*drag*) iz prozora s elementima postupka dodaju na panel u aktivni protok poslova. Na radnoj površini ti se elementi pozicioniraju i međusobno povezuju. Pomoću prozora za prikaz svojstava omogućeno je postavljenje svojstava trenutno aktivnog elementa postupka.

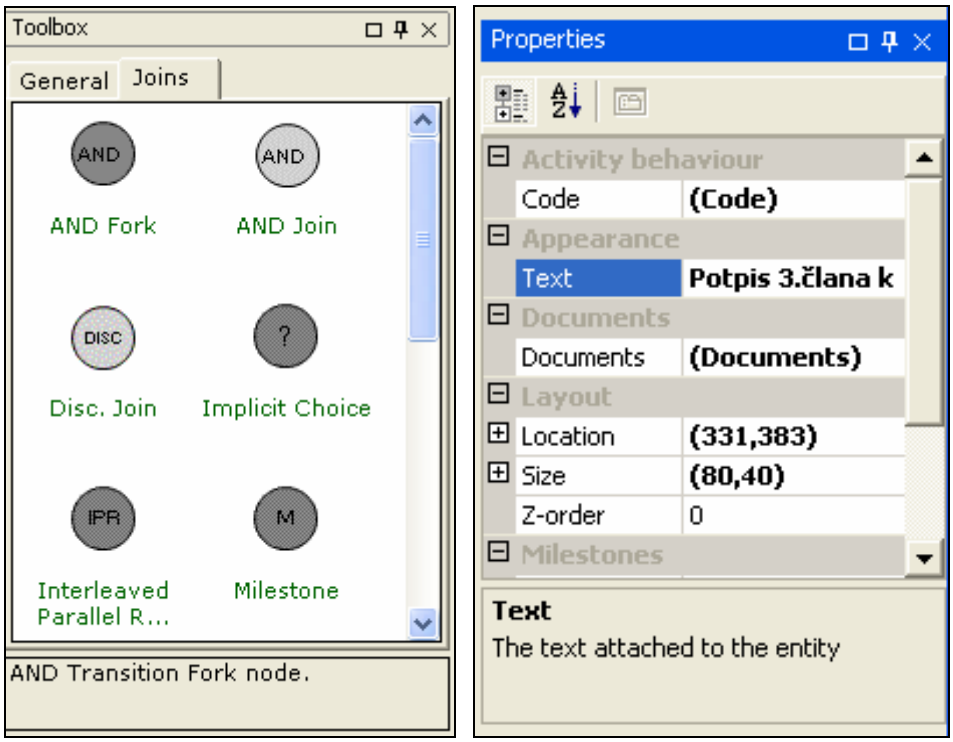


Slika 5.5 : Dijagram protoka poslova pri uređivanju postupka

5.1.4. Predlošci protoka poslova i uređivanje svojstava

Prozor s predlošcima protoka poslova (alatnica) sadrži sve moguće elemente postupka. Posebno su odvojeni osnovni elementi a posebno elementi koji predstavljaju moguće prijelaze. Dovlačenjem na radnu površinu prozora za rad s postupkom pojedinog elementa stvara se struktura postupka.

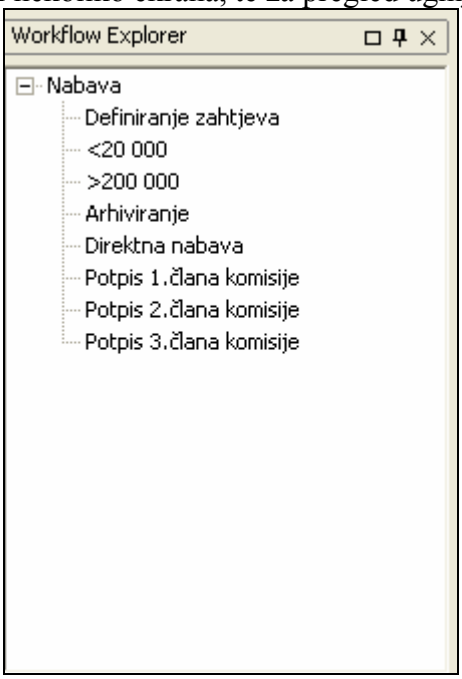
Prozor za uređivanje svojstava omogućuje pregled i izmjenu svojstava pojedinog elementa postupka. Isto tako, pomoću tog prozora moguće je mijenjati svojstva i ostalih elemenata projekta kao primjerice naziv projekta, naziv postupka i slično.



Slika 5.6 : Predlošci protoka poslova i uređivanje svojstava

5.1.5. Struktura postupka

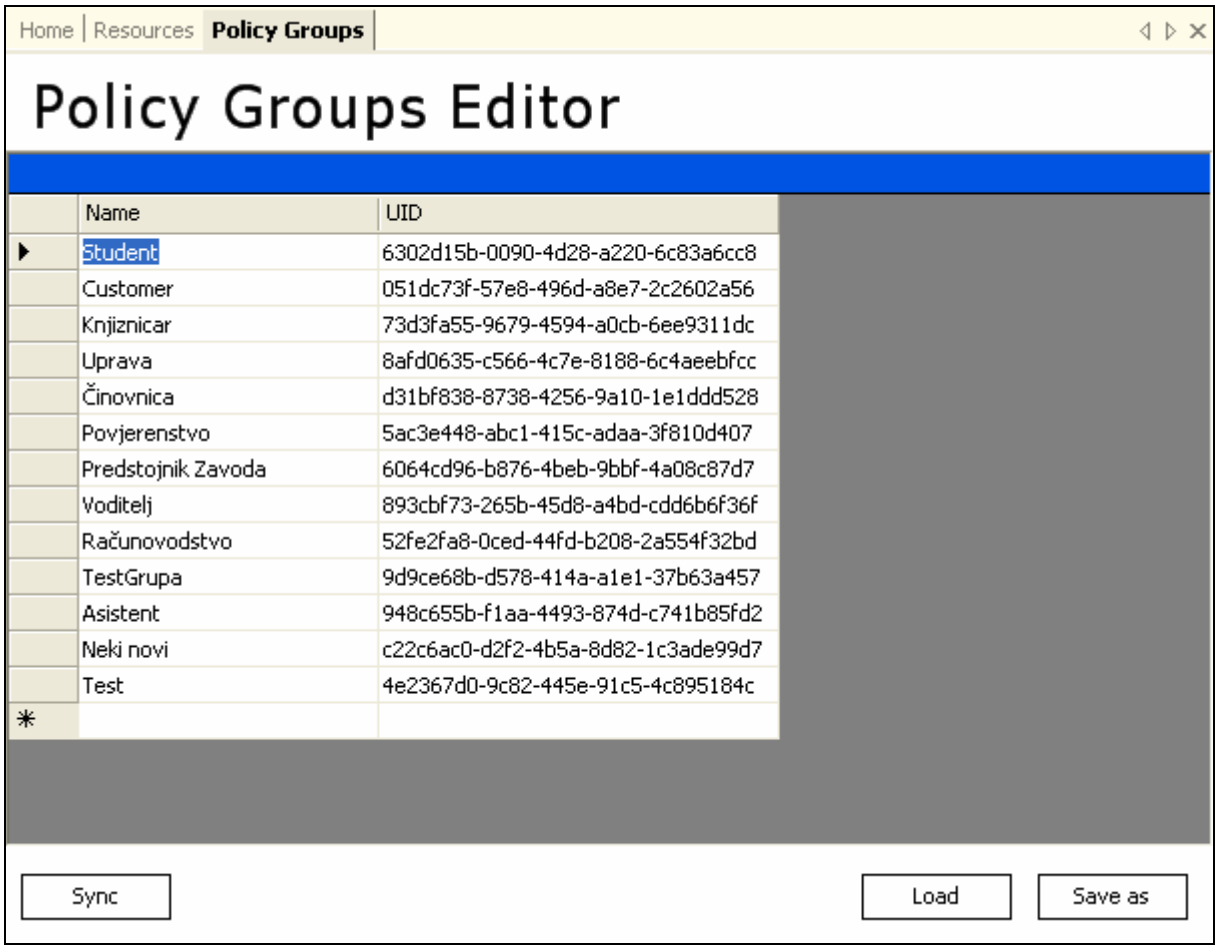
Prozor za prikaz postupka sadrži još jedan pogled na postupak gdje se elementi postupka prikazuju u stablastoj hijerarhiji. Ovaj prozor je prikladan za navigaciju po strukturi postupka koja grafički prikazana sadrži nekoliko ekrana, te za pregled ugniježđenih postupaka.



Slika 5.7 : Struktura postupka

5.1.6. Definiranje grupa korisnika

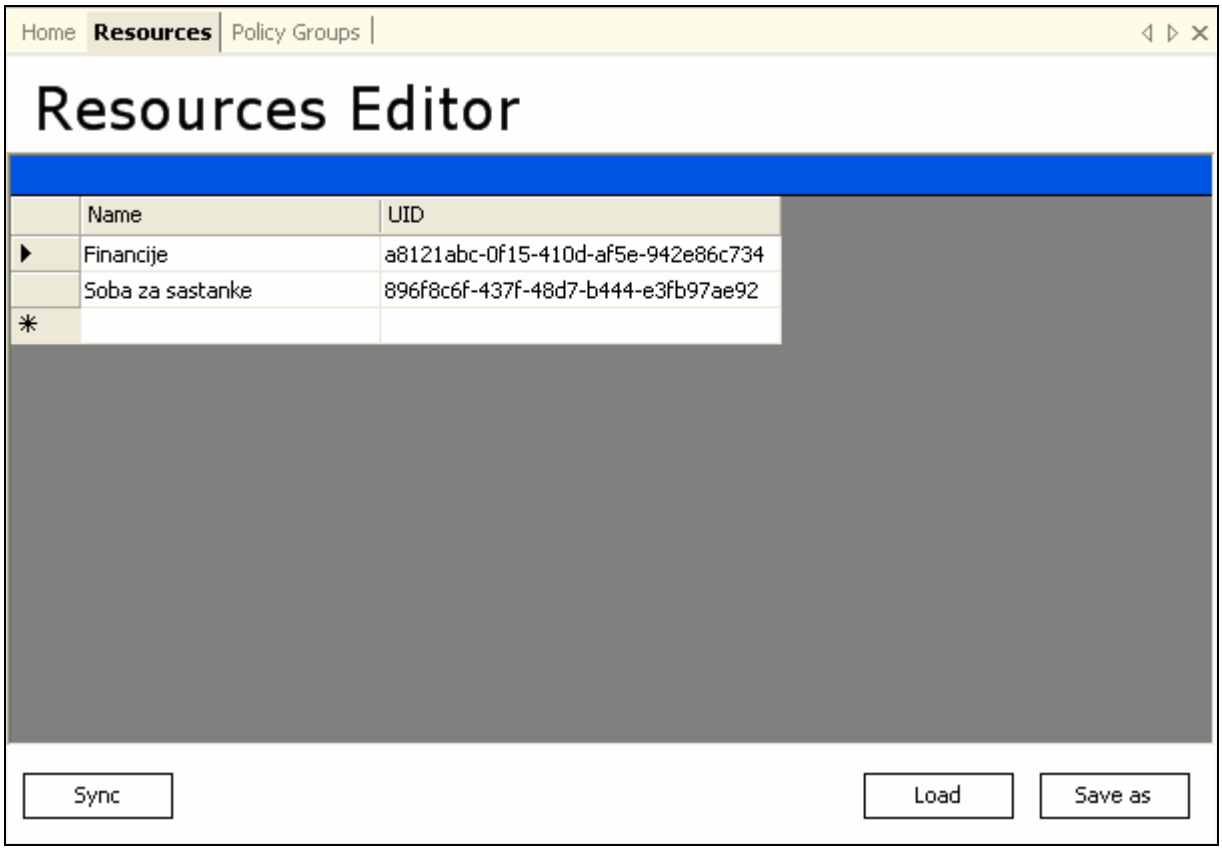
U posebnom radnom prozoru omogućeno je stvaranje, izmjena i brisanje grupa korisnika koje imaju pravo na pojedine elemente postupka. Moguća je i sinkronizacija s poslužiteljem.



Slika 5.8 : Uređivanje grupa korisnika

5.1.7. Definiranje resursa

U posebnom radnom prozoru omogućeno je stvaranje, izmjena i brisanje resursa koji se koriste prilikom izvršavanja pojedinog elementa postupka. Moguća je i sinkronizacija s poslužiteljem.



Slika 5.9 : Uređivanje resursa

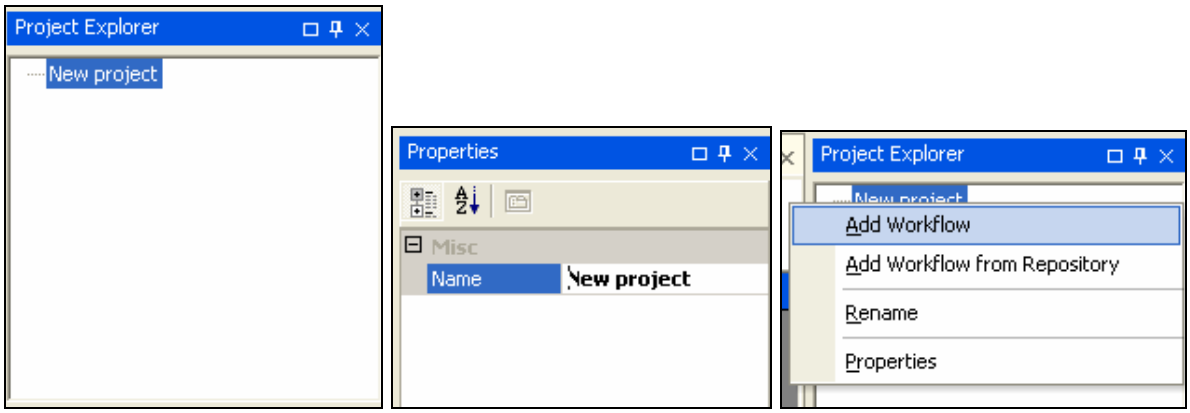
5.1.8. Rukovanje projektom

Osnovni radni dokument SUSAP klijent aplikacije je projekt. Projekt je moguće stvoriti na dva načina:

- pomoću početne stranice i
- pomoću glavnog izbornika aplikacije.

Novostvoreni projekt je prikazan u prozoru za prikaz projekta.

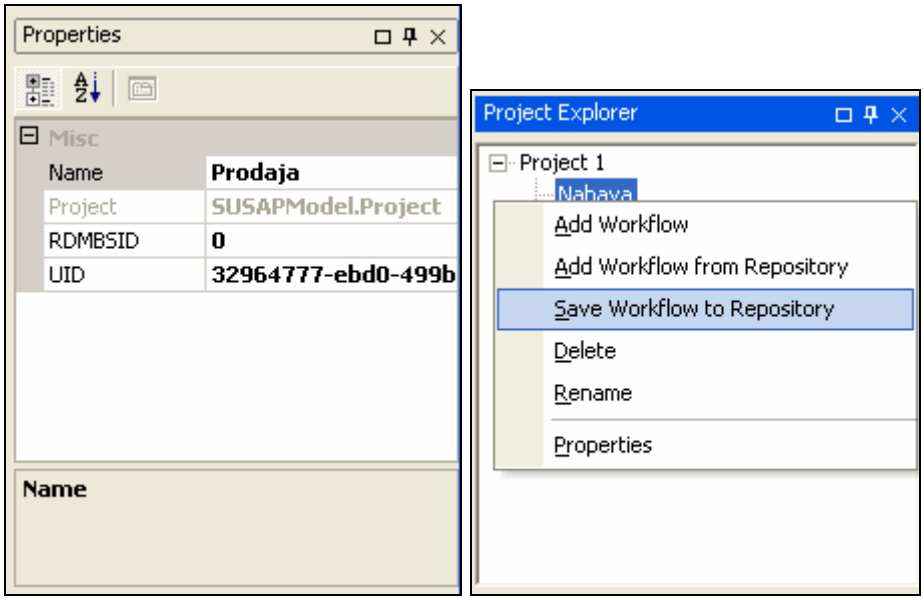
Odabirom naziva projekta u prozoru za prikaz projekta svojstva projekta prikazuju se u prozoru za uređivanje svojstava. Izmjenom svojstva *Name* moguće je mijenjati naziv projekta.



Slika 5.10 : Primjer rada s projektom

Upotrebom kontekstnog izbornika prozora za prikaz projekta te odabirom opcije *Add workflow* stvara se nova definicija postupka. Ukoliko je postupak već smješten na poslužitelju moguće ga je dodati u projekt pomoću komande *Add workflow from repository*.

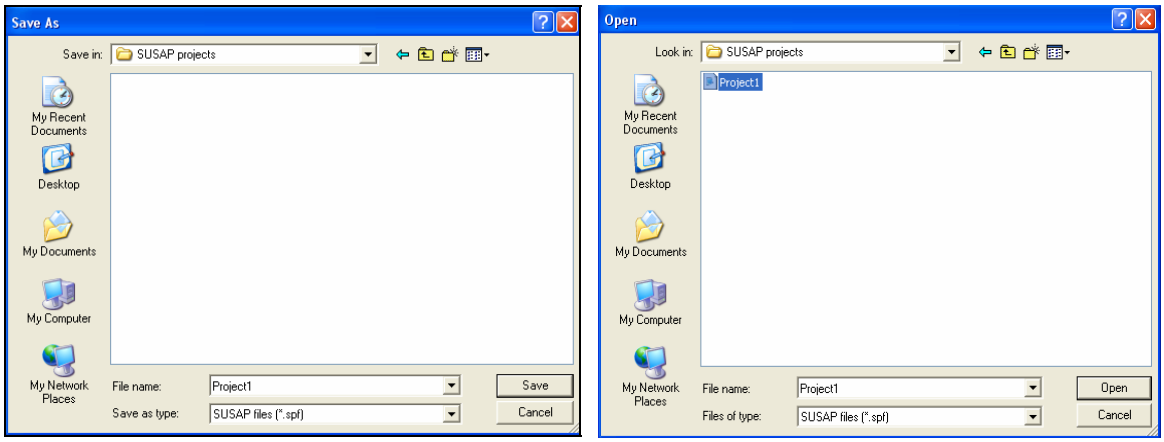
Odabirom postupka u prozoru za prikaz projekta moguće je mijenjati njena svojstva u prozoru za uređivanje svojstava.



Slika 5.11 : Definiranje postupka

Dvostukim klikom miša na pojedini postupak ona se otvara u novom prozoru. Kontekstni izbornik sadrži opcije *Save Workflow to repository* za spremanje postupka na poslužitelj.

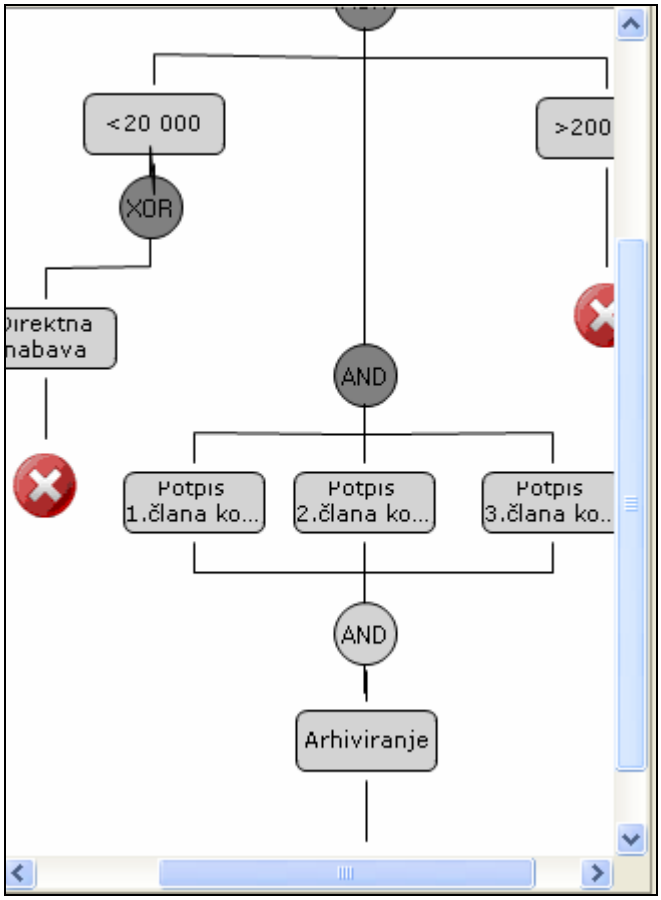
Koristeći opciju glavnog izbornika *Save* ili *Save as* projekt se sprema kao datoteka na čvrsti disk. Slično kao i kod stvaranja novog projekta, postojeći projekt moguće je otvoriti pomoću početne stranice te putem *Open* opcije glavnog izbornika.



Slika 5.12 : Dijalozi za rukovanje datotekom projekta

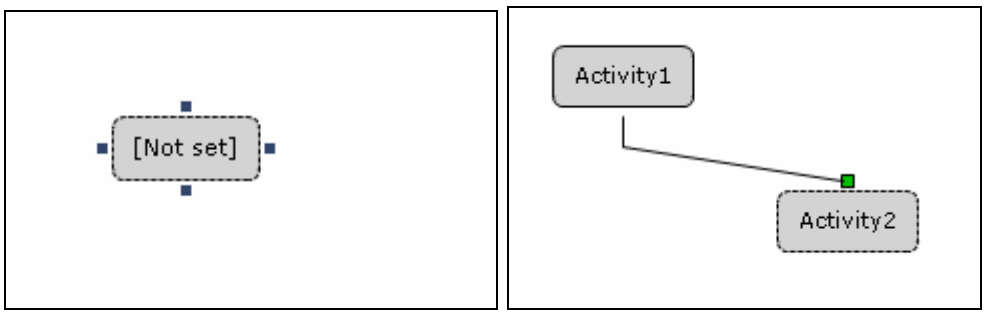
5.1.9. Rukovanje postupkom

Najveći dio vremena korištenja aplikacije odnosi se na uređivanje postupaka. Ukoliko grafički prikaz protoka poslova prelazi više ekrana moguće povlačenjem traka za pomak (*scrollbars*) mijenjati trenutno prikazani dio postupka.



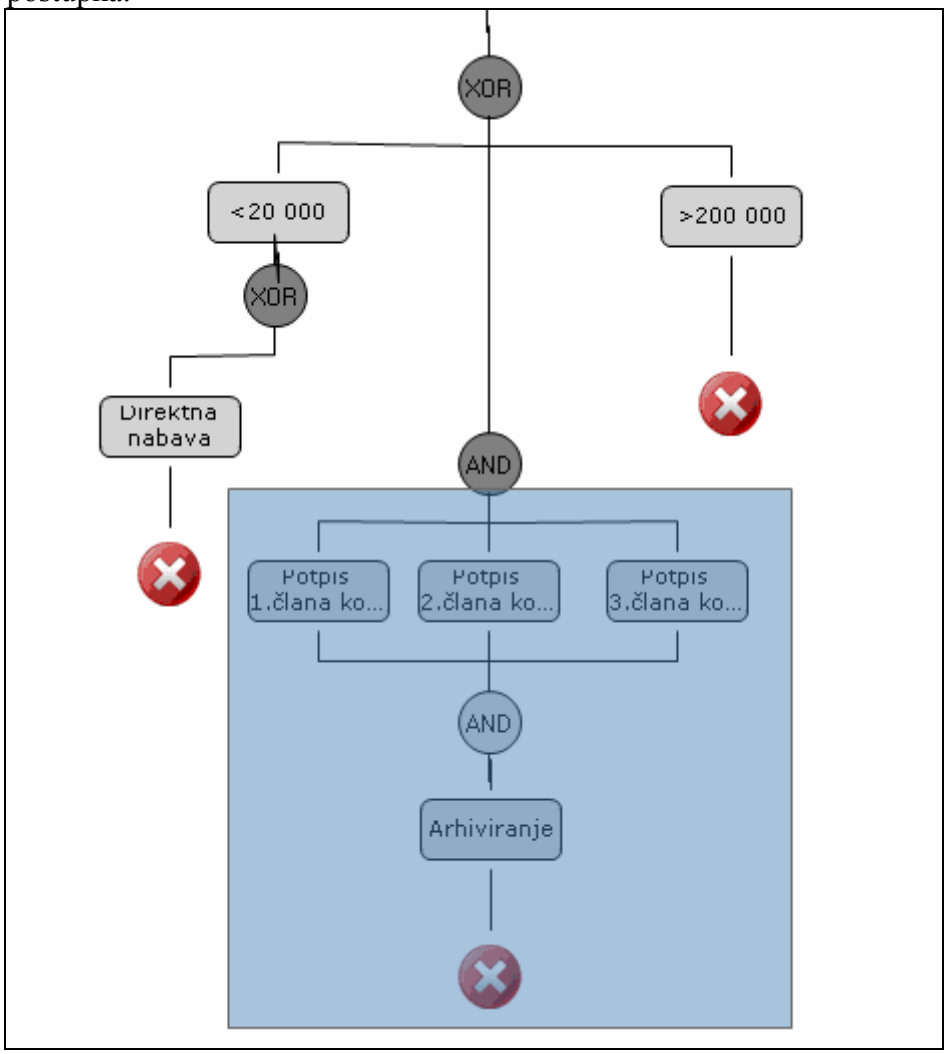
Slika 5.13 : Rukovanje postupkom

Elementi postupka dodaju se preko alatnog prozora jednostavnim odabirom u alatnom prozoru i dovlačenjem pomoću miša na radnu površinu grafičkog prikaza. Novostvoreni element je označen u grafičkom prikazu i moguće je njegovo pozicioniranje povlačenjem pomoću miša na željenu poziciju. Dok je označen, pritiskom na tipku za brisanje (*Delete, DEL*) briše se označeni element.



Svaki element sadrži više konektora koji omogućavaju povezivanje elemenata.

Odabirom jednog konektora i odvlačenjem na drugi stvara se nova veza. Označavanjem pomoću miša i pritiskom na tipku za brisanje (*Delete, DEL*) briše se označena veza. Označavanjem više elemenata i povlačenjem mišem moguće je grupno repozicioniranje elemenata postupka.



Slika 5.14 : Selekcija više elemenata

5.1.10. Svojstva elemenata dijagrama protoka poslova

Dvostrukim klikom na element postupka moguće je mijenjati svojstva pojedinog elementa kao što je ranije objašnjeno.

Aktivnost (*Activity*)

Svojstva	
Name	Naziv elementa
Description	Opis aktivnost
Location	Pozicija elementa
Size	Veličina elementa
Z-order	Z koordinata
Documents	Dokumenti koje aktivnost kreira
Nested worflow	Naziv ugniježđenog postupka
Milestones	Prekretnice o kojima ovisi aktivnost
Policy Group	Korisnička grupa za pokretanje aktivnosti
Code	Programski kod aktivnosti

Bilješka (*Notes*)

Svojstva	
Name	Naziv elementa
Text	Tekst bilješke
Alignment	Poravnanje teksta
Location	Pozicija elementa
Size	Veličina elementa
Z-order	Z koordinata

Start/Stop prijelazi (*Start/Stop Transitions*)

Start i *Stop* prijelazi su specifična vrsta prijelaza pomoću kojih se određuje početak i kraj postupka.

Svojstva	
Name	Naziv elementa
Location	Pozicija elementa
Size	Veličina elementa
Z-order	Z koordinata

Prijelazi (*Transitions*)

Obični prijelazi (predlošci protoka poslova) imaju jednaki skup svojstava, prikazan u narednoj tablici.

- Sequence
- AND Fork i AND JOIN
- OR Fork i AND JOIN
- XOR Fork i XOR JOIN
- Discriminator
- Interleaved parallel routing
- Multiple instances
- Implicit Choice
- Milestone

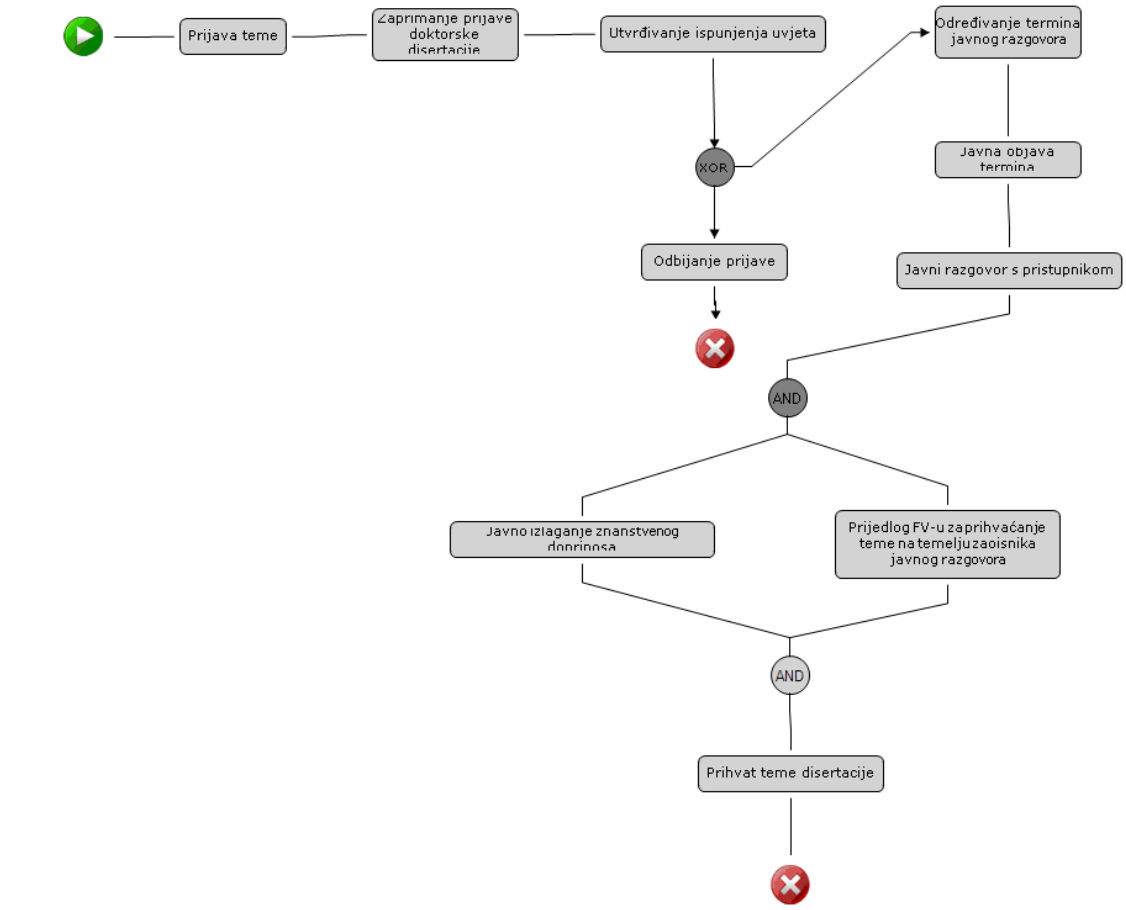
Svojstva	
Name	Naziv elementa
Location	Pozicija elementa
Size	Veličina elementa
Z-order	Z koordinata
Condition	Uvjeti prijelaza

M out of N

Svojstva	
Name	Naziv elementa
Location	Pozicija elementa
Size	Veličina elementa
Z-order	Z koordinata
Condition	Uvjeti prijelaza
M	Vrijednost M

5.2. Web aplikacija za upravljanje poslovanjem

Opis grafičkog sučelja u nastavku je prikazan kroz jednostavni primjer koji je javno dostupan na web adresi http://apps.zpr.fer.hr/susap. Primjer predstavlja postupak prijave teme doktorske disertacije i određivanje termina javnog razgovora. Sljedeća slika prikazuje postupak nacrtan u grafičkom uređivaču postupka, napravljenom u sklopu ovog projekta.

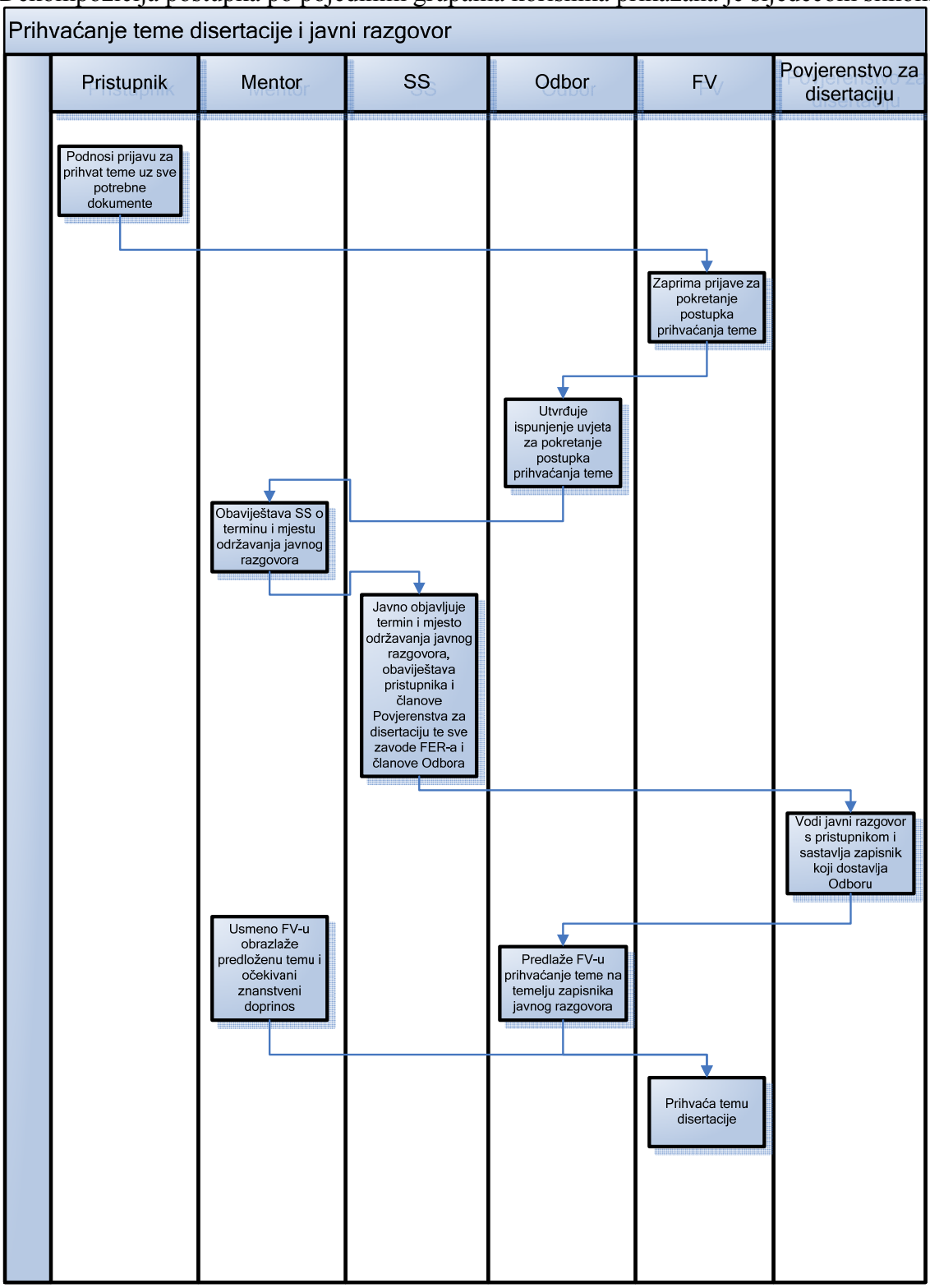


Slika 5.15 : Postupak prijave teme doktorske disertacije i određivanje termina javnog razgovora

U svrhu probnog testiranja mogu se koristiti sljedeći probni korisnički računi:

Korisničko ime	Zaporka	Članstvo u grupama
student	student	Student
mentor1	mentor1	Mentor
mentor2	mentor2	Mentor
ss	ss	Studentska služba
fv	fv	Fakultetsko vijeće
odbor	odbor	Odbor za poslijediplomski studij
pzd	pzd	Povjerenstvo za disertaciju

Dekompozicija postupka po pojedinim grupama korisnika prikazana je sljedećom slikom:

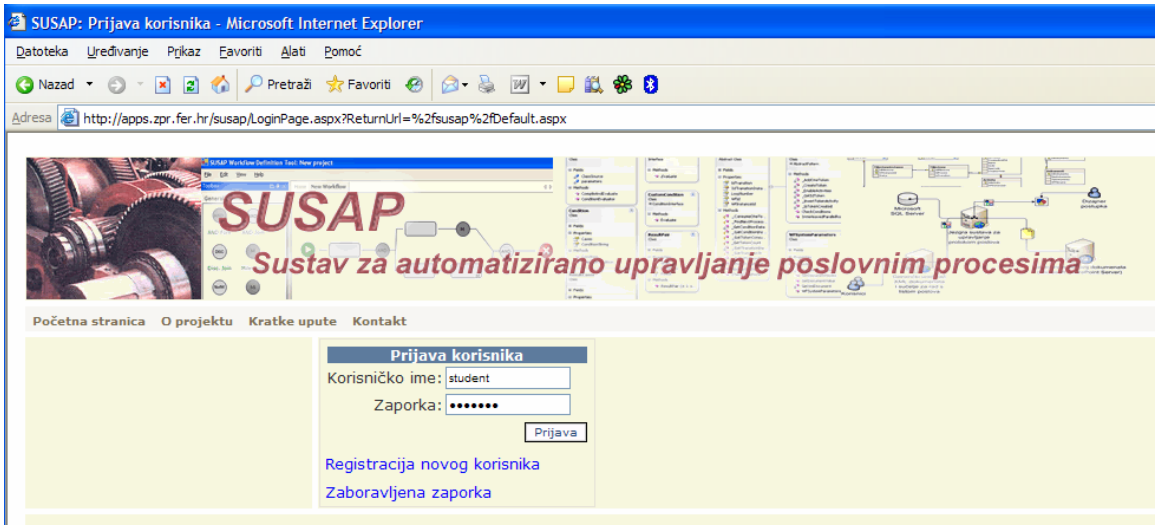


Slika 5.16 : Prikaz tijeka postupka s naglaskom na grupe kojima izvršitelji pripadaju

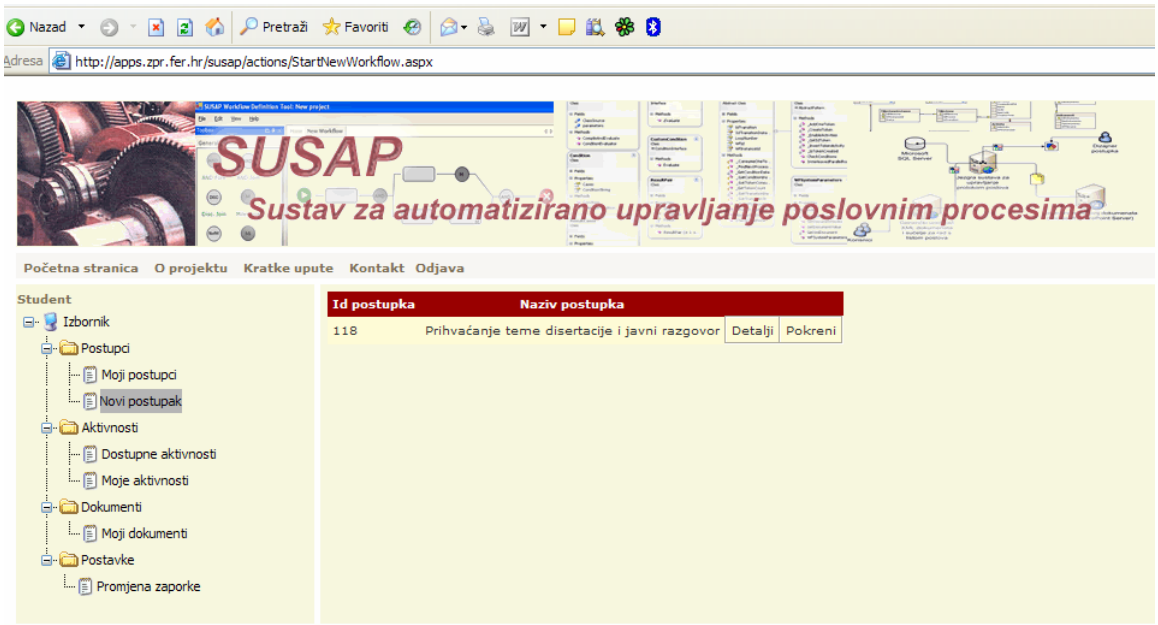
Dokument korišten unutar postupka opisan je XML shemom i XML dokumentom koji sadrži popis prava na pristup pojedinom elementu dokumenta u pojedinom trenutku, kao što je opisano u sljedećem poglavlju. Primjer dokumenta s pravima nalazi se u prilogu, na kraju ovog dokumenta.

5.2.1. Prijava korisnika i izbornik akcija

Rad na sustavu počinje prijavom na sustav nakon čega se prijavljenom korisniku s desne strane pojavljuje izbornik akcija.



Slika 5.17 : Prijava na sustav



Slika 5.18 : Izbornik akcija

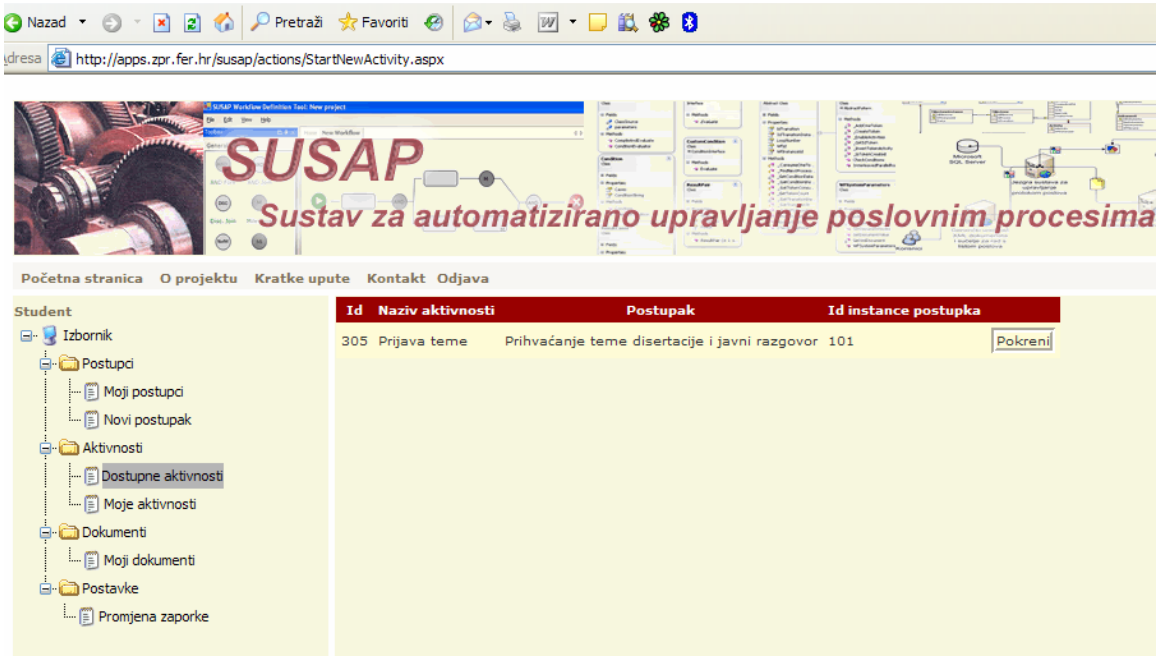
Izbornik sadrži sljedeće elemente:

- *Moji postupci*: Prikazuje listu postupaka koje je korisnik započeo te omogućava uvid u stanje pojedinog postupka.
- *Novi postupak*: Prikazuje listu postupaka koje korisnik može pokrenuti.
- *Dostupne aktivnosti*: Lista nepreuzetih aktivnosti koje su ponuđene trenutnom korisniku. Odabirom elemente iz liste aktivnost će se vezati isključivo za tog korisnika.

- *Moje aktivnosti*: Lista aktivnosti koje su pridružene trenutnom korisniku. Za svaki element u listi aktivnosti korisnik može dohvatiti popis dokumenata korištenih u toj aktivnosti.
- *Moji dokumenti*: Lista svih dokumenata na kojima korisnik može raditi. Lista nije ograničena pojedinom aktivnosti, ali su uključeni samo oni dokumenti koji pripadaju trenutno pokrenutim aktivnostima.
- *Promjena zaporke*: omogućava promjenu zaporke trenutnom korisniku (onemogućeno za ranije navedene probne korisničke račune).

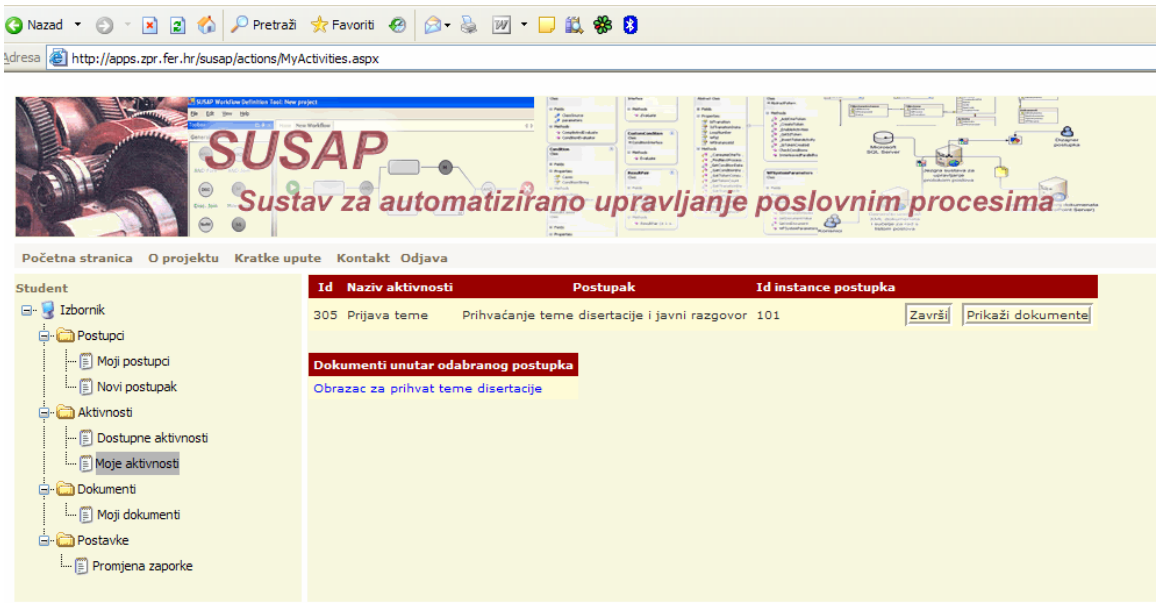
Slika 5.19 : Prikaz detalja o postupku koji se može pokrenuti

Ukoliko korisnik želi pokrenuti neki od postupaka, to će učiniti tako da iz izbornika odabere opciju *Novi postupak* i zatim pokrene jedan od ponuđenih postupaka. Klikom na gumb *Detalji*, korisnik, prije samo pokretanja postupka može vidjeti njegovu sliku ukoliko takva postoji. Pokretanjem postupka u listi dostupnih aktivnosti pojavit će se prva aktivnost unutar tog postupka. U ovisnosti o pripadnosti grupi korisnika stvorena aktivnost će se prikazati određenim korisnicima. Aktivnost se može pronaći unutar izbornika *Dostupne aktivnosti*. Aktivnosti unutar navedene liste dostupne su za preuzimanje i postat će aktivne tek nakon što ih neki od korisnika, koji za to imaju pravo, pokrene klikom na gumb *Pokreni*. Tako pokrenuta aktivnost pridjeljuje se isključivo korisniku koji ju je pokrenuo i vidi se u njegovoj privatnoj listi pod opcijom izbornika *Moje aktivnosti*.



Slika 5.20 : Popis aktivnosti koje korisnik može pokrenuti

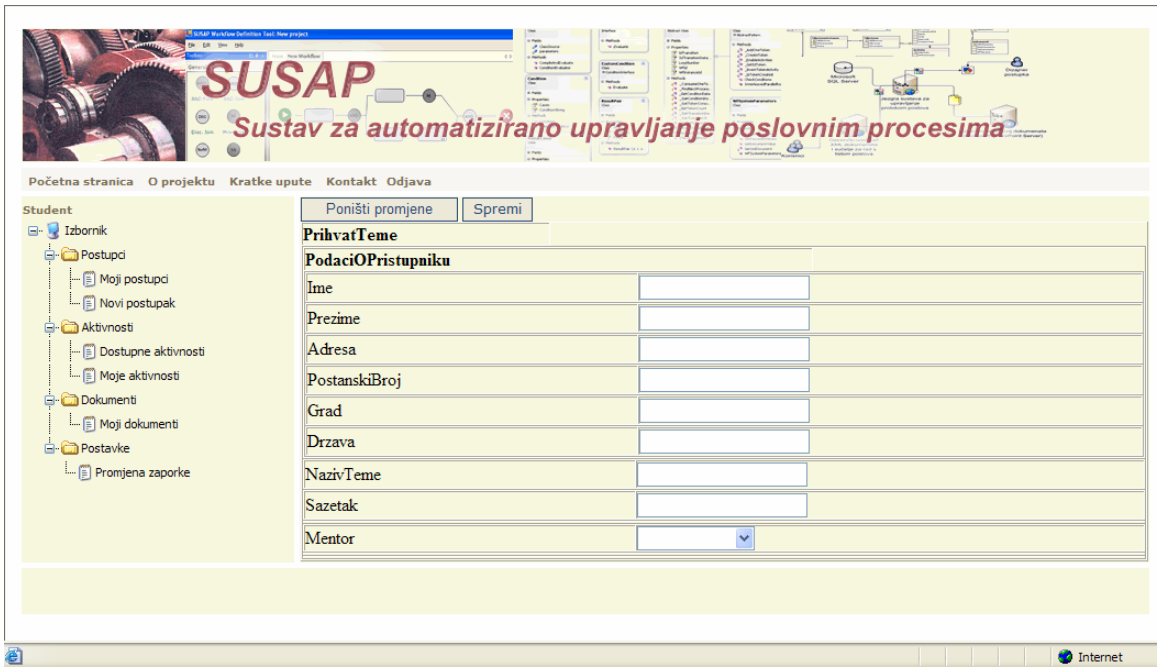
Za svaku od aktivnosti navedenih u popisu pod izbornikom *Moje aktivnosti* korisnik može dobiti listu dokumenata koje se koriste unutar određene aktivnosti.



Slika 5.21 : Aktivnosti pojedinog korisnika

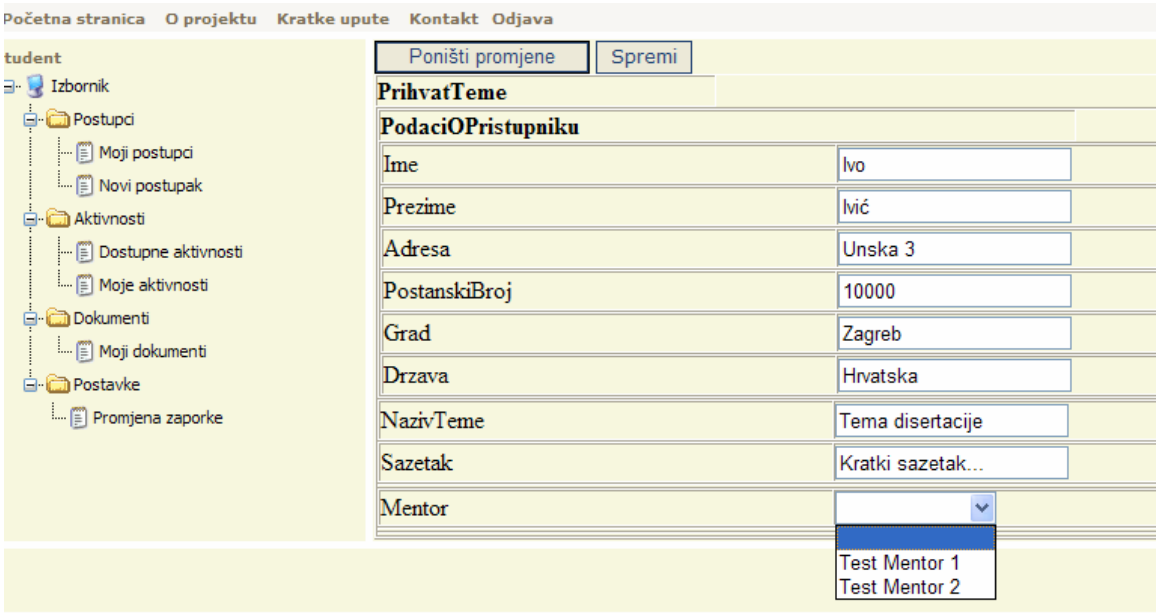
5.2.2. Ažuriranje dokumenata

Klikom na naziv dokumenta otvara se pripadajuća forma (generički obrazac) s automatski generiranim poljima za unos podataka u dokument i to u ovisnosti o pravima pojedinog korisnika u ovisnosti o trenutnoj aktivnosti.



Slika 5.22 : XML uređivač

U ovisnosti o tipu podatka definiranom unutar XML sheme za svaki element se kreira potreban broj kontrola za unos. Primjerice, za sve tipove podataka koji su definirani kao *SUSAPType_NazivGrupe* aplikacija će kreirati padajuću listu i popuniti je sa članovima zadane grupe (u danom primjeru su to dva test mentora).

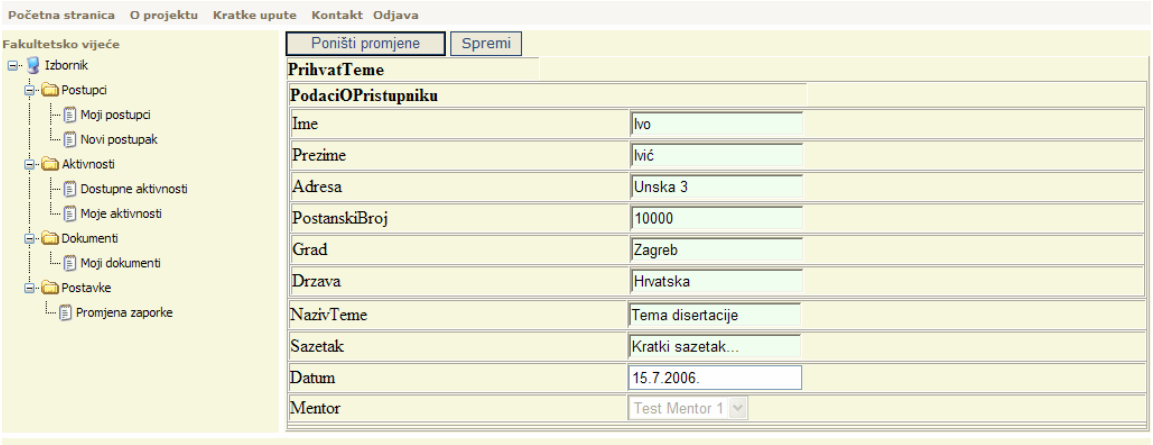


Slika 5.23 : Padajuća lista popunjava se u ovisnosti o pripadnosti korisnika nekoj grupi

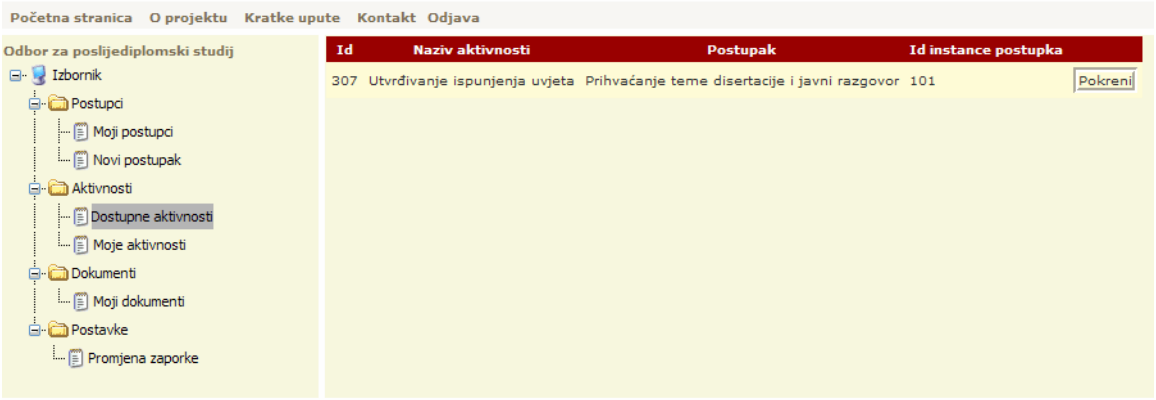
Nakon što korisnik popuni sve podatke potrebno je kliknuti na gumb *Spremi*. Popunjavanjem navedenih elemenata aktivnost nije završena (primjerice, moglo je biti više ovakvih dokumenata u obradi) te je treba eksplicitno proglasiti završenom, za što je potrebno otići pod *Moje Aktivnosti* i kliknuti *Završi*. Sustav će po završetku zadane aktivnosti tijek postupka preusmjeriti dalje u ovisnosti o navedenim pravilima. Primjerice, u probnom postupku, aktivnost će se preusmjeriti na grupu korisnika koji propadaju *Fakultetskom vijeću*. U tom

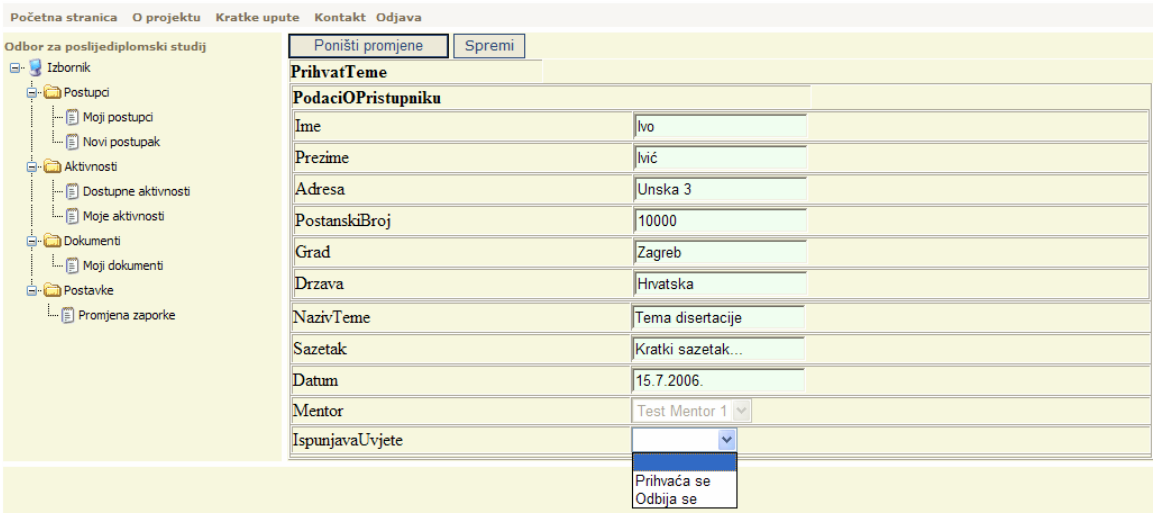
trenutku korisniku *fv* koji predstavlja *Fakultetsko vijeće* pojavit će se nova aktivnosti u listi dostupnih aktivnosti

Budući da je tako definirano unutar XML datoteke s popisom prava unutar XML uređivača pojavit će se dodatno polje za unos datuma zaprimanja prijave, dok će ostala polja biti onemogućena.

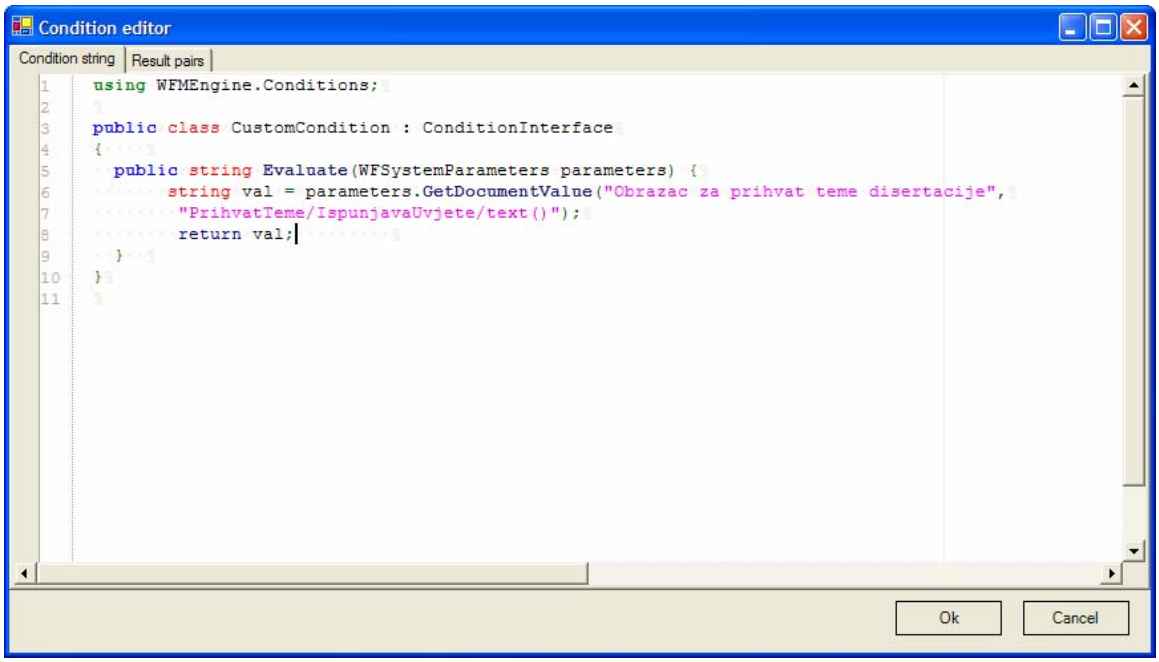


Poštujući pravilnosti navedene prilikom izrade postupka, nakon završetka neke od aktivnosti, nove aktivnosti će se pojavljivati u listama drugih korisnika. Ovisno o tome te ovisno o navedenim pravima, polja koja će XML uređivač prikazati na web stranicama izgledat će drugačije. Jedan od takvih primjera je situacija u kojoj korisnik član grupe *Odbor za poslijediplomski studij* ima mogućnost popuniti polje koje odlučuje da li pristupnik ispunjava uvjete ili ne.

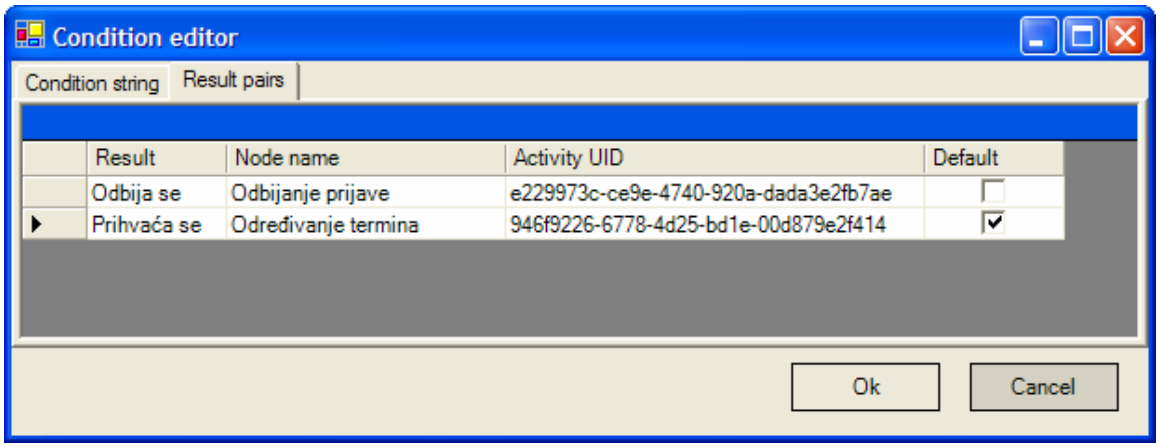




Nakon što završi ova aktivnosti, prema definiciji postupka (Slika 5.15 : Postupak prijave teme doktorske disertacije i određivanje termina javnog razgovora
) dolazi do odabira jedne od dviju mogućih izlaznih grana. Sustav će u tom trenutku dinamički kompilirati kod koji je naveden unutar prijelaza te će na osnovu vrijednosti postavljene unutar dokumenta donijeti odluku o tome koju od izlaznih grana odabrati.

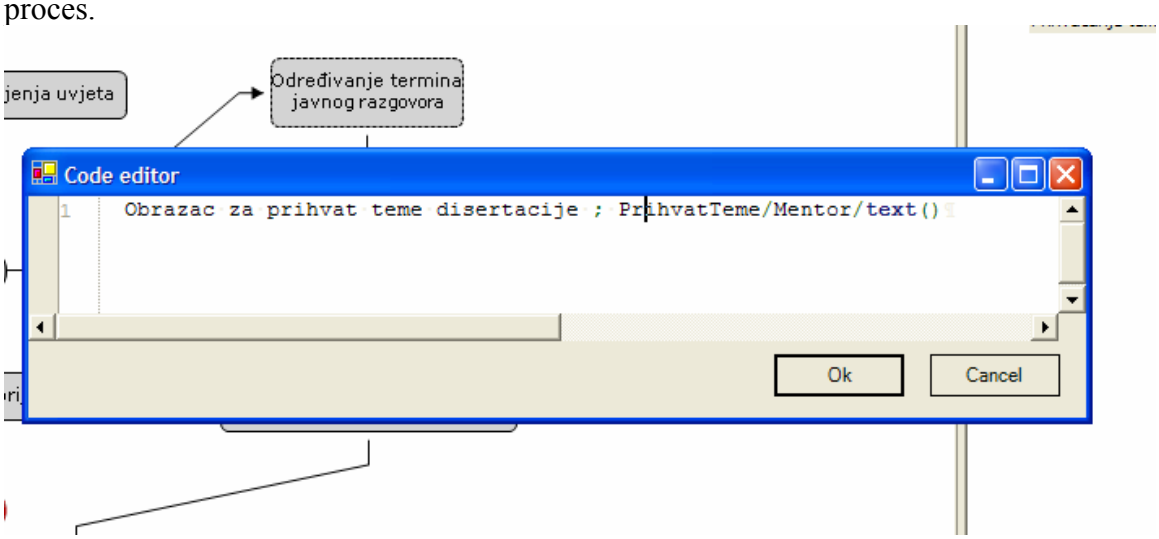


Slika 5.24 : Kod prijelaza koji će se dinamički kompilirati



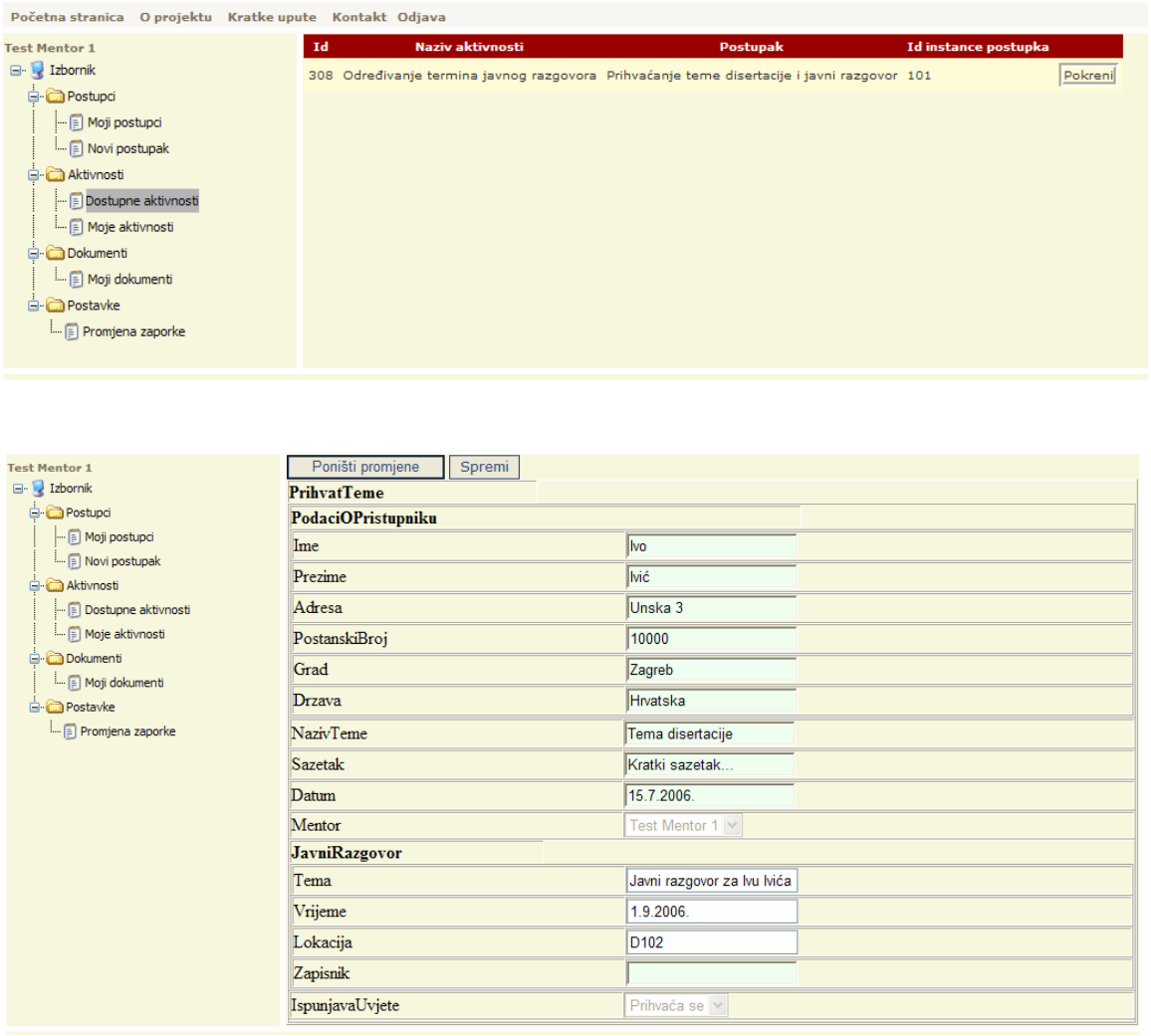
Slika 5.25 : Odabir izlazne grane u ovisnosti o vrijednosti rezultata metode Evaluate

Ukoliko je odbor prihvatio prijavu teme, to jest, ako je kandidat ispunio uvjete, postupak pokreće novu aktivnosti. Definicija postupka govori sustavu da se aktivnosti ne nudi svim korisnicima iz grupe Mentor, već točno onome koji je naveden unutar dokumenta. Navedeni podatak potreban sustavu postavljen je prilikom dizajna postupka u editoru koda za navedeni proces.



Slika 5.26 : Vrijednost određenog elementa unutar korištenog dokumenta poslužit će za odabir izvođača navedene aktivnosti

Ukoliko je pristupnik ispunio uvjete stvorit će se aktivnost koju će moći izvršiti samo mentor koji je naveden u dokumentu. Prema postavkama navedenim u dokumentu s popisom prava pristupa pojedinim elementima dokumenta, mentor će imati mogućnost unosa podataka o javnom razgovoru. Potrebno je primijetiti da će tu mogućnost imati samo u ovoj aktivnosti, a ne i aktivnosti koja će predstavljati proces javnog izlaganja znanstvenog doprinosa koji će uslijedi nešto kasnije.



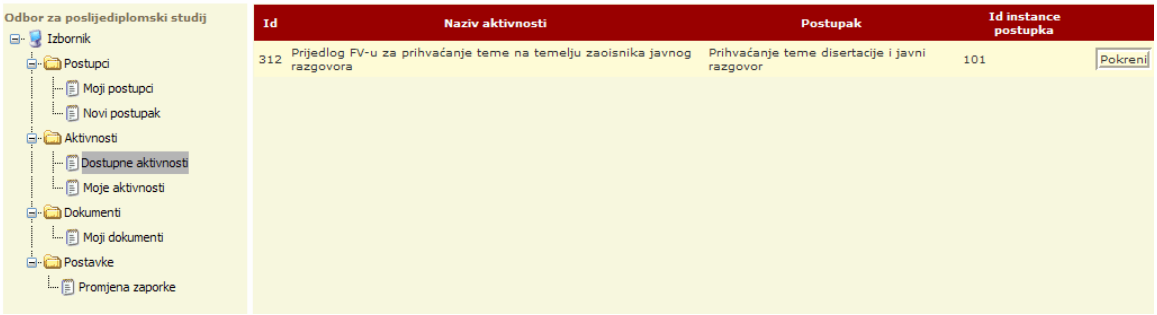
Slika 5.27 : Precizno definirana prava omogućuju mentoru da unese podatke o javnom razgovor, ali ne i njegov zapisnik.

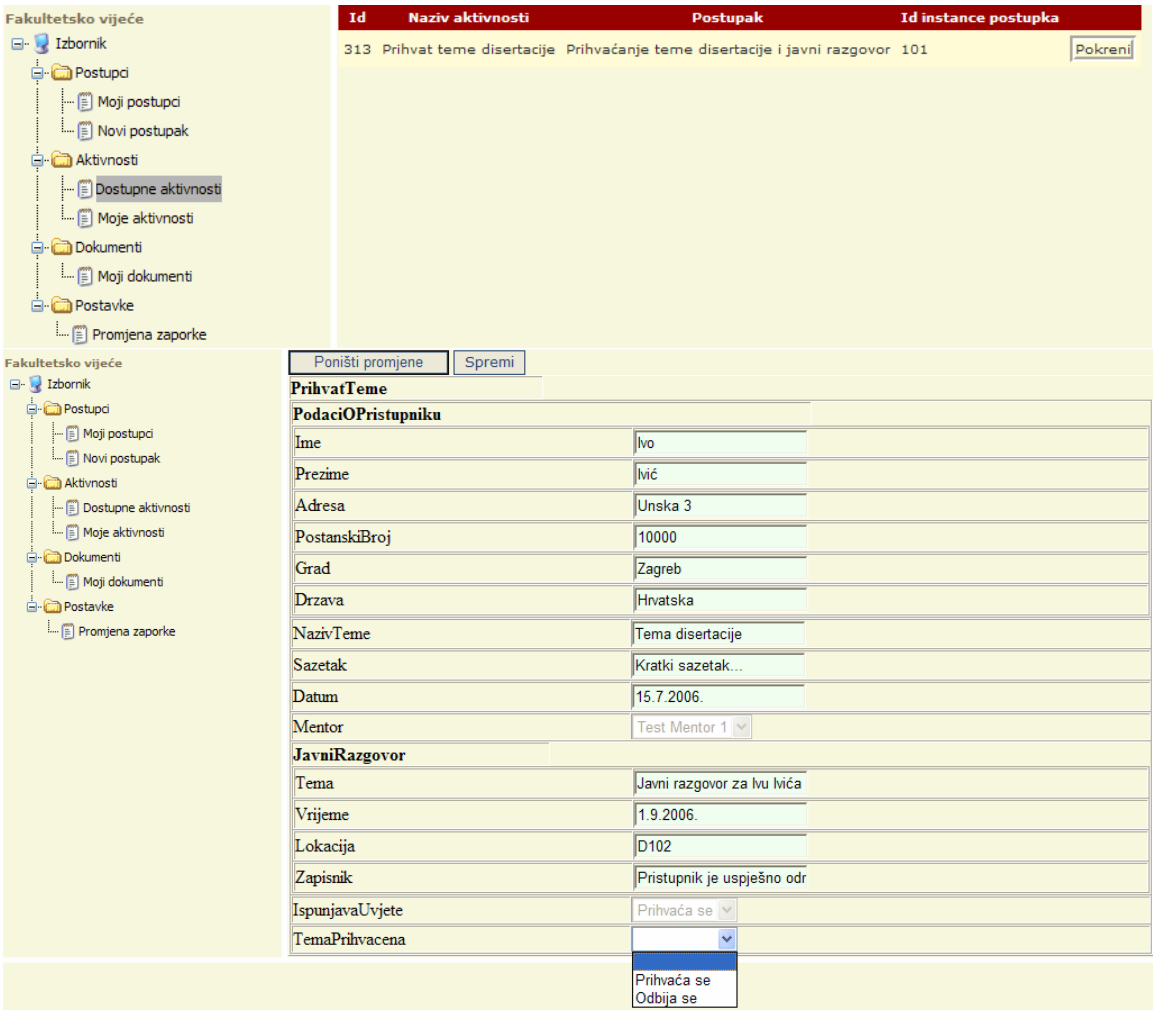
Sljedećih nekoliko slika prikazuje daljnji tijek postupka i izgled XML web uređivača kako ga vidi pojedini korisnik.


Povjerenstvo za disertaciju
Izbornik
Postupci
Moji postupci
Novi postupak
Aktivnosti
Dostupne aktivnosti
Moje aktivnosti
Dokumenti
Moji dokumenti
Postavke
Promjena zaporke
Id
Naziv aktivnosti
Postupak
Id instance postupka
310 Javni razgovor s pristupnikom Prihvaćanje teme disertacije i javni razgovor 101
Pokreni

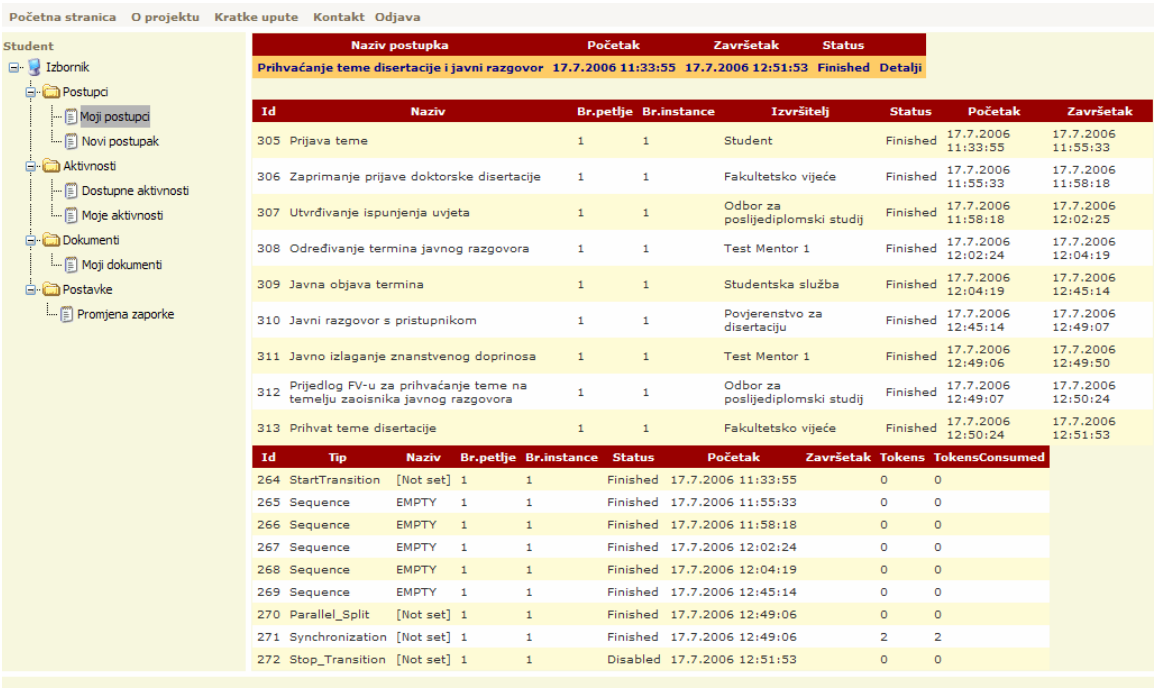

Početna stranica O projektu Kratke upute Kontakt Odjava
Povjerenstvo za disertaciju
Poništi promjene
Spremi
PrihvatTeme
PodaciOPristupniku
Ime Ivo
Prezime Ivić
Adresa Unska 3
PostanskiBroj 10000
Grad Zagreb
Drzava Hrvatska
NazivTeme Tema disertacije
Sazetak Kratki sazetak...
Datum 15.7.2006.
Mentor Test Mentor 1
JavniRazgovor
Tema Javni razgovor za Ivu Ivića
Vrijeme 1.9.2006.
Lokacija D102
Zapisnik Pristupnik je uspješno odr
IspunjavaUvjete Prihvaća se


Test Mentor 1
Izbornik
Postupci
Moji postupci
Novi postupak
Aktivnosti
Dostupne aktivnosti
Moje aktivnosti
Dokumenti
Moji dokumenti
Postavke
Promjena zaporke
Id
Naziv aktivnosti
Postupak
Id instance postupka
311 Javno izlaganje znanstvenog doprinosa Prihvaćanje teme disertacije i javni razgovor 101
Pokreni


Odbor za poslijediplomski studij
Izbornik
Postupci
Moji postupci
Novi postupak
Aktivnosti
Dostupne aktivnosti
Moje aktivnosti
Dokumenti
Moji dokumenti
Postavke
Promjena zaporke
Id
Naziv aktivnosti
Postupak
Id instance postupka
312 Prijedlog FV-u za prihvaćanje teme na temelju zaoisnika javnog razgovora
Prihvaćanje teme disertacije i javni razgovor
101
Pokreni



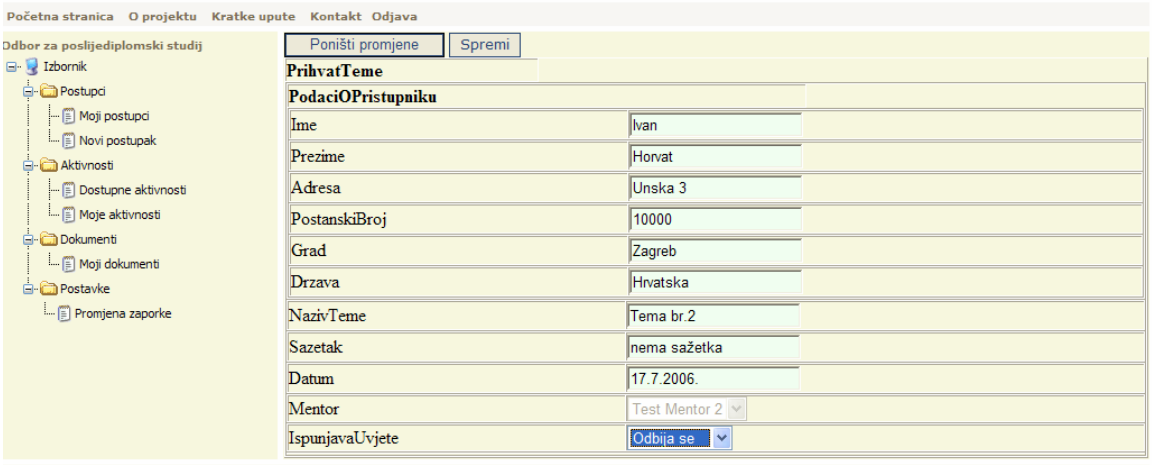
Korisnik koji je započeo postupak, što je u ovom primjeru pristupnik, može vidjeti stanje postupka koje je evidentirano kroz popis stvorenih aktivnosti i prijelaza. Uz svaku od aktivnosti naveden je podatak o vremenu stvaranja aktivnosti, vremenu završetka (ukoliko je završena) kao i o imenu izvršitelja. Ukoliko je postupak imao ponavljanja lakše praćenje tijeka postupka omogućit će podatci *LoopNumber* (broj ponavljanja) i *InstanceNumber* (broj instance ukoliko je postojao prijelaz s višestrukim instancama).

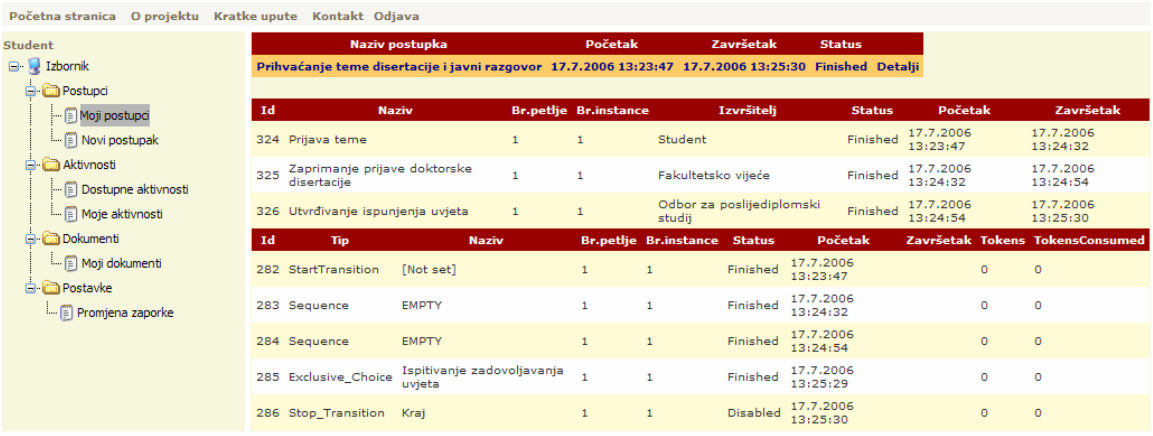


Slika 5.28 : Izvršeni prijelazi

Popis izvršenih prijelaza sadrži i podatke o tokenima korištenim za sinkronizaciju (broj aktivnosti koje se čekaju i broj završenih aktivnosti koje prethode točkama sinkronizacije).

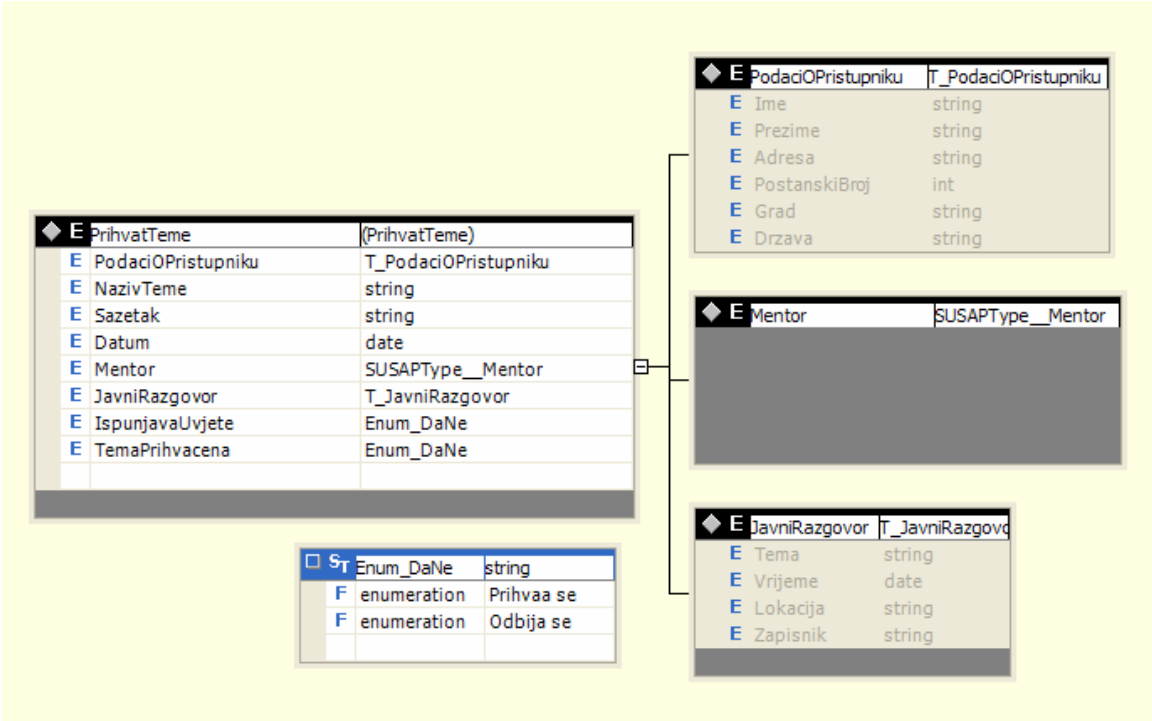
Ukoliko je *Odbor za poslijediplomski studij* odbio prijavu kandidata postupak bi odmah završio i u listi aktivnosti bi se prikazale dotad izvršene aktivnosti.





Slika 5.29 : Popis aktivnosti izvršenih do trenutka kad je prijava kandidata odbačena

5.2.3. Shema dokumenta



Slika 5.30 : Shema dokumenta

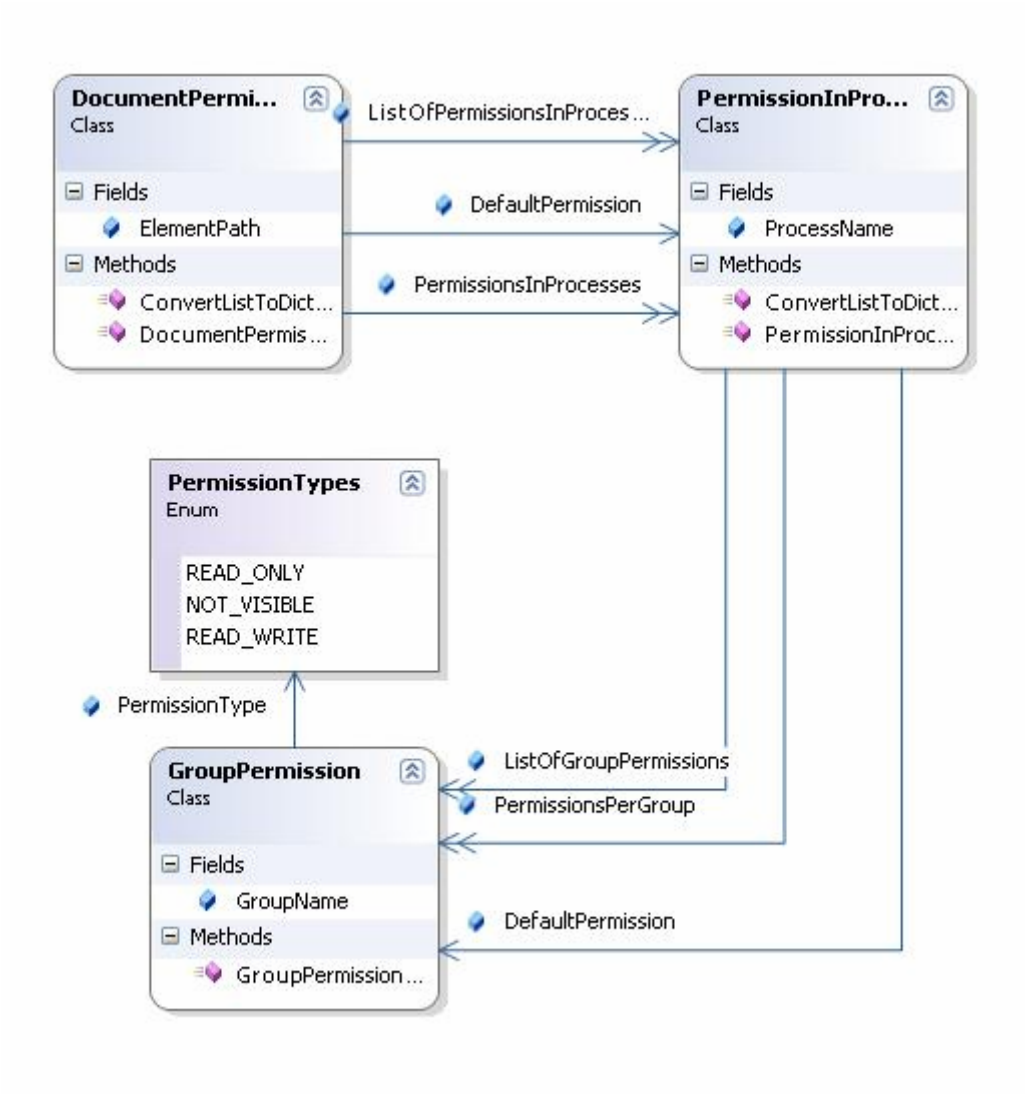
Obrazac za prijavu teme doktorske disertacije opisan je shemom koja se može prikazati gornjom slikom. Međutim, sam opis dokumenta nije dovoljan te je potrebno izraditi dokument s popisom prava. U ovisnosti o podacima unutar tako kreiranog XML dokumenta web uređivač će različito prikazati ili uopće neće prikazati određene elemente. Prava koja se mogu postaviti na pojedini element su:

- mogućnost čitanja i izmjene sadržaja nekog elementa,
- element je samo za čitanje,
- element nije vidljiv određenoj osobi u određenom trenutku.

Međutim, pojedini element može imati različito pravo pristupa ovisno o trenutnom korisniku i u ovisnosti o trenutnom procesu (to jest aktivnosti koja predstavlja instancu procesa).

Primjerice, u postupku prijave teme za doktorsku disertaciju mentor određuje vrijeme javnog razgovora. Međutim, mentor to smije obaviti tek nakon što je utvrđeno da je pristupnik ispunio sve uvjete te tako uneseni podatak više ne smije kasnije mijenjati. Dakle, iako se radi o istom elementu unutar istog dokumenta i o istom korisniku, pravo pristupa je opisano ne samo korisnikom, već i stanjem u kojem se cijeli postupak nalazi.

Kako bi opis postupka određivanja prava pristupa pojedinom elementu bilo jednostavniji, na slici je dana shema razreda korištenih za određivanje prava. Vrijednosti članova navedenih razreda pohranjeni su serializacijom unutar za to predviđenog XML dokumenta. XML dokument s pravima treba imati naziv koji se sastoji od imena dokumenta korištenog za XML shemu uz sufiks `_Permissions`, te se u repozitoriju dokumenata nalazi unutar iste mape kao i XML shema.



Slika 5.31 : Razredi korišteni za određivanje prava

Nakon uspješnog dohvata navedenog XML dokumenta pretvorbom iz XML-a u instance navedenih razreda dobit će se skup pravila za određene elemente. Implementacijski je taj skup izveden kao rječnik zbog bržeg dohvata (`Dictionary<string, DocumentPermission>`)

Rječnik kao ključ sadrži putanje elemenata. Svaka vrijednost dohvaćena po navedenom ključu je skup pravila za dani element. Ukoliko neki element nema navedena pravila za pristup, algoritam će potražiti prvog pretka u hijerarhiji koji ima definirana pravila pristupa. Ako takav ne postoji smatrat će se da je element namijenjen i za čitanje i za pisanje svim korisnicima u svim procesima unutar tog postupka.

Prilikom određivanja prava pristupa za pojedini element dolazi do traženje stavke koja najbliže odgovara traženoj stavci. Algoritam u prvom koraku traži stavku koja za putanju ima upravo putanju elementa za koji je potrebno odrediti pravo. Ukoliko takva stavka ne postoji, traži se stavka s putanjom roditelja navedenog elementa, a zatim ukoliko ni ona ne postoji, onda se traži stavka daljnjeg pretka i tako dalje, sve dok se neka od stavki ne pronađe ili se iscrpe sve mogućnosti (to jest, niti traženi element, niti njegovi preci nemaju definiranu stavku s pravilom prava pristupa).

Ukoliko su sva prava iscrpljena smatra se da se mora dodijeliti pretpostavljeno pravo pristupa, a ono je u ovom slučaju pravo čitanja i pisanja.

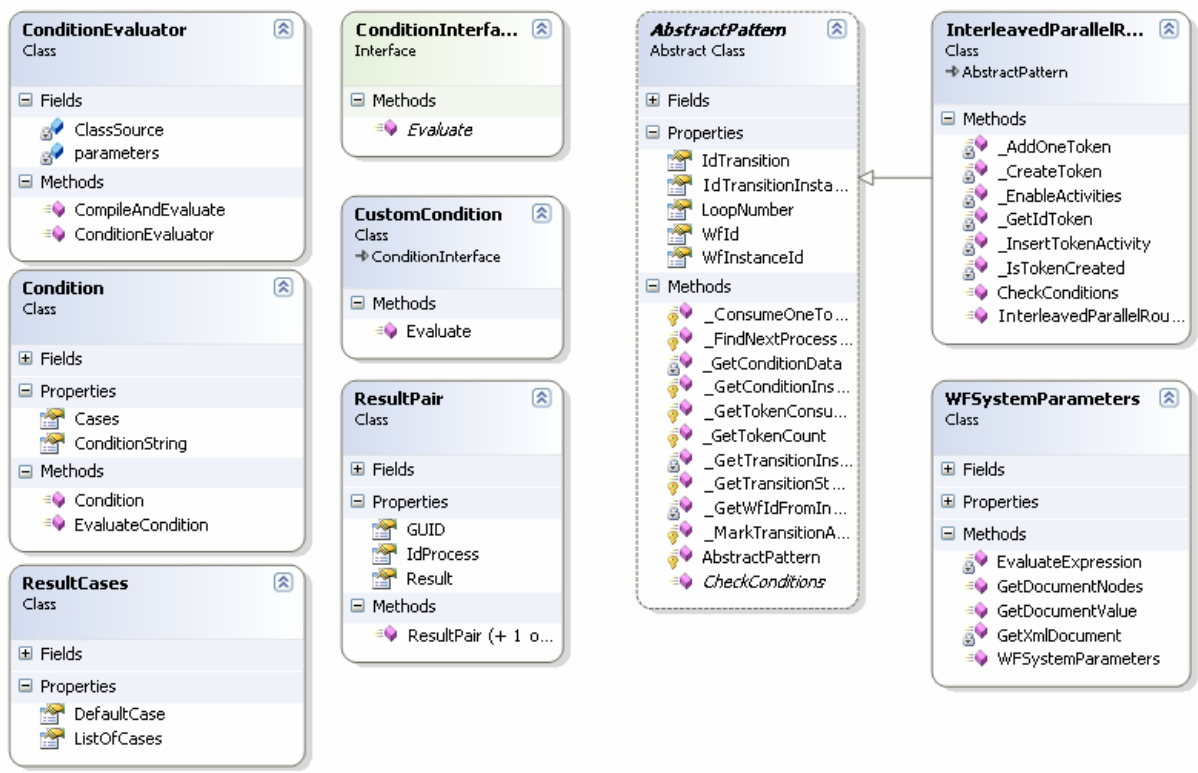
Ukoliko se pronađe željena stavka, tada dolazi do provjere da li se dana stavka odnosi na trenutni proces. Ukoliko stavka nema zapis za zadani proces, koriste se pretpostavljeno pravo, ako takvo postoji, a inače se element ne prikazuje.

5.3. Prijelazi između aktivnosti i evaluacija izraza prijelaza

Ukoliko nakon završetka neke od aktivnosti treba donijeti odluku o odabiru jedne ili više alternativa, odnosno odluku o broju instanci koje treba pokrenuti, potrebno je definirati uvjet prijelaza. Često je prilikom pisanja uvjeta prijelaza potrebno imati vezu s varijablama iz sustava za protok poslova, kao što su vrijeme izvođenja procesa, vrijeme početka procesa, podatak o osobi koja izvodi dani proces, vrijednost nekog polja u dokumentu koji se koristi kroz sustav ili pak neki drugi podatak koji se može dohvatiti iz javno dostupnih podataka zapisanih u tablicama sustava. Također, od sustava se zahtjeva da promjena uvjeta prijelaza ne mora uzrokovati izmjenu programskog koda sustava, što bi uzrokovalo privremeni prekid rada sustava, prevođenje programskog koda te ponovno pokretanje i nastavak rada sustava od stanja u kojem je prethodno bio zaustavljen. Pored navedenog pisanje uvjeta prijelaza mora biti jednostavno i omogućavati određenu sigurnost po pitanju sintaktičke ispravnosti. Neki od postojećih sustava koriste vlastite skriptne jezike koji nisu uvijek sintaktički provjerljivi, nisu jednostavni za korištenje, nisu toliko fleksibilni ili jednostavno ne sliče na neke uobičajene jezike kojim se dizajner postupka inače koristi.

Uzimajući navedene zahtjeve u obzir, kao rješenje odabran je programski jezik C# i dinamička kompilacija dijelova programskog koda za evaluaciju uvjeta prijelaza.[7] Svi prijelazi unutar sustava nasljeđuju razred AbstractPattern i implementiraju postupak CheckConditions koji sustav pokreće prilikom izvršavanja svakog od prijelaza. Ukoliko je prijelaz po prirodi takav da mora donijeti odluku o izboru jedne ili više grana, izraz prijelaza će se nalaziti u bazi u tablici Transition u polju Condition. Sadržaj ovog polja u bazi je serijalizirani sadržaj razreda Condition i postupak CheckConditions vrši njegovu deserijalizaciju i pomoću ConditionEvaluator razreda vrši dinamičku kompilaciju, a zatim instancira objekt zadanog tipa i pokreće postupak Evaluate.

[7] Obzirom na mogućnosti .NET frameworka i način korištenja dinamičke kompilacije i dinamičkog instanciranja tako napisanih razreda, prijelaze je moguće napisati u bilo kojem .NET jeziku.



Slika 5.32: Razredi korišteni za analizu i raščlambu postupka

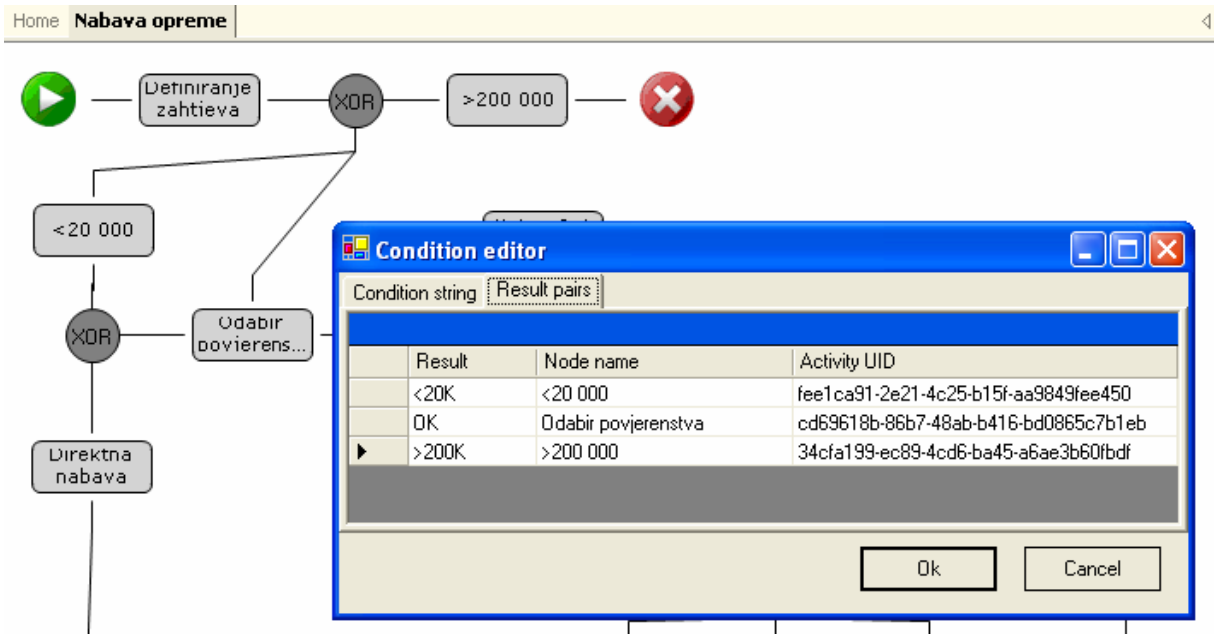
Dizajner postupka pri pisanju izraza prijelaza mora napisati vlastiti razred koji će implementirati sučelje ConditionInterface i apstraktni postupak Evaluate. Postupak Evaluate kao argument prima parametre sustava (instanca razreda WFSystemParameters) kroz koje može dohvatiti vrijednosti relevantne za odluku, poput vrijednosti nekog elementa u određenom dokumentu što je omogućeno postupkom GetDocumentValue. Danom postupku je potrebno navesti naziv pod kojim je dokument zaveden u bazi i XPath izraz kojim se može dosegnuti pojedini element iz XML dokumenta. Ukoliko se radi o složenijem upitu, postupkom GetDocumentNodes moguće je dohvatiti više čvorova, a ne samo jednu vrijednost. Što razred WFSystemParameters bude "bogatiji" dizajner sustava će na raspolaganju imati veću fleksibilnost u izboru podataka na osnovu kojih će moći graditi izraz prijelaza. Primjeri podataka koji bi se trebali moći dosegnuti kroz ove parametre su podaci o prethodnim aktivnostima, njihovim trajanjima, podaci o izvođačima i slično. Unos izraza prijelaza može se obaviti primjerice koristeći razvojnu okolinu MS Visual Studio ili neki drugi alat, u kojem se lako može provjeriti sintaktička ispravnost ili u kojem postoji dodatak za navođenje pri pisanju programskog koda kao što je Intelisense[8]. Izraz naknadno treba kopirati u za to predviđeno polje unutar uređivača postupka.

Sljedeći primjer prikazuje uvjet prijelaza koji se koristi u postupku nabave pri grananju postupka ovisno o vrijednosti nabave. Primjerice, ukoliko je vrijednost nabave veća od 200.000 Kn postupak Evaluate će vratiti vrijednost ">200K" što je u okviru za odabir procesa (Slika 5.33) povezano s procesom koji ima naziv ">200 000". Vrijednost nabave unesena je tijekom postupka i nalazi se zapisana unutar dokumenta Zahtjev_za_nabavu u elementu VrijenostNarudzbe koji je dijete elementa Zahtjev_za_nabavu.

[8] Da bi dizajner mogao kompilirati napisani razred mora ga prethodno povezati sa dinamičkom bibliotekom (dll datoteka) koja sadrži razred WfSystemParameters i sučelje ConditionInterface

```
using WFMEngine.Conditions;

public class CustomCondition : ConditionInterface
{
  public string Evaluate(WFSystemParameters parameters) {
      string val = parameters.GetDocumentValue("ZahtjevZaNabavu",
      "number(Zahtjev_za_nabavu/VrijednostNarudzbe/text())");
      double vrijednost = double.Parse(val);
        if (vrijednost>200000)
            return "<200K";
        else if (vrijednost>20000)
            return "OK";
        else
            return "<20K";
     }
}
```



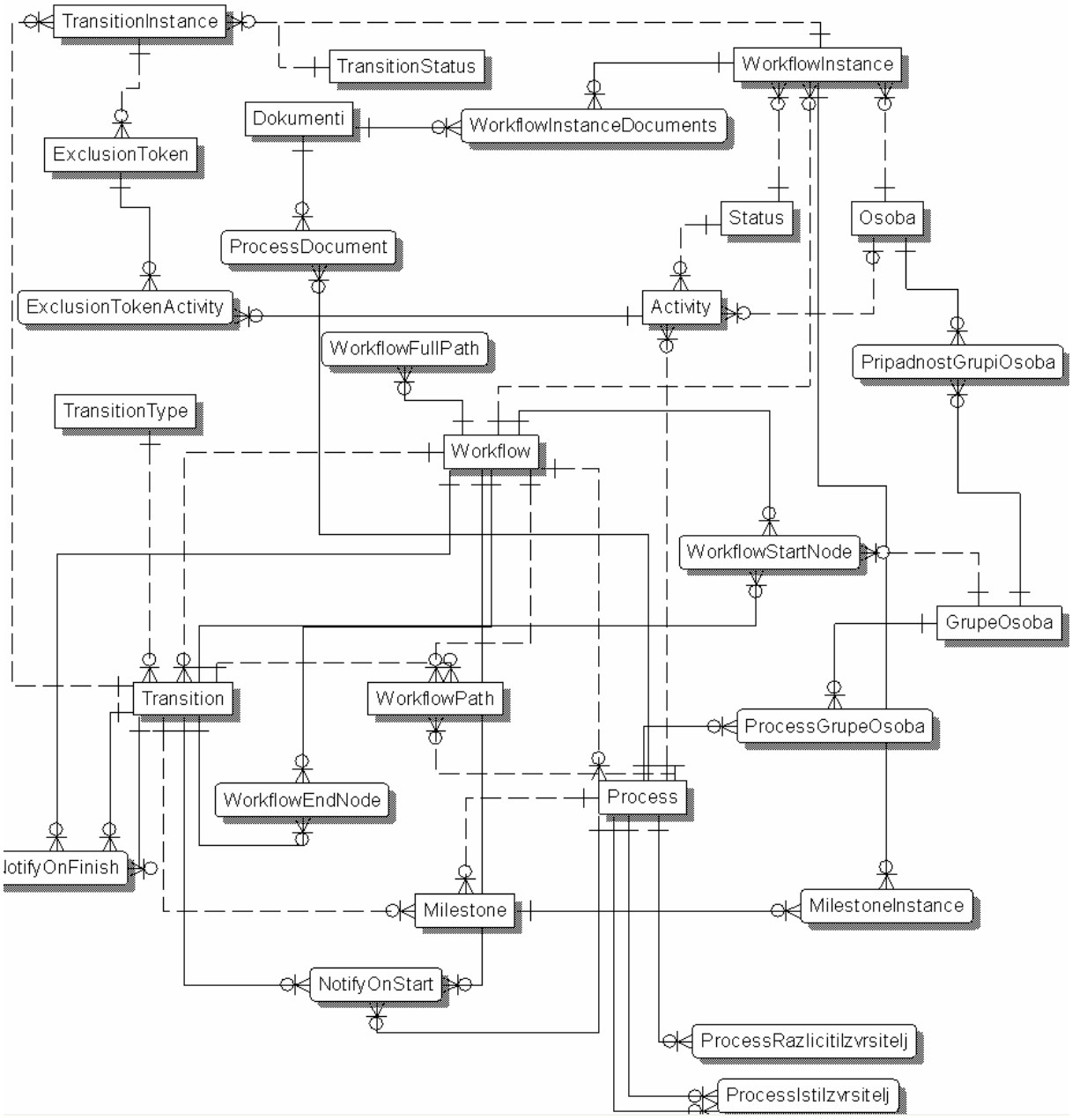
Slika 5.33 : Okvir za odabir procesa koji treba pokrenuti ovisno o rezultatu

U ovisnosti o rezultatu postupka Evaluate pokrenut će se onaj proces koji je u listi rezultata naveden pored traženog rezultata. Parovi tipa ResultCases navedeni su unutar unutar kolekcije Cases u razredu Condition.

6. Elementi tehničke dokumentacije

6.1. Osnovni model protoka poslova

6.1.1. Konceptualni model podataka



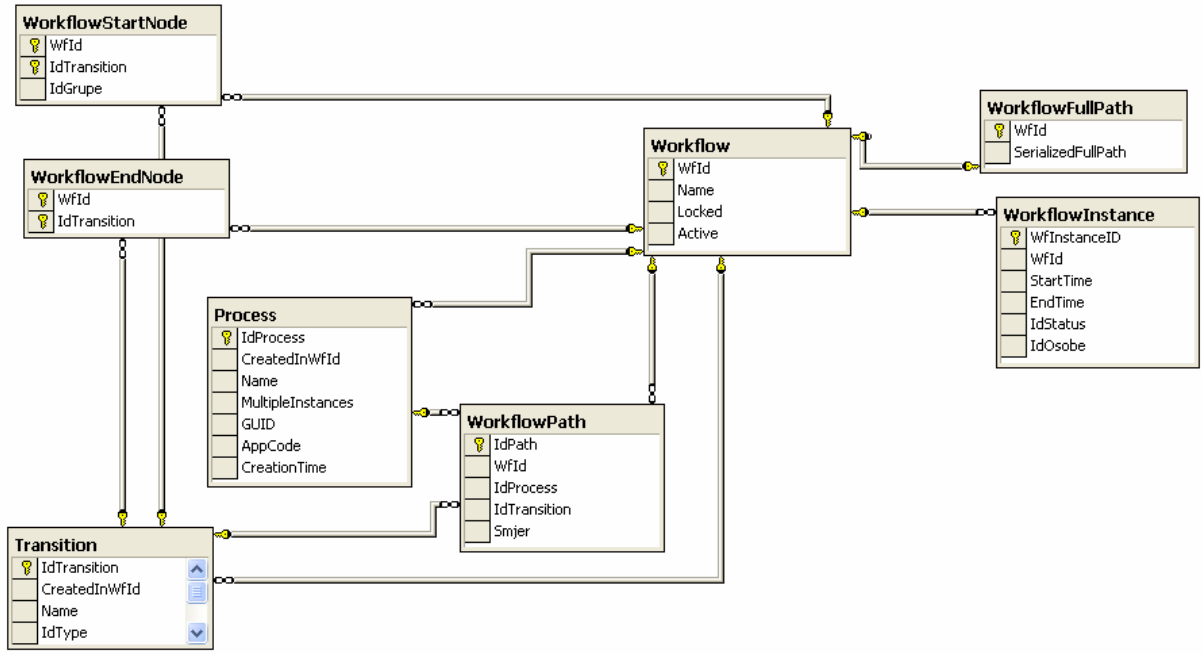
Slika 6.1 : Osnovni model baze podataka za sustav za upravljanje protokom poslova

Slika 6.1 prikazuje cjelokupni osnovni model baze podataka korištene unutar sustava za upravljanje protokom poslova. Ovisno o naglasku na rad pojedinog dijela sustava određene tablice bit će prikazane detaljnije u narednim poglavljima, a Tablica 6-1 sadrži kratki opis korištenih tablica.

Naziv tablice	Svrha
Workflow	Osnovni podaci o postupcima
WorkflowPath	Veze između procesa i prijelaza
WorkflowStartNode	Početni prijelazi u postupcima
WorkflowEndNode	Završni prijelazi u postupcima
WorkflowFullPath	Podaci specifični za prikaz postupka unutar aplikacije za grafičko uređivanje postupka.
WorkflowInstance	Osnovni podaci o svim pokrenutim instancama postupaka
Transition	Podaci o prijelazima i uvjetima prijelaza
TransitionInstance	Podaci o svim pokrenutim instancama prijelaza
TransitionStatus	Šifrarnik mogućih status instanci prijelaza
TransitionType	Šifrarnik vrsta prijelaza
Dokumenti	Popis dokumenata i njihovih naziva korištenih u centralnom repozitoriju dokumenata
Osoba	Popis korisnika sustava
GrupeOsoba	Grupe (uloge) osoba unutar sustava
PripadnostGrupiOsoba	Definira pripadnosti osobe pojedinoj grupi
Process	Osnovni podaci o procesima
Activity	Popis pokrenutih instanci pojedinih procesa
ProcessDocument	Popis dokumenata korištenih u pojedinom procesu
ProcessGrupeOsoba	Definira grupu kojoj izvršitelj procesa treba pripadati
ProcessIstiIzvrsitelj	Povezivanje procesa koji trebaju imati istog izvršitelja
ProcessRazlicitiIzvrsitelj	Omogućava funkcionalnost u kojoj dva procesa moraju imati različite izvršitelje
ExclusionToken	Ugradnja sinkronizacijskih točki i međusobno isključivanje procesa
ExclusionTokenActivity	Praćenje stanja pojedine aktivnosti u svrhu valjanje izvedbe sinkronizacijskih točaka i međusobnog isključivanja
NotifyOnStart	Popis procesa koji imaju utjecaj na točke sinkronizacije
NotifyOnFinish	Popis prijelaza koji imaju utjecaj na točke sinkronizacije
Milestone	Popis prekretnica i procesa koji ovise o prekretnicama
MilestoneInstance	Praćenje statusa pojedine prekretnice u nekoj instanci postupka
Status	Šifrarnik statusa aktivnosti i instance postupaka

Tablica 6-1: Kratki pregled korištenih tablica

6.1.2. Postupak – definicija i instanca



Svaki poslovni postupak (Workflow) je opisan imenom i ima jedinstveni identifikator (WfId). Ovisno da li je aktivan ili ne (atribut Active), moguće je započeti novu instancu postupka što će se zabilježiti u tablici WorkflowInstance. Ukoliko je postupak pokrenut barem jednom, vrijednosti stupaca Active i Locked bit će postavljena na 1, čime će se onemogućiti promjene u definiciji postupka kako se ne bi narušilo pravilno izvođenje postojećih instanci, te kako bi se održala konzistentnost u izvještajima o dotad izvedenim postupcima.[9] Naravno, kopija izmijenjenog postupka može se pospremiti pod nekim drugim nazivom.

WorkflowInstance		
WfInstanceId	int,autonumber	Jedinstveni broj instance pokrenutog postupka
WfId	int	Identifikator postupka koji je upravo pokrenut (instanciran)
StartTime	datetime	Vrijeme pokretanja postupka
EndTime	datetime	Vrijeme završetka postupka
IdStatus	int	Šifra stanja postupka
IdOsobe	Int	Identifikator osobe koja je započela danu instancu postupka

Svaki postupak mora ima svoj početni i završni prijelaz koji su zapisani u tablicama WorkflowStartNode i WorkflowEndNode. Put i prijelazi unutar nekog postupka zapisani su u tablici WorkflowPath koja sadrži trojke proces, prijelaz i orijentaciju prijelaza.

[9] Moguće je da će u skoroj budućnosti, nakon izvršenih daljnjih testiranja rada sustava, biti dopuštena promjena postupaka ukoliko promjene ne utječu na redoslijed procesa i prijelaza već se odnose samo na modifikacije uvjeta prijelaza ili na liste dokumenata koje se koriste te je iz tog razloga predviđeno postojanje, trenutno jednakih po funkciji, stupaca Active i Locked.

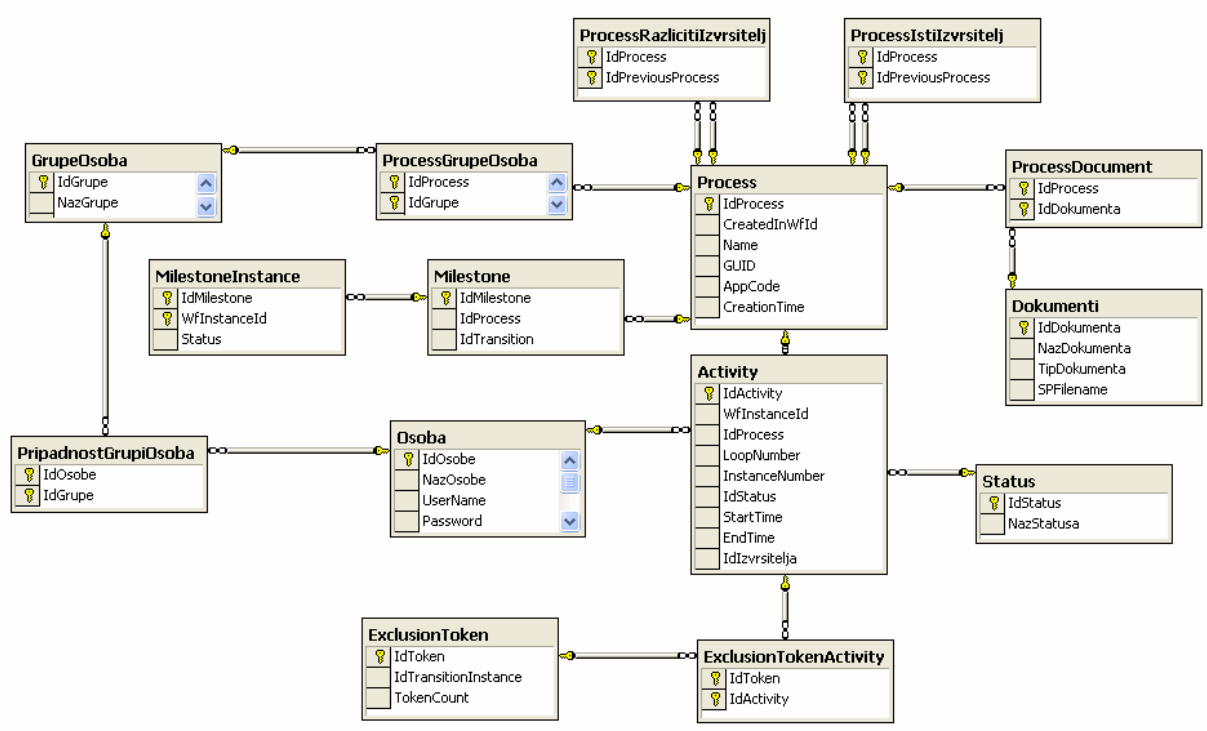
WorkflowPath		
IdPath	int,autonumber	Jedinstveni broj pojedinog luka
WfId	int	Identifikator postupka za koji se definira par u postupku
IdProcess	int	Identifikator procesa
IdTransition	int	Identifikator prijelaza
Smjer	int	Može biti -1 ili 1. Ukoliko je vrijednost -1 u tom slučaju orijentacija luka je od prijelaza prema procesu. Ukoliko je 1 u tom slučaju proces 'ulazi' u prijelaz.

Pojedini proces može biti prisutan u nekoliko različitih postupaka[10], pa tablica WorkflowPath sadrži i podatak o kojem se postupku radi. Nazovemo li danu spojnicu između postupka i prijelaza lukom (kao u terminima teorije grafova) onda smjer predstavlja orijentaciju luka i njegova je vrijednost 1 ako luk ulazi u prijelaz ili -1 ako luk izlazi iz prijelaza i ide prema procesu. Iako grafički na nekim mjestima, posebno kod sekvence, vizualno predočenje nije najpogodnije zbog viška elemenata u grafu, ovo pravilo mora biti zadržano zbog konzistentnosti modela. Ukoliko bi se dopustila mogućnost da se direktno mogu spojiti dva procesa ili dvije tranzicije, kvaliteta i efikasnost modela bila bi značajno narušena. Naravno, prilikom vizualnog prikaza „prazni“ procesi i sekvence mogu se izostaviti iz prikaza. Način rješavanja ovog problema izložen je u 6.3.1.

Grafički uređivač protoka poslova surađuje sa središnjim dijelom korištenjem posebnih modula za transformaciju grafičkog prikaza (provjera, unos i renumeraciju čvorova) u opisani model. Obzirom da unutar samog sustava za upravljanje poslovnim procesima koordinate prikaza koje su korištene u dizajnu postupka nisu bitne, u svrhu reverznog procesa (dakle ponovnog grafičkog prikaza) zamišljena je tablica WorkflowFullPath u kojoj se nalazi serijalizirani čitav grafički prikaz u formatu koji je prikladan grafičkom uređivaču protoka poslova.

[10] Slučaj je moguć ukoliko se neki postupak ugnijezdi unutar nekog drugog postupka. Promjena tako ugniježđenog postupka ne bi bila moguća iz danog postupka, a svaka njegova promjena reflektirala bi se u svim postupcima koji ga sadrže. Ako bi se dopustio samo takav scenarij, onda tablica WorkflowPath ne bi bila normalizirana jer bi IdPath jedinstveno identificirao pojedinu trojku (proces, postupak, smjer) te bi se trebala stvoriti nova tablica sa zapisima IdPath, WfId, čime bi prikazana tablica bila zamijenjena odgovarajućim pogledom (eng. view). Međutim, zbog mogućnosti da se u nekom trenutku omogući ponovo iskorištavanje nekih postojećih procesa, ali spojenih s različitim prijelazima unutar različitih postupaka, ovakav izgled tablice biva opravdan.

6.1.3. Procesi i aktivnosti



Okosnicu ovog dijela modela baze čini tablica Process koja sadrži podatke o procesima.

Process		
IdProcess	int, autonumber	Jedinstveni broj pojedinog procesa
Name	varchar	Naziv procesa koji će se prikazati korisniku
CreatedInWfId	int	Veza s postupkom u kojem je proces originalno kreiran. Brisanjem odnosno promjenom postupka mijenja se i dani proces
GUID	varchar	Jedinstveni identifikacijski broj procesa pridijeljen od strane alata za definiciju postupka. Koristi se u izrazima prijelaza kod izraza odluke
AppCode	text	Služi za posebne postavke procesa, npr. u svrhu određivanja izvršitelja kada nije određen pripadnošću grupom.
CreationTime	datetime	Vrijeme stvaranja procesa.

Svaki proces mora obavljati neka osoba. Izbor osobe se vrši na osnovu algoritama za raspodjelu poslova, a osoba mora biti iz neke od grupa koje su označene kao moguće grupe za izvršavanje tih poslova. Popis grupa osoba koje mogu izvršavati neki proces dan je tablicom ProcessGrupeOsoba.

ProcessGrupeOsoba	
IdProcess	Identifikator procesa za koji se definira grupa osoba koja ga može izvršavati

IdGrupe	Identifikator grupe koja može izvršiti dani proces (vezna tablica GrupeOsoba)

Svaka grupa ima svoj naziv i vezu s tablicom osobe, čime se definira pripadnost osobe nekoj od grupe. Pojedina osoba može pripadati u više grupa što opisuju tablice PripadnostGrupiOsoba i tablica Osoba. Tablica Osoba se može po potrebi proširiti dodatnim stupcima ili povezati dalje na tablice vezane za djelatnost i sl., ali daljnje povezivanje osoba ne utječe na organizaciju i izvršavanje procesa.

Ukoliko je iz nekog razloga potrebno da neki proces izvrši ista osoba, a da ta osoba inicijalno nije poznata u tu svrhu poslužit će tablica ProcessIstiIzvršitelj te u tom slučaju za dani IdProcess ne mora postojati zapis u tablici ProcessGrupaResursa.

Primjer: Ukoliko je npr. naručitelj knjige netko iz skupine Customers, onda plaćanje, koje, gledajući resurse, također pripada skupini Customer, mora obaviti ista osoba koja je izvršila naručivanje.

Analogno ovom slučaju, moguća je i potreba da neki proces mora izvršiti različita osoba od one koje je izvršavala neki prethodni proces što je evidentirano u tablici ProcessRazličitiIzvršitelj.
Primjer: Osoba koja se nalazi u statusu recenzenta može poslati svoj rad na konferenciju, ali ne može ocijeniti vlastiti rad.

ProcessIstiIzvrsitelj	
IdProcess	Identifikator procesa za koji se mora odabrati isti/različiti izvršitelj u odnosu na neki od prethodnih procesa.
IdPreviousProcess	Identifikator procesa čiji se izvršitelj mora uzeti u obzir prilikom odluke.

U nekim slučajevima, primjerice kod postupka nabave opreme, izvršitelj pojedinog procesa određuje se na osnovu podataka sadržanih u dokumentima koji su tijekom procesa popunjavani. U takvim slučajevima izvršitelj se pronalazi evaluiranjem XPath upita na danom dokumentu.

Svaki proces može raditi s određenim dokumentima, bez obzira o kojem se tipu dokumenta radi. Ukoliko je dokument u formi generičkog obrasca onda se rukovanje tim dokumentom može provesti generički alatom što je opisano u 6.5 i 6.6. S kojim dokumentima neki proces radi zapisano je u tablici ProcessDocument koji kao stranu vezu ima tablicu Dokumenti koja sadrži podatke o tipu dokumenta i njegovom jedinstvenom nazivu unutar repozitorija dokumenata. Potreba za jedinstvenim nazivom nastala je zbog jednostavnijeg pisanja uvjeta prijelaza u kojem se referenciraju dokumenti, kao i radi preglednijeg evidentiranja unutar SharePoint-a koji je iskorišten kao repozitorij dokumenata.

Prilikom pokretanja procesa u nekoj instanci postupka, kreira se novi zapis u tablici Activity.

Activity		
IdActivity	int, autonumber	Identifikator aktivnosti
WfInstanceId	int	Identifikator instance postupka u kojem je dana aktivnost stvorena
IdProcess	int	Identifikator procesa čija je ovo aktivnost
LoopNumber	int	Broj koji predstavlja koliko se puta isti proces ponovio u ovom postupku
InstanceNumber	int	Broj instance danog procesa unutar istog ponavljanja
IdStatus	int	Veza sa tablicom Status – predstavlja trenutno stanje aktivnosti
StartTime	datetime	Vrijeme kada je aktivnost kreirana (ne nužno i pokrenuta)
EndTime	datetime	Vrijeme završetka izvođenja aktivnosti
IdIzvrsitelja	int	Identifikator osobe koja izvršava ovu aktivnost

Ukoliko kreirana aktivnost predstavlja proces koji se već izvršavao unutar konkretnog postupka (npr. u slučaju ponavljanja) onda se danoj aktivnosti pridjeljuje odgovarajući broj ponavljanja (LoopNumber) kako bi se mogao razlikovati od prethodnog ponavljanja. Samim tim (vidi implementaciju ciklusa u 6.2.11) i sve sljedeće aktivnosti i prijelazi će nositi dani LoopNumber. Ukoliko se unutar postupka nađe predložak koji koristi višestruke instance nekog procesa, a zahtjeva sinkronizaciju, odgovarajući podatak o broju instance će se zapisati u polje InstanceNumber kako bi se međusobno mogli razlikovati, dok će istovremeno imati isti LoopNumber. Prilikom samog stvaranja aktivnosti, osim u slučaju kada se izvršitelj bira na osnovu podataka zapisanim u AppCode, izvršitelj još uvijek nije poznat, pa tek onog trenutka kad neka osoba prihvati (ili joj se dodijeli izvršavanje aktivnosti) danu aktivnost, ovo polje će biti ažurirano.

Činjenicu da neki proces ovisi o nekom prijelazu, to jest o prekretnici koja se postavlja u određenom prijelazu evidentirat će tablica Milestone. Obzirom da dana tablica definira samo ovisnost procesa, za konkretnu instancu postupka, bit će kreirana i instanca prekretnice zajedno sa stupcem Status koji može imati vrijednost 1 ukoliko je prekretnica dosegnuta. Vrijednost stupca Status 0 ili nepostojanje zapisa ukazivat će da prekretnica još uvijek nije dosegnuta.

Ukoliko je potrebno osigurati da se neki procesi ne izvode istovremeno kao što je to slučaj kod predloška Naizmjeničnog paralelnog izvršavanja, onda će se prilikom stvaranja aktivnosti koje predstavljaju te procese odmah napuniti i tablice ExclusionTokenActivity i ExclusionToken.

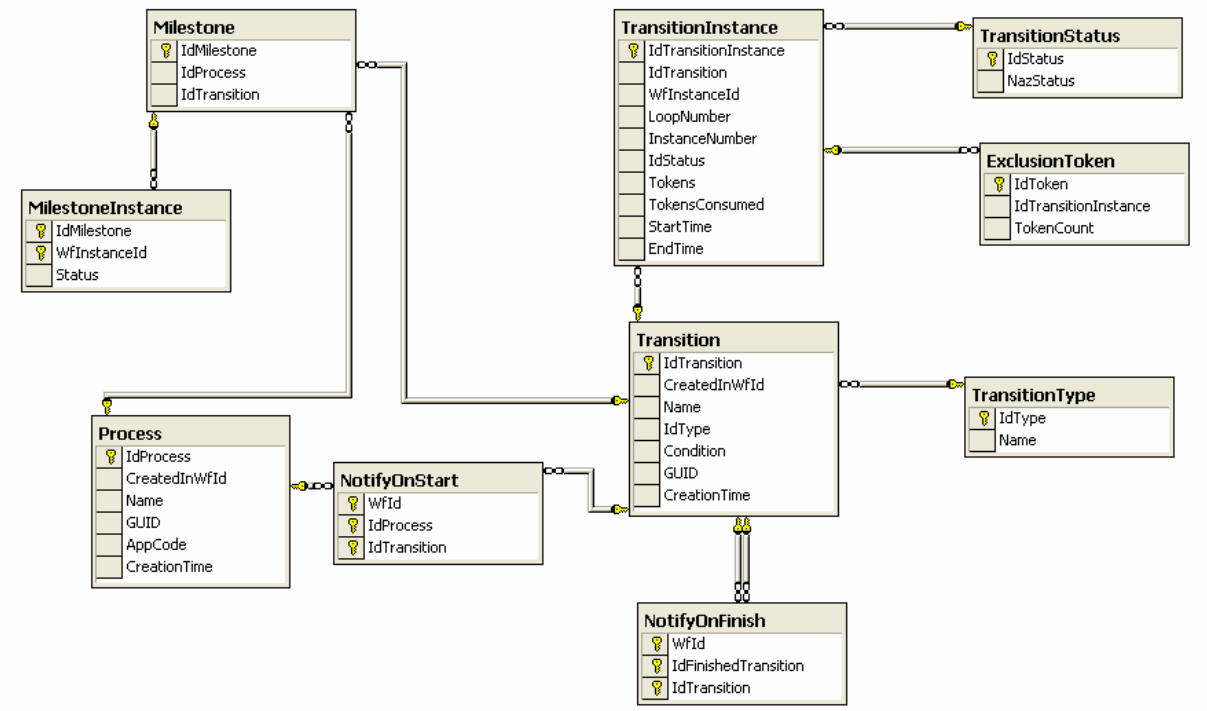
ExclusionTokenActivity	
IdTokena	Identifikator tokena koji služi za međusobno isključivanje dvaju ili više aktivnosti.
IdActivity	Identifikator aktivnosti uključene u međusobno isključivanje.

ExclusionToken

IdToken	Identifikator tokena za koji se vodi evidencija o stanju tokena.
IdTransitionInstance	Identifikator instance prijelaza iza kojeg slijede procesi čije se izvođenje mora međusobno isključiti
TokenCount	Stanje (brojilo) tokena.

Za stvaranje zapisa i ažuriranje vrijednosti tokena zadužen je odgovarajući prijelaz. TokenCount predstavlja brojač raspoloživih tokena i implementacijski može poprimiti vrijednosti 0 ili 1 te ima ulogu binarnog semafora, iako se po potrebi može poopćiti kako bi poprimao i druge vrijednosti. Ako je vrijednost tokena 1 znači da se jedna od međusobno isključivih aktivnosti može početi izvršavati. Obzirom da se najčešće radi o korisnikovom odabiru pitanje 'pravednog' odabira aktivnosti koja će iskoristiti token je u ovom trenutku nebitno. Sigurnost implementacije kod istovremenog izvršavanja je osigurana zaključavanjem kritičkih odsječaka.

6.1.4. Prijelazi i veza s procesima



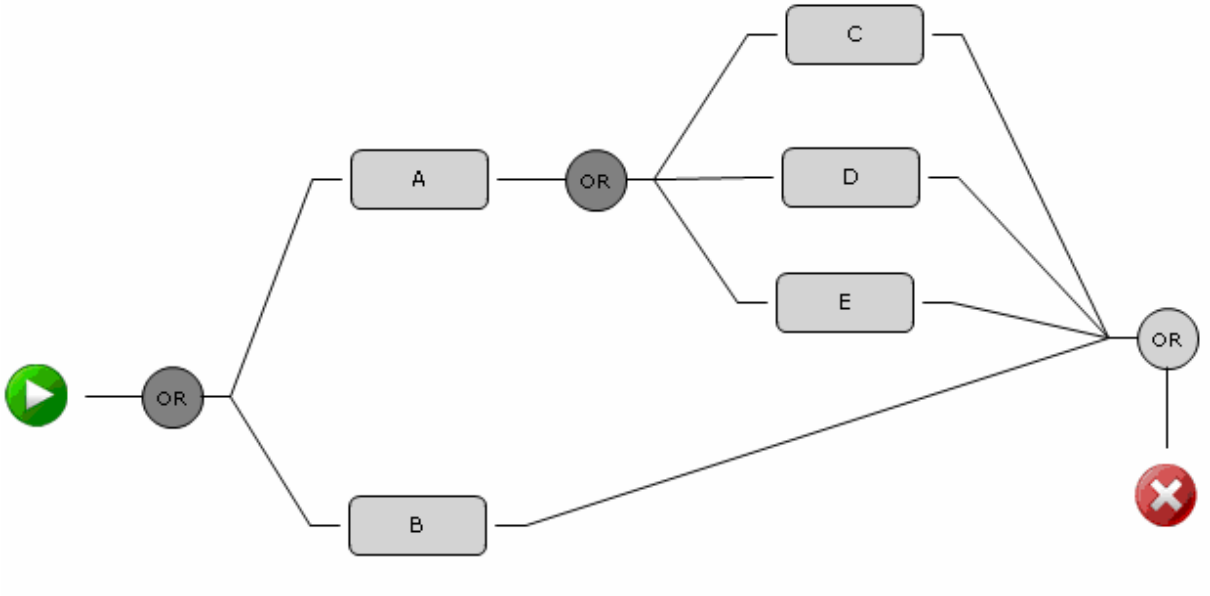
Svaki tip prijelaza (tranzicije) predstavlja jedan od u 3. poglavlju navedenih predložaka za kontrolu toka u poslovnim postupcima. Svaki prijelaz definiran je svojim imenom, tipom i uvjetom prijelaza (Condition).

Transition		
IdTransition	int, autonumber	Jedinstveni broj prijelaza pod kojim je zaveden u sustavu
Name	varchar	Naziv prijelaza. Može biti bilo što, ne prikazuje se korisniku, ali dizajneru poslovnog procesa olakšava razumijevanje i izradu postupka
IdType	int	Veza za tablicom TransitionType. Predstavlja jedan od 21 + 2 (start i stop prijelaz) moguća predloška
Condition	text	Serijalizirana klasa Condition koja u sebi sadrži uvjet prijelaza koji se evaluira s danim podacima kao i popis mogućih odabira kod grananja.
GUID	Varchar	Jedinstveni identifikaciji broj procesa pridijeljen od strane alata za definiciju postupka. Koristi se u izrazima prijelaza kod izraza odluke
CreationTime	Datetime	Vrijeme stvaranja procesa.

Posebno zanimljiv je stupac Condition koji predstavlja serijaliziranu klasu Condition koja u sebi sadrži uvjet prijelaza koji se u danom trenutku evaluira te se na osnovu zapisanog uvjeta i postojećih podataka u postupku određuje vrijednost rezultata. Ova klasa u sebi također sadrži i parove (vrijednost, proces) koji služe za odabir procesa koji će se izvršiti u slučaju grananja. Više o ovoj klasi može se vidjeti u Implementaciji predložaka (6.2).

Svaki put kada neki proces, zapravo njemu odgovarajuća aktivnost, završi potrebno je obavijestiti prijelaze koji slijede iza te aktivnosti. Povezanost se može ustanoviti iz tablice WorkFlowPath. Za pojedinu tranziciju potrebno je pronaći da li postoji odgovarajuća instanca te tranzicije obzirom na trenutnu instancu postupka i na broj ponavljanja. Potrebni podaci čuvaju se u tablici TransitionInstance.

TransitionInstance		
IdTransitionInstance	int, autonumber	Identifikator instance prijelaza
IdTransition	Int	Identifikator prijelaza za kojeg je stvorena dana instanca
WfInstanceId	Int	Pripadnost konkretnoj instanci postupka
LoopNumber	Int	Broj koji predstavlja koliko se puta dani prijelaz ponovio u ovom postupku
IdStatus	Int	Veza sa tablicom Status – predstavlja trenutno stanje aktivnosti
Tokens	Int	Broj tokena koji predstavljaju koliko aktivnosti se spaja u ovom prijelazu, odnosno koliko je prethodnik aktivnosti potrebno pričekati da završe, prije nego se može prijeći na sljedeću aktivnost
TokensConsumed	int	Broj konzumiranih tokena koji predstavlja broj završenih aktivnosti koje prethode ovom prijelazu.
StartTime	datetime	Vrijeme kada je instanca prijelaza stvorena
EndTime	datetime	Vrijeme završetka obrade prijelaza



Slika 6.2 : Primjer potrebe korištenja tokena za notifikaciju

Slika 6.2 prikazuje primjer u kojem postoje 2 grananja i samo jedna točka sinkronizacije te je potrebno iznaći rješenje za problem implementacije točke sinkronizacije, jer su ovisno o stanju vrijednosti moguće različite kombinacije pokrenutih procesa jer neki od procesa prethodnika uopće ne moraju biti pokrenuti. Rješenje ovog problema pronađeno je u korištenju tokena. Za primjer koji prikazuje Slika 6.2 potrebno je pri kreiranju svake od aktivnosti koja slijedi nakon višestrukog izbora (OR) povećavati broj tokena za sinkronizaciju. Ukoliko nakon drugog višestrukog izbora ne dođe do brisanja jednog tokena doći će do pogrešne sinkronizacije, to jest u ovom primjeru do nemogućnosti sinkronizacije, što dovodi do potrebe korištenja dviju tablica NotifyOnStart i NotifyOnFinish.

Pomoću vrijednosti polja Tokens svaki od prijelaza može znati koliko aktivnosti treba sinkronizirati bez obzira u kojem statusu su prethodne aktivnosti te da li su neposredni prethodnici uopće pokrenuti. Vrijednost polja TokensConsumed označava koliko je aktivnosti dovršeno. U 6.3.1 se detaljnije obrazlaže problem i način na koji se popunjavaju dane tablice.

NotifyOnStart	
WfId	Identifikator poslovnog postupka u kojem se koristi ova notifikacija
IdProcess	Identifikator procesa čije pokretanje mora doprinosi povećanju broja procesa koji u određenom trenutku čekaju na sinkronizaciju
IdTransition	Identifikator prijelaza koji predstavlja točku sinkronizacije, a jedan od uzročnika je i dani proces

NotifyOnStart tablica sadrži one procese koji slijede neposredno iza nekog od prijelaza koji predstavlja grananje. Obzirom da u trenutku grananja nije uvijek poznat broj aktivnosti koje slijede, a zbog specifičnosti nekih od predložaka obavijest o zahtjevu za povećanje tokena je smještenu u aktivnost, a ne u proces.

Tablica NotifyOnFinish sadrži podatke o onim prijelazima čiji završetak uzrokuje smanjenje vrijednosti tokena za jedan u nekom drugom prijelazu.

NotifyOnFinish	
WfId	Identifikator poslovnog postupka u kojem se koristi ova notifikacija
IdFinishedTransition	Identifikator prijelaza koji predstavlja točku sinkronizacije ili grananja čiji završetak uzrokuje smanjivanje broja tokena na nekom drugom prijelazu
IdTransition	Identifikator prijelaza koji predstavlja točku sinkronizacije

6.2. Implementacija predložaka za kontrolu protoka poslova

Dva najčešća problema koja se nameću prilikom dizajniranja modela za upravljanje poslovnim procesima su problem petlji, tj. proizvoljnih ciklusa i implementacije točaka sinkronizacije, bilo da se radi o običnom paralelnom izvođenju prilikom grananja, bilo prilikom izvođenja višestrukih instanci pojedinih procesa.

Problem petlji je riješen uvođenjem dodatnog polja InstanceNumber u tablice TransitionInstance i Activity čime je omogućeno linearno razvijanje petlje. Uvođenjem ovog polja, predložak za petlju se može zamijeniti s predloškom koji predstavlja ekskluzivni odabir gdje je jedan od mogućih izlaznih procesa neki već izvršeni proces, čime dolazi do ponavljanja, dok je drugi, koji se izvršava u slučaju da uvjet ne vrijedi izlazak iz petlje i ujedno prvi proces van nje. Prilikom ponavljanja određenog ciklusa pri svakom kreiranju instance procesa, to jest aktivnosti i instance prijelaza provjerava se da li je dotični proces odnosno prijelaz već postojao u konkretnoj instanci poslovnog postupka. Ako jest, dodjeljuje mu se za jedan veći broj instance čime se svaki korak zapravo pretvara u linearni nastavak poslovnog procesa. Ukoliko takav nije postojao, dodjeljuje mu se broj 1. Svaki put kada pojedina aktivnost završi dolazi do pokretanja odgovarajućeg prijelaza na osnovu broja instance poslovnog postupka i broja vlastite instance.

Na ovaj način elegantno je riješen problem diskriminatora koji prema svojoj definiciji treba pričekati završetak svojih pokrenutih prethodnika kako bi se mogao poništiti i ponovo iskoristiti u slučaju ciklusa. Na ovaj način nema potrebe za takvu implementaciju diskriminatora jer će nastati onoliko instanci diskriminatora koliko se puta ciklus ponovio. Svaki od nastalih diskriminatora imat će drugačiji broj ponavljanja (LoopNumber). Problem sinkronizacije, osim klasičnih problema sinkronizacije, pati i od problema evidencije svih prethodnika i trenutka njihovog izvršavanja. Kao što je već ranije navedeno u 6.1.4 problem je riješen korištenjem tokena. U nastavku slijedi opis implementacije svih predložaka za kontrolu toka.

6.2.1. Slijed

Implementacijski algoritam:

```
1. Pronaći proces sljedbenik ovog prijelaza.
2. Kreirati aktivnost za pronađeni sljedbenik.
3. Označiti prijelaz kao završen.
```

6.2.2. Paralelno grananje
Implementacijski algoritam:

```
1. Pronaći sve procese sljedbenike ovog prijelaza.
2. Za svaki od nađenih procesa kreirati novu aktivnost.
3. Označiti prijelaz kao završen.
```

6.2.3. Sinkronizacija
Budući da je predložak Sinkronizacija pojednostavljeni slučaj Sinkroniziranog spajanja, ovaj predložak nasljeđuje implementaciju navedenu u 6.2.7.

6.2.4. Ekskluzivni odabir
Implementacijski algoritam:

```
1. Dohvatiti uvjet prijelaza i izračunati vrijednost
rezultata. Rezultat je tipa string.


2. Unutar liste sljedbenika pronaći prvi par (rezultat,
proces) koji zadovoljava vrijednost rezultata i kreirati
aktivnost za pronađeni proces.


3. Ako takav par nije pronađen tada:


    3A. provjeriti da li postoji pretpostavljeni (default)
    proces koji se treba pokrenuti ukoliko nijedan od uvjeta
    nije zadovoljen.

    3B. Ako pretpostavljeni proces postoji stvoriti aktivnost
    za dani proces.


4. Ako je u koracima 1-3 došlo do pokretanja određene
aktivnosti označiti prijelaz kao dovršen, a inače odgoditi
odluku i označiti stanje prijelaza oznakom 'Naknadno donijeti
odluku'.[11]
```

6.2.5. Jednostavno spajanje
Obzirom na pretpostavku da se izvršavala samo jedna od mogućih grana, ovdje nema potrebe za sinkronizacijom te se ovaj predložak prilikom izvršavanja time pretvara u sekvencu.

6.2.6. Višestruki izbor
Implementacijski algoritam:

```
1.Dohvatiti uvjet prijelaza i izračunati vrijednost rezultata.
```

[11] Sustav će periodički provjeravati da li su se uvjeti za ovaj prijelaz promijenili te može li biti donesena dana odluka

2.Unutar liste sljedbenika pronaći sve parove (rezultat,proces) koji zadovoljavaju vrijednost rezultata i stvoriti aktivnosti za svaki od pronađenih procesa.

3. Ako takav par nije pronađen tada:

3.A provjeriti da li postoji pretpostavljeni (default) proces koji se treba pokrenuti ukoliko nijedan od uvjeta nije zadovoljen.

3.B Ako pretpostavljeni proces postoji stvoriti aktivnost za dani proces.

4. Ako je u koracima 1-3 došlo do pokretanja određene aktivnosti označiti prijelaz kao dovršen, a inače odgoditi odluku i označiti stanje prijelaza oznakom 'Naknadno donijeti odluku'.[11]

6.2.7. Sinkronizirano spajanje

Budući da kod ovog predloška nije dovoljno provjeriti stanje samo neposrednih prethodnika već cijeli niz prethodnika, rješenje koristi tokene za praćenje broja završenih prethodnika i ukupnog broja prethodnika.

Implementacijski algoritam:

1. Pri svakom pokretanju prijelaza uslijed završetka neke od aktivnosti konzumirati jedan token.
2. Ako je broj konzumiranih tokena jednak broju pridijeljenih tokena za ovaj prijelaz pronaći proces sljedbenik i stvoriti aktivnost za dani proces te označiti prijelaz kao završen.

6.2.8. Višestruko spajanje

Budući da će za svaki završetak aktivnosti koja prethodi ovog prijelaza doći do pokretanja aktivnosti koja predstavlja instancu procesa sljedbenika, u ovom predlošku nije potrebnu brinuti se o sinkronizaciji te ovaj predložak nalikuje sekvenci. Međutim, obzirom da sustav mora podržavati Implicitni završetak, potrebno je voditi evidenciju o stanju svih aktivnosti i instanci prijelaza pa je implementacijski algoritam za ovaj predložak sljedeći:

1. Pri svakom pokretanju prijelaza uslijed završetka neke od aktivnosti konzumirati jedan token
2. Pronaći proces sljedbenik i pokrenuti novu instancu procesa. Nova instanca treba imati isti broj ponavljanja (LoopNumber) kao i trenutni prijelaz
3. Ako je broj konzumiranih tokena jednak broju pridijeljenih

```
   tokena označiti prijelaz završenim
```

Alternativno rješenje problema Implicitnog završetka moglo bi se riješiti i označavanjem predloška višestrukog spajanja završenim već nakon prvog pokretanja, pa bi Implicitni završetak ovisio o stanju procesa prethodnika. Takvo rješenje je korektno, iako bi korisnika moglo navesti na krivi dojam prilikom pregleda stanja postupka.

6.2.9. Diskriminator

Implementacija:

```
1.Provjeriti da li je prijelaz označen kao dovršen.

2.Ako uvjet pod 1 nije istinit, pronaći i kreirati aktivnost
koja predstavlja instancu pronađenog procesa sljedbenika i
označiti prijelaz kao završen.
```

Ovakva implementacija ne pati od problema opisanog u [4] koji bi se mogao manifestirati kod ciklusa. Naime, ovakvom implementacijom ne treba pratiti da li su sve prethodne aktivnosti ispred diskriminatora završile jer će u slučaju ciklusa sve ponovljene aktivnosti dobiti novi broj instance, pa će se pojaviti i novi diskriminator.
Budući da se kod prvog pokretanja diskriminator označava kao dovršen, problem implicitnog završetka je riješen te time nema potrebe za korištenjem tokena. Za razliku od višestrukog spajanja gdje bi takvo označavanje pri prikazu stanju cijelog postupka djelomično odstupalo od stvarnog stanja, ovdje je takvo rješenje sasvim korektno jer je diskriminator dovršen svojim prvim pokretanjem i sva ostala pokretanja se mogu ignorirati.

6.2.10. Čekanje M od N

Ovaj predložak je varijanta sinkroniziranog spajanja u kojoj ne treba čekati sve prethodnike već samo njih M od mogućih N
Algoritam implementacije:

```
1. Pri svakom pokretanju prijelaza uslijed završetka neke od
aktivnosti konzumirati jedan token

2. Ako je broj konzumiranih tokena jednak M, pronaći proces
sljedbenik i pokrenuti odgovarajuću aktivnost te označiti
prijelaz završenim.

3. Ako je prijelaz označen kao završen završiti s obradom ovog
prijelaza.
```

Zadnji korak u implementaciji može se premjestiti na početak algoritma, jer je sa stajališta definicije prijelaz završen onog trenutka kada M njegovih prethodnika završi. Da li će se evidentirati ukupni broj završenih prethodnika ovisi o potrebama izvještaja o stanju sustava.

6.2.11. Proizvoljni ciklusi

Opisani model dozvoljava proizvoljna ponavljanja sve dok se poštuje pravilo da se spojnicom povezuju prijelaz i proces. Ukoliko jedan od lukova koji vode iz prijelaza prema procesu vodi na već izvršeni proces onda će prilikom kreiranja dane aktivnosti dotičnoj aktivnosti biti pridijeljen za jedan veći broj ponavljanja (LoopNumber). Sa stajališta jezgre sustava ovakvo ponavljanja bit će tretirano kao da se petlja raspetljala linearno onoliko puta koliko se puta

ponavljanje izvelo. Gore navedeni način izvedbe, uz korištenje brojača ponavljanja omogućava implementaciju ovog predloška identičnu kao kod predloška Ekskluzivni odabir.

6.2.12. Implicitni završetak

Ovaj predložak je implementiran na način da jezgra sustava za upravljanje protokom poslova periodički provjerava stanje pojedinog postupka te ukoliko su sve aktivnosti unutar postupka završene, sustav će proglasiti postupak završenim.

6.2.13. Višestruke instance bez sinkronizacije

Implementacijski algoritam:

```
1. Dohvatiti uvjet prijelaza i izračunatu vrijednost
pretvoriti u broj.

2. Pokrenuti onoliko aktivnosti istog procesa kolika je bila
vrijednost izračunatog broja. Pokrenute aktivnosti imaju isti
broj ponavljanja, ali različiti broj instance.

3. Označiti prijelaz kao završen.
```

6.2.14. Višestruke instance s unaprijed poznatim brojem instanci

Implementacija ovog predloška se svodi na implementaciju višestrukih instanci s brojem instanci poznatim u trenutku izvođenja.

6.2.15. Višestruke instance s brojem instanci poznatim u trenutku izvođenja

Implementacija:

```
1. Prilikom aktivacije provjeriti stanje u kojem se prijelaz
nalazi.


2. Ako je prijelaz u stanju čekanja (situacija u kojoj su
instance kreirane te je neka od instanci upravo završila) tada
konzumirati jedan token.


    2A. Ako je broj konzumiranih tokena jednak broju
    pokrenutih tokena pokrenuti aktivnost za proces
    sljedbenik i označiti prijelaz završenim.


3. Ako prijelaz nije završen i ako nije u stanju čekanja
(situacija u kojoj je potreno pokrenuti instance).


    3A. Evaluirati izraz prijelaza na osnovu kojeg se dobije
    potreban broj instanci n.


    3B. Pokrenuti n instanci istog procesa (stvorene
    aktivnosti imaju isti broj ponavljanja i različite
```

```
    brojeve instanci)

    3C. broj stvorenih tokena (pridijeljenih ovom prijelazu)
    postaviti na n.
```

6.2.16. Višestruke instance s nepoznatim brojem instanci

Obzirom da se nove instance mogu pokrenuti u bilo kojem trenutku neovisno o stanju dotad pokrenutih, implementacija ovog predloška je malo složenija. Kod ovog predloška potrebno je razlikovati proces za koji dolazi do pokretanja višestrukih instanci i proces sljedbenik koji će se pokrenuti nakon što se sve željene instance prethodno navedenog procesa pokrenu i završe. Također potrebno je omogućiti pokretanje minimalnog, tj. inicijalnog broja instanci ukoliko je takva vrijednost zadana.

Implementacijski algoritam je sljedeći:

```
1. Pri aktivaciji prijelaza provjeriti da li je dosad stvorena
neka instanca.

2. Ako nema stvorenih instanci procesa provjeriti da li je
zadan broj inicijalnih instanci.
  2A. Stvoriti traženi broj instanci i postaviti odgovarajući
  broj tokena pridijeljenih ovom prijelazu

  2B. Ako nema ograničenja po minimalnom broju instanci,
  stvoriti novu instancu i omogućiti je u formi predloška
  Neposredni izbor

3. Ako već postoje instance (situacija u kojoj je prijelaz
aktiviran uslijed završetka neke od instanci) tada:

    3A. Provjeriti da li je broj konzumiranih tokena jednak
    broju stvorenih tokena. Ako da, omogućiti neposredni
    odabir između stvaranja nove instance i prelaska na
    proces sljedbenik.
```

6.2.17. Neposredni izbor

Implementacija ovog predloška realizirana je stvaranjem tokena za međusobno isključivanje. Algoritam se sastoji od dva dijela, gdje je prvi vezan za izvršavanje prijelaza, a drugi za postupak provjere stvorenog tokena prilikom pokretanja neke aktivnosti. Algoritam je sljedeći:

```
1. Pri aktivaciji prijelaza stvoriti novi token za međusobno
isključivanje.

2. Za sve procese sljedbenike stvoriti odgovarajuće aktivnosti
```

```
i unijeti podatke u tablicu međusobnih isključivanje pomoću
stvorenog tokena.


3. Prilikom pokretanja aktivnosti provjeriti status tokena.
Postupak je sinkroniziran tako da ne može doći do istodobnog
izvršavanja kritičnih odsječaka.


4. Ako je vrijednost tokena veća od 0 tada
     4A. Smanjiti vrijednost tokena
     4B. Trajno onemogućiti aktivnosti koje su bile ovisne o
     tom tokenu.


5. Ako je vrijednost tokena 0
     5A. Postaviti vlastito stanje na Onemogućen
     5B. i odgoditi/onemogućiti vlastito izvršavanje
```

Gore navedeni postupak većinom se obavlja unutar pohranjene procedure. Poziv procedure nalazi se unutar postupka koji je označen kao kritični odsječak te je primijenjeno međusobno isključivanje zaključavanjem kritičnog odsječka. Prilikom pokretanja svake od aktivnosti dolazi do provjere vrijednosti tokena za međusobno isključivanje te se u slučaju da je vrijednost tokena jednaka nula, što predstavlja slučaj u kojem se aktivnost ne smije započeti, početak odgađa na način da se dana aktivnost onemogući. Ukoliko je vrijednost tokena veća od nule (odnosno jednaka 1), tada se prije samog početka aktivnosti, sve ostale aktivnosti ovisne o istom tokenu onemoguće postavljanje stanja na *Onemogućen* (*Disabled)*.

6.2.18. Naizmjenično paralelno izvođenje

Implementacija ovog predloška je slična neposrednom izboru, ali onemogućavanje ostalih aktivnosti nije trajno, već po završetku neke aktivnosti može doći do aktiviranja ostalih aktivnosti uključenih u međusobno isključivanje. Također kod ovog predloška osim procesa koji se trebaju izvoditi naizmjenično, postoji i proces sljedbenik koji će se izvršiti nakon što se svi prethodni procesi izvrše.

Implementacijski algoritam:

1. Pri aktivaciji prijelaza provjeriti da li je token za međusobno isključivanje već stvoren.

   1A. Ako nije, stvoriti novi token i za sve procese sljedbenike stvoriti odgovarajuću aktivnosti i unijeti podatke u tablicu međusobnih isključivanje pomoću stvorenog tokena

   1B. Prilikom pokretanja aktivnosti provjeriti status tokena. Postupak je sinkroniziran tako da ne može doći do istodobnog izvršavanja kritičnih odsječaka

   1C. Ako je vrijednost tokena veća od 0, smanjiti vrijednost tokena i onemogućiti aktivnosti koje su ovisile o tom tokenu

   1D. Ako je vrijednost tokena 0, onemogućiti vlastito stanje i odgodi vlastito izvršavanje

2. Ako je token bio stvoren (to implicira da je prijelaz aktiviran uslijed završetka jedne od aktivnosti koje su uključene u međusobno isključivanje) tada.

   2A. Vrijednost tokena povećati za jedan i omogućiti aktivnosti koje su u statusu čekanja.

   Napomena: Prva aktivnost koja bude pokrenuta onemogućit će ostale kao što je opisano u prethodnom koraku. Ukoliko nema aktivnosti koje čekaju na preuzimanje tokena za međusobno isključivanje, pokrenut će se aktivnost za proces koji slijedi iza ovog prijelaza.

6.2.19. Prekretnica

Za ugradnju prekretnice koriste se tablice Milestone i MilestoneInstance. Implementacijski algoritam je sljedeći:

```
1. Kreirati novu instancu prekretnice (unijeti zapis u tablicu
MilestoneInstance) i postaviti Status na 1.


2. Pronaći sve aktivnosti koje su stvorene u danoj instanci
postupka a ovise o prekretnici, a u statusu su čekanja na
prekretnicu.


3. Za pronađene aktivnosti promijeniti status kako bi se mogle
početi izvršavati.
```

Napomena: Prilikom kreiranja aktivnosti, postupak kreiranja aktivnosti će provjeriti da li proces čija je aktivnost stvorena ovisi o nekoj prekretnici. Ukoliko je to slučaj i prekretnica nije aktivna, to jest. zapis o instanci dane prekretnice ne postoji u bazi ili ima vrijednost 0, aktivnosti neće biti pridijeljen uobičajeni status već će joj se pridijeliti status čekanja na prekretnicu.

6.2.20. Otkazivanje aktivnosti

Otkazana aktivnost ponovo se mora dodijeliti nekoj osobi na izvođenje, tj. preciznije rečeno, neka od osoba iz grupe mora preuzeti aktivnost), a trenutna verzija korištenih dokumenata se vraća na prethodnu verziju (prije preuzimanja) aktivnosti.

6.2.21. Otkazivanje slučaja

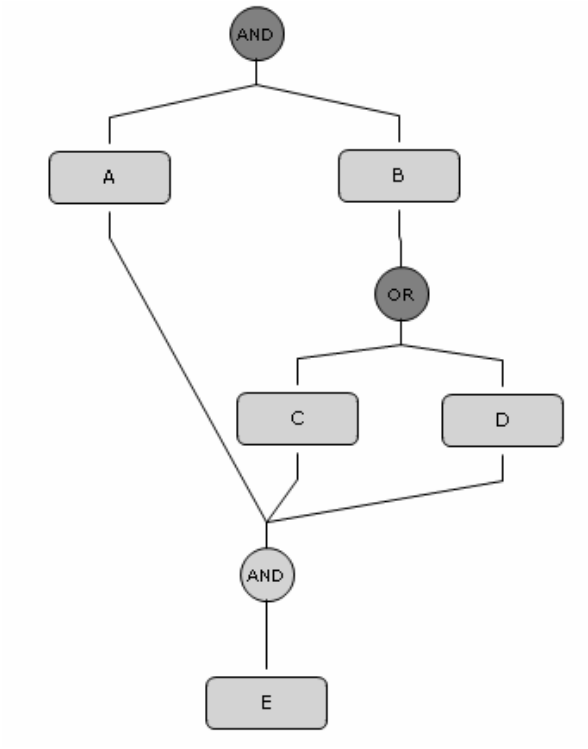
Status instance postupka se postavlja vrijednost koja simbolizira otkazani postupak (tj. poprima vrijednost cancelled), a dokumenti zadržavaju stanje koje su u tom trenutku imali.

6.3. Analiza i raščlamba poslovnih postupaka

6.3.1. Problem analize i raščlambe kreiranog postupka

Ulazni postupak može se promatrati kao varijanta usmjerenog grafa za koji treba provjeriti poštuje li zadana ograničenja te, ukoliko je to ostvareno treba pospremiti u bazu sve njegove elemente i međusobne veze i ovisnosti pojedinih elemenata. Iako je obilazak grafa klasični problem iz teorije grafova, postupak u ovom slučaju je nešto složeniji zbog međusobne ovisnosti nesusjednih čvorova u grafu.

Za ilustraciju problema poslužit će jednostavan primjer (Slika 6.3). Neka u nekom postupku postoji pet procesa označenih s A, B, C, D i E, paralelno grananje označeno tamno sivim krugom u kojem piše AND, jedan višestruki izbor (tamno sivi krug OR) i jedno sinkronizirano spajanje (svjetlo sivi krug AND).



Slika 6.3 : Jednostavan primjer problema točke sinkronizacije

Prikazani postupak zadovoljava sva unaprijed opisana pravila te je "samo" potrebno izvršiti njegovu raščlambu u odgovarajuće tablice u bazi podataka. Promotri li se postupak lako se može uočiti da ponašanje postupka nije jednoznačno određeno.

Dok se prijelaz 'Višestruki odabir' ne obavi, nije moguće odrediti hoće li se izvesti proces C, proces D ili i C i D. Prijelaz dopušta sve tri mogućnosti, ali odabir ovisi o vrijednostima unutar postupka koje su mogle biti postavljen primjerice u procesu B. Proces E će se izvesti tek nakon što se izvrši prijelaz 'Sinkronizirano spajanje' koji mora pričekati završetak svih procesa koji mu prethode.

Neka je na primjer proces A upravo završio i svojim završetkom aktivirao prijelaz koji slijedi, to jest Sinkronizirano spajanje. U tom trenutku stanje postupka može se podijeliti u nekoliko kategorija:

1. B je u postupku izvršavanja
2. B završen, a višestruki prijelaz je aktivirao:
    a. proces C
    b. proces D
    c. proces C i proces D

Pridoda li se u sumu proces A, sinkronizirano spajanje u slučajevima 2a i 2b mora čekati završetak dvaju procesa, dok u slučaju 2c mora čekati završetak triju procesa. Međutim upravo zbog situacije broj 1 nije moguće gledati samo stanja neposrednih prethodnika, jer bi to izazvalo nepravilno ponašanje sustava ukoliko neki od procesa neposrednih prethodnika uopće nije započet ili možda neće biti aktiviran.

Kako bi se izbjegla situacija da sustav u trenutku donošenja odluka vrši opsežnu analizu grafa postupka stvarajući ga ponovno iz normaliziranih tablica i vraćajući se unaprijed nepoznat broj koraka unatrag, uveden je princip brojača, to jest tokena.

Svaka aktivacija neke instance sinkroniziranog spajanja konzumirat će jedan token, a nastavak će uslijediti tek nakon što je broj konzumiranih tokena jednak ukupnom broju tokena pridijeljenih danom prijelazu. U gore opisanom postupku broj tokena za sinkronizirano spajanje je 2 u slučajevima 2a i 2b, odnosno 3 u slučaju 2c.

Problem pronalaska ukupnog broja tokena je moguće analizirati na istom primjeru. Logično je za pretpostaviti da će svako grananje u ukupnom broju tokena doprinijeti s onoliko tokena koliko je procesa nakon grananja kreirano. Na taj način, prvo grananje u postupku sa slike bi sinkroniziranom spajanju pridodalo 2 tokena.

Međutim, ukoliko bi i višestruki izbor prilikom kreiranja obaju procesa dodao sinkronizaciji 2 tokena, ukupan broj tokena bi narastao na 4 što je nemoguće, tj. postupak nikad ne bi završio, jer točka sinkronizacije nikad ne bi konzumirala četiri tokena.

Iz toga razloga, svako grananje u onom dijelu grafa koji je već uzrokovao povećanje broja tokena u nadolazećoj sinkronizaciji će stvoriti za jedan manje token nego što je broj kreiranih procesa. Uvjet se, uz pravilnu sinkronizaciju rada s tokenima, može preformulirati na način da će takvo grananje izbrisati jedan token, a zatim ih stvoriti onoliko koliko je to grananje kreiralo procesa.

6.3.2. Algoritam analize i raščlambe kreiranog postupka

U skladu s opisom problema i predloženim rješenjem u prethodnom odlomku, slijedi opis algoritma za raščlambu postupka. Algoritam se provodi u dvije faze. U prvoj fazi obavlja se šetnja po grafu postupka pri čemu se u bazu upisuju podaci o svakom čvoru te se upisuju njihove međusobne veze.

Budući da je u grafičkom prikazu moguće međusobno direktno spojiti dva prijelaza ili dvije aktivnosti u danom obilasku se na takva mjesta zbog konzistentnosti ubacuju tzv. prazni proces (proces koji ne zahtijeva izvođača i završava odmah nakon pokretanja) odnosno prijelaz Sekvenca.

Prilikom prve faze također dolazi do evidentiranja n-torki koje predstavljaju tokene za točke sinkronizacije kao i do evidentiranja svih prekretnica i svih procesa koji ovise o nekoj od prekretnica. Druga faza se sastoji od unosa prikupljenih n-torki tokena i prekretnica.

Stvarni model sadrži više podataka, ali zbog jednostavnosti opisa postupka može se pretpostaviti da svaki čvor sadrži sljedeće elemente:

- `Data` : opći podaci
- `Type` : tip čvora
- `Next` : lista izlaznih čvorova (popis čvora koji slijede iza danog čvora)
- `Processed` : (istina ili laž) označava da li je čvor već evidentiran tijekom šetnje (npr. kod slučajeva kad se više grana spaja u jednu točku sinkronizacije)
- `Inserted` : (istina ili laž) označava da li je čvor upisan u bazu ili ne

Unutar algoritma bit će korišteni sljedeći elementi:

- `NotifyTokens` : trenutna lista procesa koji donose povećanje tokena prvoj sljedećoj točki sinkronizacije na koju obilazak grafa naiđe.
- `RemoveTokens` : trenutna lista prijelaza koji uzrokuju smanjenje vrijednosti tokena za jedan prvoj sljedećoj točki sinkronizacije na koju obilazak grafa naiđe.
- `NTokens` : skup svih uređenih parova (proces , točka sinkronizacije)
- `RTokens` : skup svih uređenih parova (prijelaz , točka sinkronizacije)
- `HandleToken` : (istina ili laž) – da li trenutni proces uzrokuje povećanje tokena nekoj točki sinkronizacije
- `Milestones` : skup svih prekretnica
- `MilestoneDependencies` : skup svih uređenih parova (proces , prekretnica) koji sadrže sve procese koji ovise o nekoj prekretnici.
- `InsertIntoDB` : (istina ili laž) – označava da li se u danom prolazu vrši ubacivanje elemenata u bazu ili ne

Budući da je algoritam rekurzivan pri svakom pozivu se prenose kopije listi[12], dok se skupovi uređenih parova i prekretnica prenose po referenci. U opisu algoritma u Tablica 6-2 pretpostavlja se da liste i skupovi ne mogu sadržavati duplikate, što se u implementaciji dodatno provjerava.

[12] Ostvareno korištenjem razreda Queue i korištenjem postupka Clone()

```
WalkOnGraph(element,predecessor,InsertIntoDB,NotifyTokens,RemoveTokens,
            HandleToken,NotifyTokens,RemoveTokens,NTokens,RTokens,
            Milestones, MilestoneDependencies)

Ako je InsertIntoDB istina:
    Ako je element.Inserted laž:
          Spremiti opće podatke elementa u bazu
          i postaviti Inserted = istina
    Ako prethodnik postoji:
          Evidentirati vezu prethodnik (predecessor) -element.
          Postupak će provjeriti tip i poredak prethodnika i elementa
          te ukoliko su istog tipa dodati prazni prijelaz ili sekvencu.
Inače:
     Ako je tip elementa grananje:
          HandleToken = istina
          Ako je broj elemenata u listi NotifyTokens>0:
              Dodati element u listu RemoveTokens
     Ako je tip elementa neka od sinkronizacija:
          Prekopirati sadržaj iz NotifyTokens u NTokens
          Isprazniti NotifyTokens
          Prekopirati sadržaj iz RemoveTokens u RTokens
          Isprazniti RemoveTokens
     Inače ako je HandleToken istina:
          Dodati trenutni element u NotifyTokens
     Inače:
          HandleToken = laž
     Ako je element prekretnica:
          Dodati element u Milestones
     Ako je element proces ovisan o prekretnici:
          Dohvatiti GUID milestonea o kojem je proces ovisan i
          ubaciti par (prekretnica,element) u MilestoneDependencies
Ako je vrijednost Processed za dani element istina:
     Završiti poziv funkcije
Inače:
     Postaviti Processed = istina

Lista sljedbenika = element.Next
Za svaki element iz liste sljedbenika pozvati WalkOnGraph
```

Tablica 6-2 : Algoritam analize i raščlambe postupka

Gornji algoritam se poziva dva puta, a nakon toga slijedi spremanje zapisa o tokenima u bazu podataka.

6.4. Praćenje statusa pojedinih postupaka

Status pojedinih postupaka moguće je dohvatiti korištenjem za to predviđenih web servisa. Autorizirani korisnik će kroz uzastopne pozive web servisa odabrati za koji postupak želi pogledati popis stanja nakon čega će dobiti popis instanci danog postupka za određeno vremensko razdoblje. Odabirom određene instance, korisniku će se grafički prikazati postupak pri čemu će u ovisnosti o statusu prijelazi i procesi biti obojani odgovarajućim bojama.

Osim grafičkog prikaza koji se ostvaruje kroz vanjski modul, ovlaštenom korisniku će biti omogućen i uvid u sadržaj pojedinog dokumenata, odnosno dijelova dokumenta što će biti izvedeno kroz ugrađeni uređivač XML dokumenata, ukoliko se radi o generičkom obrascu, odnosno bit će mu ponuđena datoteka za preuzimanje ukoliko je dokument u nekoj drugoj formi.

6.5. Rad s dokumentima

Unutar svakog poslovnog postupka korisnici rade s dokumentima koji mogu biti različitih tipova i namjena te pritom treba voditi računa o nekoliko raznorodnih zahtjeva. Sasvim je razumljivo da neke dokumente mogu vidjeti samo određeni korisnici i to samo u određenom trenutku, tj. u određenom procesu. Ukoliko se implementiraju prethodno navedeni predlošci za kontrolu toka, onda je jasno da može doći do otkazivanja pojedinih aktivnosti te da stanja dokumenata u tom slučaju treba vratiti na prethodnu verziju. Pored toga, za administriranje sustava i generiranje izvješća poželjno je da se svaka promjena unutar dokumenata evidentira kako bi se pružio bolji uvod u dane promjene. Za rješenje koje ovaj rad prikazuje koristi se kombinacija vlastitog rješenja i mogućnosti koje po pitanju verzija pruža Microsoft SharePoint. Potonji je odabran zbog mogućnosti praćenja verzija dokumenata te iz strateških razloga integracije različitih rješenja unutar SUSAP projekta.

Također, velik broj dokumenata koji se koriste unutar postupaka su različite vrste obrazaca koje sadrže jednostavna polja nalik onima na papirnatim obrascima koje korisnici popunjavaju. Osim što bi bilo efikasnije te podatke na određeni način normalizirati, ponekad kontrola toka može ovisiti o vrijednosti unutar pojedinog polja u takvom obrascu. Za takve dokumente, koji predstavljaju obrasce, razvijen je XML Web uređivač koji omogućava jednostavno popunjavanje tih podataka, naravno u ovisnosti o dozvolama pojedinog korisnika u određenom trenutku. Sam izgled dokumenta opisuje se XML Shemom nakon čega se stvara odgovarajući XML dokument.

Svi dokumenti koji se koriste unutar rješenja za upravljanje protokom poslova pospremaju se u za to predviđene repozitorije na SharePoint serveru. Unutar baze podataka vodi se evidencija o trenutnoj poveznici na dani dokument te o poveznicama na predložak dokumenta. Prilikom svakog dohvata dokumenta u svrhu promjene radi se odjava (Check-Out), odnosno prijava (Check-In) pri spremanju dokumenta natrag u riznicu. SharePoint automatski evidentira povijest svake promjene uz odgovarajući komentar.

6.6. XML Web uređivač

Budući da je velik dio dokumenata koji se upotrebljavaju u nekom poslovnom postupku u formi nekog generičkog obrasca (na primer odabir knjiga za posudbu, odabir izbornih predmeta, odabir smjera prilikom upisa više godine, zahtjev za nabavu i sl.) razumno je dane dokumente opisati pomoću XML sheme te na osnovu sheme dinamički izgraditi polja koja bi se mogla popunjavati.

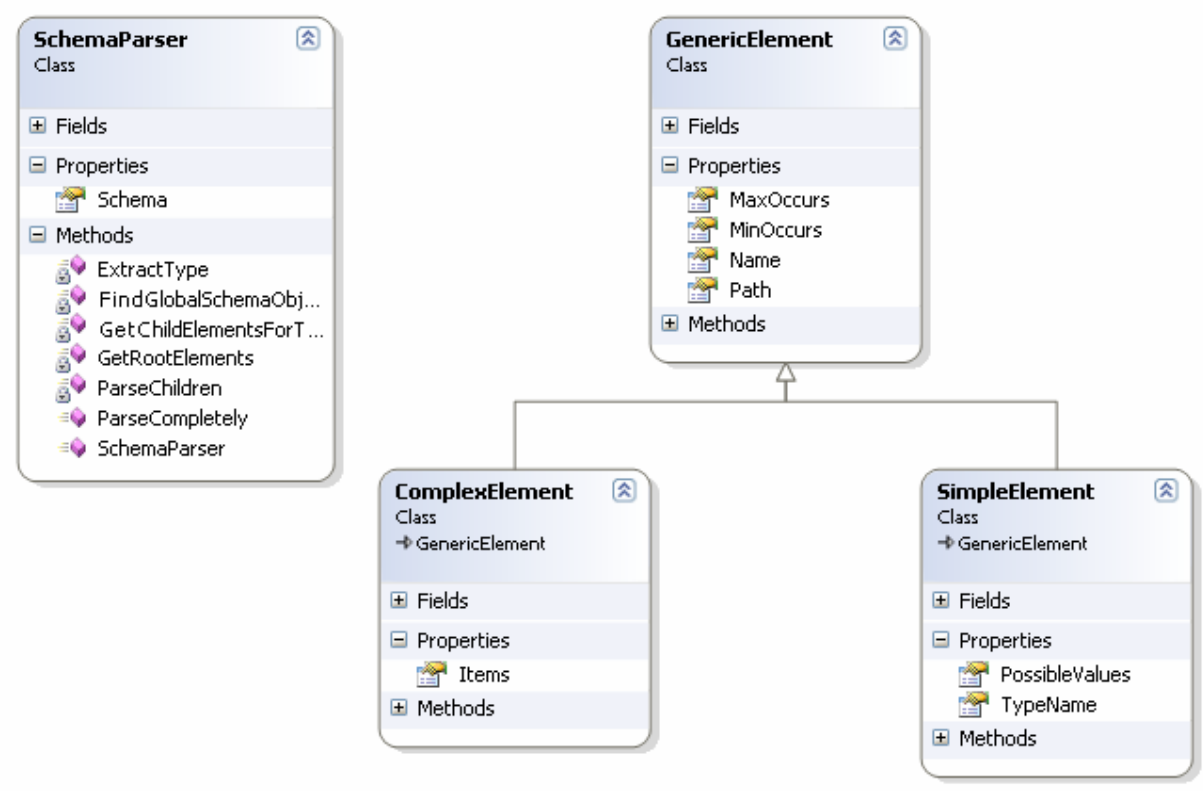
Iako .NET Framework omogućava jednostavnu integraciju XML dokumenata i DataSeta kroz XML shemu, nažalost zbog ograničenja da neka tablica (to jest neki element) ne može biti dijete dvaju ili više tablica [16][17], takvo rješenje nije bilo prihvatljivo te je razvijen vlastiti uređivač XML dokumenata. Uređivač je izveden kao web aplikacija, što mu omogućava jednostavnu integraciju sa sučeljem za rad s listama poslova te pristup neovisan o platformi pojedinog korisnika sustava.

Da bi se nekom korisniku prikazao određeni obrazac, potrebno je da je dotični dokument opisan odgovarajućom XML shemom (XSD). Na osnovu takve sheme web aplikacija generira odgovarajuća polja za unos kao što su tekstualna polja, padajući izbornici, gumbi za kontrolu trenutno prikazanog elementa grupe, gumbi za dodavanje i brisanje elemenata iz grupe i slično.

Algoritam koji bi opisivao rad uređivača XML dokumenata može se podijeliti u tri koraka. Prvi korak je izdvajanje ključnih podataka iz XML sheme te pretvorba u prikladniji oblik za aplikaciju kao i određivanje pravilnog poretka i veza među elementima. U drugom koraku, dolazi do stvaranja kontrola za svaki od elemenata i povezivanja s postojećim vrijednostima unutar XML dokumenta. Treći korak predstavlja ažuriranje promijenjenih vrijednosti.

6.6.1. Analiza XML shema i pretvorba u odgovarajući format

Budući da je obilazak XML sheme djelomično neprikladan prilikom kreiranja polja za unos tijekom rada web aplikacije, podaci iz sheme prevode se u prikladniji oblik. Povratna vrijednost postupka za parsiranje sheme *ParseCompletely* je niz generičkih elemenata. Svaki generički element može biti složeni element (*ComplexElement*) koji će u sebi sadržavati kolekciju generičkih elemenata ili jednostavni element koji je određen nazivom tipa i svim dozvoljenim vrijednostima, ukoliko takvo ograničenje postoji.



Slika 6.4 : Dijagram razreda korištenih u analizi XML sheme

Algoritam pretvorbe XML sheme u niz generičkih elemenata koji se obavlja u postupku ParseCompletely može se opisati sljedećim koracima:

```
1. Dohvatiti sve korijenske elemente (svi elementi unutar XML
sheme koji se nalaze na vrhu hijerarhije

2. Za svaki korijenski element stvoriti složeni element koji
će za svoje elemente (svojstvo Items iz gornje slike) imati
rezultat poziva postupka ParseChildren

3. Pozivatelju vratiti niz tako stvorenih složenih elemenata
```

Postupak ParseChildren može se opisati kao niz sljedećih koraka:

```
1.Dohvatiti sve elemente iz sheme koji su djeca trenutnog
elementa.

2.Za svaki od tako dohvaćenih elemenata stvoriti generički
element i postaviti mu osnovne podatke

3.Evidentirati apsolutnu poziciju danog elementa unutar
hijerarhije dokumenta kako bi se kroz XPath upit mogle
dohvatiti vrijednosti iz XML dokumenta koji će se naknadno
učitati.

4.Ako je maksimalni broj pojavljivanja veći od 1, tada
parametrizirati putanju s dodatnim nizom znakova koji će
predstavljati poziciju unutar istog roditelja.

5.Ako element nije jednostavnog tipa, pozvati postupak
ParseChildren na tom elementu.

6. Ako nije vrijedila prethodna tvrdnja ili je poziv postupka
vratio praznu kolekciju, element pretvoriti u jednostavni
element i provjeriti da li ima ograničen skup vrijednosti,
inače ga pretvoriti u složeni element.
```

6.6.2. Stvaranje kontrola za uređivanje XML dokumenta

Nakon što je prethodnim postupkom stvoren niz generičkih elemenata, web aplikacija može prijeći u izgradnju kontrola za uređivanje dokumenata unutar web sučelja i njihovo povezivanje s trenutnim vrijednostima dokumenta.

Za svaki složeni element iz niza složenih elemenata aplikacija će kreirati poseban okvir koji će se dinamički dodati u za to predviđen prostor na stranici (*PlaceHolder*). Za svaki element iz kolekcije koja predstavlja članove složenog elementa, potrebno je napraviti provjeru da li se radi od složenom elementu ili jednostavnom elementu te je potrebno provjeriti maksimalni mogući broj pojavljivanja takvog elementa. Ukoliko je maksimalni broj pojavljivanja veći od jedan stvorit će se zasebni okvir na čijem vrhu će se nalaziti navigacija namijenjena samo tim elementima.

Prije samog stvaranja elementa doći će do provjere prava vidljivosti i mogućnosti ažuriranja elementa, što se nalazi u zasebnom XML dokumentu. Ukoliko u dokumentu s popisom prava nema podataka za dani element, naslijedit će se vrijednosti od prvog pretka u stablu za kojeg bude postojao zapis. Postupak kreće od roditelja elementa, a ukoliko nijedan od predaka ne

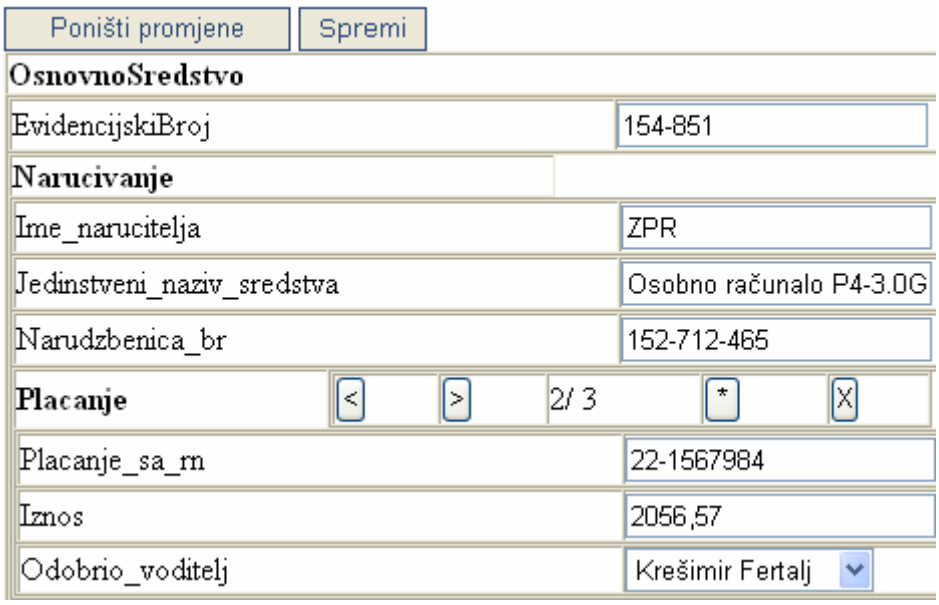
bude imao zapis o pravima čitljivosti i ažuriranja smatrat će se da je dozvoljeno i čitanje i pisanje.

Ako je element iz kolekcije bio jednostavni element potrebno je provjeriti o kojem tipu jednostavnog elementa se radi. Ukoliko je zadan skup vrijednosti koje element smije sadržavati potrebno je kreirati padajuću listu u kojoj će se nalaziti isključivo nabrojane vrijednosti.

Ukoliko naziv tipa počinje sa *SUSAPType_* to sugerira da se radi o nekom od unaprijed definiranih tipova podataka namijenjenih za korištenje u generičkim obrascima te će za takav tip biti također kreirana padajuća lista, a moguće vrijednosti dobit će se pozivom za to predviđenog web servisa koji će u ovisnosti o nazivu tipa vratiti odgovarajuće vrijednosti. U preostalim slučajevima web aplikacija će kreirati tekstualno polje. Ukoliko se radi o ažuriranju već postojećeg XML dokumenta, onda je potrebno dohvatiti trenutnu vrijednost za dani element, što će se izvesti postavljanjem XPath upita u kojem će se osim putanje navesti i redni broj pojavljivanja, ukoliko se trenutni element, ili neki od njegovih prethodnika u hijerarhiji, mogao pojavljivati više puta.

Ako je, pak, element iz kolekcije bio složeni element, onda će doći do rekurzivnog poziva postupka za kreiranje kontrola, što će uzrokovati stvaranje podokvira čime se opisani postupak ponavlja. Kao i kod jednostavnih elemenata, ukoliko se složeni element može pojavljivati više puta, na vrhu tako stvorenog okvira pojavit će se gumbi za navigaciju.

Sljedeća slika prikazuje izgled web uređivača za dokument pod nazivom Osnovno sredstvo korišten u postupku nabave. Slika 6.6 prikazuje XML shemu koja opisuje dani dokument.



Slika 6.5 : Isječak iz uređivača XML dokumenata na primjeru obrasca 'Osnovno sredstvo'[13]

[13] Nazivi prikazani na slici dobiveni su iz sheme na Slika 6.6. Korisniku prihvatljivi nazivi, primjerice nazivi koji uključuju praznine, mogu se dobiti dodavanjem dodatnih atributa elementima uz pravilno označavanje posebnih znakova kako bi se očuvala valjanost XML datoteke.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema                targetNamespace="http://tempuri.org/XMLSchema.xsd"
elementFormDefault="qualified"      xmlns="http://tempuri.org/XMLSchema.xsd"
xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="OsnovnoSredstvo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="EvidencijskiBroj" type="xs:string" />
        <xs:element name="Narucivanje" type="T_Narucivanje" />
        <xs:element name="Placanje" type="T_Placanje" />
        <xs:element name="Preuzimanje" type="T_Preuzimanje" />
        <xs:element name="Knjizenje" type="T_Knjizenje" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="T_Narucivanje">
    <xs:sequence>
      <xs:element name="Ime_narucitelja" type="xs:string" minOccurs="1" />
      <xs:element     name="Jedinstveni_naziv_sredstva"     type="xs:string"
minOccurs="1" />
      <xs:element name="Narudzbenica_br" type="xs:string" />
      <xs:element        name="Placanje"        type="T_NarucivanjePlacanje"
minOccurs="1" maxOccurs="unbounded" />
      <xs:element   name="Narudzba_odobrena_Predstojnik_Zavoda_datum_potpis"
type="xs:string" minOccurs="1" />
      <xs:element        name="Narudzba_odobrena_racunovodstvo_datum_potpis"
type="xs:string" />
      <xs:element               name="Narudzba_odobrena_Uprava_datum_potpis"
type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="T_NarucivanjePlacanje">
    <xs:sequence>
      <xs:element name="Placanje_sa_rn" type="xs:string" minOccurs="1" />
      <xs:element name="Iznos" type="xs:decimal" minOccurs="1" />
      <xs:element     name="Odobrio_voditelj"     type="SUSAPType_Voditelj"
minOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SUSAPType_Voditelj">
    <xs:sequence />
  </xs:complexType>
  <xs:complexType name="T_Placanje">
    <xs:sequence>
      <xs:element name="Racun_br." type="xs:string" />
      <xs:element name="Predracun_br." type="xs:string" />
      <xs:element name="Placeno_dana" type="xs:date" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="T_Preuzimanje">
    <xs:sequence>
      <xs:element name="Formular_preuzet" type="xs:string" />
      <xs:element name="Sredstvo_preuzeo" type="xs:string" />
      <xs:element name="Datum" type="xs:date" />
```

```
      <xs:element  name="Formular_o_preuzimanju_prilozen"  type="xs:string"
/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="T_Knjizenje">
    <xs:sequence>
      <xs:element name="Preuzeo_rukovatelj_imovine" type="xs:string" />
      <xs:element name="Datum" type="xs:date" />
      <xs:element name="Potpis" type="xs:string" />
      <xs:element name="Inventarni_broj" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Slika 6.6 : XML shema za obrazac 'Osnovno sredstvo'

Prilikom kreiranja svaka od kontrola povezana je s postupkom koji će se izvršiti ukoliko se vrijednost elementa promijeni. Promjene se evidentiraju unutar sjednice (eng. session), a nakon klika na gumb *Spremi*, promjene se evidentiraju i u centralnom repozitoriju što se obavlja kroz poziv web servisa.

6.7. Korištena tehnologija

Sistemski softver

- Windows 2003 Server, Service Pack 1
- SQL Server 2000, Service Pack 4
- Microsoft .NET Framework 2003

Korištene komponente

- Windows Sharepoint Services http://www.microsoft.com/windowsserver2003/technologies/sharepoint/default.mspx
- *Open-source* grafička knjižnica Netron, http://sourceforge.net/projects/netron/
- Razvojni okvir Atlas za izradu sučelja web aplikacija http://atlas.asp.net

Reference

[1] W. van der Aalst, K. van Hee: *Workflow management. Models, Methods, and Systems,* The MIT Press, 2004

[2] M. zur Muehlen: *Workflow-based Process Controlling. Foundation, Design and Application of workflow-driven Process Information Systems*, Logos, Berlin 2004[14]

[3] Workflow Patterns Web site, URL: http://www.workflowpatterns.com, 2005

[4] B. Kiepuszewski: *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*, doktorska dizertacija, Faculty of Information Technology, Queensland University of Technology, 2002

[5] Workflow Management Coalition: *The Workflow Reference Model, dokument br. TC00-1003 Issue 1.1*, 1995.

[6] J. Mettraux: *Open source WorkFlow Environment-Chapter 7: Workflow Patterns*, URL: http://www.openwfe.org/docbook/build/ch07.html

[7] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D.Smith, S. Thatte, I. Trickovic, S. Weerawarana: *Business Process Execution Language for Web Services Version 1.1*, 2003, URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp

[8] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros : *Workfow Patterns*, URL: http://is.tm.tue.nl/research/patterns/download/wfs-pat-2002.pdf

[9] Skelta Workflow.NET 2004 User's Guide, 2004, URL: http://www.skelta.com

[10] W.M.P. van der Aalst ,N. Russell, A.H.M. ter Hofstede, D. Edmond: *Workflow Resource Patterns*

[11] W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, P. Wohed: *Pattern Based Analysis of BPML (and WSCI),* FIT Technical Report FIT-TR-2002-05

[12] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede: *Pattern Based Analysis of BPEL4WS*, Technical Report FIT-TR-2002-04, QUT

[13] YAWL Web site: URL: http://www.yawl.fit.qut.edu.au/

[14] XML Path Language: URL : http://www.w3.org/TR/xpath

[15] JavaScript documents, Calling Service Methods, URL: http://www.webreference.com/js/column97/index.html

[16] Microsoft Knowledge Base: PRB: Error Message When You View Multi-Dimensional XSD Schemas in the Visual Studio .NET XML Schema Designer, URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;325695

[17] Microsoft Knowledge Base: You receive an error message when you view multidimensional XML data in the Visual Studio .NET XML Designer URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;325696

[18] L. Fisher: *Workflow Handbook 2002*, Future Strategies inc.

[19] D. Chaffey: *Groupware, Workflow and Intranets: Reengineering the Enterprise with Collaborative Software*, 1998. , Digital Press

[20] R.B. Walford: *Business Process Implementation for IT Professionals*, 1999. Artech House

[21] W3C, XML Schema Part 2: Datatypes, URL: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#typesystem

[22] XPath Tutorial, URL: http://www.w3schools.com/xpath/

[14] PDF Verzija knjige dostupna na http://www.workflow-research.de/Publications/Book/index.html

[23] BizTalk Overview, URL: http://www.microsoft.com/biztalk/evaluation/overview/biztalkserver.mspx
[24] Undestanding BizTalk Server, URL: http://www.microsoft.com/biztalk/techinfo/whitepapers/understanding.mspx
[25] BizTalk Server: How to Buy, URL: http://www.microsoft.com/biztalk/howtobuy/default.mspx
[26] J. Buyens: *Microsoft Windows SharePoint Services*, Microsoft Press, 2005
[27] Open source workflow engines in Java, URL: http://java-source.net/open-source/workflow-engines
[28] WorkflowPatterns, comparison of various standards, URL: http://is.tm.tue.nl/research/patterns/standards.htm
[29] W3C, XML, URL: http://www.w3.org/XML/
[30] DTD Tutorial, URL: http://www.w3schools.com/dtd/default.asp
[31] YAWL Home Page: URL: http://www.yawl.fit.qut.edu.au/
[32] SharePoint Products and Technologies 2003 Sofware Development Kit Help
[33] Petri Nets World: Online Services for the International Petri Nets Community, URL: http://www.informatik.uni-hamburg.de/TGI/PetriNets/
[34] Petri Nets, URL: http://pdv.cs.tu-berlin.de/~azi/petri.html
[35] YAWL Engine User Manual – Beta 7
[36] Skelta Workflow.NET 2004 WhitePapers: Workflow Managment For An Effective Organization, URL: http://www.skelta.com/customers/downloads/whitepapers/Workflow_management_for_an_effective_organization.pdf
[37] Skelta Preview Edition, URL: http://www.skelta.com/products/Workflow/registration.aspx?x=Eval
[38] Skelta Worklow.NET User's Guide
[39] WfMC: Document of Understanding, URL: http://www.wfmc.org/membership/Document_of_Understanding_2000.htm

Dodatak: XML dokument s popisom prava korištenih u postupku Prijave teme doktorske disertacije.

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfDocumentPermission      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <DocumentPermission>
    <ElementPath>PrihvatTeme</ElementPath>
    <DefaultPermission>
      <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
      </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>
  <!-- podatke o pristupniku mijenja samo pristupnik i to samo u pocetku
procesa -->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/PodaciOPristupniku</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Student</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>
  <!-- takodjer je ista stvar s nazivom teme i sazetkom te mentorom-->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/NazivTeme</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Student</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
```

```
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>
  <DocumentPermission>
    <ElementPath>PrihvatTeme/Sazetak</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Student</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>
  <DocumentPermission>
    <ElementPath>PrihvatTeme/Mentor</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Student</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>
```

```
  <!-- datum zaprimanja prijave postavlja Fakutetsko vijece i vidi se tek
nakon tog koraka -->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/Datum</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Zaprimanje prijave doktorske disertacije</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Fakultetsko vijeće</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>
  <!--  da  li  pristupnik  zadovoljava  uvjete  odredjuje  odbor  za
poslijediplomski studij i vidi se tek nakon tog koraka -->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/IspunjavaUvjete</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Utvrđivanje ispunjenja uvjeta</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Odbor za poslijediplomski studij</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
```

```
      <PermissionInProcess>
        <ProcessName>Zaprimanje prijave doktorske disertacije</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
    </DefaultPermission>
  </DocumentPermission>

  <!-- da li je tema prihvacena odlucuje fakultetsko vijece i ne vidi se
nigdje osim na kraju postupka -->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/TemaPrihvacena</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Prihvat teme disertacije</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Fakultetsko vijeće</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
            </DefaultPermission>
      </DefaultPermission>
  </DocumentPermission>

  <!-- Javni razgovor (vidi se tek nakon sto mentor odredi mjesto i
vrijeme) -->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/JavniRazgovor</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Određivanje termina javnog razgovora</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Mentor</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
```

```
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Zaprimanje prijave doktorske disertacije</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Utvrđivanje ispunjenja uvjeta</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
      </DefaultPermission>
  </DocumentPermission>

  <!-- Zapisnik javnog razgovora ispunjava Povjerenstvo za disertaciju -->
  <DocumentPermission>
    <ElementPath>PrihvatTeme/JavniRazgovor/Zapisnik</ElementPath>
    <ListOfPermissionsInProcesses>
      <PermissionInProcess>
        <ProcessName>Javni razgovor s pristupnikom</ProcessName>
        <ListOfGroupPermissions>
          <GroupPermission>
            <GroupName>_Povjerenstvo za disertaciju</GroupName>
            <PermissionType>READ_WRITE</PermissionType>
          </GroupPermission>
        </ListOfGroupPermissions>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
```

```
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Zaprimanje prijave doktorske disertacije</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Utvrđivanje ispunjenja uvjeta</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
      <PermissionInProcess>
        <ProcessName>Prijava teme</ProcessName>
        <ListOfGroupPermissions/>
        <DefaultPermission>
          <GroupName />
          <PermissionType>NOT_VISIBLE</PermissionType>
        </DefaultPermission>
      </PermissionInProcess>
    </ListOfPermissionsInProcesses>
    <DefaultPermission>
        <ProcessName />
            <DefaultPermission>
          <GroupName />
          <PermissionType>READ_ONLY</PermissionType>
            </DefaultPermission>
      </DefaultPermission>
  </DocumentPermission>
</ArrayOfDocumentPermission>
```