Implementacija simulatora Petrijevih mreža

Milanović, Matej

Sveučilište Jurja Dobrile u Puli Fakultet
informatike u Puli

MATEJ MILANOVIĆ

IMPLEMENTACIJA SIMULATORA PETRIJEVIH MREŽA

Završni rad

Pula, 2021.

Sveučilište Jurja Dobrile u Puli Fakultet
informatike u Puli

MATEJ MILANOVIĆ

IMPLEMENTACIJA SIMULATORA PETRIJEVIH MREŽA

Završni rad

JMBAG: 0303061692 , redoviti student
Studijski smjer: Informatika

Predmet: Modeliranje poslovnih procesa

Znanstveno područje: Informatika
Znanstveno polje: Društvene znanosti
Znanstvena grana: Informacijske i komunikacijske znanosti
Mentor: doc. dr. sc. Darko Etinger

Pula, 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Matej Milanović, ovime izjavljujem da je ovaj završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio završnoga rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli,_____

SADRŽAJ

1. UVOD

Kako bismo mogli uspješno upravljati organizacijom i povećati njezinu učinkovitost i ostvarivanja ciljeva te unaprjeđivanja poslovanja, potrebno je izvrsno poznavati unutrašnji ustroj te način na koji ona djeluje. Tu nastupa korištenje modeliranja poslovnih procesa za koje možemo reći da prikazuju strukture nekih poslovnih procesa uz uporabu grafičkih metoda i programskih alata u cilju što točnijeg prikaza tih poslovnih procesa. Modeliranjem poslovnih procesa često se nailazi na probleme zbog složenosti nekog sustava pogotovo kada se radi o timskom radu u koji su uključeni stručnjaci iz različitih područja zbog čega je potrebno korištenje grafičke metode. Napredovanje u simuliranju poslovnih procesa dovelo je do simulacije poslovnih procesa zbog toga što nam simulacija uključuje čimbenike iz okoline te tako možemo koristiti trajanje aktivnosti, vrijeme uporabe resursa, vrijeme čekanja te utjecaj slučajnih varijabli i ostalo. Za Petrijeve mreže možemo reći kako su jedna od najpoznatijih tehnika za specificiranje poslovnih procesa, a ime su dobile po Carlu Adamu Petriju koji ih je razvio 1962. godine. Iznimno su važne jer osim što omogućuju grafički prikaz, pomoću njih je moguće istodobno uvođenje matematičkih pravila za utvrđivanje ponašanja sustava. Sastoje se od četiri glavne grafičke oznake, a to su: krug-mjesto, pravokutnik-prijelaz, lukovi-konektori, točka-značka. Međutim, postoje i proširenja u Petrijevim mrežama, a to su vremenski prijelazi, lukovi s težinom i luk sprječavanja. Temeljna klasa Petrijevih mreža su mreže uvjetnog događaja, a kod tih događaja svako mjesto može imati najviše jednu značku. Isto tako, i mreže uvjetnog događaja imaju svoje proširenje, a ono se naziva mjesta prijelaza jer se u bilo kojem stanju Petrijevih mreža broj znački može nalaziti na bilo kojem mjestu. U slučaju postavljanja prijelaznih mreža nemoguće je razlikovati značke jedne od drugih. Takav se problem u Petrijevim mrežama rješava značajkom boje koja omogućava značkama da imaju svoje vrijednosti kao što imaju varijable u programskim jezicima. Kod analize Petrijeve mreže dostupan je širok spektar tehnika analiza što nije slučaj kada se koriste neformalne metode. Mogu se koristiti za provjeru posjeduje li Petrijeva mreža određena svojstva, primjerice, da bude bez zastoja. Carl Adam Petri iznio je dvije rečenice koje

nam kažu da „konkurentnost treba biti uključena kao polazište, a ne kao misao“ i „tehnika modeliranja treba biti u skladu sa zakonima fizike“.

Isto tako, za Petrijeve mreže možemo reći da su bile prvi model koji je adekvatno objedinio istovremenost, što igra važnu ulogu u poslovnim procesima. Sve više tvrtki kreće na korak ponovnog inženjeringa poslovnih procesa BPR i upravljanja tijekom rada WFM pri čemu se javlja potreba za tehnikama za izgradnju i analizu poslovnih postupaka. Mreža BP je mreža slobodnog izvora s dva posebna mjesta *i* i *o* koji se koriste za označavanje početka i kraja postupka no o tome ćemo više u samome radu.

Rad sadrži uvod, zaključak i četiri poglavlja. U prvome je poglavlju objašnjena definicija Petrijevih mreža te na koji način one funkcioniraju od čega se sastoje, te vrste Petrijevih mreža. U trećem su poglavlju objašnjene klase Petrijevih mreža te analiziranje poslovnih procesa. Četvrto poglavlje prikazuje dva softverska alata za izradu Petrijevih mreža i njihovu upotrebu. Peto poglavlje govori o implementaciji jednostavne Petrijeve mreže koju možemo vidjeti na dva primjera.

2. PETRIJEVE MREŽE

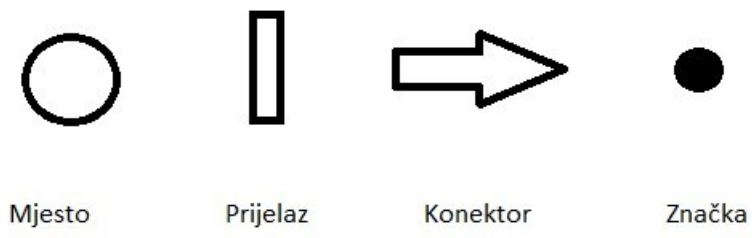
2.1 DEFINICIJA I POJAM PETRIJEVIH MREŽA

Petrijeve su mreže jedna od najpoznatijih tehnika za specificiranje poslovnih procesa na formalni i apstraktni način i kao takve vrlo su važna osnova za procesne jezike. Za njih možemo reći da su apstraktne jer zanemaruju izvršno okruženje poslovnog procesa na način da svi aspekti, osim funkcionalne i procesne perspektive, nisu pokriveni. One su, zapravo, grafički i matematički alat za modeliranje koji je primjenjiv na različite vrste sustava. Njihov idejni tvorac, po kojemu su i dobile ime,C. A. Petrij razvio ih je 1962. godine, a važne su jer omogućuju grafički prikaz ponašanja sustava, uz mogućnost istodobnog uvođenja matematičkih pravila za utvrđivanje ponašanja sustava. Mogu se koristiti za modeliranje dinamičkih sustava sa statičkom strukturom, a sastoje se od mjesta, prijelaza i usmjerenih lukova koji povezuju mjesta i prijelaze. Određeni događaj *t* povezuje se s mjestima (uvjetima) direktnim lukovima koji moraju biti ispunjeni da bi se zbio taj događaj i s uvjetima koji će biti ispunjeni nakon nastupa tj. paljenja događaja.

Grafičke oznake:

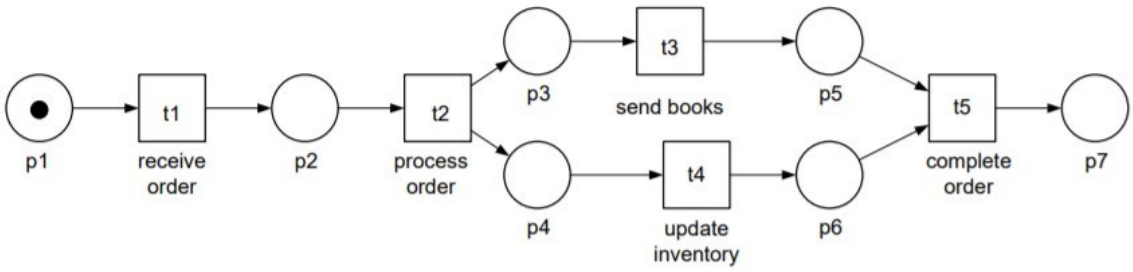
- Krug – mjesto
- Pravokutnik – prijelaz
- Lukovi – konektori
- Točka - značka

Slika 1. Grafičke oznake Petrijevih mreža



Dinamika sustava, predstavljena Petrijevom mrežom, modelirana je pomoću znački koje borave na mjestu. Iako je struktura fiksna značke mogu promijeniti svoj položaj prema pravilima pucanja, a raspodjelu znački među mjestima određuje stanje Petrijeve mreže i sustava koji je modeliran. Prijelaz se može aktivirati samo ako je omogućen, a omogućen je ako postoji značka na svakom od njegovih ulaznih mjesta. Ako se prijelazi aktiviraju jedna značka se uklanja sa svakog ulaznog mjesta i svakom izlaznom mjestu se dodaje po jedna značka. Kretanje znački u Petrijevim mrežama prema pravilima pucanja je nazvana „token play“. Igra značkama često se smatra protokom znački iako za to ne možemo reći da je točno zbog toga što se značke uklanjaju s mjesta unosa i dodaju na izlazna mjesta – ne idu od ulaznih mjesta prema izlaznim mjestima. S obzirom na to da prijelazi mogu promijeniti stanje Petrijeve mreže one se smatraju kao aktivne komponente koje predstavljaju događaje, operacije, transformacije ili prijenose. Budući da opisuju strukturu sustava one predstavljaju model poslovnog procesa, a njegovi prijelazi predstavljaju modele aktivnosti. Razina instance bilježi se značkama što znači da prijelaz predstavlja instancu aktivnosti. Svaka instanca je predstavljena najmanje jednom značkom, a zbog podijeljenih i pridruženih čvorova broj znački koje se zajedno karakteriziraju jedan se postupak može razlikovati tijekom „životnog vijeka“ procesa.

Slika 2. Uzorak Petrijeve mreže koja predstavlja instancu jednog procesa (Matias Weske, 2007. Business Process Management, 158)



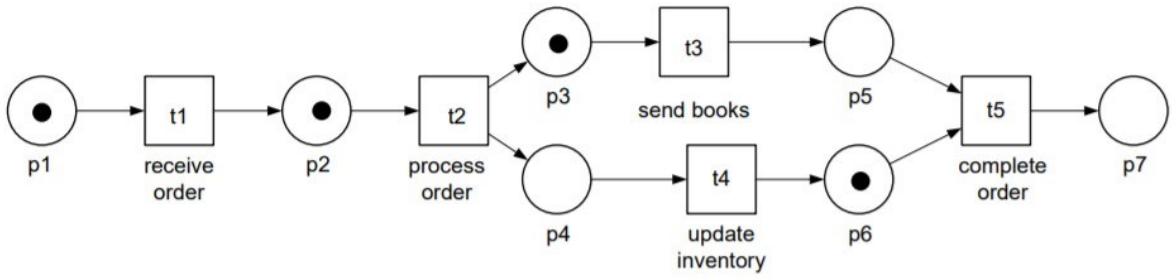
Slika 2. predstavlja nam uzorak Petrijeve mreže koja predstavlja pojedinačnu instancu procesa. Prijelazi u ovoj mreži odgovaraju primjercima aktivnosti dok mjesta i lukovi karakteriziraju ograničenja izvršavanja instanci. Proces počinje kada se značka

stavi na mjesto p1., a predstavljena je crnom točkom na tom mjestu. S obzirom na to da na svim ulazima primljene narudžbe postoji žeton ovaj prijelaz je omogućen.
Jednom kada se prijelaz iz primljene narudžbe aktivira, značka se uklanja iz p1 i na p2 se stavlja token koji predstavlja izvršavanje aktivnosti nakon primljene narudžbe. Vrijeme izvođenja ove instance nije postavljeno; Petrijeve se mreže aktiviraju trenutno. Nakon aktiviranja prijelaza istodobno u otvorene „grane“, a s obzirom na aktiviranje prijelaza stavljaju značke na obje p3 i p4 te omogućuju nalog za slanje i ažuriranje prijelaza inventara. Prijelaz reda potpuno je omogućen samo kada su oba prijelaza aktivirana. Kada se završi postupak na p7 se nalazi jedna značka. Grafički prikazi Petrijevih mreža se mogu preslikati u skup (*P,T,F*) i obrnuto. Tako možemo Petrijevu mrežu prikazanu na Slici 4. predstaviti :

- *P = {p1,p2,p3,p4,p5,p6,p7},*
- *T= {t1,t2,t3,t4,t5},*
- *F= {(p1, t1),(t1, p2),(p2, t2),(t2, p3),(t2, p4),(p3, t3),(p4, t4),(t3, p5), (t4, p6),(p5, t5),(p6, t5),(t5, p7)}.*

Stanje Petrijeve mreže karakterizira raspodjela znački na mjesta mreže. Postoje različita pravila aktiviranja za različite klase Petrijevih mreža.

Slika 3. Uzorak Petrijeve mreže koja predstavlja više instanci procesa (Matias Weske, 2007. Business Process Management, 160)



Na slici iznad vidimo primjer mreže s predstavljanjem više instanci procesa. Oznaka Petrijeve (*P, T, F*) mreže definirana je funkcijom *M: P* $\longrightarrow$ *N* koja preslikava skup mjesta na prirodne brojeve gdje je N skup prirodnih brojeva uključujući 0. Oznaka mreže zapravo predstavlja njezino stanje. Stanje Petrijeve mreže prikazano na slici iznad (Slika 3.) predstavljeno je kao M (p1) = M (p3) = M (p6) = 1 i M (p2) = M (p4) = M (p5) = M(p7) = 0. Ako su mjesta potpuno uređena njihovim identifikatorom (p1,p2,p3,p4…p7) oznaka se može izraziti nizom.

2.2 DES

Pojam simulacija diskretnih događaja (DES) uspostavljen je kao krovni pojam koji podrazumijeva različite vrste računalnih simulacijskih pristupa, a temelji se na ideji izrade računskog modela stvarnog sustava zamišljenog kao diskretni dinamički sustav uz postavljanje svog stanja uz pomoć (diskretnih i kontinuiranih) varijabli stanja i modeliranja njegove dinamike modeliranjem događaja koji su odgovorni za njegove diskretne promjene stanja. Simulacije diskretnih događaja dijelimo u pojmove od kojih se ona sastoji, sve definirane pojmove možemo vidjeti u Tablici 1.

Tablica 1. Osnovni pojmovi simulacije diskretnih događaja (Vesna B.V., Tomislav H., Andrej K. 2013. upravljanje poslovnim procesima, 166)

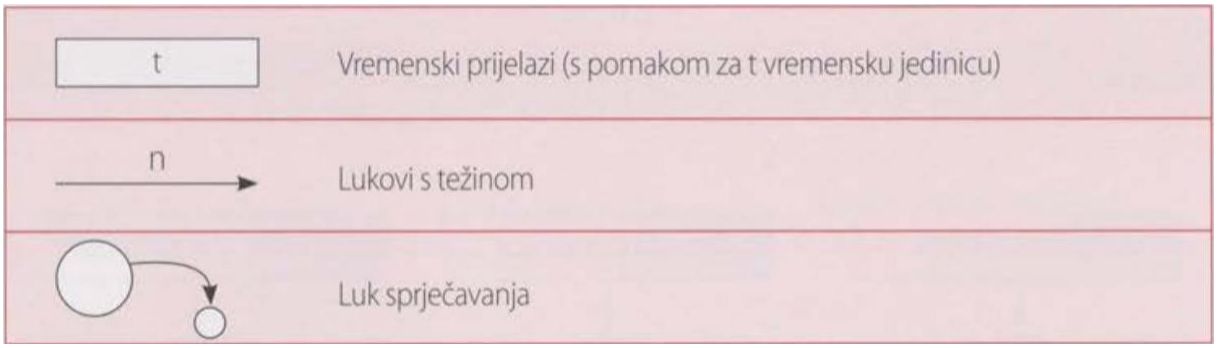
Entiteti (objekti)	Komponente sustava koje se mogu individualno referencirati, a dijele se na stalne entitete (resurse, entitete koji pripadaju sustavu: strojevi, službenici) i privremene entitete (entiteti koji ulaze u sustav iz njegove okoline, sudjeluju u obavljanju aktivnosti i izlaze iz sustava: narudžbe kupaca, kupci).
Atributi entiteta	Svojstva koja opisuju entitet (broj narudžbe).
Klase entiteta	To su grupe istovrsnih entiteta (narudžbe).
Skupovi entiteta	Dio je entiteta iste klase koja imaju jednaka svojstva
Redovi čekanja	Potrebna je vrsta entiteta koji čekaju da se oslobodi neki resurs (Primljena narudžba čeka obradu, a obrada narudžbe može početi ako komercijalist nije zauzet nekim drugim aktivnostima).
Stanje sustava	Skup je svih informacija nužnih za opis sustava

Događaj	Promjena stanja sustava (ulaz/izlaz entiteta iz sustava, promjene vrijednosti atributa nekih entiteta: dolazak narudžbe, dolazak kupca).
Uvjetni događaji	Mogu se dogoditi tek kad je ispunjen neki uvjet (početak aktivnosti ovisi o tome je li ispunjen uvjet zauzimanja resursa: obrada narudžbe počinje ako je komercijalist slobodan).
Bezuvjetni (planirani) događaji	Događaju se nakon prolaza nekog vremena
Aktivnosti	Međudjelovanje entiteta koje traje određeno vrijeme: aktivnost se prikazuje promjenom stanja entiteta u početnom i u završnom događaju aktivnosti (primljena narudžba kupca pokreće aktivnost obrade narudžbe, a rezultat je završetka obrade obrađena narudžba).
Proces	Niz je logički povezanih događaja kroz koji prolazi neki privremeni entitet.

U standardnoj metodi vremenskog napredovanja DES-a koja se naziva vremenska progresija sljedećeg događaja nema promjena stanja između uzastopnih događaja. Tako vrijeme simulacije može izravno skočiti na vrijeme pojave sljedećeg događaja. U alternativnom pristupu, koji se naziva vremenska progresija s fiksnim priraštajem, vrijeme se dijeli na male vremenske odsječke i stanje sustava ažurira se prema skupu događaja koji se događaju u vremenskom odsječku. Ne postoji opće prihvaćena definicija pojma „simulacija diskretnih događaja“, umjesto toga postoji niz različitih formalizama DES-a kao što su Petrijeve mreže, grafikoni događaja ili DEVS, ali postoje i različiti DES alati i okviri.

Temeljnim modelima Petrijevih mreža mogu se prikazati samo jednostavni sustavi, međutim, za simulaciju i modeliranje složenijih sustava uvode se proširenja. U sklopu proširenja dodaju se elementi kao što su simboli lukova s težinom, vremenski prijelazi i lukovi sprječavanja koji su prikazani u tablici na Slici 4.

Slika 4. Elementi DES mreža (Vesna B.V., Tomislav H., Andrej K. 2013. upravljanje poslovnim procesima, 169)



- Vremenski prijelazi – pokreću se nakon određenog vremena od trenutka kada su ostvareni uvjeti za njihovo ispaljivanje.
- Lukovi s težinom – ekvivalent su nizu paralelnih lukova koji povezuju mjesta i prijelaze u istom smjeru. Oni uvelike pojednostavljuju model jer se smanjuje broj lukova koji se rabe u modelu. Za prikaz simultanog kretanja većeg broja znački do mjesta i iz mjesta u grafu broj se znački upisuje uz odgovarajući luk, a kako bi se označio veći broj raspoloživih resursa, broj znački ucrtava se na odgovarajuće mjesto.
- Lukovi sprječavanja – proširenje su Petrijevih mreža koje omogućuje modeliranje prioriteta u sustavu. Oni su usmjereni od mjesta prema prijelazu. Prijelaz je onemogućen (zabranjen) sve dok mjesto koje je prijelazom povezano lukom zabrane sadržava značke. Ispaljivanje je moguće samo onda kad je to ulazno mjesto prazno.

Zbog lakšeg razumijevanja i primjene potrebno je određivanje metode koja sadrži dobre karakteristike poznatih metoda Petrijevih mreža, primjerice, mogu se uzeti DES mreže (eng. Discrete Event Simulation Nets) koje su prototip standarda proširenih Petrijevih mreža za simulaciju diskretnih događaja. Od dodatnih elemenata možemo spomenuti lukove s napomenama, pravila odlučivanja, niz značaka, boje značaka,

varijable i uvijete prijelaza. Za uklanjanje konflikata koristimo pravila odlučivanja, a to su: pravilo prioriteta, pravilo vjerojatnosti i uvjetno pravilo što je pojašnjeno u Tablici 2.

Tablica 2. Pravila odlučivanja u DES mrežama (Vesna B.V., Tomislav H., Andrej K. 2013. upravljanje poslovnim procesima, 170)

Pravilo prioriteta	Služi za dodjeljivanje prioriteta izlaznim prijelazima onda kada se oni trebaju izvoditi simultano (pretpostavka je Petrijevih mreža da se u jednom trenutku izvodi samo jedan prijelaz). Pritom niža vrijednost prijelaza označuje viši prioritet.
Pravilo vjerojatnosti	Pridružuje vjerojatnost izlaznim prijelazima. Zbroj vjerojatnosti uvijek je jednak 1. Međutim, odluka o tome koji će se prijelaz izvršiti donosi se slučajnim odabirom, u skladu s deklariranim vjerojatnostima
Uvjetno pravilo	Ispituje se u trenutku donošenja odluke. Ovisno o rezultatu ispitivanja (je li uvjet zadovoljen ili nije), odabire se odgovarajući prijelaz.

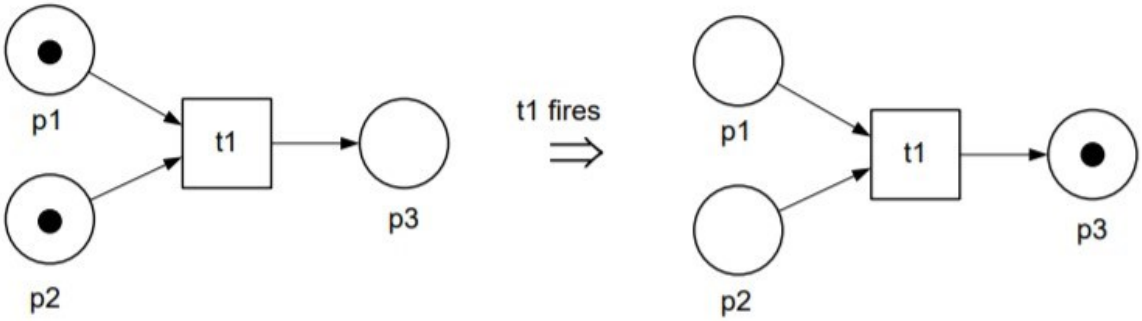
2.3. MREŽE UVJETNOG DOGAĐAJA

Mreže uvjetnog događaja su temeljna klasa Petrijevih mreža. U svakom trenutku svako mjesto može imati najviše jednu značku. Značke su nestrukturirane, nemaju identitet i stoga se ne mogu razlikovati jedna od druge. Ako je značka na mjestu p tada je uvjet p zadovoljen. Kod aktivacije prijelaza dogodi se događaj i mijenja se stanje.

Petrijeva mreža (P, T, F) je mreža uvjetnog događaja ako je M (P) ≤ 1 za sva mjesta p ∈ P i za sva stanja M. Prijelaz t je omogućen u stanju M ako je M (p) = 1 za sva mjesta unosa p od t i M (q) = 0 za sva izlazna mjesta q do t koja nisu ulazna mjesta u isto vrijeme. Aktiviranjem prijelaza t u stanju M rezultira stanjem M0 gdje $(\forall p \in \bullet t) M0$

(p) = M(p) − 1 ∧ (∀p ∈ t•)M0 (p) = M(p) + 1. Budući da je po definiciji M (p) 0 1 za sva ulazna mjesta p od t i M(q) = 0 za sva izlazna mjesta q do t, slijedi stanje M0 postignuto tom aktivacijom pod pretpostavkom da su mjesta izlaza i mjesta ulaza međusobno povezana (∀p ∈ •t)M0 (p) = 0 ∧ (∀p ∈ t•)M0 (p) = 1.

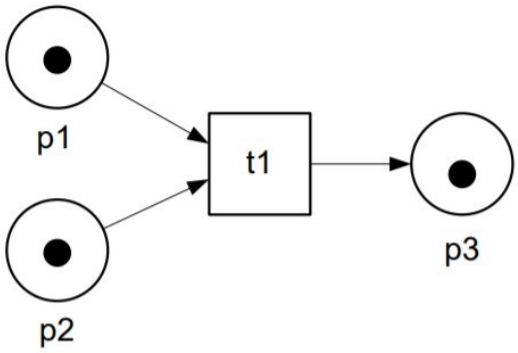
Slika 5. Aktiviranje uvjetnog događaja (Matias Weske, 2007. Business Process Management, 161)



Na slici 5. vidljivo je da su uvjet p1 i p2 ispunjeni, ali uvjet za p3 nije zadovoljen, dakle t1 je omogućen. Puštanjem t1 povlače se značke iz unosa i stavlja se značka na izlazno mjesto.

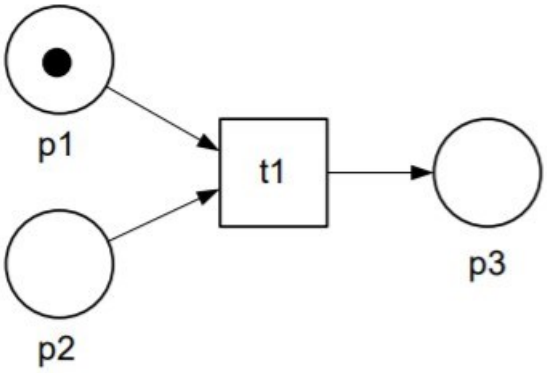
.

Slika 6. Aktiviranje uvjetnog događaja (Matias Weske, 2007. Business Process Management, 161)



t1 nije omogućen jer je izlaz ispunjen.

Slika 7. Aktiviranje uvjetnog događaja (Matias Weske, 2007. Business Process Management, 161)



t1 nije omogućen jer nisu ispunjeni svi uvjeti unosa.

Slike 5., 6. i 7. nam prikazuju ponašanje aktiviranja uvjetnog događaja. Prijelaz t1 je omogućen pri čemu samo sva mjesta za ulaz imaju jednu značku dok sva mjesta za izlaz nemaju značku. To znači da su uvjeti predstavljeni ulazom mjesta prijelaza koji su zadovoljeni, a uvjeti se odražavaju na izlazu mjesta prijelaza koja nisu ispunjena. Aktiviranjem t1 povlači se značka sa svakog ulaznog mjesta i stavlja se žeton na svakom izlaznom mjestu (Slika 5.). Prijelaz t1 nije omogućen ako u njemu postoji značka u jednom od njegovih izlaznih mjesta (Slika 6.) ili ako nemaju sva ulazna mjesta značku (Slika 7.). Budući da se značke ne mogu međusobno razlikovati, a zbog činjenice da je broj znački ograničen na po jedno na svakom mjestu, nije baš pogodno za modeliranje poslovnih procesa.

2.4. MJESTA PRIJELAZA MREŽE

Mjesta prijelaza mreže su proširenje uvjetnih mrežnih događaja jer u bilo kojem stanju Petrijevih mreža broj znački može se nalaziti na bilo kojem mjestu. Dakle, mogu poslužiti kao brojači. Kakogod značke su nestrukturirani objekti koji se ne mogu međusobno razlikovati. Da bi s uzelo u obzir više znački na svakom mjestu omogućavanje prijelaza mora biti preispitano. Tako se može iskoristiti i povući više znački s ulaznog mjesta kada se prijelaz aktivira, a više se znački može proizvesti kada se prijelaz aktivira prema pridruženim težinama lukova povezanih s prijelazom.

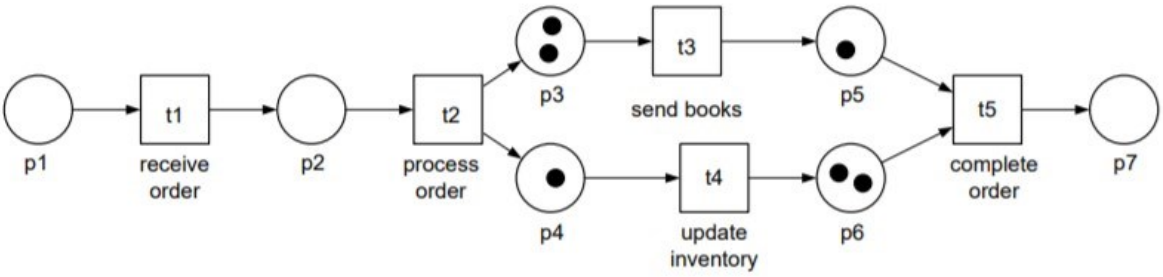
Ovo se proširenje može grafički prikazati višestrukim lukovima od ulaznog mjesta do prijelaza ili lukovima označenim s prirodnim brojevima koji označavaju njihovu težinu.

(P, T, F, ω) je mjesto prijelaza mreže ako je (P, T, F) Petrijeva mreža i ω: F → N je funkcija koja dodjeljuje svakom luku prirodni broj. Dinamičko ponašanje mjesta prijelaza je definirano:

- prijelaz za mjesto prijelaza omogućen je ako je svako ulazno mjesto p od t sadrži barem broj znački definirani kao težina spojnih lukova ako je $M(p) \geq \omega((p, t))$.
- Kada se prijelaz t aktivira broj znački povlači se iz njegovih ulaznih mjesta i broj znački dodaje se na izlazna mjesta dotičnim težinama luka. Sa svakog ulaznog mjesta p do t ω ((p, t)) značke se povlače, a ω ((t, q)) značke dodaju se na svako izlazno mjesto q do t
- Aktiviranje prijelaza t u stanju M rezultira stanjem M0 gdje
  $(\forall p \in \bullet t) M0(p) = M(p) - \omega((p, t)) \wedge (\forall p \in t\bullet) M0(p) = M(p) + \omega((t, p))$.

Pod pretpostavkom da je Petrijeva mreža prikazana na slici 2. u dodatak t3 (koji je također omogućen ako je Petrijeva mreža uvjetnog događanja) omogućeni su prijelazi t1 i t2. aktiviranje t1 koristi značku od p1 i dodaje značku p2, tako da dvije značke borave u p2., dakle t5 nije omogućen zbog toga što p5 ne sadrži značku. Svaki luk ima težinu jedan. Iako mjesta prijelaza mreže dopuštaju više znački na svakom mjestu to još uvijek nije korisno za predstavljanje poslovnih procesa zbog toga što se značke ne mogu međusobno razlikovati. Razlog ovog problema možemo vidjeti prikazanim primjerom (Slika 8.) gdje su višestruki procesi instance predstavljeni u Petrijevoj mreži s više znački. Promatranjem mreže i njezine distribucije značke može se otkriti koje značke pripadaju kojim procesnim instancama. Aktivne su tri instance procesa. Jedna instanca procesa predstavljena je značkom na mjestu p1, dok je druga instanca procesa već izvršila nalog narudžbe i stoga postavlja značku u p2. Aktivnost procesnog naloga rezultira dvije istodobne niti. Stoga značke u p3 i p6 moraju pripadati istoj instanci procesa. Kada je aktivnost slanja narudžbe dovršena, značka se stavlja u p5 i dovršava se aktivnost narudžbe dovršavajući treću instancu postupka.

Slika 8. Postavljanje mreže prijelaza s više instanci procesa (Matias Weske, 2007. Business Process Management, 163)



Iako ukazuje da vanjski promatrač u nekim slučajevima može identificirati tokene koji pripadaju jednoj instanci procesa to u konačnici ipak nije moguće. Na slici 8. vidimo da opisuje isti sustav, ali u nekom kasnijem vremenu. Točnije prva i druga procesna instanca ima ulaz u iste „grane“. Na slici 8. nije jasno koje značke pripadaju određenoj procesnoj instanci. Pravilo otpuštanja mjesta prijelaza mreža definira da se kompletna narudžba prijelaza može aktivirati čim postoji značka na mjestu p5 i značka na mjestu p6. To dovodi do problema jer bi se kompletna aktivnost narudžbe mogla provest i za dvije različite instance procesa. Ako prva instanca procesa naručuje knjige A i B, a druga instanca procesa naručuje knjige C i D, tada bi se mogla dogoditi situacija u kojoj bi slanje knjiga A i B te ažuriranje inventara za knjige C i D moglo pridružiti, što je neprihvatljivo. Stoga značke trebaju „nositi“ vrijednosti na način da je u svakom stanju Petrijeve mreže jasno koja značka pripada kojoj instanci procesa.

2.5. OBOJENE PETRIJEVE MREŽE

U slučaju stanja mreža i postavljanja prijelaznih mreža nemoguće je razlikovati značke jedne od drugih. Ovaj nedostatak u jednostavnim vrstama Petrijevih mreža rješava se značajkom boje, koja omogućava značkama da imaju vrijednosti. Poput varijabli u programskim jezicima, i značke imaju napisane vrijednosti. Tip podataka znački definira domenu vrijednosti i operacije koje su valjane nad podacima. U kontekstu obojenih Petrijevih mreža, vrste podataka nazivaju se i skupovima boja, s time da se razumije da skup boja znački predstavlja skup vrijednosti koje značka može imati. U obojenim Petrijevim mrežama omogućenost prijelaza određuje se, ne samo

brojem znački na ulaznim mjestima prijelaza, već i vrijednostima tih znački. Hoće li se preduvjet prijelaza ispuniti ili ne, ovisi o prisutnosti i vrijednostima znački koji se trebaju potrošiti. Ovo se ponašanje ostvaruje dodavanjem izraza prijelazima koji se vrednuju kako bi se odlučilo je li prijelaz omogućen.

Slično tome, vrijednosti znački proizvedenih prijelaznim aktivacijom mogu ovisiti o vrijednostima potrošenih znački, to također, znači da ne dobivaju sva izlazna mjesta značke nakon pokretanja prijelaza, ostvarujući izbor grana mreže na temelju vrijednosti utrošene značke i uvjeta vezanih uz prijelaz. Specifično aktiviranje prijelaza određeno je u postkondiciji prijelaza. Kao rezultat toga, grafički prikaz obojenih Petrijevih mreža nije potpun, jer izrazi koji označavaju preduvjete i protuslove, kao i vrijednosti znački, nisu prikazani. Iako postoji nekoliko inačica obojenih Petrijevih mreža, razvio ih je Kurt Jensen, ponašanje prijelaza vodi se pomoću znački na ulaznim mjestima, u „čuvarima“ pričvršćenim za prijelaze, a u izrazima pričvršćenim za lukove, izrazima luka. Izrazi luka koriste se za utvrđivanje je li prijelaz omogućen, a izračunavaju se u više skupova, pri čemu više skupova može sadržavati više identičnih elemenata. Ovi multiskupovi određuju značke uklonjene s ulaznih mjesta i dodane na izlazna mjesta prijelaza kada se aktivira.

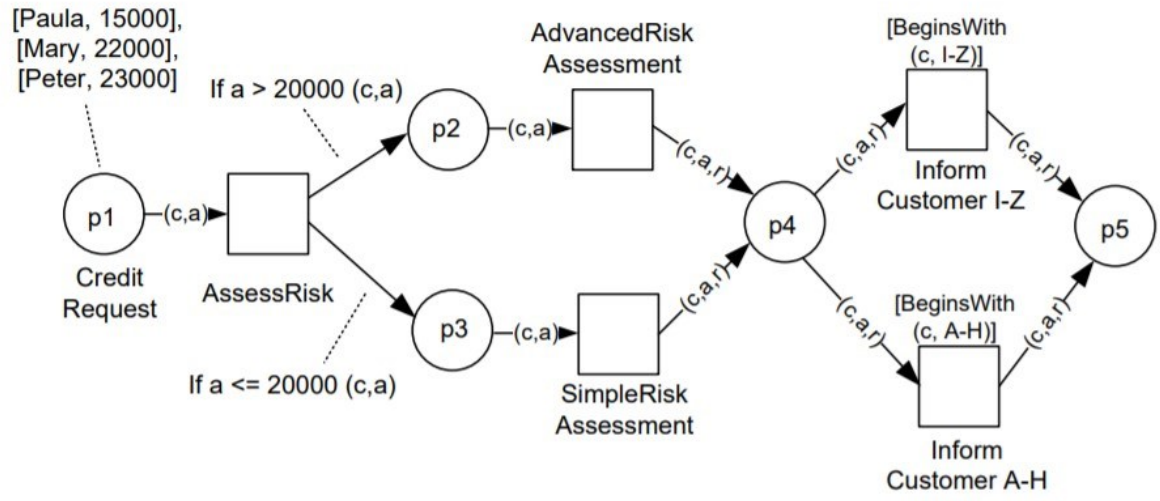
Obojena Petrijeva mreža je nabor (Σ, P, T, A, N, C, G, E, I) takav da:

- Σ je konačni skup nepraznih vrsta, koji se nazivaju skupovi boja
- P je konačan skup mjesta
- T je konačan skup prijelaza
- A je konačan skup identifikatora luka, takav da je $P \cap T = P \cap A = T \cap A = \emptyset$
- N: $A \rightarrow (P \times T) \cup (T \times P)$ je funkcija čvora koja preslikava svaki identifikator luka u par (početni čvor, krajnji čvor) luka
- C: $P \rightarrow \Sigma$ je funkcija boje koja svako mjesto povezuje s nizom boja
- G: T → BooleanExpr je zaštitna funkcija koja mapira svaki prijelaz u iskaz
- E: A → Izraz je izraz luka koji se izračunava kao multi-set preko skupa boja mjesta
- I je početno označavanje obojene Petrijeve mreže

Vezovi se koriste za pridruživanje vrijednosti podataka varijablama, tj. boja značkama. Na primjer, vrijednost "Paul" može se vezati za varijablu "name". U obojenim Petrijevim mrežama omogućavanje prijelaza ovisi o vezivanju. Prijelaz t je omogućen na povezu B ako njegova mjesta za unos sadrže značke koji zadovoljavaju izraze koji su obvezujući pod b i ako uz to, zaštitna funkcija prijelaza procijeni na vrijednost true.

Ako je prijelaz omogućen, može se aktivirati. Ovisno o procjeni izraza luka, odgovarajuće značke i uklanjaju se s ulaznih mjesta. Vođeni izrazima luka odlaznih bridova, značke se relativno dodaju na izlazna mjesta. Za formalnu obradu Petrijevih mreža u boji, čitatelj se poziva na bibliografske napomene na kraju ovog poglavlja. Da bismo ilustrirali Petrijeve mreže u boji, možemo vidjeti sliku 9. koja prikazuje obojenu Petrijevu mrežu zahtjeva poslovnog procesa za odobrenje kredita. Ovaj primjer ilustrira definiciju statičkog i dinamičkog aspekta boje Petrijeve mreže.

Slika 9. Uzorak obojene Petrijeve mreže (Matias Weske, 2007. Business Process Management, 165)



Obojena Petrijeva mreža pridružila je skup boja koji se sastoji od parova vrijednosti [Kupac, Količina], pri čemu je Kupac niz, a iznos cjelobrojna vrijednost. Obojene značke predstavljaju određene parove vrijednosti [Kupac, Količina]. U početku se nalaze tri značke na mjestu Zahtjeva za kredit p1, koji predstavljaju zahtjeve za kredit Paule, Marije i Petera i njihove odgovarajuće iznose. Prijelaz AssessRisk je omogućen jer se umjesto njega nalazi ulazni znak i s tim nije povezana dodatna zaštitna funkcija tranzicije. Prijelaz AssessRisk omogućen je pod vezivanjem c = Paula i a = 15000, jer na P1 postoji značka [Paula 15000] koja zadovoljava izraz luka. Budući

da s tim prijelazom nije povezan dodatni zaštitni izraz, omogućen je i može se aktivirati. Tranzicijska procjena rizika odlučuje o tome koje od njegovih izlaznih mjesta treba staviti značku kada se aktivira. Odgovarajući lukovi označeni su dodacima postcon-158 4 Process Orchestrations koji definiraju ponašanje pucanja. Pod pretpostavkom da se troši značka koja nosi vrijednost [Paul, 15 000], pa je taj iznos ispod 20 000, prijelaz AssessRisk stavlja žeton na p3, a ne na p2, što omogućuje prijelaz na SimpleRiskAssessment prijelaz. Ovakvo je ponašanje obojene Petrijeve mreže posljedica izraza povezanih s izlaznim rubovima prijelaza. Kada se prijelaz aktivira, procjenjuju se izrazi luka oba odlazeća ruba. Budući da je traženi iznos ispod 20 000, samo se izraz luka ako je <= 20 000 (c, a) procjenjuje na istinito, tako da se značka stavlja na p3, a nikakva značka na p2. Kada se aktivira prijelaz SimpleRiskAssession, značka se stavlja na p4. Token, također, uključuje vrijednost koja procjenjuje rizik traženog kredita; označeni su izrazom (C, A, R), gdje R predstavlja varijablu koja nosi procijenjeni rizik. Pod pretpostavkom da postoje dvije razine rizika za niske I i h za visoke i Paula dobiva procjenu visokog i niskog rizika, značka [Paula, 15000, l] stavlja se na P4. Ovisno o prvom slovu imena tražitelja kredita, omogućen je prijelaz odgovarajućeg kupca. Ovom odlukom vladaju funkcije „čuvara“, koji predstavljaju funkcije postavljene na vrhu prijelaza. U primjeru, BeginsWith (ime, interval) vraća true ako i samo ako ime započinje slovnim intervalom. Budući da je značka za Paulu u P4, omogućen je prijelaz Obavijesti kupca 1-Z. Nakon što je kupac obaviješten, postupak se dovršava.

Da rezimiramo, ponašanje se prijelaza u obojenu Petrijevu mrežu definira zaštitnom funkcijom, značkama koje se nalaze na njezinim ulaznim mjestima i izrazima luka. Stoga, prijelazno aktiviranje u obojenim Petrijevim mrežama pokazuje složeno i vrlo prilagodljivo ponašanje pa je valjano reći da svaki prijelaz u obojenoj Petrijevoj mreži ima svoje prijelazno ponašanje, što pruža veliku izražajnu snagu za ovu klasu Petrijevih mreža. Istodobno, grafički prikaz više nije dovoljan za hvatanje i razumijevanje semantike Petrijeve mreže. Kada se prijelaz aktivira, bilo koji broj tokena s bilo kojim vrijednostima može se staviti na izlazna mjesta prijelaza, određena izrazima luka. Budući da Petrijeve mreže u boji pružaju ovu izražajnu snagu i mogu obrađivati podatke, one su vrlo pogodne za predstavljanje modela poslovnih procesa.

2.6 ANALIZA PETRIJEVE MREŽE

Kada koristimo neformalne modele, nemoguće ih je koristiti za analizu procesa. Srećom za Petrijeve mreže dostupan je širok spektar tehnika analize. Tradicionalno se najveći dio Petrijevih istraživanja fokusirao na analize utemeljene na modelima. Nadalje, najveći udio tehnika analize temeljenih na modelima ograničen je na funkcionalna svojstva. Generičke tehnike poput provjere modela mogu se koristiti za provjeru ima li Petrijeva mreža određena svojstva, primjerice, da bude bez zastoja. Specifični pojmovi Petrijevih mreža su zamke, nepromjenjivi prijelazi, grafikoni konvertibilnosti često se koriste za provjeru željenih funkcionalnih svojstava.

Analiza koja se temelji na modelu također se može usredotočiti na nefunkcionalna svojstva kao što su vrijeme protoka, vrijeme odaziva, troškovi, rizici, upotreba i dostupnost. Takva svojstva su od najveće važnosti za BPM. Za određene klase Petrijevih mreža može se koristiti Markovian analiza, npr. stohastičke Petrijeve mreže s negativnim eksponencijalnim razdiobama kašnjenja mogu se prevesti u Markov chain koji se mogu analizirati kako bi se utvrdilo vrijeme protoka vjerojatnosti. Za složenije modele procesa i pitanja jedan treba pribjegavati simulaciji, stoga postoje mnogi BPM alati koji omogućuju neki oblik simulacije.

Interes za analizu temeljenu na podacima potiče sve veća dostupnost podataka o događajima i interes organizacija za analizu koja se temelji na činjenicama BPM-a na temelju dokaza. Izraz procesno rudarstvo koristi se za označavanje tehnika koje izvlače znanje iz zapisnika događaja. Tehnike rudnog procesa čine obitelj tehnika aposteriori analize koja koristi informacije zabilježene u revizorskim zapisima, bazama podataka, zapisima transakcija. Procesno rudarstvo uključuje otkrivanje procesa.

2.7 UTJECAJ PETRIJEVIH MREŽA NA JEZIKE I SUSTAVE

Možemo reći da postoji neprekinuti niz novih bilješki procesa modeliranja. Neki su od tih zapisa utemeljeni i postoje već desetljećima (npr. Petrijeve mreže) dok su ostale specifične ili su popularne samo kratko vrijeme. Neprekidne rasprave o raznim konkurentnim zapisima često skrivaju više temeljnih pitanja. Oznake se kreću u rasponu od jezika koji imaju za cilj pružiti formalnu osnovu (Petrijeve mreže i procesne

algebre) do specifičnih oznaka isporučitelja (Različiti jezici radnog rijeka koje koriste dobavljači BPM-a), industrijski standardi kao što je BPEl (jezik izvršenja poslovnih procesa) i BPMN-a obično su samo djelomično usvojeni, isporučitelji podržavaju samo podskupove ovih standarda, a korisnici imaju tendenciju upotrebe samo malog dijela ponuđenih koncepata. Postoji malo konsenzusa o modelu koji se koristi. To je rezultiralo „Babilonskom kulom procesnih jezika " dakle mnoštvom sličnih, ali suptilno različitih jezika koji sprječavaju učinkovitu i objedinjenu podršku i analizu procesa.

Usprkos BPM, Petrijeve mreže igrale su važnu ulogu u razvoju terena. Gotovo svi jezici za modeliranje poslovnih procesa i BPM/WFM sustava koriste semantiku temeljenu na značkama Petrijevih mreža. Iako su Petrijeve mreže često skrivene postoje i primjeri BPM/WFM sustava i alata koji prikazuju Petrijeve mreže izravno korisniku. COSA je jedan od vodećih alata WFM u 90-ima, a u potpunosti se temelji na Petrijevim mrežama: COSA modeler, COSA engine i COSA simulator svi koriste Petrijeve mreže.

2.8 PRAVA GRAĐA POSLOVNIH PROCESA

Carl Adam Petri iznio je dvije rečenice koje kažu da „konkurentnost treba biti uključena kao polazište, a ne kao misao (lokalitet djelovanja) „ i „tehnika modeliranja treba biti u skladu sa zakonima fizike". Petrijeve mreže bile su prvi model koji je adekvatno objedinio istovremenost. Možemo reći da istodobnost igra važnu ulogu u poslovnim procesima. Npr. može biti puno resursa (ljudi, strojeva) koji rade istovremeno, a u bilo kojem trenutku može biti puno pokretačkih slučajeva. C. A. Petri bio je zainteresiran za odnos između modeliranja procesa i fizike. U kontekstu BPM-a treba obratiti pozornost i na odnos između modela procesa i stvarnih karakteristika poslovnih procesa. Poslovni procesi imaju izrazito paralelnu tendenciju. Stoga su sekvencijalni modeli neadekvatni. Ne može se izolirati pažnja na pojedinačnu procesnu instancu (kao što je slučaj u BPMN-u, EPC-u), ali mogu postojati složeni odnosi mnoštva prema broju između naloga, linija za narudžbu, može biti više isporuka povezanih s istim nalogom, a isporuka se može odnositi na redove narudžbe različitih naloga. Tradicionalni pristupi modeliranju imaju problema s takvim složenim ovisnostima dok praktična iskustva s miniranjem procesa pokazuju da su interakcije između artefakta ključnih za analizu procesa. Empirijska priroda rudarstva procesa

pomaže menadžerima, konzultantima i analitičarima procesa da bolje razumiju građu poslovnih procesa pa tako i vide ograničenja konvencionalnih jezika modeliranja procesa. Izazov je povezati elegantne kratke formalne modele poput Petrijeve mreže i ponašanja koje se stvarno promatra u stvarnosti.

3. KLASA PETRIJEVIH MREŽA ZA MODELIRANJE I ANALIZIRANJE POSLOVNIH PROCESA

3.1 BP MREŽE

Svjedoci smo kako sve više tvrtki kreće na korak ponovnog inženjeringa poslovnih procesa (BPR) i upravljanja tijekom rada (WFM). Takvi koraci nam otkrivaju potrebu za tehnikama za izgradnju i analizu poslovnih postupaka. Mreža BP je mreža slobodnog izvora s dva posebna mjesta I i O, a koriste se za označavanje početka i kraja postupka. Obrada slučaja započinje onog trenutka kada značku postavimo na mjesto „i“ završava onog trenutka kada se značka pojavi na mjestu o. U svakom slučaju, postupak će se na kraju „i“ završiti i postupak se završava. Na mjestu „o“ i svim ostalim se postavlja znak da su mjesta prazna. Možemo reći da se ova tehnika temelji na bogatoj teoriji razvijenoj za Petrijeve mreže slobodnog izbora. BP imaju zanimljiva svojstva, za njih možemo reći da je BP-mreža valjana ako i samo ako je malo izmijenjena verzija ove mreže aktivna i ograničena. Također, nastojat ćemo pokazati kako postoji sveobuhvatan skup pravila transformacije koji čuvaju stabilnost. Pravila transformacije pokazuju kako se valjani postupak može pretvoriti u drugi. Kada gledamo iz ugla WFM i BPR gdje se postupci moraju često ili radikalno mijenjati ta su pravila transformacije vrlo korisna. Zajedničko upravljanje tijeka rada i preuređivanje poslovnih procesa je usredotočenost na poslovne procese. Sustavi upravljanja tijekom rada usmjereni su oko definicije poslovnog procesa koja se naziva tijek rada.
Reinženjering poslovnih procesa uključuje izričito preispitivanje i redizajn poslovnih procesa. Cilj poslovnog procesa je obrada predmeta. Za potpuno definiranje poslovnog procesa moramo navesti dvije stvari:

- Procedura: djelomično uređeni skup zadataka
- Raspodjela resursa zadacima

Postupak nam određuje skup zadatka potrebnih za uspješnu obradu predmeta. Dakle, postupak određuje redoslijed kojim se ovi zadaci moraju izvršiti, mogu biti neobavezni ili obavezni i izvode se paralelno. Raspodjela resursa zadacima potrebna je kako bi se utvrdilo tko će izvršiti zadatak za određeni slučaj. Dakle, svaki resurs može obavljati određene funkcije i svaki zadatak zahtijeva određene funkcije. Resursu

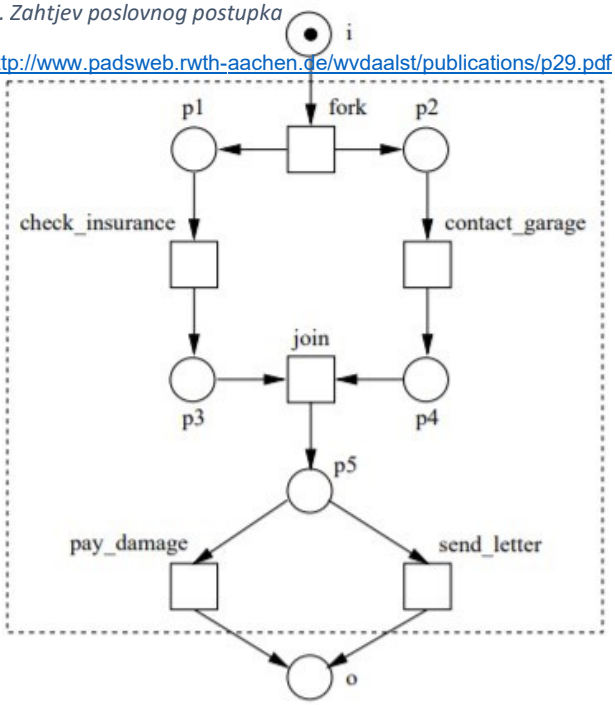
se može dodijeliti zadatak ako resurs nudi potrebne funkcije. No ovdje se koncentriramo na modeliranje poslovnih postupaka tj. apstrahiramo od resursa potrebnih za provođenje tih postupaka.

3.2 MODELIRANJE POSTUPAKA

Petrijeve mreže koristimo za modeliranje i analizu poslovnih postupaka. Dakle, postupak je djelomično uređeni skup zadataka zbog čega je prilično lako preslikati postupak na Petrijevu mrežu. Zadaci se modeliraju prijelazima, a odnosi prioriteta modeliraju se mjestima. Ako uzmemo u obzir zahtjev za poslovnog postupka, zadaci provjeravaju osiguranje, kontaktiraju garažu, plaćaju štetu i pismo za slanje modeliraju prijelazima. Budući da se dva zadatka provjere osiguranja i kontaktne garaže mogu izvoditi paralelno postoje dva dodatna prijelaza: račvanje i pridruživanje. Mjesta p1, p2, p3, p4, p5 koriste se za pravilno usmjeravanje slučaja kroz postupak. Što možemo vidjeti na slici 10.

Slika 10. Zahtjev poslovnog postupka

Izvor:(http://www.padsweb.rwth-aachen.de/wvdaalst/publications/p29.pdf)



- Slučajevi se obrađuju neovisno, tj. zadatak izvršen za neki slučaj ne može utjecati na zadatak izveden za drugi zadatak. Bez obzira na vrijeme prolaska predmeta može se povećati ako se za isti resurs natječu mnogi drugi slučajevi. Oznaka na mjestu „i“ na slici 10. odgovara jednom slučaju. Tijekom obrade slučaja može biti nekoliko znački koje se odnose na isti slučaj, no ako se aktivira prijelaz račvanja tada postoje dvije značke, jedna u p1 i jedna u p2 koje se odnose na isti zahtjev. Da bi za Petrijevu mrežu mogli reći da je BP onda mora ispunjavati sljedeće uvijete:
- Uvijek imaju dva posebna mjesta „i“ i „o“ koja odgovaraju početku i prestanku obrade predmeta.
- Petrijeva mreža koja predstavlja poslovni postupak uvijek je Petri mreža slobodnog izbora.
- Za svaki prijelaz *t* trebao bi biti usmjeren put od mjesta i do o preko t.

Petrijeva mreža koja zadovoljava ova tri zahtjeva naziva se mrežom poslovnih postupka (BP-mreža).

3.3.VALJANE PROCEDURE

Tri uvjeta koja su nam potrebna kako bi znali radi li se o BP-mreži mogu se provjeriti statistički, točnije, odnose se na strukturu Petrijevih mreža. No, postoji i četvrti uvjet koji bi se trebao zadovoljiti:

- U svakom slučaju postupak će se na kraju *i* izvršiti i postupak se završava, a na mjestu *o* i svim ostalim postavlja se znak da su mjesta prazna.

Ovo svojstvo nazivamo svojstvom valjanosti. Procedura modelirana s BPmrežama Petrijeve mreže = (P, T, F) je valjano samo ako:

- Za svako stanje M koje je dostupno iz stanja *i* tada postoji redoslijed aktiviranja vodeći iz stanja M u stanje *o.*

$$\forall M\ (i * \rightarrow M) \Rightarrow (M * \rightarrow o)$$

- Stanje *o* je jedino stanje do kojeg se može doći iz stanja *i* s najmanje jednom značkom.

$$\forall M\ (i * \rightarrow M \wedge M \geq o) \Rightarrow (M = o)$$

Svojstvo valjanosti odnosi se na dinamiku BP-mreže. Ako kažemo da je počevši od početnog stanja uvijek moguće doći do stanja jednom značkom na mjestu *o,* pritom treba imati na umu da postoje preopterećenja zapisa, simbol *i* koristi se za označavanje mjesta *i* i stanja sa samo jednom značkom na mjestu *i.* Prvi uvijek podrazumijeva da se s vremenom postiže stanje *o* zbog toga što prijelaz koji je omogućen često se aktivira. Drugi uvjet kaže da se u trenutku kada se značka na mjesto *o* sva ostala mjesta trebaju biti prazna.

3. 4.VIŠESTRUKI SLUČAJEVI

Kako je rečeno da pojedinačni slučajevi ne utječi jedan na drugoga zbog toga što se apstrahiramo od resursa dovoljno je razmotriti jedan slučaj pojedinačno za provjeru ispravnosti postupka, ali ako želimo modelirati postupak koji se koristi za obradu više slučajeva istodobno moramo pribjeći Petrijevoj mreži visokog nivoa. Petrijeva mreža na visokoj razini organizirana je tako da svaka značka ima vrijednost koja se odnosi na slučaj kojem pripada i prijelazi mogu koristiti samo značke koje pripadaju istom slučaju. Svojstvo valjanosti možemo proširiti i na situaciju u kojoj postoji proizvoljni broj slučajeva.

Jedno od osnovnih svojstava Petrijeve mreže slobodnog izbora je činjenica da se ona može podijeliti u klastere. Klaster bi definirali: ako je *t* prijelaz u Petrijevu mrežu slobodnog izbora. Klaster *t* označen sa [t], je skup $\bullet t \cup \{t \in T \mid \bullet t = \bullet t\}$. klaster mjesta *p*, koji je također označen sa [p], je skup $p \bullet \cup \{p \in P \mid (p \bullet \cap p \bullet) = \emptyset\}$.

Mjesto *p* i prijelaz *t* pripadaju istoj grupi. Dakle, Petrijeve mreže imaju sljedeće svojstvo, ako je omogućen prijelaz *t* omogućen je bilo koji prijelaz u [t]. Klaster c naziva se omogućenim ako su prijelazi u *c* omogućeni.

4. SOFTVERSKI ALATI

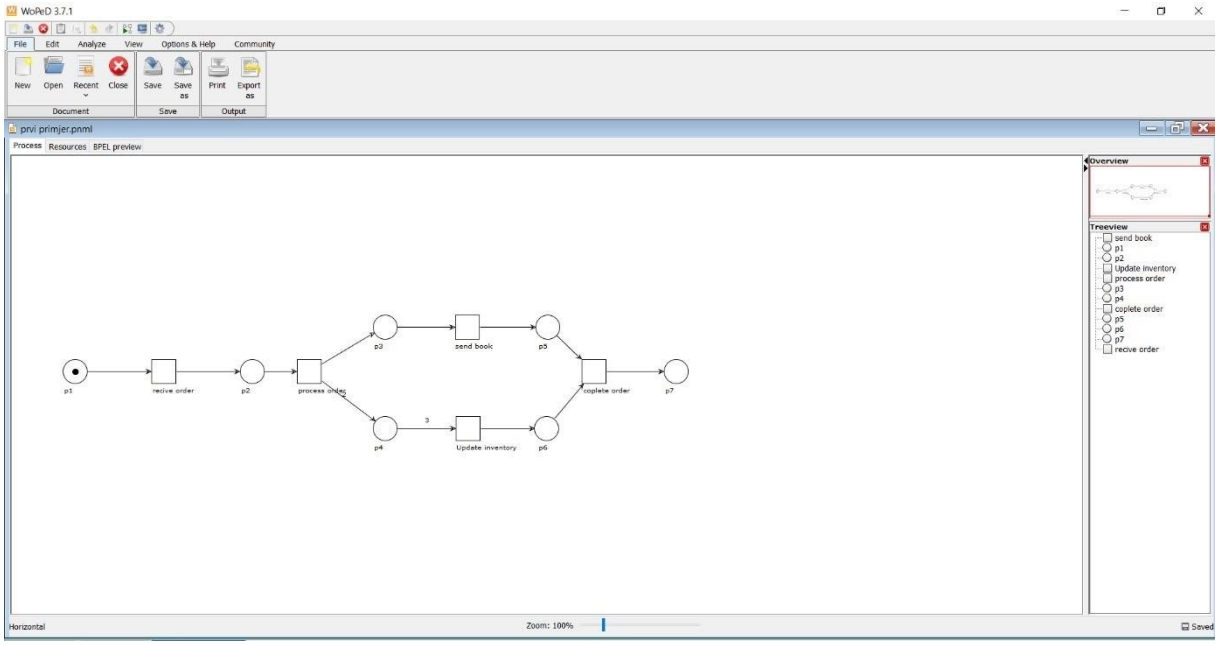
4.1 WoPeD

WoPeD ili skraćeno Workflow Petri Net Designer je softver otvorenog izvora. Tijekom godina postao je vrlo popularan alat u svijetu. Razvijen je 2003. godine i od tada je neprekidno nadograđivan. Ovaj program pruža grafički uređivač za osnovne Petrijeve mreže i mreže toka rada i interaktivne simulatore. Također, podržava verifikaciju modela i vizualizaciju imovine.

Pomoću njega možemo vršiti transformacije u druge formate modela tako i iz njega. Pokazao se kao vješt i vrlo jednostavan za korištenje. U njemu se može crtati raditi bilješke, upravljati i izvoziti uobičajena mjesta i prijelaze Petrijevih mreža. Podržava razmjenu podatka s drugim alatima koji se koriste za izrađivanje Petrijevih mreža. Korisnik ima mogućnost zatražiti pokretanje provjere u čarobnjaku gdje kroz vizualno objašnjenje vodi kroz sve korake provjere. Također, moguće je koristiti simulator igranja koji nam omogućuje navigaciju preko Petrijevih mreža. Ponašanje koje se odnosi na Petrijevu mrežu može se pokazati grafom pokrivenosti uz ručni pregled i navigaciju.

Sučelje same aplikacije je vrlo jednostavno sastoji se od pet tablica. U tablici edit odabiremo elemente koji trebamo koristiti za Petrijevu mrežu te klikom na prazni dokument postavljamo jedan od elemenata. Ako želimo postaviti značku na element „Place“ desnim klikom na „Place“ odabiremo dodaj značku. Na desnoj strani aplikacije možemo vidjeti izgled drveta napravljene Petrijeve mreže.

Slika 11. Sučelje WoPeD-a



4.2 PIPE2

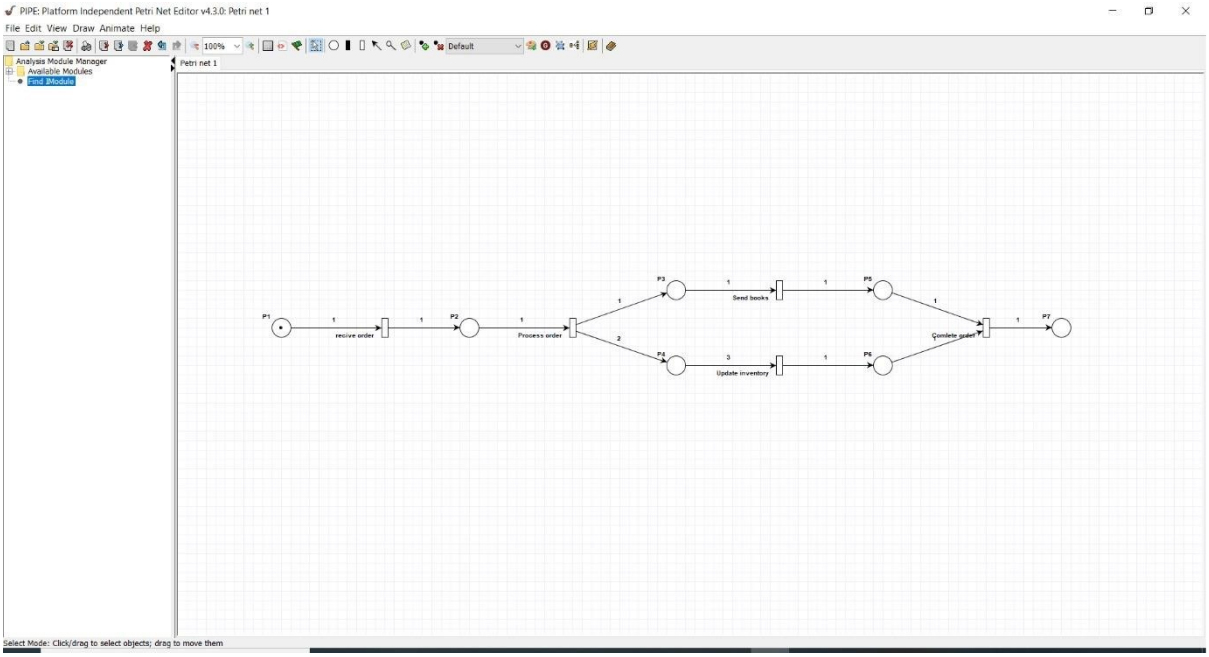
PIPE2 (Platform-Independent Petri Net Editor 2) alat je temeljen na Java Programu koji se najprije koristi za analizu i konstrukciju modela generaliziranih stohastičkih Petrijevih mreža. Izumili su ga studenti s Odjela za računarstvo s jednog prestižnog Sveučilišta u Londonu te se tijekom sljedećih godina alat konstantno unaprjeđivao.

Uz manipulacije, animacije i stvaranja obične Petrijeve mreže može se koristiti i za izračun vremena integracije nove funkcionalnosti preko modula analize koji se može priključiti na uređaj. Te na osnovu toga možemo reći da je to nešto što ga izdvaja u odnosu na ostale alate za izradu Petrijevih mreža. Također, crta se na platnu k koristeći alatnu traku. Osim funkcionalnosti osnovnog dizajna, dizajnersko sučelje omogućava značajke kao što su izvoz, uređivanje karticama, animacija i zumiranje. Način animacije osobito je koristan za pomoć korisnicima u intuitivnoj provjeri ponašanja njihovih modela. PIPE2 koristi Petrijev Net Markup Language(PNML) kao format datoteke koji dopušta kompatibilnost s drugim alatima za izradu Petrijeve mreže kao što su P3, WoPeD i Woflan.

Arhitektura PIPE2 podržava module te tako omogućava produženje vremena izvođenja s kodom koji je korisnik postavio. Također možemo se koristiti mnogim

specijaliziranim analizama koje postižu strukturnu i analizu performansi na GSPN modelima.

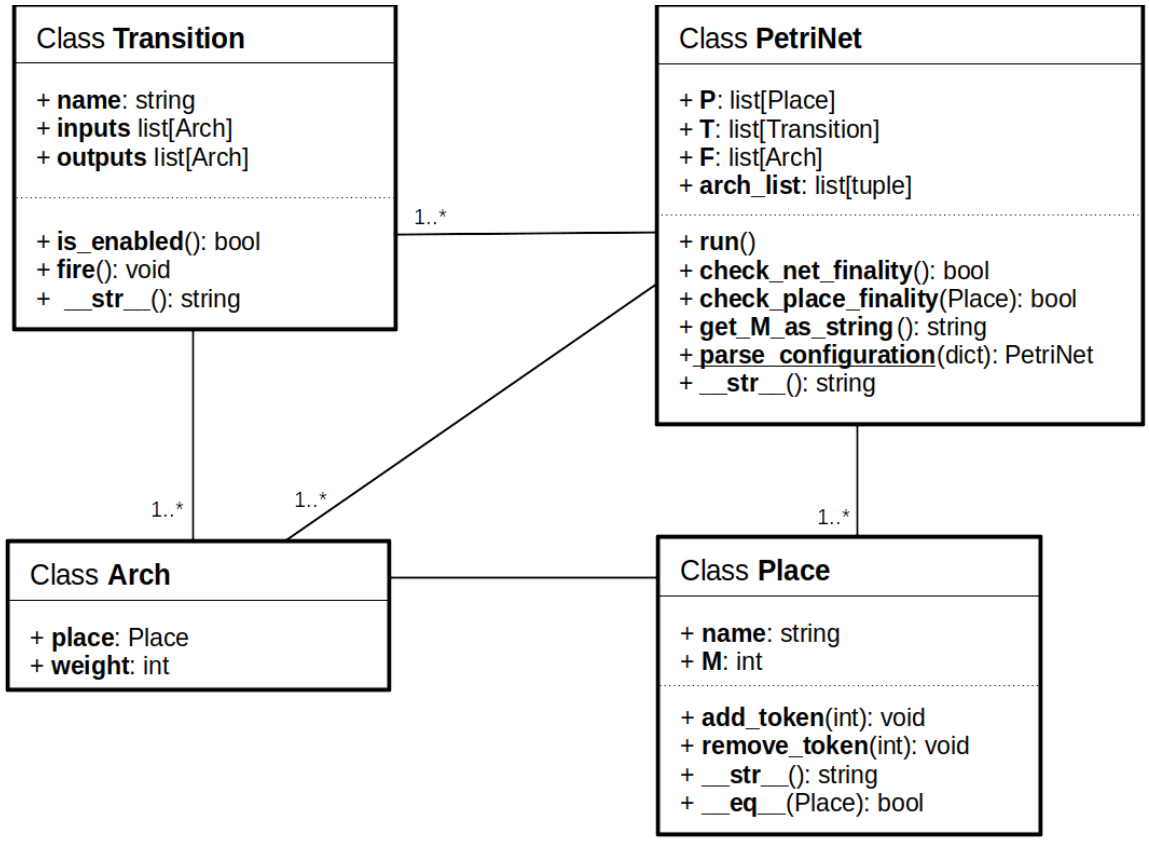
Slika 12. Sučelje PIPE2 -a



5.IMPLEMENTACIJA SIMULATORA OSNOVNE PETRIJEVE MREŽE

Kao praktični primjer u sklopu ovog rada implementiran je jednostavni simulator osnovne formulacije Petrijeve mreže, opisane u poglavlju x. Simulator je implementiran u programskom jeziku *Python*, korištenjem isključivo standardnih biblioteka. Petrijeva mreža implementirana je u objektno orijentiranoj paradigmi, gdje su svi njeni elementi apstrahirani razredima. Tako je mjesto definirano razredom *Place,* prijelaz razredom *Transition*, luk razredom *Arch*, a sama mreža razredom *PetriNet* u modulu *petri_net.py.* Dijagram razreda koji opisuje ovu implementaciju prikazan je na slici 13.

Slika 13 UML dijagram



Povezivanje elemenata, odnosno mjesta I prijelaza lukovima je glavna implementacijska odluka kod modeliranja Petrijeve mreže. U ovom radu primijenjena je arhitektura u kojoj mjesto ne sadrži nikakvu informaciju o tome s kojim je elementima povezano. Lukovi čuvaju informaciju o mjestu s kojim su povezani,, dok prijelazi sadrže popis ulaznih I izlaznih lukova. Ovim pristupom povezivanje elemenata ostvareno je uz to da svaki element sadrži minimalnu količinu informacije. Zbog ove arhitekture većina logike koja se koristi prilikom simuliranja prijelaza ugrađena je u metode *fire* i *is_enabled* razreda *Transition,* opisane sljedećim pseudokodovima:

Metoda **prijelaz_omogućen** | povratna vrijednost: *omogućenost(bool) {*
za svaki *luk* ***u*** *ulazni_lukovi* ***ako*** *luk.mjesto.M < luk.težine:*
vrati *F* ***vrati***
T
}

Metoda **aktiviraj** { ***ako***
*!**prijelaz_omogućen**():* ***vrati*** *null*
za svaki *luk* ***u*** *ulazni_lukovi:*
*luk.**ukloni_tokene**(luk.težina*
) ***za svaki*** *luk* ***u*** *izlazni_lukovi:*
*luk.**dodaj_tokene**(luk.težina)*
}

te Python kodom:

```
def is_enabled(self):
        """
```

```
            Check if a transition has met the conditions to be fired.


            Returns
-------
is_enabled: bool
                True if transition is enabled, false otherwise.
        """         for arch in
self.inputs:             if arch.place.M <
arch.weight:
                return False
return True
     def
fire(self):
        """
            Transition firing procedure.
        """         if not
self.is_enabled():
            return
         print('--Prijelaz {} je aktivirannn--
'.format(self.name))
              for arch in
self.inputs:
            arch.place.remove_token(arch.weight)
print('Uklanjanje značaka (količina = {}) sa mjesta {}'.\
format(arch.weight, arch.place))
                  for arch in
self.outputs:
            arch.place.add_token(arch.weight)
print('Postavljanje značaka (količina = {}) na mjesto {}'.\
format(arch.weight, arch.place))

         print('--Prijelaz {} je izvršen--
\n'.format(self.name))
     def __str__(self):
return self.name
```

iz ovog programskog koda vidljivi su implementacijski detalji, koji se u slučaju Pythona ne razlikuju pretjerano od pseudokoda. U programskom kodu možemo vidjeti pristupanje metodama objekta razreda *Place* preko objekta razreda *Arch* u metodi razreda *Transition*, ponašanje koje je omogućeno ranije opisanom arhitekturom u kojoj prijelaz sadrži logiku i većinu informacija, dok luk i mjesto sadrže samo najpotrebnije informacije. Možemo istaknuti i formatiranje akcija za ispis simulatora, te dokumentaciju koja je napisana po uzoru na popularnu biblioteku otvorenog koda *numpy*.

Simulator podržava proizvoljnu definiciju konfiguracije s imenima mjesta, imenima prijelaza, uređenim parovima koji definiraju lukove I tako specificiraju poveznice mjesta s prijelazima te težine tih poveznica, kao što je prikazano naprimjeru u poglavlju 2.1. Uz ove informacije u konfiguraciji je potrebno navesti početno stanje mreže, odnosno količinu značaka koja se u početnom trenutku nalazi na svakome od mjesta. Sve ove informacije zapisuju se u Pythonov razred *dict* koji modelira strukturu podataka rječnik, intuitivno rješenje za pohranjivanje definicije kakva je prikazana u poglavlju 2.1. Prikaz definicije ove konfiguracije u programskom kodu možemo vidjeti ispod.

```
# configuration example configuration
= {
  'P': ['p1', 'p2', 'p3', 'p4', 'p5', 'p6', 'p7'],
  'T': ['t1', 't2', 't3', 't4', 't5'],
  'F': [('p1', 't1', 1), ('t1', 'p2', 1), ('p2', 't2', 1), ('t2', 'p3', 1),
       ('t2', 'p4', 1), ('p3', 't3', 1), ('p4', 't4', 1), ('t3', 'p5', 1),
('t4', 'p6', 1), ('p5', 't5', 1), ('p6', 't5', 1),('t5', 'p7', 1)],
    'M': [1, 0, 0, 0, 0, 0, 0]
}
```

U razredu *PetriNet* implementirana je metoda koja parsira ovaj oblik definicije mreže te vraća objekt koji predstavlja jedan primjerak Petrijeve mreže s danom konfiguracijom. Ovime je omogućeno jednostavno definiranje I simuliranje različitih konfiguracija Petrijevih mreža korištenjem modula *runner.py*. Metoda *parse_configuration* razreda *PetriNet* prikazan programskim kodom u nastavku.

```
def parse_configuration(configuration):
```

```
        """
            Class method.
            Parse a configuration dictionary and return a Petri Net object.

            Parameters
----------
configuration: dict
                Configuration in a dictionary form.
                (see any configuration in runner.py as an example)

            Returns
-------             net:
PetriNet
                A Petri Net object defined in the configuration.
        """         P = []         for i in
range(len(configuration['P'])):
            P.append(Place(configuration['P'][i], configuration['M'][i]))
        F = []         T = []
for t in configuration['T']:
            inputs = []
            outputs = []             for
f in configuration['F']:

input = False
output = False
                if f[0] == t:
place = f[1]
output = True
```

```
                if f[1] == t:
place = f[0]
input = True
                if input or
output:
                   place = [p for p in P if p.name == place][0]
arch = Arch(place, f[2])                    F.append(arch)
if input:
                        inputs.append(arch)
if output:
                        outputs.append(arch)

            T.append(Transition(t, inputs, outputs))
return PetriNet(P, T, F, configuration['F'])
```

U ovoj metodi nalazi se najveći dio proceduralnog koda, budući da parsiranje definicija u opisanom formatu zahtjeva određenu vrstu logike koju je najlakše formalizirati nizom provjera uvjeta. Provjere su potrebne kako bi se odredilo koji je luk ulazni, a koji izlazni budući da ta informacija nije specificirana u definiciji.

Implementirana inačica Petrijeve mreže definira pravilo po kojemu se samo jedan prijelaz može aktivirati u jednom diskretnom vremenskom trenutku kako bi se pojednostavilo izvršavanje I praćenje simulacije. U slučaju da više prijelaza ostvari uvjete za aktivaciju (nedeterminizam), prijelaz koji će se aktivirati bira se slučajnim odabirom. Implementacija ovog mehanizma specificirana je sljedećim pseudokom:

Metoda **simuliraj**{
ima_prijelaza = T
dok *ima_prijelaza:*

ima_prijelaza = F ***za svaki*** *prijelaz* ***u*** ***slučajno_poredaj****(prijelazi):* ***ako*** *prijelaz.****omogućen****():* *prijelaz.****aktiviraj****()* *ima_prijelaza = T* ***prekini*** *petlju*
}

, te je njen izvorni kod vidljiv ispod u nastavku:

```
def run(self):
        """
            Run the Petri Net simulation.
        """         print('\n Početak simulacije
Petrijeve mreže\n')
print('Definicija:\n{}\n'.format(self.__str__()))
        print('Početno                                stanje:\n{}\n---------
\n'.format(self.get_M_as_string()))

        transitions_available = True
         while transitions_available:
transitions_available = False
                        for transition in
random.sample(self.T,len(self.T)):
                                if
transition.is_enabled():
                    transition.fire()
transitions_available = True                        break

         print('\nViše nije moguće izvesti niti jedan
prijelaz.')
```

```
        print('Trenutno stanje mreže
jest:\n{}\n'.format(self.get_M_as_string()))


        # analize network state after ending the simulation
if self.check_net_finality():
            print('Mreža je završila u finalnom stanju u kojemu su sve
značke' +
                  ' na mjestima koja nemaju izlaznih lukova.')
else:
            print('Protok mreže je zaglavio u stanju u kojemu su značke' +
                  ' raspoređene i na mjestima koja imaju izlazne' +
                  ' lukove.')
```

Od implementacijskih detalja, potrebno je istaknuti korištenje metode *sample* standardne biblioteke *random* u liniji 168 za određivanje liste prijelaza u slučajnom pretku koja ostvaruje ranije navedeno svojstvo nedeterminizma. Kao što je i navedeno, nakon završetka simulacije koja je opisana pseudokodom, dolazi do analize stanja mreže korištenjem metode *check_net_finality*.

Lukovima je omogućeno svojstvo težine koje određuje količinu znački potrebnu da se prijelaz aktivira ili količinu znački koju će prijelaz postaviti na mjesto u slučaju aktivacije. Po završetku simulacije određuje se jesu li sve značke završile na mjestima koja nemaju izlaznih lukova, te se ta informacije ispisuje u svrhu analize ponašanja mreže u raznim uvjetima. Logika za određivanje ovih karakteristika opisana je sljedećim algoritmima.

Metoda **mjesto_konačno** | argumenti: *mjesto*(*Place*) |
Povratna vrijednost: *mjesto_konačno(bool){* ***za svaki*** *prijelaz* ***u*** *prijelazi:* ***za svaki*** *ulazni_luk* ***u*** *prijelaz.ulazni_lukovi:* *ako* *ulazni_luk.mjesto == mjesto:*

vrati F

vrati T

}

Metoda **mreža_konačna** | povratna vrijednost: *mreža_konačna(bool){* ***za svako*** *mjesto* ***u*** *mjesta:*

*ako mjesto.M > 0 I !****mjesto_konačno****(mjesto):*

vrati F

vrati T

}

Izvorni kod prikazan je ispod

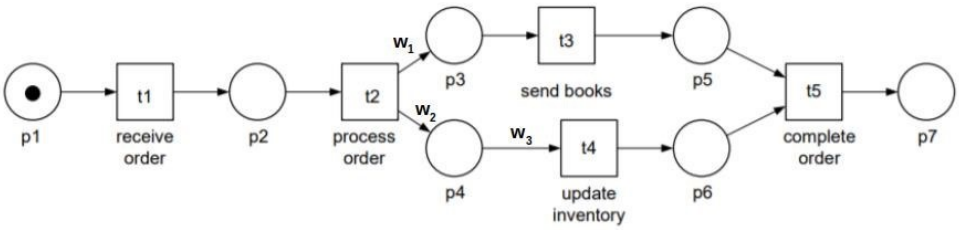
```
def check_net_finality(self):
        """
            Check if the network is in a final state.

            Returns
-------
is_final: bool
                True if Petri Net is in a final state, false otherwise.
        """         for place in self.P:             if place.M > 0
and not self.check_place_finality(place):
                return False
return True
     def check_place_finality(self,
place):
        """
            Check if a place is final (no output archs)
```

```
            Returns
-------
is_final: bool
                True if place is final, false otherwise.
        """         for t in self.T:
for input in t.inputs:
if input.place == place:
                    return False
return True
```

Testiranje ovog programskog rješenja obavljeno je bez korištenja neke standardne procedure. Svrha ovog modula jest simuliranje mreže bez dodatnih funkcionalnosti koje bi ga učinile upotrebljivim bez mijenjanja izvorno definiranog koda, poput zadavanja argumenata u naredbenoj liniji I sličnih procedura. Iz tog razloga, simuliranje je omogućeno na najjednostavniji način, uz zadavanje konfiguracije promjenom izvorno definiranog koda. Testiranje ovakvog modula obavljeno je pisanjem jednostavnih testova koji pokrivaju samo osnovni slučaj tijekom razvijanja rješenja.
Testiranje na logičke pogreške obavljeno je simuliranjem raznih konfiguracija mreže te utvrđivanjem točnosti izvođenja simulacije usporedbom sa simulacijom izvedenom na papiru.

Kao eksperiment s ovim simulatorom definirane su konfiguracije mreže koje arhitekturom odgovaraju primjerima sa slike 2 I slike 10. U prvom primjeru, onome koji odgovara mreži prikazanoj na slici 2, uvedene su promjene u vidu težina na određenim lukovima, kako bi se istražilo više scenarija ponašanja mreže, te su također definirana različita početna stanja koja predstavljaju različite slučajeve u kojima sustav može biti. Težine lukova su označene s w_1, w_2 I w_3 te su pozicije lukova s varijabilnim težinama vidljive na slici 2

Slika 2. Uzorak Petrijeve mreže koja predstavlja instancu jednog procesa (Matias Weske, 2007. Business Process Management, 158)



U tablici 5 prikazano je nekoliko slučajeva od kojih svaki prikazuje ponašanje mreže uz različito definirane težine lukove i/ili početno stanje. Ponašanje mreže opisano je navođenjem početnog I konačnog stanja (označenog s M_T) u kojemu se mreža nalazi.

Tablica 3 Ponašanje mreže uz različito definirane težine

Slučaj	w_1	w_2	w_3	M_0	M_T
A	1	1	1	[1 0 0 0 0 0 0]	[0 0 0 0 0 0 1]
B	1	1	1	[6 0 2 3 1 0 0]	[0 0 0 0 0 0 9]
C	3	2	1	[6 0 0 0 0 0 0]	[0 0 0 0 6 0 12]
D	3	2	3	[6 0 0 0 0 0 0]	[0 0 0 0 14 0 4]
E	2	4	2	[6 0 0 0 0 0 0]	[0 0 0 0 0 0 12]
F	2	4	1	[6 0 1 0 1 1 0]	[0 0 0 0 0 11 14]

Prikazani slučajevi imaju proizvoljno odabrane težine I početna stanja. Iz simulacija je vidljivo kako neuravnotežene težine i/ili početna stanja mogu lako izbaciti ovakav sustav iz stanja u kojemu će na kraju izvođenja postići željeno stanje (sve značke u p_7).

Ispis simulatora za slučaj A vidljiv je na slici 14.

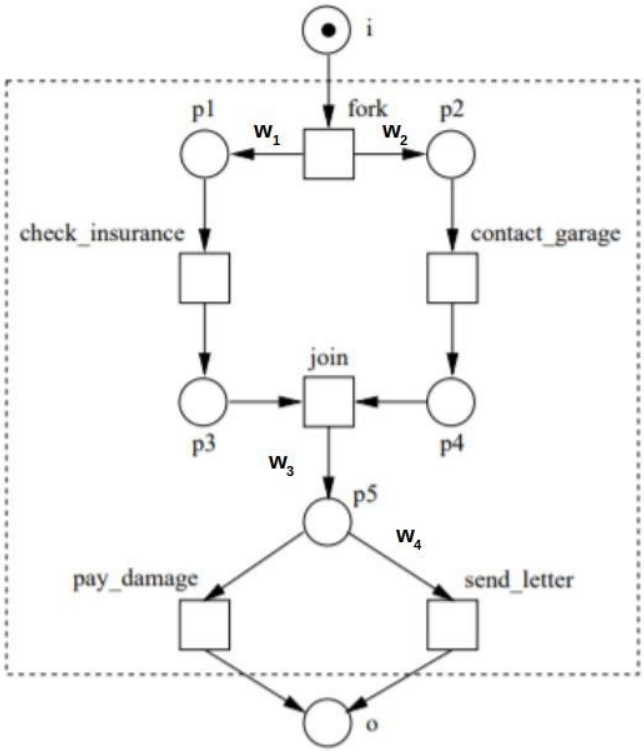
Slika 14 Ispis simulatora

```
 Početak simulacije Petrijeve mreže

Definicija:
P: p1, p2, p3, p4, p5, p6, p7
T: t1, t2, t3, t4, t5
F: (p1, t1, 1), (t1, p2, 1), (p2, t2, 1), (t2, p3, 1), (t2, p4, 1), (p3, t3, 1), (p4, t4, 1), (t3, p5, 1), (t4, p6, 1), (p5, t5, 1), (p6, t5, 1), (t5, p7, 1)

Početno stanje:
M(p1)=1
M(p2)=0
M(p3)=0
M(p4)=0
M(p5)=0
M(p6)=0
M(p7)=0
---------

--Prijelaz t1 je aktiviran--
Uklanjanje značaka (količina = 1) sa mjesta p1
Postavljanje značaka (količina = 1) na mjesto p2
--Prijelaz t1 je izvršen--

--Prijelaz t2 je aktiviran--
Uklanjanje značaka (količina = 1) sa mjesta p2
Postavljanje značaka (količina = 1) na mjesto p3
Postavljanje značaka (količina = 1) na mjesto p4
--Prijelaz t2 je izvršen--

--Prijelaz t4 je aktiviran--
Uklanjanje značaka (količina = 1) sa mjesta p4
Postavljanje značaka (količina = 1) na mjesto p6
--Prijelaz t4 je izvršen--

--Prijelaz t3 je aktiviran--
Uklanjanje značaka (količina = 1) sa mjesta p3
Postavljanje značaka (količina = 1) na mjesto p5
--Prijelaz t3 je izvršen--

--Prijelaz t5 je aktiviran--
Uklanjanje značaka (količina = 1) sa mjesta p5
Uklanjanje značaka (količina = 1) sa mjesta p6
Postavljanje značaka (količina = 1) na mjesto p7
--Prijelaz t5 je izvršen--


Više nije moguće izvesti niti jedan prijelaz.
Trenutno stanje mreže jest:
M(p1)=0
M(p2)=0
M(p3)=0
M(p4)=0
M(p5)=0
M(p6)=0
M(p7)=1

Mreža je završila u finalnom stanju u kojemu su sve značke na mjestima koja nemaju izlaznih lukova.
```

Drugi primjer definiran je nad mrežom sa slike 10, te su pozicije lukova s težinama w_1, w_2, w_3 I w_4 vidljive na slici.

Tablica 4. sadrži rezultate simulacije u nekoliko slučajeva s različito definiranim početnim vrijednostima I težinama lukova.

Izvor:(http://www.padsweb.rwth-aachen.de/wvdaalst/publications/p29.pdf)



Tablica 4 Rezultati simulacije

Slučaj	w_1	w_2	w_3	w_4	M_0	M_T
A	1	1	1	1	[1 0 0 0 0 0 0]	[0 0 0 0 0 0 1]
B	1	2	3	1	[6 0 2 3 1 0 0]	[0 0 0 0 6 0 27]
C	3	2	1	2	[6 0 0 0 0 0 0]	[0 0 0 6 0 0 10]
D	3	2	3	1	[6 0 1 1 0 0 0]	[0 0 0 6 0 0 39]
E	2	4	2	2	[6 0 0 0 0 0 0]	[0 0 0 0 12 0 18]
E’	2	4	2	2	[6 0 0 0 0 0 0]	[0 0 0 0 12 0 15]
F	2	4	3	3	[6 0 1 0 1 1 0]	[0 0 0 0 14 0 21]

Nije potrebno komentirati sve rezultate, oni samo svjedoče o točnosti rada simulatora. Rezultati u slučajevima E I E' su zanimljivi jer pokazuju jedno od svojstava ovog podskupa Petrijevih mreža, nedeterminizam. Nedeterminizam opisuje pojavu koja nastaje u trenutku kada imamo značku na mjestu p5. U slučaju da je w_4 postavljen na 1, mogu se aktivirati I prijelaz *pay_damage* I prijelaz *send_letter.* Tada se između ova dva prijelaza bira slučajnim odabirom. Kada je težina w_4 postavljena na vrijednost veću od 1, a imamo dovoljan broj značaka na mjestu p5 da se aktivira bilo koji od oba prijelaza, onda ovom nedeterminizmu možemo svjedočiti promatrajući broj značaka na mjestu o u konačnom trenutku T. U slučaju da je više puta odabran prijelaz *pay_damage*, M(o) će biti veći (slučaj E), jer je cijena prijelaza *pay_damage* jednaka 1. Budući da je cijena prijelaza *send_letter* u ovom slučaju veća od 1, ovaj će prijelaz "potrošiti" više značaka te ćemo tako imati manje značaka na mjestu o u trenutku T (slučaj E').

Cijeli kod moguće je pronaći na repozitoriju na web stranici GitHub :
https://github.com/Milas117/Implementacija-simulatora-Petrijevih-mreza.git

6. ZAKLJUČAK

Sve više poslovnih organizacija odlučuje se za modeliranje poslovnih procesa kako bi unaprijedili svoje poslovne procese. Promjenom poslovanja mijenjanju se i svi elementi arhitekture poslovanja zbog toga što oni uključuju različite metode i razvoj cjelovitog organizacijskog modela. Za Petrijeve mreže možemo reći da su jedna od tehnika simuliranja poslovnih procesa kao što je navedeno u radu uz grafički prikaz ponašanja sustava i mogućnost uvođenja matematičkih pravila.

Također se koriste za modeliranje dinamičkih sustava sa statičkom strukturom. Određeni događaj *t* povezuje se s mjestima direktnim lukovima koji moraju biti ispunjeni da bi se taj događaj dogodio i s uvjetima koji će biti ispunjeni nakon aktiviranja događaja. Budući da Petrijeve mreže opisuju strukturu sustava one predstavljaju model poslovnog procesa, a njegovi prijelazi predstavljaju modele aktivnosti.

Simulacija diskretnih događaja je pojam koji podrazumijeva različite vrste računalnih simulacijskih pristupa, a temelji se na ideji izrade računskog modela stvarnog sustava zamišljenog kao diskretni dinamički sustav. Dijelimo ih u pojmove od kojih se sastoji, a to su: entiteti, atributi, klase, skupovi, redovi čekanja, stanje sustava, uvjetni i bezuvjetni događaji, aktivnosti i proces. DES mreže su prototip standarda proširenih Petrijevih mreža za simulaciju diskretnih događaja.

Značke su nestrukturirane i nemaju identitet, stoga se ne mogu razlikovati jedna od druge. Ako je značka na mjestu p tada je uvjet p zadovoljen. Kod aktivacije prijelaza dogodi se događaj i stanje se mijenja.

Za obradu više slučajeva moramo koristiti Petrijevu mrežu visokog nivoa. Petrijeva mreža visokog nivoa organizirana je tako da svaka značka ima vrijednost koja se odnosi na slučaj kojem pripada i značke mogu koristiti samo one prijelaze koje pripadaju istom slučaju. Jedno od osnovnih svojstava Petrijevih mreža slobodnog izbora je činjenica da se može podijeliti u klastere.

U ovome je radu prikazan primjer jednostavnog simulatora. Simulator podržava proizvoljnu definiciju konfiguracije s imenima mjesta, imenima, uređenim parovima koji

definiraju lukove i tako specificiraju poveznice mjesta s prijelazima te težine tih poveznica. U implementaciji mreže definirano je pravilo po kojemu se samo jedan prijelaz može aktivirati u jednom vremenskom trenutku kako bi ih pojednostavilo. Za provjeru ovoga simulatora definirani su primjeri sa slike 2. i slike 10.

7. LITERATURA

Web izvori: https://people.etf.unsa.ba/~jvelagic/laras/dok/lekcijad13.pdf

https://core.ac.uk/download/pdf/197494277.pdf

https://repozitorij.fsb.unizg.hr/islandora/object/fsb%3A3877/datastream/PDF/view

https://www.fer.unizg.hr/_download/repository/INFMRE-2017_10.pdf

https://www.sciencedirect.com/topics/computer-science/petri-nets

https://www.fer.unizg.hr/en/course/des_a

https://www.intechopen.com/books/simulation-modelling-practice-and-theory/petri-netmodels-optimized-for-simulation

http://www.peterlongo.it/Italiano/Informatica/Petri/index.html

https://presmarymethuen.org/hr/dictionary/what-is-the-difference-between-discrete-eventsimulation-and-markov-modeling/

https://web.ma.utexas.edu/users/davis/375/popecol/lec1/petrinet.html

http://people.cs.vt.edu/kafura/ComputationalThinking/Class-Notes/Petri-Net-NotesExpanded.pdf

https://www.intechopen.com/books/petri-nets-manufacturing-and-computer-science/timedpetri-nets

Knjige:

1. Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A. Reijes: Fundamentals of Business Process Management, (2012) The M. C. Escher Company-Holland
2. Matias Weske: Buisness Process Management (2007) Springer-Verlag Berlin Heidelberg
3. Vesna Bosilj Vukšić, Tomislav Hernaus, Andrej Kovačić: Upravljanje poslovnim procesima, (2007) Školska knjiga Zagreb
4. Manuel Laguna, Johan Marklund:Buisness process modeling simulation and design, (2013) by Taylor & Francis Group, LLC

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Osnova svrha Petrijevih mreža specificiranje je poslovnih procesa te se može koristiti na formalni ili apstraktni način, te su tako važna osnova za procesne jezike. Ono što Petrijeve mreže razlikuje od drugih tehnika za modeliranje poslovnih procesa je za što su one grafički i matematički alat koji se može primijeniti na različite vrste sustava. Petrijeve mreže sastoje se od tri grafičke oznake mjesta, prijelaza i usmjerenih lukova koji ih povezuju.

Možemo ih podijeliti u klase, a to su: mreže stanja i prijelaza, to su mreže visoke razine kao što su obojene Petrijeve mreže, vremenske mreže i stohastičke Petrijeve mreže za koju možemo reći da ću mreže u kojima je najvažnija primjena Petrijevih mreža.

Ključne riječi: Petrijeve mreže, grafičke oznake, obojene Petrijeve mreže, DES

SUMMARY

The basis of the purposes of Petri's networks is to specify business processes and can be used in a formal or abstract way, and they are most important thing for process languages. What makes petri networks different from other techniques for modeling business processes is that they are a graphical and mathematical tool that can be applied to different types of systems. Petri nets consist of three graphic labels: places, transitions and directional arcs that connect them.

We can divide them into classes: networks of conditions and transitions, these are high-level networks such as colored Petri nets, time nets and stochastic Petri nets, which are at the same time the networks in which the application of Petri nets is most important.

Key words: Petri nets, graphic labels, colored Petri nets, DES