SAOBRAĆAJNI FAKULTET UNIVERZITETA U BEOGRADU

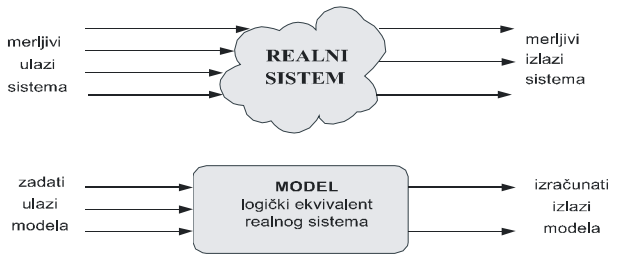
RAČUNARSKA
SIMULACIJA
- SKRIPTA -

BEOGRAD 2007.

SADRŽAJ

I MODELIRANJE I SIMULACIJA

1.MODELIRANJE I MODELI



Modeliranje je osnovni proces ljudskog uma. To je isplativo (u smislu troškova) korišćenje nečega (modela) umesto nečeg drugog (realnog sistema) sa ciljem da se dođe do određenog saznanja. Rezultat modeliranja je model.

Model je uprošćena i idealizovana slika realnosti. Model je apstrakcija realnog sistema, zadržava samo one osobine originala koje su bitne za izučavanje. Nivo apstrakije utiče na validnost modela tj. na uspešnost predstavljanja realnog sistema preko modela. Isuviše složeni modeli su skupi i neadekvatni dok istviše prosti modeli neoslikavaju posmatrani sistem na pravi način.

2.VRSTE MODELA

I podela:

1. **Mentalni** (misaoni) modeli. Konstruiše ih ljudski um i na osnovu toga deluje. Omogućavaju komunikaciju među ljudima, planiranje aktivnosti itd.
2. **Verbalni** modeli su direktna posledica mentalnih modela, njihov izraz u govornom jeziku. Uobičajno se predstavljaju u pisanom obliku i spadaju u klasu neformalnih modela.
3. **Fizički** modeli predstavljaju umanjene modele realnog sistema. Ponašaju se kao njihovi originali a prave se na osnovu teorije sličnosti ili fizičkih zakona sličnosti.
4. **Matematički** modeli se javljaju ako su veze između objekata opisane matematičkim (numeričkim) relacijama. Polazi se od verbalnog modela koji se transformiše u stanje koje se može opisati matematičkim jezikom. Spadaju u klasu apstraktnih modela a primenjuju se u naučnim i inžinjerskim disciplinama.

Različiti fizički modeli mogu imati iste matematičke modele pa se kaže da između njih postoji matematička analogija (analogija u ponašanju). To pruža mogućnost da se neki od fizičkih objekata jednog modela koristi za analizu drugog modela i tada se on naziva analogni model.

5. **Konceptualni** modeli nastaju na osnovu strukture, logike rada sistema. Zovu se još i strukturni modeli pošto u grafičkom obliku ukazuju na strukturu sistema te su zgodno sredstvo za komunikaciju. Predstavljaju osnovu za izradu računarskih modela.
6. **Računarski** (simulicioni) modeli su prikaz konceptualnih modela u obliku programa za računar korišćenjem programskih jezika i usko su vezani za razvoj računarske nauke.

Modeli se često dele na **materijalne** (hemijska struktura molekula, model aviona) i **simboličke** (matematički, konceptualni, računarski, simulacioni).

II podela:

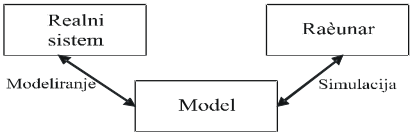
1. **Neformalni** opis modela daje osnovne pojmove o modelu i najčešće nije potpun i precizan. Zbog toga se vrši podela na:
   a. **objekte** – to su delovi iz kojih se sastoji model;
   b. **opisne promenljive** – opisuju stanje u kome se objekat nalazi u nekom vremenskom trenutku;
   c. **pravila interakcije objekata** – definišu kako objekti modela utiču jedni na druge i na opisne promenljive u cilju promena njihovog stanja.

Neformalni opis je dosta brz i lak te zbog toga može biti **nekompletan** (ne sadrži sve situacije koje mogu da nastupe), **nekonzistentan** (predviđanje dva ili više pravila za istu situaciju – kontradiktorne akcije), **nejasan** (ako nije definisan redosled akcija). Ovakve situacije se prevazilaze pravilima i konvencijama u komuniciranju zvanim formalizmi.

2. **Formalni** opis modela treba da obezbedi veću preciznost, potpunost u opisivanju modela. Omogućava i formalizovanje nekompletnosti, nekonzistentnosti i nejasnosti kao i usmeravanje pažnje na karakteristike objekata koje su od najvećeg značaja za istraživanje (apstrakcija).

Preporuke pri izradi modela: **1.**granica modela mora biti odabrana tako da model obuhvata samo fenomene od interesa; **2.**model ne sme biti suviše složen niti detaljan; **3.**model ne sme suviše da pojednostavi problem; **4.**model je razumno rastaviti na više modula radi lakše izgradnje i provere; **5.**korišćenje neke od proverenih metoda za razvoj algoritama i programa; **6.**provera logičke i kvantitativne ispravnosti i modela i modula.

3.RAČUNARSKA SIMULACIJA (modeliranje i simulacija)



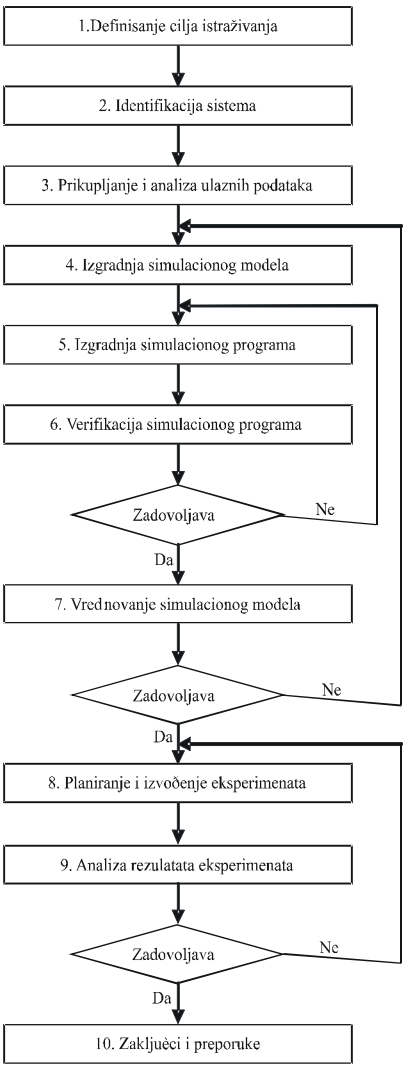
Modeliranje i simulacija izražavaju složenu aktivnost koja sadrži tri elementa:

1. **Realni sistem** je uređen, međuzavistan skup elemenata koji formiraju jednu celinu i deluju zajednički kako bi ostvarili zadati cilj. Realni sistem je izvor podataka, oblika x(t), za specifikaciju modela.
2. **Model** je apstraktni prikaz sistema, daje njegovu strukturu, komponente i njihovo uzajamno delovanje. U računarskoj tehnici model predstavlja skup instrukcija (program) koje služe da se generiše ponašanje simuliranog sistema. Model ima svoje objekte koji su opisani atributima i promenljivima.
3. **Računar** je uređaj za izvrašavanje instrukcija modela, koje generišu razvoj modela u vremenu na osnovu ulaznih podataka.

Modeliranje je proces kojim se uspostavlja veza između realnog sistema i modela. Odnosi se na validnost modela koja opisuje koliko verno model predsavlja simulacioni sistem (validacija).

Simulacija je proces koji uspostavlja vezu između modela i računara. Odnosi se na proveru da li simulacioni program verno prenosi model na računar i na tačnost kojom računar vrši instrukcije. Procena korektnosti simulatora zove se verifikacija.

4.SIMULACIONI PROCES (dijagram toka simulacionog procesa)



Simulacioni proces je struktura rešavanja stvarnih problema pomoću simulacionog modeliranja. Sastoji se iz više koraka i nije strogo sekvencijalna, moguć je povratak na korake procesa.

Koraci simulacionih procesa:

1. Definicija cilja simulacione studije;
2. Identifikacija sistema (opis komponenata, način rada, veza sa okolinom, formalni prikaz sistema);
3. Prikupljanje podataka o sistemu i njihova analiza;
4. Izgradnja simulacionog modela (stvaranje konceptualnog modela koji adekvatno opisuje sistem);
5. Izgradnja simulacionog programa (izbor programskog jezika i stvaranje simulacionog programa);
6. Verifikacija simulacionog programa (da li nam program verno prenosi model);
7. Validacija (vrednovanje) simulacionog modela (da li model adekvatno predstavlja realni sistem);
8. Planiranje simulacionih eksperimenata i njihovo izvođenje;
9. Analiza rezultata eksperimenata (najčešće statistička analiza);
10. Zaključci i preporuke.

5.PODELA SIMULACIONIH MODELA

Prva podela je prema vrsti promenljivih u modelu:

- **Deterministrički modeli** su modeli čije se stanje može predvideti tj. novo stanje je potpuno određeno prethodnim. Pr: stanje sistema S_n se menja pod uticajem aktivnosti A, determinističkog trajanja od 45 sekundi, u stanje S_{n+1}.
- **Stohastički modeli** čije se ponašanje ne može unapred predvideti ali se mogu predvideti verovatnoće promena stanja. Za stohastičke modele je karakteristično slučajno ponašanje, postojanje slučajnih promenljivih. Pr: stanje S_n se pod uticajem aktivnosti A može promeniti u stanja S'_{n+1}, S''_{n+1} ili S'''_{n+1} prema uniformnoj raspodeli verovatnoća stanja.

Druga podela je prema načinu na koji se stanje modela menja u vremenu:

- **Diskretni modeli** u kojima se stanje sistema menja samo u pojedinim tačkama u vremenu, nema kontinualne promene stanja. Te promene se nazivaju događaji.
- **Kontinualni modeli** u kojima se promenjive stanja menjaju kontinualno u vremenu. Na digitalnim računarima se ne mogu izvoditi kontinualne promene veličina već se moraju aproksimirati skupom diskretnih vrednosti.
- **Kontinualno – diskretni modeli** sadrže i kontinualne i diskretne promenljive.

6.VRSTE SIMULACIONIH MODELA

Postoje četiri osnovne vrste simulacionih modela:

1. **Monte Karlo** simulacija je statička simulacija kod koje se u rešavanju problema koristi stvaranje uzoraka iz raspodela slučajnih promenljivih, a problemi mogu biti i determinističkog i stohastičkog oblika.
   Tipovi primene Monte Karlo simulacije:
   - ***deterministički problemi koje je teško ili skupo rešavati***. Pr: računanje integrala koji se ne mogu izračunati anlitički. Generiše se niz slučajnih tačaka x_j, y_j sa jednakim verovatnoćama unutar određenog pravougaonika i zatim ispituje koliko je generisanih tačaka unutar površine koja odgovara integralu.
   - ***složeni fenomeni koji nisu dovoljno poznati***. Karakterističnost je da nije poznat način uzajamnog delovanja između elemenata već su poznate samo verovatnoće ishoda, koje se koriste za izvođenje niza eksperimenata koji daju uzorke mogućih stanja zavisnih promenljivih. Statističkom analizom dobija se raspodela verovatnoća zavisnih promeljivih. Primena kod analiziranja društvenih ili ekonomskih fenomena.
   - ***statistički problemi koji nemaju analitička rešenja***. Pr: testiranje novih hipoteza, procena kritičnih vrednosti. Prilikom rešavanja takvih problema takođe se koristi generisanje slučajnih brojeva i promenljivih.
2. **Kontinualna simulacija** (dinamička) se koristi za dinamičke probleme kod kojih se promenljive stanja menjaju kontinualno u vremenu. Postoje dve klase problema koji se rešavaju ovom metodom:
   - jednostavni problemi koji su opisani detaljno i kod kojih su promene 'glatke', opisuju se diferencijanim jednačinama. To su problemi iz fizike, biologije i inženjerstva.
   - problemi koji nastaju opisom veoma složenih sistema u agregiranom obliku, u kom se niz elemenata sistema redukuje na manji broj komponenata a promene u sistemu se aproksimiraju konstantnim brzinama promene. To su najčešće problemi iz ekonomije i društvenih nauka.

   Postoje tri tipa ovih simulacionih modela:
   - ***modeli koji se opisuju običnim diferencijanim jednačinama*** (postoji jedna nezavisna promenljiva). To su problemi kretanja, fizički, hemijski, biološki i drugi procesi gde se radi o jednoj nepoznatoj funkciji $y = y(t)$ jedne nepoznate promenljive (t). Izražavaju se matematičkim jednačinama u kojima pored nezavisne promenljive i nepoznate funkcije javlja i izvod te funkcije (dy/dt). To su obične diferencijalne jednačine.
   - ***modeli koji se opisuju sistemima parcijanim diferencijalnim jednačinama.*** Postoji više od jedne nezavisne promenljive po kojima se traže izvodi zavisne promenljive. Njima se opisuju problemi aerodinamike, hidrodinamike i meteorologije.
   - ***modeli dinamike sistema***. Modeliraju sisteme sa povratnom vezom.
3. **Simulacija diskretnih događaja** (dinamička) se bavi modeliranjem sistema koji se mogu predstaviti skupom događaja. Događaj je diskretna promena stanja entiteta sistema. Nastupa u određenom trenutku a te promene stanja entiteta se dešavaju diskontinualno u vremenu, tj. samo u nekim trenucima. Između dva uzastopna događaja stanje sistema se ne menja.
4. **Mešovita simulacija (kontinualno – diskretna)** se uvodi da bi se modelirali (simulirali) oni sistemi koji sadrže procese i događaje. Procesi teku kontinualno dok događaji dovode do diskontinuiteta u ponašanju sistema. Tako je razvijena mešovita simulacija koja integriše diskontinualnu i kontinualnu. To se postiže uvođenjem dva tipa događaja. **1.*Vremenski događaji*** koje generiše mehanizam upravljanja događajima. Oni mogu da izazovi trenutnu promenu stanja kontinualne promenljive. **2.*Događaji stanja*** koja aktivira mehanizam pomaka vremena sa konstantnim prirastom. Oni mogu da aktiviraju događaje diskretnog dela modela. Poznati simulacioni jezici za mešovitu simulaciju su GASP i SLAM.

Pri izboru simulacionog modela najvažnije je da model bude što jednostavniji, što zbog njegovog razvoja tako i zbog potrebe da ga korisnik lakše razume.

II KLASIFIKACIJA MODELA

1.Klasifikacija u odnosu na promenljive

Kod svakog modela moguće je identifikovati opisne promenljive značajne za njegovo razumevanje, opis i upravljanje. Opisne promenljive se dele na one koje je moguće i nemoguće posmatrati (meriti). One mogu biti ulazne, izlazne i promenljive stanja. Svaka promenljiva ima svoj opseg (domen) i jednu funkciju kojom se opisuju te promene u vremenu.

U pogledu promenljivih stanja:

- **Modeli bez memorijom (trenutne funkcije)** nemaju ni jednu promenljivu stanja. Njihove izlazne promenljive zavise od ulaznih u trenutku posmatranja.
- **Modeli sa memorijom** imaju barem jednu promenljivu stanja.

U pogledu ulaznih promeljivih:

- **Autonomni** su bez ulaznih promenljivih.
- **Neatonomni** su sa ulaznim promenljivima. Oni se dele na modele sa i bez izlaznih promenljivih.

U pogledu izlaznih promenljivih:

- **Zatvoreni** modeli su autonomni modeli koji ne sadrže izlaznu promenljivu.
- **Otvoreni** modeli su autonomni modeli sa izlaznim promenljivama i svi neautonomni modeli.

2.Klasifikacija u odnosu na prirodu opsega vrednosti promenljivih modela

Pod opsegom vrednosti promenljivih se podrazumeva skup svih vrednosti koje može da uzme promenljiva. Vrednosti promenljivih mogu biti iz prebrojivog (diskretnog) i neprebrojivog (kontinualnog) skupa. U skladu sa tim razlikujemo tri klase modela:

1. ***Modeli sa diskretnim stanjima*** – sve opisne promenljive uzimaju vrednost iz skupa diskretnih vrednosti.
2. ***Modeli sa kontinualnim stanjima*** – sve opisne promenljive uzimaju vrednosti iz podskupa realnih brojeva (iz kontinualnog skupa).
3. ***Modeli sa mešovitim stanjima*** – neke opisne promenljive uzimaju vrednosti iz diskretnog a ostale iz kontinualnog skupa.

3.Klasifikacija u odnosu na prirodu opsega vrednosti promenljive 'vreme'

Skup vrednosti promenljive 'vreme' može biti prebrojiv ili neprebrojiv. Stoga razlikujemo:

1. Modele sa **kontinualnim** vremenom (vreme kontinualno teče)
   - i kontinualnim promenama stanja;
   - i diskretnim promenama stanja (promene stanja se dešavaju samo u diskretnim skokovima).
2. Modeli sa **diskretnim** vremenom (vreme se povećava u inkrementima koji ne moraju biti ekvidistantni)
   - i kontinualnim promenama stanja;
   - i diskretnim promenama stanja.

4.Klasifikacija u odnosu na vremensku zavisnost modela

1. **Varijantan** (vremenski promenljiv) model – ako struktura modela zavisi od vremena.
2. **Invarijantan** (vremenski nepromenljiv) model – ako struktura modela ne zavisi od vremena.

5.Klasifikacija u odnosu na determinizam

1. **Deterministički** modeli – vrednost promenljivih stanja i ulaznih promeljivih u jednom trenutku jednoznačno određuju vrednosti promenljivih stanja u sledećem trenutku. Ovakvi modeli ne sadrže slučajne promenljive.
2. **Nedeterministički** (stohastički) modeli – postoji bar jedna slučajna promeljiva.

6.Klasifikacija u odnosu na predviđanje budućnosti

1. **Anticipatorski** modeli koji za izračunavanje promenljivih stanja uzimaju u obzir i buduće vrednosti ulaznih promenljivih.
2. **Neanticipatorski** modeli kod kojih prethodno nije slučaj.

7.Klasifikacija u odnosu na linearnost

Linearni sistemi menjaju stanja i daju izlaze poštujući zakonitosti linearnih transformacija. Jedna linearna transformacija $L : U \to Y$ zadovoljava princip superpozicije ako za $u_1, u_2 \in U$ i c_1, c_2 važi:

$$L[c_1 u_1(t) + c_2 u_2(t)] = c_1 L[u_1(t)] + c_2 L[u_2(t)]$$

Odnosno, ako su ulazi u sistem: $u_1(t)$ i $u_2(t)$, izlazi iz sistema $y_1(t)$ i $y_2(t)$, a c_1 i c_2 skalari, tada važi: $u(t) = c_1 u_1(t) + c_2 u_2(t)$ i $y(t) = c_1 y_1(t) + c_2 y_2(t)$

8.Klasifikacija prema vrsti računara

Tri vrste računara se mogu koristiti za simulaciju: **analogni**, **digitalni** i **hibridni**. Skoro svi modeli se mogu simulirati na digitalnim i hibridnim računarima dok se na analognim računarima mogu simulirati samo kontinualni modeli sa kontinualnim vremenom.

9.Klasifikacija u odnosu na formalni opis model

Formalni opis modela podrazumeva precizan, matematički opis modela.

Modeli sa kontinualnim promenama vremena se opisuju diferencijalnim jednačinama i spadaju u kontinualne vremenske modele. Oni ne moraju da menjaju svoje stanje kontinualno.

Diskontinualnu modeli su oni modeli u kojima je najmanje jedna promenljiva stanja i/ili njen izvod diskontinualan.

Diskretni modeli menjaju svoje stanje u diskontinualnim vremenskim trenucima.

FORMALNA SPECIFIKACIJA MODELA

Teorija skupova omogućava konstruisanje formalizama koji se koriste za opisivanje objekata modela. Svaka klasa može se predstaviti odgovarajućim formalizmom koji definiše njene parametre i ograničenja. Da bi se definisao poseban objekat neke klase u okviru nekog formalizma, paramertima formalizma dodeljuju se vrednosti koje zadovoljavaju ograničenja.

Jedan objekat iz klase zadate formalizmom, definiše se na taj način što se parametrima dodeljuju konkretne vrednosti koje zadovoljavaju ograničenja. Struktura formalizma odnosi se kako na parametre, tako i na ograničenja.

Klase objekata su najčešće povezane tako da se te veze mogu formalizovati kao preslikavanje iz jedne klase u drugu. Posebno su interesantne tri vrste takvih preslikavanja: apstrakcija, asocijacija i specifikacija.

Apstrakcija je proces kojim se vrši razdvajanje 'bitnih' od 'nebitnih' osobina, kako bi se ukazalo na suštinu nekog objekta. To je preslikavanje objekta jedne klase u drugu, manje složenu klasu. Izvorna klasa naziva se ***konkretnom***, a ciljna ***apstraktnom***.

Asocijacija je vrsta preslikavanja od višeg ka nižem nivou u hijerarhiji specifikacije sistema. Inverzno preslikavanje nazivamo ***realizacijom*** ili ***implementacijom***.

Specifikacija – za datu klasu moguće je definisati podklase, uvođenjem novog formalizma za svaku podklasu, čije je objekte moguće koristiti unutar novih formalizama. Da bi uspostavili veze između objekata različitih podklasa potrebno je definisati preslikavanje koje prevodi jedan poseban formalizam u drugi, generalniji. Ako takvo preslikavanje postoji, onda svi koncepti i akcije koji se mogu primeniti nad objektima definisanim u generalnom formalizmu, primenjivi su i nad objektima definisanim u posebnom formalizmu.

FORMALNI MODEL ULAZNO – IZLAZNOG SISTEMA

Posmatrajmo ulazno – izlazni sistem kao skupovnu strukturu:

$$M = \{T, U, \Omega, S, Y, \delta, \lambda\}$$

gde su:

T - **vremenska baza** je skup vrednosti vremena u modelu na osnovu kojih se planira redosled događaja[1].

U - **skup ulaza** je deo interfejsa preko kog okruženje utiče na sistem[2].

Ω - **skup ulaznih segnemata** opisuje oblik ulaza u sistem za neki vremenski period[3].

S - **skup internih stanja** predstavlja memoriju sistema, tj. uticaj prethodnih događaja na njegov sadašnji i budući odziv. Od izbora internih stanja i njihovog karaktera zavisi struktura modela.

δ - **funkcija prelaza stanja** je preslikavanje $\delta : SxU \to S$.[4]

Y - **skup izlaza** predstavlja deo interfejsa kojim sistem utiče na okruženje.

λ - **funkcija izlaza** predstavlja preslikavanje $\lambda : S \to Y$ koje povezuje pretpostavljeno stanje sistema sa uticajem sistema na njegovu okolinu. Opštija funkcija izlaza je preslikavanje $\lambda : SxUxT \to Y$. Često λ nije jednoznačno preslikavanje, tako da se iz okruženja, stanje sistema ne može direktno posmatrati.

---

[1] Domen vremenske baze je oblast celih I^+ ili realnih R brojeva.

[2] Domen je najčešće R^n za neko $n \in I^+$, koje predstavlja broj ulaznih promenljivih. Domen može biti $U_m \cup [\emptyset]$, gde je U_m skup eksternih događaja a $\emptyset$ je prazan skup.

[3] Određen je okruženjem sistema i definisan je preslikavanjem $\omega : \{t_0, t_1\} \to U$, gde je $\{t_0, t_1\}$ interval vremenske baze između početnog i krajnjeg trenutka. Skup svih ulaznih segmenata naziva se (U,T), znači $\omega \subset (U,T)$. U praksi je Ω skup kontinualnih segmenata $T \in R$ i $U \in R^n$ ili skup diskretnih segmenata $U \in U_m$ i $T \in R$. Segment diskretnih ulaznih događaja je preslikavanje $\omega : \{t_0, t_1\} \to U_m \cup \{\emptyset\}$, takvo da $\omega(t) = \emptyset$ isključuje mogućnost ograničenog skupa događaja $\{\tau_1, \tau_2, ..., \tau_n\} \subseteq \{t_0, t_1\}$. Kada je $T \in I^+$, Ω predstavlja skup konačnih sekvenci.

[4] Kada se na sistem sa stanjem S u vremenu t_0 dovede ulazni segment $\omega : \{t_0, t_1\} \to U$, tada $\delta(S, \omega)$ preslikava stanje sistema sa vremenom t_1, interno stanje u početnom trenutku i ulazni segment od tog trenutka, nadalje jedinstveno određuju stanje sitema na kraju segmenta. Za svako $s \in S$ i $\omega \in \Omega$ i t u oblasti ω važi ***aksiom semi grupe***: $\delta(S, \omega) = \delta[\delta(S, \omega_{t>}), \omega_{<t}]$, gde su $\omega_{t>} = \omega \mid \{t_0, t_1\}$ deo ω između t_0 i t_1 a $\omega_{<t} = \omega \mid \{t, t_1\}$ deo ω između t i t_1. Ovo zahteva da stanje $S_t = \delta(S, \omega_{t>})$ koje pripada bilo kom trenutku t, sumira potrebne prethodne događaje, tako da nastavljanje eksperimenta od tog stanja rezultuje istim krajnjim stanjem. Pronalaženje odgovarajućeg i korisnog prostora stanja je srž modeliranja.

Na osnovu definisanog formalizma moguće je odrediti pojam ponašanja sistema. To je spoljašnja manifestacija njegove interne strukture, pa je relacija pomoću koje se može odrediti ponašanje sistema zadata sa $(U,T)x(Y,T)$. Ova relacija se određuje tako što svakom stanju $s \in S$ i ulaznom segmentu $\omega : \{t_0, t_1\} \to U$ u Ω pridružena je jedinstvena trajektorija stanja $S_{traj_{s,\omega}} : \{t_0, t_1\} \to S$, tako da je $S_{traj_{s,\omega}}(t_0) = s$ i $S_{traj_{s,\omega}}(t) = \delta(S, \omega_{t>})$, za $t \in \{t_0, t_1\}$. Takva trajektorija je rezultat analognog rešenja ili je izračunata u toku izvršenja simulacije na računaru. Osmotriva projekcija ove trajektorije je trajektorija izlaza koja se definiše zajedno sa $y \in Y$ i $\omega \in \Omega$ kao $O_{traj_{s,\omega}} : \{t_0, t_1\} \to Y$. U slučaju da proste funkcije izlaza $\lambda(S)$ $O_{traj_{s,\omega}}(t) = \lambda(S_{traj_{s,\omega}}(t))$. Tada se ponašanje sistema definiše na osnovu ulazno – izlazne relacije $R_s = \{(\omega, \rho) \mid \omega \in \Omega, \rho \in O_{traj_{s,\omega}} \text{ za } s \in S\}$. Svaki segment $(\omega, \rho) \in R_s$ naziva se par ulazno – izlaznog segmenta i predstavlja rezultat posmatranja ili eksperimenta na sistemu u kome je ω ulaz u sistem, dok je ρ posmatrani izlaz sistema. Za jedan ulazni segment može postojati više izlaznih.

III OCENA PARAMETARA MODELA

Ocena parametara modela je postupak eksperimentalnog određivanja vrednosti parametara koji se pojavljuju u matematičkom opisu modela. Polazi se od pretpostavke da je struktura modela, odnosno veza između promenljivih modela i parametara, eksplicitno data. Uglavnom se polazi od pretpostavke da svi parametri imaju ne nulte vrednosti.

Posmatrajmo sistem definisan matematičkim modelom u prostoru stanja:

$$s' = f(s,u,p,t),\ y = g(s,u,p,t),\ s(t_0) = s_0$$

Gde su:

s - vektor stanja dimenzije n,

u - vektor ulaza dimenzije m,

y - vektor izlaza dimenzije k,

p - vektor sastavljen od n_p nepoznatih parametara,

f i g su odgovarajuće vektorse funkcije,

s_0 - početni uslovi.

Potrebno je za dati model i skup ulaznih i izlaznih podataka, odrediti vektor nepoznatih parametara p, pri čemu moraju biti poznati početni uslovi. Pitanje je da li se parametri mogu identifikovati, tj. da li ih je moguće matematički odrediti?

Postoji više različitih pristupa problemu ocene parametara. Klasičan pristup je čisto statistički i njegova primena zahteva ispunjenje odgovarajućih uslova. Moguće je problem određivanja parametara posmatrati i kao optimizaciju pogodno definisane funkcije cilja, što zahteva manje početnih pretpostavki. Jedinstveni pristup oceni parametara modela nije moguće realizovati. Stoga se za ocenu parametara za pojedine klase modela koriste algoritmi čija struktura zavisi od:

1. **formalizma modela**: **a)**kontinualno – vremenski; **b)**diskretno – vremenski; **c)**linearni ili nelinearni; **d)**deterministički, stohastički.
2. **konteksta modeliranja**: **a)**tip promenljivih stanja; **b)**apriorno znanje; **c)**svrha modela.
3. **filozofija procene**: **a)**kriterijumi; **b)**numerička procedura; **c)**prilaz izračunavanju.

OCENE PARAMETARA DETERMINISTIČKOG MODELA

Za model sa poznatom strukturom mogu se uvesti sledeće definicije:

1. Za skalarni parametar p_j kaže se da je identifikabilan na intervalu $[t_0, T]$, ako postoji konačan broj rešenja za p_j koja proizilaze iz relacije datog modela. Skalarni parametar je neidentifikabilan ako postoji beskonačan broj rešenja za p_j, koja proizilaze iz relacija modela.
2. Opis modela je sistemski identifikabilan, ako su svi parametri idetifikabilni. U suprotnom, ako je barem jedan parametar neidentifikabilan, opis modela je sistemski neidentifikabilan.
3. Skalarni parametar p_j je jedinstveno idenifikabilan na intervalu $[t_0, T]$, ako postoji jedinstveno rešenje za p_j koje proizilazi iz relacija modela sa datim ulazom.
4. Specifikacija modela je parametarski identifikabilna na intervalu $[t_0, T]$ ako su svi parametri jedinstveno identifikabilni.

Ove definicije su primenljive i na diskretne vremenske modele i dovoljne su da opišu sve uslove identifikabilnosti ali ne pokazuju kako se identifikabilnost može proveriti. Identifikabilnost sistema i parametara zavisi od dva faktora: **1.**Specifičnosti primenjenog ulaza i početnih uslova; **2.**Strukture jednačina i ograničenja.

OCENE PARAMETARA MODELA STOHASTIČKIH SISTEMA

Kod ovakvih sistema problem se usložnjava zahtevom za uključivanje parametara pojedinih realizacija procesa. Identifikabilnost kod stohastičkih sistema zavisi od tri faktora: **1.***strukture* (stavlja parametre jedne sa drugima kao i sa ulazima i izlazima); **2.***ulaza*; **3.***procedure procene* (moraju da konvergiraju ka stvarnim vrednostima parametara – konzistentnost).

STATISTIČKI PRISTUP PROCENI PARAMETARA STATISTIČKIH MODELA

Statističke ocene parametara mogu se realizovati u prostoj sekvencijalnoj formi ili rekurzivnoj formi (za prikupljanje podataka u realnom vremenu).

Ocena srednje vrednosti slučajne promenljive

Sekvencijalni metod – pod uslovom da su svi rezultati dostupni u vreme izračunavanja, sva merenja se sabiraju i rezultat se deli sa brojem uzoraka: $\overline{y_n} = \frac{1}{N}\sum_{i=1}^{N} y_i$.

Rekurzivni metod – karakteristike su da se baza podataka proširuje tokom računanja i da su međurezultati dostupni i teže sekvencijalnom rešenju kako izračunavanje odmiče. Računa se po proceduri: $\hat{a}_0 = 0$; $\hat{a}_k = \hat{a}_{k-1} - \frac{1}{k}(\hat{a}_{k-1} - y_k)$, gde je y_k tekući uzorak $(k = 1,2,...,N)$.

OCENA NEPOZNATOG PARAMETRA PO METODI NAJMANJIH KVADRATA

Za model kod koga je veza između parametara modela i rezultata eksperimenta data relacijom: $y_i = x_i a + e_i$ za $i = 1,2,...,N$, pretpostavljajući da su x_i poznate vrednosti a e_i greške merenja, potrebno je proceniti nepoznati parametar a.

Sekvencijalno rešenje – cilj je da se minimizira odstupanje između podataka koje generiše realni sistem i podataka koji se dobijaju na osnovu modela, odnosno da se minimizira kvadrat greške: $J = \sum_{i=1}^{N}[y_i - x_i a]^2 = \sum_{i=1}^{N} e_i^2$. U ovom slučaju imamo jedan nepoznat parametar, pa $\frac{dJ}{da} = 0$ dobija se $2\sum_{i=1}^{N}[y_i - x_i a]\cdot(-x_i) = 0$ odakle sledi $\left[\sum_{i=1}^{N} x_i^2\right]\cdot a = \sum_{i=1}^{N} x_i y_i$ iz čega se dobija parametar a.

Rekurzivno rešenje – uvodimo oznake: $\hat{a}_k = p_k b_k$; $p_k = \left[\sum_{i=1}^{k} x_i^2\right]^{-1}$ i $b_k = \sum_{i=1}^{k} x_i y_i$. U rekurzivnoj formuli: $p_k = p_{k-1} - p_k p_{k-1} x_k^2$ i $b_k = b_{k-1} + b_k y_k$, definisanjem $k_k = p_{k-1} x_k \left[1 + p_{k-1} x_k^2\right]^{-1}$, tada se ocena parametra a može izraziti kao: $\hat{a}_k = \hat{a}_{k-1} - k_k(\hat{a}_{k-1} - y_k)$.

IV VALIDACIJA I VERIFIKACIJA

Validacija i verifikacija su postupci kojim ispitujemo koliko verno i precizno jedan model predstavlja realni sistem. One se konceptualno razlikuju ali se najčešće simultano sprovode, odnosno kaže se da su u dinamičkoj povratnoj sprezi. *Verifikacija* se odnosi na proveru da li je simulacioni program (računarski kod) bez grešaka i konzistentan sa modelom (koncepcijom). *Validacija* se odnosi na proveru da li je model precizna reprezentacija realnog sistema. To je iteraktivna procedura kojom se poredi ponašanje modela i realnog sistema sve dok se ne dobije tačnost modela koja zadovoljava.

VALIDACIJA SIMULACIONIH MODELA

Validacija je postupak kojim se određuje da li je model precizna reprezentacija realnog sistema. Ponašanje modela se poredi sa ponašanjem realnog sistema i uočene razlike se koriste za ispravku modela.

Pri stvaranju modela unose se mnoge aproksimacije realnog sistema (zadržavaju se samo osobine originala koje su bitne). Najčešće primenjivane aproksimacije su:

1. **Funkcionalna aproksimacija** – nelinearne funkcije se aproksimiraju linearnim funkcijama koja mora biti približna originalnoj u oblasti gde će sistem verovatno funkcionisati. Ako program mora funkcionisati u oblasti gde je poklapanje funkcija slabo, potrebno je da program štampa poruku upozorenja.
2. **Aproksimacija raspodele** – realne verovatnoće raspodela se aproksimiraju jednostavnijim raspodelama (normalna, eksponencijalna). Najekstremniji primer je kada se slučajna promenljiva zameni konstantom.
3. **Aproksimacija nezavisnosti** – model se pojednostavljuje tako što se pretpostavlja da su različite komponente (opisane slučajnim promenljivama) statistički nezavisne.
4. **Aproksimacija agregacije** – kada više elemenata posmatramo kao jednu clinu. Primeri agregacije su:
   a. ***Vremenska agregacija*** – interval vremena kao što je dan tretira se kao jedan pojedinačni period (svi događaji koji su se desili tokom dana, pretpostavlja se da su se desili istovremeno). Karakteristična je za sve diskretno – vremenske modele.
   b. ***Među – sektorska agregacija*** – više odeljenja, firmi, proizvodnih linija se posmatra kao jedna celina.
   c. ***Agregacija pomoćnih sredstava*** – kada više pomoćnih sredstava posmatramo kao jedno.
5. **Aproksimacija stacionarnosti** – činjenica da se parametri i druge karakteristike sistema ne menjaju u vremenu pojednostavljuje stvar. Za neke fizičke procese (astronomske pojave) može biti početna aproksimacija, ali za političke, ekonomske, organizacione i socijalne sisteme ona je neodrživa jer su te pojave nestacionarne prirode.

Cilj procesa validacija

- Da proizvede model koji predstavlja ponašanje realnog sistema i koji je dovoljno blizak realnom sistemu tako da se može koristiti za eksperimente.
- Da se pouzdanost modela poveća na prihvatljiv nivo tako da se model može koristiti za donošenje odluka.

Proces validacije se mora posmatrati kao integralni i nezamenljivi deo razvoja modela. Iteraktivno poređenje modela i realnog sistema se vrši pomoću različitih testova koji mogu biti subjektivni (učešće onih koji dobro poznaju sistem) i objektivni (koriste podatke o ponašanju sistema koji se obrađuju).

Praktični pristup procesu validacije se sastoji iz tri faze:

1. Izgraditi model koji verno predstavlja realni sistem.
   U izgradnji modela potrebno je da učestvuju i korisnici i ljudi koji poseduju znanja o realnom sistemu. Za proveru validnosti može se koristiti *analiza osetljivosti* (testiranje na različite ulazne veličine).
2. Potrvrditi pretpostavke modela.
   Dve kategorije: pretpostavke o ***strukturi*** (funkcionisanje sistema često obuhvata pojednostavljenja i apstrakcije) i pretpostavke o ***podacima*** (pouzdanost podataka i ispravna statistička analiza: 1.identifikacija raspodela podataka; 2.procena parametara raspodele; 3.validacija statističkog modela nekim testom: χ^2, Kolmogorov – Smirof test).
3. Poređenje (validacija) ulazno – izlaznih transformacija modela i realnog sistema.
   Najobjektivniji test modela je provera sposobnosti modela da predvidi buduće ponašanje realnog sistema.

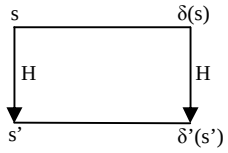
Formalni kriterijum za utvrđivanje validnosti modela

Realni sistem treba posmatrati kao sistem na višem nivou a model kao sistem na nižem nivou.

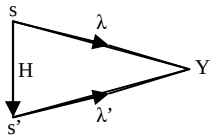
Homomorfizam (*gr. homo* – sličan, *morph* – struktura) je formalni kriterijum za utvrđivanje valindnosti modela za date eksperimentalne uslove. Radi jednostavnosti izlaganja ograničićemo se na dokazivanje validnosti diskretnih vremenskih modela.

Suština formalnog postupka za proveru validnosti je u tome da se nađe preslikavanje **H** pomoću kog je moguće iz svakog stanja osnovnog modela **s** preći u odgovarajuće stanje uprošćenog modela **s'**. Ako ovakvo preslikavanje postoji, tada se zaključuje da su ulazno – izlazna ponašanja osnovnog i uprošćenog modela ista za date eksperimentalne uslove. Da bi se utvrdio homomorfizam, moraju biti ispunjeni sledeći uslovi:

1. Očuvanje funkcije nastupanja vremena
   Stanja **s** i **s'** moraju imati istu funkciju nastupanja vremena, odnosno da se menjaju u istim vremenskim trenucima.
2. Očuvanje funkcije prelaza stanja
   Ako stanju **s** odgovara stanje **s'** primenom preslikavanja **H**, onda i sledeća stanja u koja će model preći u narednom trenutku posmatranja moraju odgovarati jedno drugom. Sledeće stanje osnovnog modela će biti **δ(s)** a uprošćenog modela **δ'(s')**, što znači da primenom preslikavanja **H** na stanje **δ(s)** treba da se dobije stanje **δ'(s')**. Grafički se može predstaviti preko komutativnog dijagrama.



3. Očuvanje izlazne funkcije
   Primena izlazne funkcije osnovnog modela **λ** (za one eksperimentalne uslove za koje je izvršeno uprošćevanje da bi se dobio uprošćeni model za stanje **s**) daje isti izlaz **Y** kao i primena izlazne funkcije uprošćenog modela **λ'** na stanje **s'**. Uočava se da se validnost uprošćenog modela utvrđuje i u odnosu na zadate eksperimentalne uslove. Ako je preslikavanje **H** tip 1:1, u pitanju je izomorfizam.



Ako su zadovoljeni svi prethodno definisani uslovi onda je uprošćeni model validan u odnosu na osnovni model za date eksperimentalne uslove.

VERIFIKACIJA SIMULACIONIH MODELA

Proces verifikacije je ispitivanje da li je i u kojoj meri, konceptualni model na odgovarajući način predstavljen računarskim kodom. U postupku verifikacije nema standardnog recepta. Zato je potrebno izvršiti više različitih provera:

- **Ručna verifikacija** logičke ispravnosti: model se izvesno vreme propušta na računaru ručno, a potom se porede dobijeni rezultati.
- **Modularno testiranje**: pojedinačno testiranje svakog modula kako bi se ustanovilo da li daje razumne izlaze za sve moguće ulaze.
- **Provera u odnosu na poznata rešenja**: podesimo model tako da predstavlja sistem čija su rešenja poznata i upoređujemo ih sa rezultatima modela.
- **Testiranje osetljivosti**: variramo jedan parametar, dok ostali ostaju nepromenjeni i proveravamo da li je ponašanje modela osetljivo na promenu tog parametra.
- **Testiranje na poremećaje**: postavimo parametre modela na neprirodne vrednosti i proveravamo da li se model ponaša na neshvatljiv način. Na taj način se mogu otkriti greške u programu koje se vrlo teško mogu uočiti na drugi način.

V SIMULACIJA DISKRETNIH DOGAĐAJA

Simulacija diskretnih događaja je metoda simulacionog modeliranja sistema kod kojih se diskretne promene stanja u sistemu ili njegovom okruženju događaju diskontinualno u vremenu. Koristi se uglavnom za analizu dinamičkih sistema sa stohastičkim karakteristikama.

Događaj je diskretna promena stanja entiteta sistema i nastupa u određenom trenutku vremena. Sistemi sa diskretnim događajima su: banke, auto servisi, pošte, samoposluge jer se stohastički karakter ogleda u slučajnoj prirodi veličina (događaja) za opis ovakvog sistema.

Prema Šanonu, opis diskretnih događaja je sledeći:

Entiteti, koji se opisuju **atributima** i uzajamno deluju u **aktivnostima**, pod određenim **uslovima** stvaraju **događaje** koji menjaju **stanje sistema**.

FORMALNI OPSI SISTEMA SA DISKRETNIM DOGAĐAJIMA

Formalni opis sistema diskretnih događaja može se predstaviti uređenom šestorkom M_d:

$M_d = \{U,S,Y,\delta,\lambda,\tau\}$, gde su:

U – skup eksternih događaja,

S – skup sekvencijalnih stanja diskretnih događaja,

Y – skup izlaza,

δ – kvazi – prenosna funkcija koja se zadaje sa funkcijama

δ^{Φ} koja preslikava S→S; pokazuje u koje će stanje preći sistem iz stanja **s**, ukoliko ne nastupi ni jedan eksterni događaj.

δ^{ex} koja preslikava UxSxT→S; pokazuje u koje će stanje preći sistem iz stanja **s**. kada nastupi eksterni događaj **u** u vremenskom trenutku **t**,

λ – izlazna funkcija koja preslikava S→Y

τ – funkcija nastupanja vremena koja preslikava $S \rightarrow R^{+}_{0,\infty}$; pokazuje koliko će dugo sistem ostati u stanju **s**, pre nego što nastupi naredna promena stanja, pod pretpostavkom da neće nastupiti ni jedan eksterni događaj.

DOGAĐAJ, AKTIVNOST I PROCES

Kod modela sa diskretnim događajima, pored koncepata koji opisuju strukturu kao što su objekti, relacije između njih i njihovi atributi, uvedeni su i koncepti za opis dinamike: događaj, aktivnost i proces.

Događaj predstavlja diskretnu promenu stanja entiteta u sistemu ili njegovom okruženju. Između dva uzastopna događaja stanje sistema se ne menja. Događaj može nastupiti zbog ulaska/izlaska privremenog entiteta u/iz sistema (dolazak/odlazak klijenta) – **eksterni**; ili zbog promene atributa pojedinih objekata sistema (opslužilac u sistemu postaje slobodan ili zauzet) - **interni**. ***Uslovni događaji*** mogu nastupiti kada je ispunjen uslov njihovog nastupanja. Obično su povezani sa zauzimanjem nekog resursa. ***Bezuslovni događaji*** su oni čiji je jedini uslov da tekuće vreme simulacije bude jednako vremenu njegovog nastupanja. Obično su povezani sa oslobađanjem nekog resursa odnosno planiranim završetkom neke aktivnosti.

Aktivnost je skup događaja koji menjaju stanje jednog ili više entiteta. Trajanje aktivnosti može biti unapred definisano (determinističko) ili da zavisi od ispunjenja uslova pa je vreme završetka takve aktivnosti nepoznato (stohastičko).

Proces je niz uzastopnih, logički povezanih događaja kroz koje prolazi neki privremeni objekat. To je hronološki uređena sekvenca događaja koja opisuje jednu pojavu od nastajanja do terminiranja. On može da obuhvati deo ili celokupan život privremenog entiteta.

Ključni elementi razvoja simulacije diskretnih događaja su: **1.**mehanizam pomaka vremena, **2.**pristup generisanju događaja i **3.**osnovne strategije simulacije.

Mehanizam pomaka vremena

U simulaciji diskretnih događaja koriste se dva osnovna mehanizma pomaka vremena: **1.**pomak vremena za konstantani priraštaj i **2.**pomak vremena na naredni događaj.

Pomak vremena za konstantni priraštaj – vreme u simulacionom modelu se menja tako da se uvek dodaje konstantan priraštaj **Δt**. Nakon svakog pomaka vremena, odnosno ažuriranja vrednosti simulacionog sata, ispituje se da li je u prethodnom intervalu vremena trebalo da dođe do nastupanja nekih događaja. Ukoliko jeste, tada se ti događaji planiraju za kraj intervala. *Nedostatak*: pomeranjem događaja na kraj vremenskog intervala uvodi se greška u simulaciji. Događaji koji nisu istovremeni u ovom se pristupu prikazuju kao istovremeni, a potom se određuje redosled njihovog izvođenja (koji se može razlikovati od stvarnog redosleda). Smanjenjem vremenskog prirasta te se greške smanjuju, ali se povećava vreme koje se troši na izvođenje simulacije kao i porast broja vremenskih intervala u kojima nema događaja.

Pomak vremena na naredni događaj – simulacioni sat se pomera na vreme u kom će nastupiti prvi naredni događaj (ili više njih). Simulacija se završava kada nema više događaja ili kada je zadovoljen neki unapred definisan uslov završetka simulacije. Na ovaj način se izbegava greška u vremenu izvođenja događaja a ujedno se preskaču intervali u kojima nema događaja. Ovaj princip je složeniji ali i efikasniji pa svi ključni simulacioni jezici koriste ovaj mehanizam.

Generisanje događaja

Događaj se opisuje sa više atributa, koji formiraju **slog** događaja. S obzirom na promenljiv broj događaja u vremenu, slogovi događaja se memorišu u **listama događaja**.

Slog događaja

Kod događaja	Vreme događaja	Par 1	Par 2	...	Par k

Lista događaja

Slog događaja 1
Slog događaja 2
Slog događaja 3
...
Slog događaja n

Postoje dva pristupa generisanja događaja:

1. **Definisanje događaja unapred** – svi događaji su unapred poznati i definisani, a lista događaja sadrži slogove svih događaja.

2. **Definisanje narednog događaja** – poznat je jedino prvi naredni događaj, a lista događaja sadrži samo jedan slog, slog poznatog događaja. Pri izvršavanju događaja, planira se i ubacuje u listu njegov naslednik.

Događaje možemo svrstati u dve kategorije u odnosu na mesto nastajanja (generisanja): **1.*Eksterni*** događaji su oni događaji koji ne zavise od modela i predstvljaju uticaj okoline na sistem (dolazak kupaca u samoposlugu, vozača u auto servis); **2.*Interni*** događaji zavise od modela i u njemu se generišu (dolazak kupaca u red na kasu).

Strategija izvođenja simulacije

Te strategije su: **1.**raspoređivanje događaja; **2.**skaniranje aktivnosti i **3.**interakcija procesa.

Raspoređivanje događaja podrazumeva da se događaji planiraju unapred i drže u listi budućih događaja, najčešće sortirani po vremenu nastupanja i prioritetu. *Procedura planiranja događaja*: generiše se slog događaja; dodele se vednosti njegovih atributa (vreme nastajanja, prioritet); događaj se stavlja u listu budućih događaja koja je uređena po vremenu nastupanja događaja i prioritetu. *Funkcionisanje simulatora*: sa liste budućih događaja uzima se prvi; ažurira se simulacioni sat na vreme njegovog nastupanja; na osnovu tipa izabranog događaja poziva se odgovarajuća procedura koja izvršava sva ažuriranja u modelu i simulatoru; kada se izvrše svi događaji koji imaju isto vreme nastupanja, simulacioni sat se ažurura na vreme sledećeg događaja iz liste budućih događaja.

Skaniranje aktivnosti podrazumeva da se događaji implicitno raspoređuju tako da se promena stanja izvršava preko funkcija koje se nazivaju aktivnosti. Svaka aktivnost ima ***uslov*** i ***akciju***. Za svaki vremenski korak **Δt**, aktivnosti se skaniraju i traži se prva aktivnost koja ima zadovoljen uslov. Tada se izvršava odgovarajući programski segment koji specificira akciju za zadatu aktivnost. Proces skaniranja traje sve dok sve aktivnosti ne budu blokirane. Tada i samo tada se simulacioni sat ažurira na sledeći vremenski korak. Skaniranje aktivnosti je bolje od raspoređivanja događaja kada je broj aktivnosti u modelu mali, a broj događaja u okviru aktivnosti veliki, jer se štedi na računarskim operacijama koje se odnose na listu i njegovo skidanje sa liste događaja.

AKTIVNOSTI	USLOVI	AKCIJE
A1	Uslovi 1	Akcije 1
A2	Uslovi 2	Akcije 2
A3	Uslovi 3	Akcije 3

Tabela aktivnosti

Interakcija procesa predstavlja tehniku simulacije koja je nastala kombinacijom raspoređivanja događaja i skaniranja aktivnosti. **Proces** možemo posmatrati kao skup isključivih aktivnosti, povezanih tako da terminiranje jedne aktivnosti dozvoljava inicijalizaciju neke druge aktivnosti iz skupa aktivnosti procesa. Glavni problem je sinhronizacija procesa jer je model sistema skup paralelnih procesa od kojih neki mogu biti uzajamno isključivi. Da do ovoga ne bi došlo uvode se dve naredbe WAIT i DELAY i to i u uslovnom i u bezuslovnom kontekstu.

WAIT naredba u uslovnom kontekstu ima oblik: WAIT UNTIL <uslov> – ako je uslov zadovoljen, proces koji čeka na taj uslov nastavlja svoj tok, a u suprotnom biće blokiran. Tada se ispituje koji od ostalih procesa može da se aktivira i on se izvršava.

DELAY naredbom se odlaže nastavak procesa za određeni broj vremenskih jedinica. U tom slučaju, ispituje se koji od ostalih procesa može da ostvari svoj tok. U bezuslovnom kontekstu naredba DELAY označava se i sa ADVANCE <broj vremenskih jedinica>.

U okviru ***procesora*** održavaju se dve liste događja:

1. **Lista tekućih događaja** (LTD) koja sadrži uslovne događaje i sortirana je po prioritetu.
2. **Lista budućih događaja** (LBD) koja sadrži raspoređene događaje koji tek treba da nastupe i sortirana je po vremenu nastupanja.

Pri svakom premeštanju događaja iz LBD u LTD planira se njegov naslednik koji se stavlja u LBD gde čeka svoje vreme nastupanja. Pre početka simulacije, za svaki proces generiše se prvi događaj i stavlja se u LBD a zatim se vreme simulacije ažurira na vreme događaja sa vrha LBD.

Rad procesora odvija se u dve faze:

1. ***Faza ažuriranja vremena simulacije***, simulacioni sat se postavlja na vrednost vremena nastupanja prvog događaja iz LBD, a zatim se prebace svi događaji sa tim vremenom nastupanja u LTD. Pri tome se za svaki premešten događaj planira naslednik i stvalja se u LBD.
2. ***Faza skaniranja liste tekućih događaja***, skanira se LTD i ukoliko ima neblokiranih događaja oni se izvršavaju. Kada nema događaja u LTD ili su svi blokirani (WAIT naredbom), tada se prenosi prvi događaj iz LBD i ažurira se simulacioni sat na njegovo vreme.

Simulacija se odvija sve dok se ne zadovolji kriterijum za završetak simulacije, kada se vrši završna obrada rezultata simulacije i procesor završava rad.

VI GPSS JEZIK

GPSS (General Purpose Simulation System) predstavlja interpreterski jezik za simulaciju diskretnih – stohastičkih sistema. Struktura modela se definiše a simulacija izvršava pomoću naredbi ugrađenog jezika. Nakon simulacije na raspolaganju su statistički pokazatelji o ponašanju modela u toku simulacije. GPSS je jezik orijentisan na procese, model se predstavlja dijagramom toka koji se konstruiše uz pomoć blokova. Svakom bloku odgovara jedna linija. **Blokovi** su statički objekti.

Entitete (objekte modela) možemo podeliti u četiri klase:

1. **Dinamički entiteti**

   Transakcije su dinamički objekti GPSS jezika koje se kreću kroz blokove. One imaju niz karakteristika koje nazivamo ***parametrima***. One su aktivni objekti i može postojati više transtakcija u svakom trenutku na različitim mestima u blok dijagramu. Predstavljaju saobraćajne jedinice u modelu (klijenti na opsluživanju, telefonski pozivi, vozila na servisu). Transakcije mogu da rezervišu određene resurse u modelu.
2. **Statički entiteti**

   To su elementi opreme, pasivni su i predstavljaju resurse na koje se deluje transakcijama. Tu spadaju: ***uređaji*** (mogu da usluže samo jednu transakciju u nekom vremenskom trenutku), ***skladišta*** (mogu da usluže više transakcija istovremeno u zavisnosti od kapaciteta) i ***logički prekidači*** (stanje prekidača se postavlja prolaskom transakcije kroz njega; mogu biti u tri stanja uključeno, isključeno i invertovano; služe za testitanje stanja opreme i zavisno od toga transakcija se preusmerava).
3. **Statistički entiteti**

   U ovu grupu spadaju ***redovi*** i ***tabele***. Ako je uređaj zauzet ili skladište puno tada transakcija formira red. Na osnovu statistike GPSS automatski formira tabele sa vrednostima. Korisnik može i sam da formira tabele sa statističkim vrednostima koje su mu potrebne. Ona sadrži frekvencijske klase sa numeričkim vrednostima nekog atributa.
4. **Entiteti operacija**

   Oni se nazivaju ***blokovima*** i pružaju logiku sistema, dajući instrukcije transakcijama gde treba da idu i šta dalje da rade. Blok dijagram predstavlja operacije koje se vrše unutar datog sistema. Blok pokazuje tip operacije koju obavlja a linije između blokova ukazuju na tok saobraćaja. Unutar blok dijagrama mogu da nastanu četiri osnovna tipa događaja:
   1. stvaranje ili uništavanje transakcija;
   2. promena numeričkih atributa entiteta;
   3. kašnjenje transakcije za određeno vreme;
   4. promena toka transakcije kroz blok dijagram.

Osnovni koncept GPSS jezika

Transakcije generiše blok GENERATE. Ona se kreće kroz model sve dok ne naiđe na blok koji nema uslova da je primi ili ne naiđe na blok TERMINATE koji uklanja transakciju iz modela. Takođe, postoje blokovi koji zadržavaju transakciju za određeni period simulacionog vremena. Ako neki blok nema uslova da primi transakciju, onda ona čeka da se ispuni uslov daljeg kretanja kroz model. Prolaz transakcija kroz određene blokove izaziva promene koje utiču na stanje modela i njegovo okruženje.

Transakcije u GPSS-u se čuvaju u listi tekućih događaja (LTD) i listi budućih događaja (LBD). Transakcije iz LTD nastoje da se kreću kroz blok dijagram. Transakcije iz LBD nisu spremne za kretanje, već kasnije, kada se ažurira simulacioni sat (ispunjeni uslovi kretanja), ove transakcije se prebacuju u LTD, kako bi nastavile kretanje kroz model.

U svakoj tački simulacionih vremena (simulacioni sat zadat numeričkim atributom C1 uzima samo celobrojne vrednosti), GPSS skanira sve transakcije u LTD i svaka koja ima uslov daljeg kretanja to i čini. Kretanje transakcija se prekida zbog jednog od tri razloga:

1. transakcija je uništena,
2. blok odbija da primi transakciju,
3. transakcija je ušla u blok koji je zadržava izvestan period simulacionog vremena.

Ukoliko je pitanju treći razlog, transakcija se prebacuje iz LTD u LBD. Ukoliko je kretanje transakcije izazvalo promenu stanja modela, GPSS ponovo obavlja skaniranje, ali od početka LTD; u suprotnom naredna transakcija iz LTD se skanira.

Kada nastupi trenutak da nijedna transakcija iz LTD ne može da se kreće, GPSS ispituje LBD. LBD je uređena tako da se na njenom početku nalaze transakcije koje uslov za kretanje kroz model stiču ranije, dok su

na kraju one transakcije koje moraju duže da čekaju. Kada nijedna transakcija iz LTD ne može da nastavi kretanje, GPSS ažurira simulacioni sat na vreme prve transakcije iz LBD. Sve transakcije iz te liste, koje su sada stekle uslov za kretanje, kopiraju se iz LBD u LTD i skaniranje LTD se ponavlja.

Svaka transakcija ima svoj prioritet, koji se može menjati sa protokom simulacionog vremena i trenutnim položajem transakcije u blok dijagramu. LTD je sortirana po opadajućem redosledu prioriteta, tako da se transakcije sa većim prioritetom kreću pre onih čiji je prioritet manji u svakom trenutku simulacionog vremena.

Vrste naredni u GPSS-u

1. ***Deklaracione naredbe entiteta*** definišu atribute pojedinih permanentnih entiteta u programu. Tu spadaju naredbe za deklaraciju uređaja, skladišta, tabela (histograma), funkcija i dr. Mogu se pisati bilo gde ali je uobičajeno da se pišu na početku programa. U polju LOKACIJE specificira se posebno definisano skladište, tabela i sl. a u polju VARIJABLE mogu da budu upisani različiti argumenti.
2. ***Blok naredbe*** služe za specifikaciju modela sistema. One se izvršavaju kada transakcija prilikom kretanja kroz model naiđe na blok. Model se sastoji od niza blok naredbi povezanih u obliku blok dijagrama. Svaka blok naredba može da ima identifikacioni broj, koji se piše u polju lokacije. Ako korisnik želi da se pozove na neki blok on može da u polje LOKACIJE unese određeni mnemonički simbol. U polju OPERACIJE piše se tip blok naredbe, a u polju VARIJABLE pišu se argumenti.
3. ***Kontrolne naredbe*** služe za kontrolu izvršenja simulacije, a mogu imati uticaj na statistiku o ponašanju entiteta u toku simulacije. Simulator zahteva neke informacije upravljanja kao što su dužina simulacije, kada je kraj naredbi programa i dr. U polju OPERACIJA piše se tip naredbe, u polje VARIJABLE unose se argumenti specificirane naredbe a polje LOKACIJE se ne koristi. To su naredbe SIMULATE, START, RESET, CLEAR, END.

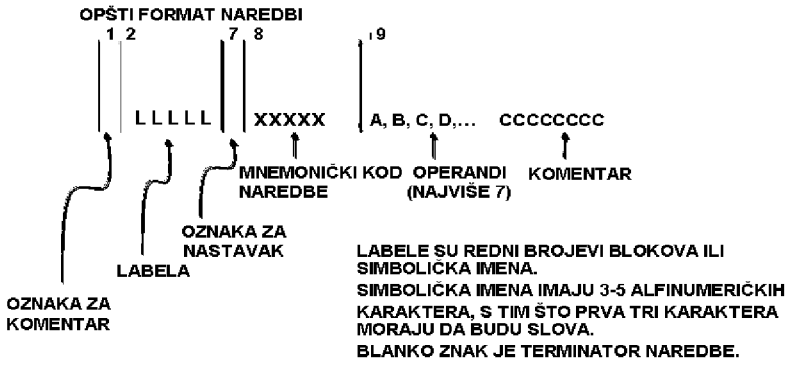
Trajanje simulacije može biti ograničeno brojem transakcija koje su prošle kroz model, što se kontroliše pogodnom upotrebom naredbi START i TERMINATE ili vremenski (fiksno), što je određeno tzv. tajmerom koji čini par naredbi GENERATE i TERMINATE.

TRANSAKCIJE I BLOKOVI ZA RAD SA TRANSAKCIJAMA

Transakcije su dinamički entiteti koje se kroz model kreću. Transakcije su aktivni objekti (pr: klijenti, pozivi, avioni, vozila). Transakcije mogu da se: stvaraju, umnožavaju (copy), spajaju (ujedinjuju), uništavaju (udaljuju iz modela). Transakcija nosi određeni broj parametara koji definišu njena svojstva. Obično ih je 12. Transakcija može imati nivo prioriteta, koji se kreće od 0 do 127. Prvo se opslužuju transakcije sa višim nivoom prioriteta. Naredbe kojima se vrši obrada transakcija mogu biti:

- naredbe za kašnjenje transakcije za neko vreme,
- naredbe za stvaranje ili uništavanje transakcije,
- naredbe za promenu vrednosti parametara transakcije,
- naredbe za stvaranje kopija transakcije,
- blokovi za sinhronizaciju kretanja transakcija.

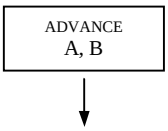
NAREDBE GPSS JEZIKA



!!!NAPOMENA: U opisu naredbi GPSS jezika operadni navedeni u uglastim zagradama, [], mogu se izostaviti, a oni koji nisu označeni uglastim zagradama moraju da se specificiraju!!!

1.Naredba za vremensko zadržavanje transakcija

Blok **ADVANCE A[,B]**



Predstavlja blok za kašnjenje transakcije za neki vremenski period. To je blok opsluživanja. Kretanje transakcije kroz model se prekida za specificiran broj vremenskih jedinica i to je vreme čekanja na mestu opsluživanja.

Kada transakcija naiđe na blok ADVANCE, ona se prebacuje iz LTD u LBD. Tek nakon što se simulacioni sat ažurira za specificirani broj vremenskih jedinica (specificiran u naredbi ADVANCE), transakcija se vraća u LTD i nastavlja se njeno napredovanje kroz model.

A – srednje vreme zadržavanja transakcije

B – modifikator srednje vrednosti zadržavanja transakcije

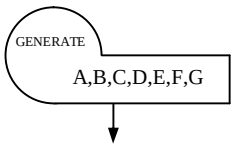
Kada se **B** izostavi, tada je zadržavanje tranakcije za vreme **A**=const. Modifikator **B** može da bude:

- u obliku **poluintervala**. Vreme zadržavanja je po uniformnoj raspodeli u nekom poluopsegu.
  PR: ADVANCE 10,5 znači da je zadržavanje uniformno na intervalu od 5 do 15 vremenskih jedinica. Brojevi se biraju slučajno sa jednakim verovatnoćama i dodeljuju srednjem vremenu zadržavanja.
- u obliku **funkcije**. Vremensko zadržavanje ima složenu raspodelu verovatnoća i računa se srednja vrednost u **A** pomnožena sa vrednošću funkcije u **B**. Vrednost funkcije u polju **B** se bira na osnovu definisane funkcije i na osnovu generatora slučajnih brojeva.
  PR: ADVANCE 100,FN5 zadržavanje je jednako 100·FN5, gde je FN5 funkcija čiji je redni broj 5.

Ukoliko je vrednost modifikatora **B** veća od srednje vrednosti zadržavanja transakcije **A**, GPSS procesor prijavljuje grešku u toku izvršenja simulacije

2.Naredbe za stvaranje/uništavanje transakcija

Blok **GENERATE A[,B][,C][,D][,E][,F][,G]**



Ovaj blok generiše transakcije. Svaki model mora da ima bar jedan blok GENERATE, a ukoliko je potrebno i više, pri čemu svaki radi nezavisno u odnosu na druge. To je jedini blok u koji transakcije ne ulaze već samo iz njega izlaze. Ne može se stvoriti transakcija dok prethodna ne izađe iz bloka GENERATE. Zato se iza ovog bloka ne sme staviti blok koji odbija transakcije već onaj koji može da ih primi ali ih ne zadržava (npr. ADVANCE 0). Stvaranje transakcija se obavlja u **regularnim** intervalima ili u **stohatičkim** intervalima (po nekoj raspodeli).

A – srednje vreme između generisanja transakcija.

B – modifikaror vrednosti iz polja **A**. On može biti:

- u obliku **poluintervala**. Tada je vreme između dolazaka ravnomerno raspoređeno u nekom opsegu.
- u obliku **funkcije**. Tada vreme između dolazaka ima složenu raspodelu verovatnoća i računa se kao proizvod vrednosti iz polja **A** i vrednosti definisane funkcije date u polju **B**.

C – Offset interval. Vreme generisanja prve transakcije dok se sve ostale generišu prema vrednostima iz **A** i **B**.

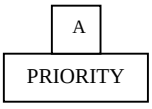
D – Ukupan broj transakcija koje stvara blok GENERATE. Ako se izostavi vrednost u ovom polju, transakcije se stvaraju do kraja simulacije.

E – Zadavanje nivoa prioriteta transakcije (od 0 do 127)**.**

F – Definisanje broja parametara koje ima transakcija. Ukoliko se vrednosti u ovom polju izostavi, transakcija će imati 12 parametara.

G – Definisanje količine memorijskog prostora koji služi za memorisanje vrednosti parametara transakcije. U ovom polju može se napisati:
H – za definisanje 16 bita memorijskog prostora ($2^{15} – 1 = 32367$)
F – za definisanje 32 bita memorijskog prostora ($2^{32} – 1 = 2147468$)

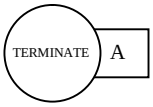
Blok **PRIORITY A**



U bloku GENERATE polje **E** daje nivo prioriteta transakcije. Za promenu vrednosti prioriteta transakcije koristi se blok PRIORITY.

A – nivo prioriteta (od 0 do 127). 8 bitova se koristi za pamćenje prioriteta.

Blok **TERMINATE [A]**



Kada transakcija izvrši predviđeni tok kroz model, ona postaje pasivna (neaktivna) i zauzima neki memorijski prostor koji služi za opis njenih atributa. Zbog toga je treba ukloniti iz modela (uništiti). Blok TERMINATE uklanja transakciju iz modela. On se koristi za predstavljanje završetka puta transakcije u modelu. Broj blokova TERMINATE je proizvoljan.

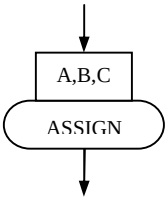
A – vrednost za koju se umanjuje terminacioni brojač (TB) kada transakcija dođe na ovaj blok. Ako se vrednost u ovom polju izostavi, transakcija se udaljuje iz modela a TB se ne modifikuje. Barem jedan blok TERMINATE mora imati definisano polje **A** ($A \geq 1$), da bi se simulacija završila.

Terminacioni brojač (TB) je jedinstvena lokacija na nivou GPSS programa, koja se umanjuje kad god neka transakcija dođe na neki od blokova tipa TERMINATE koji u polju **A** imaju neku vrednost. Kada vrednost TB-a padne na nulu, simulacija se automatski završava. Početna vrednost TB-a se definiše pomoću kontrolne naredbe START.

3.Naredbe za promenu vrednosti parametara transakcije

U momentu generisanja transakcije, svaki parametar ima vrednost 0. Kasnije im se dodeljuju vrednosti pomoću bloka ASSIGN.

Blok **ASSIGN A[±],B[,C]**



Ovim blokom se dodeljuje ili modifikuje vrednost parametra transakcije.

A – broj parametra transakcije čija se vrednost menja.

B – definiše se celobrojna vrednost koja se koristi za promenu vrednosti parametra broj **A**.

C – definiše se broj funkcije koja se koristi kao modifikator celobrojne vrednosti iz polja **B**. Vrednost funkcije se množi sa vrednošću iz polja **B** i rezultat se unosi u parametar **A**.

Simboli koji slede iza polja **A** odnose se na vrstu dodele i to:

A+ - tekućoj vrednosti parametra broj **A** ***dodaj*** vrednost iz polja **B**;

A– - tekućoj vrednosti parametra broj **A** ***oduzmi*** vrednost iz polja **B**;

A - parametru broj **A** ***dodeli*** vrednost iz polja **B**.

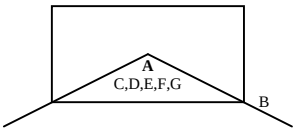
Blok **INDEX A,B**

Ovaj blok se koristi samo za rad sa parametrom p1. U polju **A** se definiše broj parametra kome se dodaje vrednost iz polja **B** i rezultat dodeljuje parametru p1. PR: `INDEX 5,30` – p5+30→p1. Blok INDEX se može realizovati sa dva bloka ASSIGN:

```
ASSIGN 1,30          ili          ASSIGN 1,P5
ASSIGN 1+,P5                      ASSIGN 1+,30
```

4.Naredbe za kopiranje i sinhronizaciju kretanja transakcija

Blok **SPLIT A,B[,C][,D][,E][,F][,G]**



Ovaj blok služi za stvaranje kopija transakcije odnosno za raspoređivanje poslova.

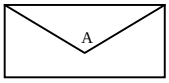
A – polje koje označava broj kopija koje stvara transakcija roditelj;

B – definiše blok u koji odlaze sve kopije, a roditelj odlazi u naredni blok.

Ako se u polju **A** zada broj **i**, onda iz bloka SPLIT izlazi **i+1** transakcija. Transakcije kopije i roditelj su ravnopravne i mogu dalje da ulaze u blokove SPLIT i stvaraju nove potomke. Sve kopije i roditelji kao i njihove kopije, pripadaju jednom ***ansamblu (familiji)***. Broj transakcija u ansamblu je proizvoljan a transakcija je član jednog i samo jednog ansambla. Kad se transakcija uništi, program je izbacuje iz ansambla ali ansambl i dalje postoji sve dok se ne uništi i poslednja transakcija. Transakcije koje pripadaju jednomansamblu mogu se dalje ***sinhronizovati*** *(objediniti)* pomoću blokova: ASSEMBLE, GATHER, MATCH. Ukoliko njih nema, onda se transakcije kreću nezavisno kroz model.

C – služi da sve potomke jednog roditelja svrstamo u niz (seriju). U ovom polju se upisuje broj parametra transakcije po kom će biti izvršena numeracija. **D**, **E**, **F**, **G** – koriste se ako želimo da kopije budu različite od roditelja po (**D** – broju parametara; **E** – prioritetu; **F** – tipu parametra; **G** – veličini mašinske reči).

Blok **ASSEMBLE A**

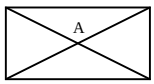


Ovaj blok ima primenu kod paralelnih procesa. Koristi se da objedini određeni broj transakcija koje se kreću po istoj putanji.

A – definiše broj transakcija koje bi bile objedinjene.

Transakcija koja uđe u ovaj blok, biće zadržana u njemu sve dok ne uđe i poslednja transakcija, do popune definisane u polju **A**. Kad je ušla poslednja transakcija u blok ASSEMBLE, tada iz njega izlazi jedna transakcija, i to ona što je prva ušla u ovaj blok, a ostale se uništavaju.

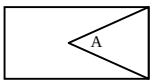
Blok **GATHER A**



Ovaj blok ima analogno dejstvo kao blok ASSEMBLE, s tim što se sve transakcije dalje propuštaju, i ni jedna se ne uništava. Ovde se takođe vrši objedinjavanje transakcija koje su članovi jednog ansambla odnosno kreću se po istoj putanji.

A – definiše broj transakcija iz istog ansambla koje moraju stići u blok GATHER i tek onda im se dozvoljava dalje kretanje kroz model.

Blok **ime MATCH A**



Prethodna dva bloka objedinjuju transakcije koje se kreću po istoj putanji. Blok MATCH služi za sinhronizaciju kretanja samo dve transakcije koje se mogu kretati i po različitim i po istim putanjama.

Za sinhronizaciju kretanja transakcija koje su na različitim putanjama potrebna su dva bloka MATCH. Za njih se kaže da su konjugovani, tj. svaki od njih treba u polju **A** da nosi ime sebi kojugovanog bloka. Kad ove transakcije uđu u blokove MATCH vrši se provera da li postoji transakcija istog ansambla u drugom bloku koji je njemu konjugovan. Kada u oba bloka postoji transakcije istog ansambla tad se one u isto vreme propuštaju kroz blokove MATCH. Ako u bloku MATCH nema ni jedne transakcije datog ansambla, onda transakcija koja je pristigla čeka dok transakcija istog ansambla ne pristigne u konjugovani blok i one odlaze u blok iza bloka MATCH.

Ako su transakcije na istoj putanji, onda je potreban jedan blok MATCH. On je konjugovan sam sebi tako da i ime i polje **A** imaju isti naziv. Transakcije se kreću po jednoj putanji i objedinjuju u bloku MATCH.

NPR: Za kretanje po različitim putanjama:

```
ime1 MATCH ime2
ime2 MATCH ime1
```

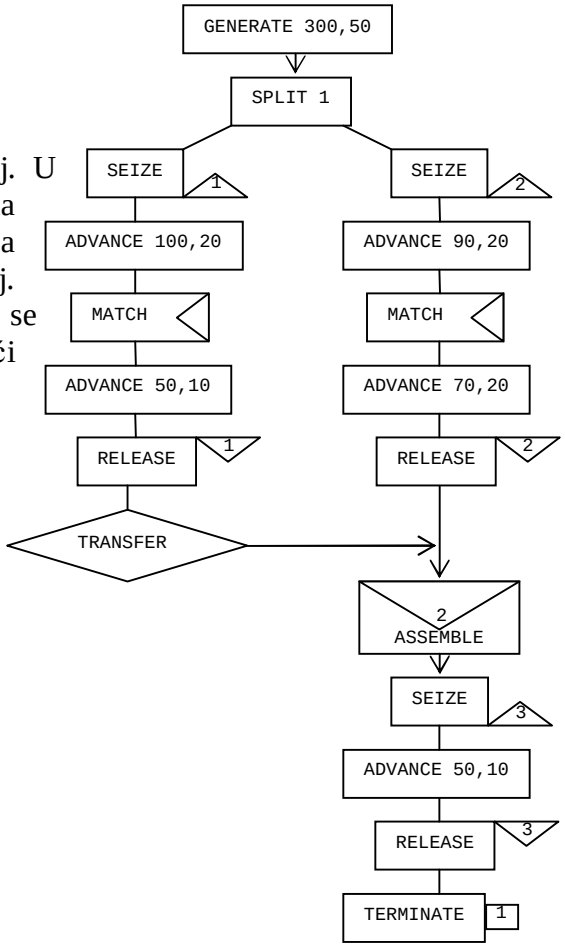
Za kretanje po istoj putanji:

```
ime  MATCH ime
```

***PRIMER:Posao obavljaju 3 radnika. Posao stiže 300±50 v.j. U Prvoj fazi posao zahteva dve operacije, povezane, i potrebno je da prvi i drugi radnik sinhronizuju svoj deo posla: Rad1 je 100±20v.j a Rad2 je 90±30 v.j. U drugoj fazi rade nezavisno: Rad1 je 50±10 v.j. a Rad2 je 70±20 v.j. Kad se završi prva i druga faza tada se objedinjuje posao i radi treći radnik: Rad3 je 50±20 v.j. Naći iskorišćenje radnika, pri 500 poslova.

```
      SIMULATE                    SEIZE RAD3
      GENERATE 300,500            ADVANCE 50,20
      SPLIT 1,DRUGI               RELEASE RAD 3
      SEIZE RAD1            TERMINATE 1
      ADVANCE 100,20        START 500
BBB   MATCH AAA             END
      ADVANCE 50,10
      RELEASE RAD1
      TRANSFER ,CCC
DGUGI SEIZE RAD2
      ADVANCE 90,20
AAA   MATCH BBB
      ADVANCE 70,20
CCC   ASSEMBLE 2
```

rad3 čeka rad1 i 2 da završe



STARTOVANJE SIMULACIJE

Kontrolna naredba **START A[,B]**

Nakon definisanja modela. da bi se izvršila simulacija, potrebno je startovati simulator. Startovanje simulatora vrši se naredbom START.

A – početna vrednost terminacionog brojača

B – koristi se za ukidanje štampanja rezultata po završetku simulacije. Da bi se to postiglo, potrebno je u polje **B** uneti ozbaku **NP** (No Print).

Naredbom START započinje se proces simulacije (izvršenje GPSS programa). Ona se nalazi na kraju paketa blok naredbi i deklaracionih naredbi.

Definisanje početka i kraja GPSS programa

Kontrolne naredbe **SIMULATE** i **END**

Kontrolna nareba SIMULATE je prva naredba GPSS programa. Ako se ona izostavi, program će se prevesti, ali se neće izvršiti (naredba START tada nema efekta).

Kontrolna naredba END je poslednja naredba GPSS programa. Ona označava kraj paketa za GPSS prevodilac i ujedno je izvršna naredba posle koje se upravljanje predaje operativnom sistemu.

PERMANENTNI ENTITETI

Permanentni entiteti (objekti) postoje u GPSS simulatoru od momenta uvođenja do završetka simulacije. Svaki od permanentnih entiteta ima svoje atribute koji se ažuriraju uvek kada transakcija pokuša (uspešno ili neuspešno) da pristupi entitetu. Atributi permanentnih entiteta štampaju se automatski po završetku simulacije, dok su u toku simulacije na raspolaganju u obliku standardnih numeričkih atributa (SNA).

Osnovni permanentni entiteti su uređaji, skladišta, logički prekidači, redovi, tabele (histogrami), blok MARK. Definišu se na dva načina:

1. Deklaracionim naredbama (STORAGE, TABLE...) i tada oni postoje od samog početka simulacije.
2. Naredbama koje označavaju pristup entitetu (ENTER, QUEUE, LINK...) i tada se entitet uvodi u model tek kada transakcija prvi put naiđe na taj blok.

UREĐAJI

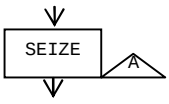
Uređaj (FACILITY) je permanentni entitet koji predstavlja samo jednog opslužioca. Uređaj može istovremeno da prihvati samo jednu transakciju. Ako transakcija pokuša da pređe u zaposednut uređaj, ona se automatski zadržava u prethodnom bloku, sve dok se uređaj ne oslobodi.

Koncept uređaja se koristi kod sistema opsluge gde postoji jedan opslužilac koji može da opslužuje jednog klijenta (jedan šalter u pošti, jedna telefonska linija).

Posebno važan koncept kod uređaja je koncept **prijemptiranja**. Koristi se kod klijenata sa različitim prioritetom. Ako je uređaj zauzet transakcijom prioriteta ***n*** a pokuša da ga zauzme transakcija prioriteta $n_1 > n$, tada se transakcija sa manjim prioritetom uklanja iz uređaja i stavlja na **stek prijemtiranja**. Uređaj zaposeda transakcija sa većim prioritetom a kad ona napusti uređaj, skida se transakcija sa vrha steka prijemtiranja i nastavlja obradu od tačke gde je bila prijemtirana. Moguće je prijemtiranje u više nivoa, zavisno od prioriteta transakcije koje dolaze na uređaj.

Zaposedanje i oslobađanje uređaja vrše naredbe SEIZE i RELEASE.

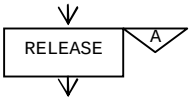
Zaposedanje uređaja: blok **SEIZE A**



Kad transakcija uđe u ovaj blok, znači da je uređaj zauzet, i ostaće sve dok transakcija ne uđe u blok RELEASE.

A – broj ili ime uređaja koji se zauzima.

Oslobađanje uređaja: blok **RELEASE A**



Kada transakcija uđe u ovaj blok uređaj se oslobađa.

A – broj ili ime uređaja koji se oslobađa.

SKLADIŠTA

Skladište (STORAGE) je permanentni entitet koji predstavlja grupu simultanih opslužioca. Ono može istovremeno da prihvati više transakcija. Ako transakcija pokuša da uđe u puno skladište, ona se automatski zadržava u prethodnom bloku, sve dok neka od transakcija koje su u skladištu ne napusti skladište i time oslobodi dovoljno jedinica skladišta.

Deklaracija skladišta: naredba **A STORAGE K**

A – ime skladišta

K – maksimalni (početni) kapacitet skladišta

Za ulazak i izlazak iz skladišta koriste se naredbe ENTER i LEAVE

Ulazak u skladište: naredba **ENTER A[,B]**

A – ime ili broj skladišta

B – broj jedinica koje se zauzimaju kad transakcija uđe u skladište. Ako se vrednost ovog polja izostavi, podrazumeva se da je **B**=1.

Oslobađanja skladišta: naredba **LEAVE A[,B]**

Ovom naredbom se oslobađa neki broj mesta kada transakcija izađe iz njega.

A – ime ili broj skadišta.

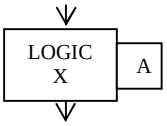
B – Broj mesta koja se oslobode kada transakcija napusti skladište.

Polja **A** i **B** moraju imati iste vrednosti (imena) i kod ENTER i kod LEAVE.

LOGIČKI PREKIDAČI

Logički prekidač (LOGIC SWITCH) je permanentni entitet koji ima dva stanja: uključen i isključen. Predstavlja ekvivalent za logičku promenljivu koja takođe može da ima dva stanja: tačno i netačno. U simulacionim modelima koristi se za označavanje jednog od dva stanja nekog entiteta u modelu (npr: telefonska linija može biti slobodna ili zauzeta, tj. svakom stanju linije se pridružuje jedno stanje logičkog prekidača).

Operacije nad logičkim prekidačima izvršavaju se kada transakcija naiđe na blok LOGIC i to su: ***uključivanje***, ***isključivanje*** i ***invertovanje*** (prebacivanje prekidača u suprotno stanje od onog u kom se nalazio).



Blok **LOGIC_{X}_A**

A – broj ili ime prekidača

X – postfiks naredba:

X = R – isključi prekidač (Reset)

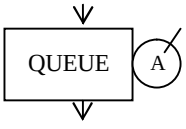
X = S – uključi prekidač (Set)

X = I – invertuj prekidač (Invert)

REDOVI

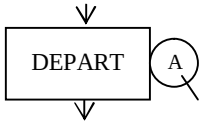
Blok QUEUE (red) je permanentni entitet koji se uvodi u model radi prikupljanja statistike o čekanju transakcije na nekom od blokova. Redovi postoje nezavisno od uvođenja ovih permanentnih entiteta. Red kao entitet ne utiče na ponašanje modela već služi samo za prikupljanje statistike o modelu. U modelu može da se uvede veći broj redova. Jedna transakcija može da bude član većeg broja redova (najviše 5).

Prijava transakcije u red: blok **QUEUE A**



A – broj ili ime reda.

Blok QUEUE bezuslovno prihvata transakciju. Ukoliko naredni blok ima uslov prijema, odmah je propušta. Ako naredni blok nema uslov prijema, transakcija ostaje u bloku QUEUE sve dok se ne ostvari uslov prijema. Ako u bloku QUEUE čeka više transakcija, odlazi prvo ona koja je najduže čekala.



Odjava transakcije iz reda: blok **DEPART A**

A – broj ili ime reda.

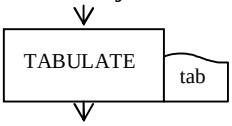
TABELE (histogrami)

Blok TABLE je permanentni entitet koji se uvodi ukoliko se želi snimanje histograma za neku veličinu (standardni numerički atribut – SNA). Ubacivanje podataka (ažuriranje tabele) u histogram vrši se dolaskom transakcije na blok TABULATE. On ne zadržava transakciju i ni na koji način ne utiče na model.

Deklaracija (definisanje) tabele: naredba **tab TABLE A,B,C,D**

tab – ime ili broj tabele

A – Definiše veličinu čiji se histogram snima. Veličina se definiše pomoću SNA.
B – Gornja granica prvog intervala histograma.
C – širina intervala
D – broj frekvencijskih intervala (klasa)
Ažuriranje tabele: blok **TABULATE tab**



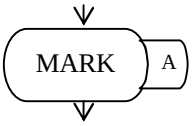
tab – broj ili ime tabele.

Blok **MARK [A]**
CLOCK TIME – kada program počne da radi vrši se otkucavanje vremenskih jedinica, jer u sebi ima časovnik. To je vreme **apsolutnog sata** (CLOCK TIME). Ovo vreme važi za sve transakcije istovremeno od trenutka ulaska transakcije u model.
MARK TIME – služi da odredimo **tranzitno vreme** za svaku transakciju.
TRANSIT TIME (tranzitno vreme) je vreme putovanja transakcije u pojedinim segmentima modela

TRANSIT TIME = CLOCK TIME – MARK TIME

MARK TIME se može računati za svaku transakciju. Program markira na početku svaku transakciju sa apsolutnim vremenom sa kojim ona ulazi u model. U tom momentu je MARK TIME = CLOCK TIME (apsolutnom vremenu) dok je TRANSIT TIME = CLOCK TIME – MARK TIME = Ø.

Ako nas interesuje vreme putovanja transakcije kroz određeni segment modela, onda je potrebno vratiti tranzitno vreme transakcije na nulu. U tu svrhu se koristi blok MARK [A].



A – operand kojim se definiše broj parametra transakcije u koji se upisuje tekuće vreme apsolutnog sata

Kad god transakcija uđe u blok MARK, MARK TIME te transakcije se izjednačuje sa apsolutnim vremenom (MARK TIME = CLOCK TIME). Operand **A** nije obavezan i u zavisnosti od toga ovaj blok radi na dva načina:

1. Ako se operand **A** izostavi ili je jednak nuli, tada se tranzitno vreme transakcije anulira, tj. vrši se resetovanje vremena putovanja transakcije, jer se MARK TIME izjednačava sa CLOCK TIME (TRANSIT TIME = CLOCK TIME – MARK TIME = 0).
2. Ako se koristi operand **A**, tada on definiše broj parametra transakcije kome će se pripisati vreme apsolutnog sata. Na taj način možemo da merimo tranzitno vreme kroz određeni segment modela. Pri tome MARK TIME transakcije se ne menja.

Ako želimo da tabeliramo vreme opsluživanja klijenata na šalteru, onda koristimo blok MARK na odgovarajućim mestima, a to se vreme može izračunati pomoću naredbe VARIABLE.
Ukoliko nas interesuje vreme putovanja transakcije od bloka 1 do bloka 2, tada iza bloka 1 resetujemo vreme putovanja transakcije postavljanjem bloka MARK.

MEMORIJSKE LOKACIJE

Memorijske lokacije su SNA koji permanentno egzistiraju tokom izvršenja GPSS programa i koji su pod direktnom kontrolom programera. To su ćelije u memoriji računara u kojima se pamti neka vrednost. Memorijske lokacije se imenuju sa $\mathbf{X_n}$ (na dužini jedne reči, 32 bita) ili $\mathbf{XH_n}$ (na dužini jedne polureči, 16 bita). Dopušta se i simboličko imenovanje kao naprimer: X$VISINA, XH$TIMER.

Kontrolna naredba: **INITIAL** $\left\{ \begin{matrix} X_n \\ XH_n \end{matrix} \right\}$**,K**

Ovaj blok dodeljuje početne vrednosti memorijskoj lokaciji i ona se aktivira odmah pri startovanju programa.
$\mathbf{X_n}$ **–** imenovanje memorijske lokacije na dužini od jedne reči (32 bita)**.**
$\mathbf{XH_n}$ – imenovanje memorijske lokacije na dužini od jedne polureči (16 bita).
K – vrednost koja se dodeljuje memorijskoj lokaciji.

Blok naredba: **SAVEVALUE A[±],B[,C]**

A – broj ili ime memorijske lokacije iza koga može da stoji + ili –, čime se definiše način rada. Ako nema ni + ni – tada se vrši zamena vrednosti memorijske lokacije iz polja **A** vrednošću definisanoj u polju **B**.

Ako stoji + tada se dodaje a za – oduzima vrednost iz polja **B** od vrednosti memorisjke lokacije iz polja **A**.

B – vrednost koja se dodeljuje, dodaje ili oduzima od vrednosti memorijske lokacije iz polja **A**.

C – definiše koliki je memorijski prostor odvojen za pamćenje memorijske lokacije (F – jedna reč od 32 bita ili H – polureč od 16 bita).

Početna vrednost memorijske lokacije se obavezno definiše (inicijalizuje) naredbom INITIAL koja mora da stoji pre naredbe SAVEVALUE. Naredba INITIAL je opisana prethodno.

MATRIČNE MEMORIJSKE LOKACIJE

Blok naredba: **ime MATRIX A,B,C**

Ovom blok naredbom se definiše matrica.

A – definiše se veličina memorijske lokacije koja se koristi za čuvanje vrednosti. Može biti **X** ako se rezerviše 32 bita (reč) ili H ako se rezerviše 16 bita (polureč).

B – definiše konstantu koja označava broj ***vrsta*** matrice

C – definiše konstantu koja označava broj ***kolona*** matrice

Blok naredba: **MSAVEVALUE A[±],B,C,D,E**

Ova naredba smešta podatke u matricu.

A – definiše ime matrične memorijske lokacije u kojoj se vrši menjanje vrednosti elementa definisanog sa poljima **B** i **C** vrednošću iz polja **D** ako nakon imena nema ni + ni –; ili se dodaje vrednost iz **D** ako je + ili oduzima ako je –.

B – broj vrste u kojoj se nalazi element čija se vrednost menja

C – broj kolone u kojoj se nalazi element čija se vrednost menja

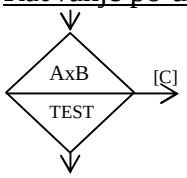
D – vrednost kojom se modificira element matrice definisane u polju **A** poljima **B** i **C**.

E – veličina memorijske ćelije: X – 32 bita (reč) ili H – 16 bita (polureč)

Pozivanje matrične memorijske lokacije se vrši: `MX$ime(i,j)` – za reč; `MH$ime (i,j)` – za polureč. Zadavanje početne vrednosti (inicijalizacija) memorijskoj lokaciji se vrši komandom INITIAL: `INITIAL MX$ime(i,j),CONST`.

BLOKOVI ZA RAČVANJE TRANSAKCIJA

Račvanje po uslovu stanja SNA: blok **TEST_{X}_A,B[,C]**



Ovim blokom se zadaju uslovi koji se proveravaju pri ulasku transakcije u ovaj blok. U zavisnosti od uslova određuje se dalji pravac kretanja transakcije. Uslov se zadaje pomoću algebarskih odnosa.

X – relacijski odnos koji se testira

X može biti:

X = L	A < B (less)
X = LE	A ≤ B (less or equal)
X = E	A = B (equal)
X = NE	A ≠ B (not equal)
X = G	A > B (greater)
X = GE	A ≥ B (greater or equal)

U polja **A** i **B** pišu se argumenti čije se vrednosti testiraju. Kao argumenti se najčešće koriste SNA, ali se u jedno od polja može staviti i konstanta (broj).

A – SNA ili konstanta sa leve strane poređenja

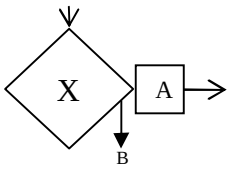
B – SNA ili konstanta sa desne strane poređenja

C – definisanje imena bloka u koji se šalje transakcija u zavisnosti od ispunjenja uslova testitanja. Ako je uslov ispunjen transakcija odlazi u sledeći (sekvencijalni) blok, a u suprotnom se šalje u nesekvencijani blok

definisan u polju **C** koji se nalazi bilo gde u modelu. Ako blok u koji se šalje transakcija nije slobodan, ona se zadržava u bloku TEST dok taj blok ne postane slobodan da primi transakciju. U zavisnosti od toga da li je polje **C** navedeno postoji:

- **režim odbijanja** – ako polje **C** nije navedeno;
  PR: `TEST LE Q6,30` – testira da li je tekuća dužina reda broj 6 manja od 30
- **režim prenosa** – ako je polje **C** navedeno.
  PR: `TEST L RN2,600,BBB` – 60% uputiti u sledći blok a 40% u blok BBB
  PR: `TEST GE V10,V20,DGUGI` – testira da li je vrednost izraza broj 10 ≥ od vrednosti izraza broj 20. Ako jeste transakcija prolazi na sledeći blok, a ako nije transakcija ide na blok DRUGI.

Račvanje po uslovu stanja opreme: naredba **GATE_{X}_A[,B]**



Ova naredba se koristi za testiranje logičkih stanja pojedinih delova sistema (uređaja, sklasišta, prekidača) pri čemu se stanje opreme koja se testira ne menja.
A – broj ili ime objekta (entiteta opreme) čije se stanje testira
B – definisanje bloka u koji transakcija odlazi ako uslov definisan u polju **X** nije ispunjen. U zavisnosti od toga da li je polje **B** navedeno postoji:

- **režim odbijanja** (režim uslovnog prolaza) – ako polje **B** nije navedeno. Blok GATE odbija ulazak transakcije sve dok se uslov definisan u polju **X** ne ispuni. Tek onda transakcija nastavlja da se kreće, a stanje uređaja se menja.
- **režim prenosa** (bezuslovni režim) – ako je polje **B** navedeno. Ako uslov definisan u polju **X** nije ispunjen transakcija ide u blok definisan u polju **B**. Ako je blok definisan u polju **B** zauzet, transakcija čeka.

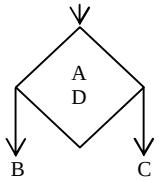
X – definisanje uslova testiranja; odnosi se na stanje opreme i može biti:

1. Za **uređaj** (FACILITY)
   - X = NU uređaj je ***slobodan*** (not in use)
   - X = U uređaj je ***zauzet*** (in use)
   - X = NI uređaj ***nije u prekidu*** (not in interupt) – u uređaju se ne nalazi transakcija većeg prioriteta
   - X = I uređaj ***u prekidu*** (interupt) – uređaj je zauzet transakcijom većeg prioriteta.
2. Za **skladište** (STORAGE)
   - X = SE skladište ***prazno*** (storage empty)
   - X = SNE skladište ***nije prazno*** (storage is not empty)
   - X = SF skladište je ***puno*** (storage full)
   - X = SNF skladište ***nije puno*** (storage is not full)
3. Za **logički prekidač** (LOGIC SWITCH)
   - X = LR logički prekidač ***isključen*** (logic switch reset)
   - X = LS logički prekidač ***uključen*** (logic switch set)
4. Za **blok MATCH** – nekada treba testirati da li je ispunjena sinhronizacija
   - X = M da li uslov sinhronizacije ***jeste*** ispunjen
   - X = NM da li uslov sinhronizacije ***nije*** ispunjen

BLOKOVI ZA PROMENU TOKA TRANSAKCIJE

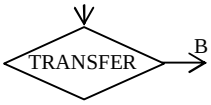
Transakcije se obično kreću sekvencijalno od bloka **n** preko **n+1** pa nadalje. Ako želimo da se kreću nesekvencijalno koristimo blokove za preusmeravanje.

Blok **TRANSFER A,B,C,D**



A – vrsta transfera (bezuslovni ili uslovni)
B i **C** – blokovi u koje ide transakcija posle grananja
D – indeksna promenljiva, koristi se samo kod operacije ALL.
Transakcija ostaje u bloku TRANSFER, ako blok u koji treba da ide (**B** ili **C**) ne može da je primi, sve dok se ne ispuni uslov prijema.

Bezuslovni transfer: blok **TRANSFER ,B**

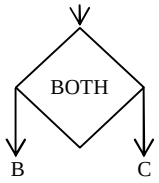


B – blok na koji transakcija ide bezuslovno. Ako blok definisan u polju **B** nije u mogućnosti da primi transakciju ona će biti zadržana u bloku TRANSFER sve dok se uslov ne ispuni.

Uslovni transfer

U polju **A** se navodi vrsta uslovnog transfera. U njemu mogu biti: **1.**BOTH, **2.**ALL, **3.**SIM ili **4.**statistiki izbor jedne od dve putanje za nastavak toka transakcije.

1.Blok **TRANSFER BOTH,B,C**



Kada transakcija naiđe na ovaj blok prvo se ispituje da li blok naveden u polju **B** može da primi transakciju. Ako može ona se tamo i šalje, a ako ne može, transakcija ide u blok definisan u polju **C**. Ukoliko se blok **B** izostavi tada se transakcija šalje na naredni blok, ukoliko je slobodan. Ako naredni blok nije slobodan transakcija se šalje u blok definisan u polju **C.** Ukoliko ni jedan blok ne može da primi transakciju ona će biti zadržana u bloku TRANSFER do ispunjenja uslova.

2.Blok **TRANSFER ALL,B,C,D**

B – prvi blok koji se testita

C – poslednji blok koji se testira

D – indeksna promenljiva i, koja dopušta korisniku da testira neke blokove unutar blokova definisanih u **B** i **C**.

B:	y	Transakcija pokušava da uđe u blok B
	y+i	transakcija pokušava da uđe u blok y+i
	y+2i	
	...	
C:	z = y+M·i	transakcija pokušava da uđe u blok C

PRIMER:

```
         TRANSFER ALL,PRVI,POSL,3
PRVI     SEIZE 1                    zauzmi uređaj 1
         ASSIGN 12,1                parametru 12 dodeli vrednost 1
         TRANSFER ,POSL+2           nastavak
         SEIZE 2                    zauzmi uređaj 2
         ASSIGN 12,2                parametru 12 dodeli vrednost 2
         TRANSFER ,POSL+2
POSL     SEIZE 3
POSL+1 ASSIGN 12,3
```

Ako je uređaj 1 zauzet ide na prvi+3, tu testira urđaj 2. Ako je slobodan zauzima ga, a ako nije testira prvi+6

3.Blok **TRANSFER SIM,B,C**

U svakoj transakciji postoji SIM indikator. Njegova vrednost može biti 1 ili 0. Kada je 1 tada transakcija ide u blok definisan u polju **C** a SIM indikator se vraća na 0. Kada je 0, transakcija ide u blok definisan u polju **B**. Ako polje **B** nije navedeno transakcija ide na naredni blok, a SIM indikator se postavlja na 1.

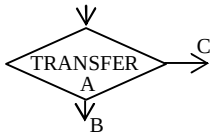
Ako treba da istovremeno bude zadovoljeno više uslova, svaki od njih se proverava i ako je svaki ispunjen SIM ostaje 0. Ako jedan od ulova nije ispunjen, SIM indikator se stavlja na 1.

PRIMER:

```
CPU    GATE NU 1              1 - CPU
       GATE NU 2              2 - KANAL
       GATE NU 15             3 - MEMORIJA
       TRANSFER SIM,,CPU
       SEIZE 15
```

Svi uređaji moraju biti slobodni za prenos. Ako bar jedan nije slobodan, transakcija se vraća na prvi uslov. U bloku TRANSFER SIM,,CPU – vrednost za SIM je 1 ako barem jedan uslov nije ispunjen, a ako su svi uslovi ispunjeni vrednost za SIM je 0.

4.Statistički transfer: blok **TRANSFER A,[B],C**



Koristi se kada je potrebno izvršiti izbor izvesnog procenta transakcija i preusmeriti ih na drugi deo modela.

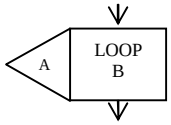
A – verovatnoća skoka na blok čije je obeležje dato u polju **C**. U ovom polju napisan je decimalni broj oblika ***.d*** ili ***.dd*** ili ***.ddd*** gde je $\boldsymbol{d} \in [\mathbf{0,1,2,...,9}]$.

B – blok u koji odlaze ostale transakcija. Ako **B** nije navedeno transakcija ide u naredni blok

C – blok u koji odlaze transakcije koje upadaju u verovatnoću koja je data u polju **A**

Ako nijedan od blokova ne može da primi transakciju ona ostaje u bloku TRANSFER do ispunjenja uslova prijema.

Blok: **LOOP A,B**



Ovaj blok obezbeđuje višestruki prolaz transakcije kroz određeni segment modela. Koristi se kada je za izradu nekog proizvoda potrebno izvršiti nekoliko operacija. Predstavlja petlju ili ciklus.

A – definiše broj parametara čija se vrednost koristi za definisanje broja ponavljanja.

B – definiše sledeći blok u koji se šalje transakcija ukoliko parametar koji je definisan u polju **A** nema vrednost ≤ 0. U svakom prolazu vrednost parametra iz **A** se smanjuje za 1.

Pre naredbe LOOP mora biti naredba ASSIGN u kojoj ćemo definisati parametar potreban za polje **A**.

PRIMERI:

```
        ASSIGN 10,5              ASSIGN 10,5
                      ═══ BLOK1 ADVANCE 20,FN10
DVA     ADVANCE 20    ═══       ASSIGN 10-,1
        LOOP 10,DVA             TEST LE P1,0,BLOK1
```

Blok: **SELECT x A,B,C,D,E[,F]**

Ovaj blok se koristi za ispitivanje skupa elemenata neke grupe (ili vrste) u cilju zadovoljenja nekih uslova. Pregled tih elemenata se vrši prema redosledu njihovog rednog broja. Kada se nađe element koji zadovoljava uslov, pregled ostalih se stopira (mnodel i dalje radi), a ako se ne nađe, pregled ide do poslednjeg elementa.

A – definiše broj parametra kome se dodeljuje redni broj elementa koji zadovoljava uslov

B – definiše element grupe sa minimalnim rednim brojem

C – definiše element grupe sa maksimalnim rednim brojem

D – definiše vrednost sa kojom se poredi vrednost atributa iz polja **E**

E – definiše atribut sa kim se poredi vrednost iz **D**

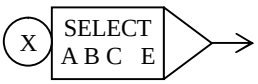
F – blok u koji ide transakcija kada uslov nije ispunjen. Ovo polje ne mora da se koristi. Ako je definisano a nije pronađen element koji zadovoljava uslov, vrednost parametra iz polja **A** se ne postavlja na 0. Ako polje **F** nije definisano i nije nađen element koji zadovoljava uslov, transakcija ide u sledeći blok a vrednost parametra iz polja **A** postaje 0.

x – definiše poređenje između SNA definisane u **E** i vrednosti definisane u **D**.

U ovom polju može se pisati: G – >; GE – ≥; E – =; L – <; LE – ≤; NE – ≠

NPR: Pregled skladišta sa rednim brojevima 5 (B), 6, 7, 8 (C) i traži se da li postoji neko skladište čiji je atribut SR manji od 250‰ (opterećenje manje od 250‰): `SELECT L 3,5,8,250,SR -` Ako je pregled uspešan, onda se u parametru 3 upisuje redni broj skladišta koje je zadovoljilo uslov i transakcija ide dalje. Ako nije pronađeno odgovarajuće skladište transakcija ide u sledeći blok a parametar 3 dobija vrednost 0.

Blok: **SELECT {x=min/max} A,B,C,,E**



Ova naredba se koristi za nalaženje člana grupe čija je vrednost nekog atributa minimalna ili maksimalna. Koristi se za davanje odgovora na pitanja: koji je najkraći red; koji uređaj ima najmanje opterećenje; koje je skladište prazno? Elementi koji zadovoljavaju uslov se pronalaze tako što se ispituje koji član iz te grupe ima min/max vrednost zadatog atributa.

A, **B**, **C**, **E** – ova polja su ista kao u prethodnoj naredbi SELECT

D – polje se ne koristi (jer nema smisla)

F – polje se ne koristi jer uvek jedan član ispunjava uslov

Ako postoje redovi sa jednakim dužinama, ova naredba proglašava red 1 za najkraći i poredi ga sa ostalima. Ako nijedan drugi nije kraći onda red1 ostaje taj, a ako naiđe na kraći njega proglašava za najkraći.

x – ima vrednosti **min** ili **max**

Atributi u polju **E** mogu biti: SR – opterećenje skladišta; FR – koeficijent iskorišćenja uređaja; R – broj slobodnih mesta u skladištu; FT – srednje vreme po transakciji provedeno na opsluzi nekog uređaja; Q – tekuća dužina reda.

PR: `SELECT MIN 1,3,10,,Q` – koji od redova od 3 do 10 ima najmanju tekuću dužinu reda. Redni broj tog reda se dodeljuje parametru 1. `SELECT MAX 5,9,14,,R` – koje skladište od 9 do 14 ima najviše slobodnih mesta i broj tog skladišta se beleži u parametru 5.

STANDARDNI NUMERIČKI ATRIBUTI (SNA)

Svakom entitetu (objektu) kao što su uređaji, skladišta, redovi, tabele odgovaraju atributu koji opisuje njegovo stanje u datom trenutku vremena. Vrednosti atributa mogu biti aritmetičke ili logičke. Neki od atributa korisnik ne može da menja a neke može. SNA su osobine (stanja) sistema koji se modelira; ujedno omogućavaju korisnicima dinamički pristup rezultatima simulacije. Svi ovi SNA imaju po jedno ili dva slova (memoriju) iza koga stoji broj (ime) entiteta na koji se on odnosi.
NPR: Q10 – tekuća dužina reda broj 10 (Q – reč; 10 – broj reda)

SISTEMSKI SNA: ostaju stalno isti i mi ih ne možemo menjati. Oni karakterišu sistem u celini (odnose se na ceo model) i nisu promenljive od strane korisnika.

Rn_j , j=1,...,8 – Generator slučajnih brojeva (nemogu se menjati veća samo kao takvi koristiti).

C1 - tečuće vreme časovnika simulatora (ne može se menjati)

SNA TRANSAKCIJA:

Pj – parametar j tekuće transakcije; (P10 - pozivamo se na vrednost parametra broj 10 transakcije);

M1 – tranzitno vreme (vreme putovanja tekuće τ. Može se modifikovati pomoću bloka MARK). M1 = CLOCK TIME – MARK TIME; Ako nema bloka MARK: M1 = C1.

MPj – parametarsko tranzitno vreme, definiše međutranzitivno vreme tekuće transakcije. Akumuliranje međutranzitnik vremena u parametru transakcije se može inicirati prolaskom transakcije kroz blok MARK koji u polju A ima definisan broj parametara. MP_j=tekuće vreme časovnika P_j.

SNA OPREME:

UREĐAJI:

Fj – numerički atribut (označava stanje uređaja j; 0 – uređaj je slobodan, 1 - zauzet);

FRj – promil iskorišćenja uređaja broj j;

FCj – broj τ koje su bile opslužene na uređaju j računajući i tekuću τ;

FTj – srednje vreme po τ provedeno na opsluzi na uređaju j. (F$PERA – j je ime ili broj)

SNA SKLADIŠTA:

Sj – tekući angažovani sadržaj;

Rj – tekući br. slobodnih mesta;

SRj – promil iskorišćenja;

SAj – srednji sadržaj;

SMj – max sadržaj;

SCj – ukupan broj τ koje su prošle kroz skladište j;

STj – srednje provedeno vreme po τ.

STATISTIČKI SNA:

REDOVI:

Qj – tekuća dužina reda;

QAj – srednja dužina reda;

QMj – max dužina reda;

QCj – ukupan br. τ koje su prošle kroz red j;

QZj – broj nultih τ (opslužene bez čekanja);

QTj – srednje vreme po τ provedeno u redu j uključujući i nulte pristupe;

QXj – srednje vreme po τ provedeno u redu j bez nultih pristupa.

TABELE:

TBj – Srednja vrednost argumenta za tabelu j;

TCj – broj pristupa tabeli j;

TDj – srednje kvadratno odstupanje za tabelu j.

SAVEVALUE:

Xj – sadržaj memorijske lokacije j;

XHj – sadržaj memoriske lokacije j (16 bita);

MXj(a,b) – sadržaj matrične memorijske lokacije j koja se nalazi u a-toj vrsti i b-toj koloni (32 bita);

MHj(a,b) – sadržaj matrične memorijske lokacije j koja se nalazi u a-toj vrsti i b-toj koloni (16 bita).

FUNKCIJE:

FNj – vrednost funkcije j. (FN5; FN$EXPO)

VARIJABLE (izrazi):
Vj – numerička vrednost aritmetičkog izraza j;
BVj – logička vrednost Bullove promenljive broja j.
KORISNIČKI REDOVI:
CHj – tekući br. τ u korisničkom redu;
CAj - srednji br. τ u korisničkom redu;
CMj – max. br. τ u k.redu;
CCj – ukupan br. τ koje su prošle kroz k.red;
CTj – sr. vreme po τ provedeno u k.redu.
BLOKOVI:
Nj – ukupan br. τ koje su prošle kroz blok j;
Wj – ukupan br. τ koje u tom trenutku čekaju ispred bloka j.

ADRESIRANJE U GPSS-U

Postoji tri načina adresiranja: **1.**direktno, **2.**relativno i **3.**indirektno.

Direktnno adresiranje znači direktno pozivanje na neki blok (tj. u polje lokacije se piše ime)

Relativno adresiranje je moguće ako postoji neki imenovani blok te se pisanjem u polje lokacije NPR: **BLOK1±n**, gde je **n** broj redova u odnosu na taj BLOK1.

Indirektno adresiranje – koristi se simbol ***n**, kojim se definiše broj parametra transakcije. Kao lokacija željenog argumenta se uvek koristi parametar. NPR:***5** - pozivamo se na vrednost parametra broj 5; **Q*5** - tekuća vrednost reda čiji je broj određen vrednossću parametra 5; **S*15** - tekući sadržaj skladišta čiji je broj određen vrednošću parametra 15.

Indirektno adresiranje se ne može koristiti zajedno sa C1, Rn_j, M1. Radi veće fleksibilnosti i mogućnosti ovog jezika treba koristiti indirektno adresiranje adresa entiteta koje bi bile sadržane u nekom od parametara transakcije. Umesto indeksa **j** pišemo ***n**, gde je **n** broj parametra. NPR: **SEIZE *10** – zauzimanje uređaja čiji je broj određen vrednossću parametra 10 tekuće transakcije; **Q*5** – tekući sadržaj reda čiji je broj određen vrednossću parametra 5.

PONAVLJANJE SIMULACIJE

Simulacija počinje sa kontrolnom naredbom START i traje sve dok terminacioni brojač ne padne na nulu, posle čega se štampaju/ne štampaju rezultati. Stanje podataka u sistemu ostaje nepromenjeno. Simulaciju je moguće ponoviti/nastaviti novom START naredbom. Pre nove START naredbe treba ubaciti RESET ili CLEAR (ili obe), koje utiču na stanje sistema. Takođe možemo ubaciti i neke deklerativne naredbe koje menjaju pojedine parametre modela (npr: SALA STORAGE 5...SALA STORAGE 8).

CLEAR i RESET su kontrolne naredbe koje daju odgovore na pitanja:
- Kako izvršiti više nezavisnih ponavljanja simulacionog modela?
- Kako da eliminišemo efekat pomerenosti opservacija zbog nereprezentativnih početnih uslova?

Ova pitanja su vezana za problem planiranja i analize simulacionih eksperimenata. Skoro uvek simulacioni modeli koriste slučajne promenljive. Za generisanje slučajnih promenljivih koriste se ugrađeni tokovi pseudoslučajnih brojeva (RN1÷RN8). Procena stvarnih vrednosti karakteristika sistema zavisi od slučajnih brojeva koji bi bili generisani za vreme simulacije. Međutim, izvršavanjem druge realizacije istog modela koja koristi drugi skup slučajnih brojeva, daće nam različitu procenu stvarnih vrednosti modela (sistema). Kako je procena tih vrednosti zavisna od slučajnih brojeva (ona sama je slučajna promenljiva), to znači da postoji određena varijabilnost unutar procene. Da bi bili u stanju da izmenimo te varijabilnosti treba da imamo više od jedne realizacije. Često je potrebno izvršiti niz izračunavanja modela na odgovarajući način. U tu svrhu se koriste kontrolne naredbe.

Kontrolna naredba **RESET**

Koristi se kad hoćemo da izbegnemo ***prelazni režim*** jer još uvek nije uspostavljena ***stacionarnost***. Rezulata dejstva naredbe RESET je da se anulira sva statistika koja je skupljena u procesu modeliranja, a transakcije ostaju u sistemu.

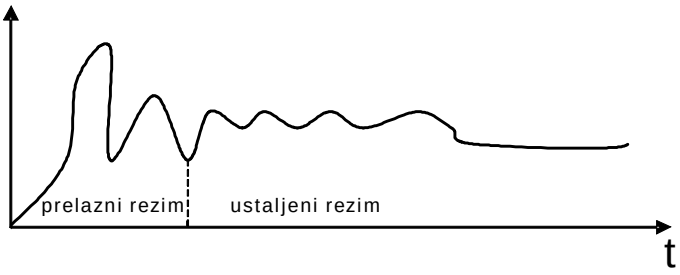
Anulira se:

- časovnik, anulira se vreme relativnog časovnika (za svaku fazu postoji relativno vreme rada);
- broj ulazaka transakcija u blokove (ali transakcije koje čekaju ispred blokova ostaju);
- upotreba uređaja se resetuje (anulira se broj ulazaka), ako je trenutno stanje uređaja 0 – slobodan; 1 – zauzet;
- upotreba skladišta se resetuje, ali se broj transakcija u skladištu izjednačava sa tekućim sadržajem skladišta;
- upotreba reda, ali broj klijenata u redu se izjednačava sa tekućim sadržajem reda

Ne menja se:

- Vrednost apsolutnog časovnika se ne menja;
- Tekuće transakcije ostaju u modelu;
- Memorijske lokacije i stanja logičkih prekidača;
- Šeme generatora slučajnih brojeva RNj. On se ne vraća na početak, već nastavlja tamo gde je stao.
- Korisnik ako želi može da zadrži vrednosti nekih atrinuta

PR: RESET F1,F3,Q1,S3 – resetovani su svi: uređaji osim 1 i 3, redovi osim 1, skladišta osim 3



U trenutku kada bi trebalo da nastupi ustaljeni režim aktiviramo RESET, jer nadalje važe formule SMO-a (sistema masovnog opsluživanja).

2 načina:

- ako sistem kratko traje, mi simuliramo i prelazni režim, simulaciju pustimo **n** puta, skupimo i procente sa **n**.
- Ako hoćemo da resetujemo statistiku prelaznog režima (stanja):

START 200,NP - simulator određuje kad se završava prelazni režim posle čega se izveštaj ne štampa
RESET - anuliraj statistiku
START 100 - simulator određuje vreme ustaljenog režima (stabilnog stanja), i štampa rezultat

Kontrolna naredba **CLEAR** [lista memorijskih lokacija]

Ova kontrolna naredba anulira sve statistike i udaljuju se sve transakcije (poništavaju se). Zajedno sa START naredbom koristi se kad je potrebno višestruko izvršiti simulacije, a ako je potrebno možemo ponoviti izvršenje simulacije sa izvesnim promenama u strukturi modela. Kad se naiđe na ovu naredbu, procesor vrši dejstvo:

- udaljuju se sve transakcije iz modela (tekuće i one koje su prošle);
- anulira sve statistike koje se odnose na entitete, sem onih koji su definisani sa naredbom CLEAR;
- anulira tekuće i ukupne pristupe blokovima;
- anulira vreme relativnog i apsolutnog časovnika (simulacija počinje iz početka)

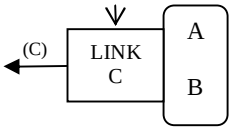
Ova naredba ne utiče na sekvence pseudoslučajnih brojeva (generatori slučajnih brojeva nastavljaju gde su stali da generišu, ne vraćaju se na polazni broj). Dejstvo ove naredbe je efektivno u trenutku njenog pojavljivanja i ona ne dejstvuje na naredbe koje slede iza nje. Ispred prve naredbe START nije potrebna naredba CLEAR. Ako želimo npr. da neka memorijska lokacija zadrži svoju vrednost i pri drugoj realizaciji modela potrebno je tu lokaciju navesti zajedno sa naredbom CLEAR (CLEAR X1 – sve lokacuije anulira sem lokacije broj 1).

Anulira se:**1.**broj svih ulaza i broj nultih čekanja; **2.**upotreba uređaja, svi se prevode u stanje slobodan (0); **3.**sve u vezi skadišta; **4.**kod redova: broj pristupa redu i broj nultih pristupa; **5.**sve informacije za tabele i memorijske lokacije; **6.**logički prekidači se isključuju (off); **7.**relativno vreme časovnika; **8.**svi blokovi GENERATE se inicijalizuju (počinju od početka). Ne menja se: **1.**Generatori slučajnih brojeva nastavljaju generisanje; **2.**ako želimo zadržati neke memorijske lokacije i pri drugoj realizaciji, stavljamo broj memorijske lokacije.

Korisnički redovi (user chains): blok naredbe LINK i UNLINK

Klasični redovi u modelu zahtevaju određeno vreme obrade jer računar skenira sve transakije u modelu. To vreme treba skartiti da bi ubrzali vreme izvršenja računara. Zbog toga se uvode ***korisnički redovi***. Oni se uvode i zbog načina opsluživanja (simulator radi na principu FIFO, a mi želimo LIFO ili neki drugi princip), jer ih koristimo za nestandardne načine opsluge (LIFO). Uvođenje ima smisla samo onda kad su redovi dugački; za kratke redove to nema smisla. Korisnik može da formira jedan ili više redova u koje stavlja transakcije, a ove transakcije postaju neaktivne sve dok one ponovo ne izađu iz reda.

Blok naredba: **LINK A,B,C**



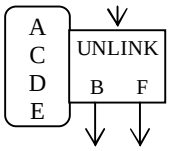
Ova naredba ima ulogu postavljanja transakcije u korisnički red.

A – definiše broj ili ime korisničkog reda

B – ukazuje na disciplinu opsluživanja. Može biti: FIFO (first in first out) – transakcija se stavlja na kraj reda; LIFO (last in first out); Pn (parametru transakcije) – transakcije se stavljaju u red tako da budu sortirane po **n**-tom parametru transakcije (ukoliko su vrednosti parametra transakcija iste one se uređuju prema FIFO).

C – definiše alternativni izlaz koji se koristi kod predstavljanja specijalnih (različitih) situacija uređivanja transakcija.

Blok naredba: **UNLINK A,B,C,D,E,F**



Služi za vađanje transakcija iz korisničkog reda da bi se ponovo aktivirale.

A – definiše broj ili ime korisničkog reda iz koga će transakcija biti izvađena

B – definiše se obeležje (broj ili ime) sledećeg bloka u koji odlazi izvađena transakcija

C – definiše broj transakcija koje se vade iz korisničkog reda. Može biti konstanta (najčešće 1) ili **ALL** (sve).

Ako se koriste **A**, **B**, **C** polja, program vadi transakcije sa početka reda. Ako se koristi i polje **D** (BACK), transakcije se vade sa kraja reda. U tom slučaju polje **E** se ne koristi.

D, **E**, **F** polja se koriste u slučaju da se transakcije vade prema vrednosti parametra.

D – definiše broj parametra čija se vrednost ispituje

E – definiše vrednost paramatra koju mora da ima transakcija koja bi bila izvađena

F – definiše lokaciju u koju će biti poslata transakcija koja je inicirala vađenje, ukoliko nije pronađena ni jedna transakcija koja zadovoljava uslov. Ako se polja **F** izostavi, transakcija ide u blok koji sledi iza bloka UNLINK, tj. u sledeći sekvencijalni blok.

RAČUNSKI ENTITETI

U računske entitete spadaju: funkcije, aritmetički izrazi (promenljive), bulovi (logički) izrazi.

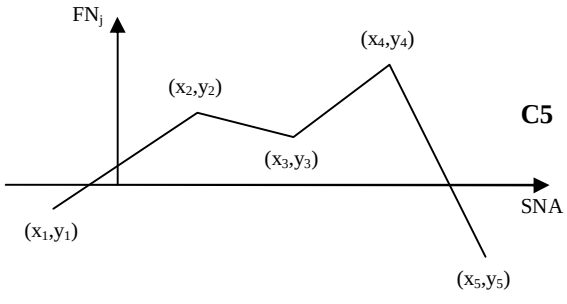
1.Funkcije

Funkcije su osnovne numeričke vrednosti koje se izračunavaju na osnovu pravila koje definiše korisnik. Postoje zakoni po kojima se vrši opsluživanje ili dolazak u sistem. Zbog složenosti tih zakona se koriste funkcije. Najčešće su to raspodele verovatnoća.

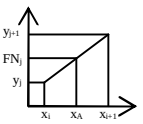
Tipovi funkcija:

- Kontinualne – **Cn**
- Diskretne – **Dn**
- Spisak Numeričkih Vrednosti – **Ln**
- Diskretne Vrednosti Atributa – **En**
- Spisak Vrednosti Atributa – **Mn**

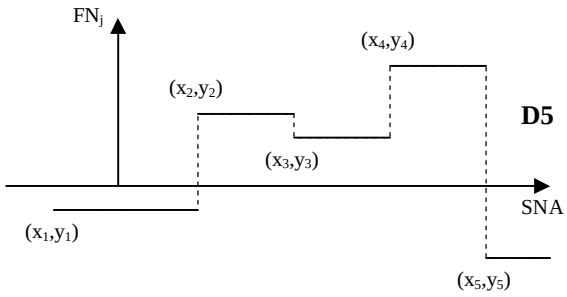
1.Kontinualna funkcija



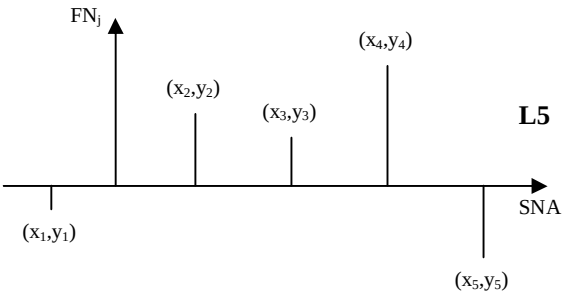
Tačke se spajaju pravim linijama. Ako je neki argument između dve tačke definisanja, tada GPSS program računa vrednost zavisne promenljive $\mathbf{FN_j}$ koristeći linearnu interpolaciju. $FN_j = y_j + \frac{y_{j+1} - y_j}{x_{i+1} - x_i} \cdot (x_A - x_i)$



2.Diskretna funkcija



3.Spisak numeričkih vrednosti



Ako je kod ovih funkcija $\mathbf{x_i<arg<x_{i+1}}$, tada je $FN\,j = \begin{cases} y_1 & \text{za arg} \le x_1 \\ y_2 & \text{za arg} > x_1 \end{cases}$

Format pisanja funkcije u GPSS-u

U GPSS-u funkcije se isključivo zadaju tabelarno. Format naredbe za pisanje funkcije je:

br/ime FUNCTION A,B

x_i y_i

Najpre se definiše broj ili ime funkcije. Funkcija se u programu poziva upotrebom SNA za funkciju: **FN$ime** ili **FNbroj**.

A – definiše se argumet funkcije (nezavisna promenljiva). Kao argumet funkcije koristi se bilo koji SNA. Ako se radi o raspodeli verovatnoća, onda se u ovom polju definiše generator slučajnih brojeva (GSB) **RNj**, j=1,2,...,8. Kada se GSB koristi pri definisanju tačaka funkcije njegove vrednosti su $0 \le RNj < 1$.

B – definiše se tip funkcije. Tu se može navesti: **Cn** – kontinualna funkcija, **Dn** – diskretna funkcija, **Ln** – spisak numeričkih vrednosti, **En** – diskretne vrednosti atributa ili **Mn** – spisak vrednosti atributa. Oznaka **n** u tipu funcije definiše broj parova tačaka kojima se funkcija zadaje.

x_i y_i – parovi tačaka kojima se definišu vrednosti funkcije. Svaka funkcija mora da ima bar 2 definisane tačke pa čak iako je vrednost funkcije u njima ista. Za pisanje parova tačaka postoje dva načina:

1.Standardni format

x_1	y_1	x_2	y_2
1-6. polja	7-12. polja	13-18. polja	19-24. polja

Podaci se pišu od 1. do 71 kolone a onda se prelazi u nov red i nastavlja se pisanje parova tačaka.

2.Slobodni (fiksni) format – češće se koristi

Tačke se definišu pisanjem parova na sledeći način: **x_1,y_1/x_2,y_2/...** do 71. kolone, nakon koje se prelazi u nov red. Pri tome parovi tačaka moraju biti u istom redu. Svaka sukcesivna vrednost **x** mora biti veća od prethodne: $x_2 > x_1$.

PRIMERI:

Kontinualna funkcija
```
ime/br FUNCTION arg,C5
       x1,y1/x2,y2/...
```

Diskretna funkcija
```
ime/br FUNCTION arg,D5
       x1,y1/x2,y2/...
```

Spisak numeričkih vrednosti
```
ime/br FUNCTION arg,L5
       [1],y1/[2],y2/...
```

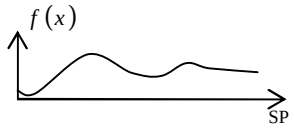
Diskretne vrednosti atributa
```
ime/br FUNCTION arg,E5
       x1,SNA1/x2,SNA2/...
```

Spisak vrednosti atributa
```
ime/br FUNCTION arg,M5
       [1],SNA1/[2],SNA2/...
```

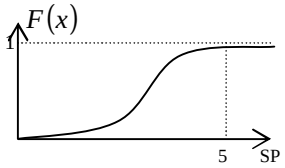
Definisanje verovatnoća u GPSS-u

Slučajne promenljive mogu biti: ***kontinualne*** (slučajna promenljiva može uzeti bilo koju vrednost iz intervala $(-\infty,+\infty)$) i ***diskretne*** (slučajna promenljiva uzima vrednosti iz skupa $\{x_1,...,x_n\}$).

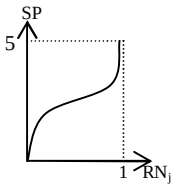
1.**Kontinualne verovatnoće**



$f(x)=P(SP=x)$ je gustina raspodele verovatnoća



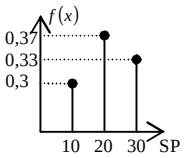
$F(x)$ - funkcija raspodele verovatnoća (kumulativna raspodela)



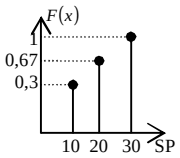
Inverzna funkcija raspodele verovatnoća na osnovu koje se generišu SP (slučajne promenljive). RN_j je generator slučajnih brojeva koji uzima vrednosti od 0 do 0,999 na osnovu kojih se dobija vrednost slučajne promenljive.

2.**Diskretne verovatnoće** (Slučajna promenljiva uzima diskretne vrednosti)

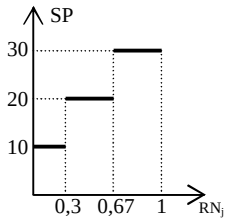
SP	10	20	30
P	0,3	0,37	0,33



$f(x)=P(SP=x)$ je gustina raspodela verovatnoća



$F(x)$ je funkcija raspodele verovatnoća (kumulativna verovatnoća)



Inverzna funkcija raspodele verovatnoća na osnovu koje se definiše funkcija raspodele u GPSS-u.

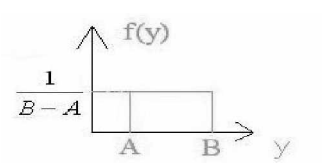
NPR:

```
1 FUNCTION RN2,D3
.3,10/.67,20/1,30
```

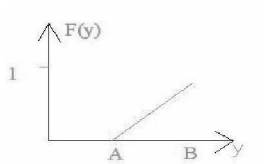
Tipične raspodele slučajnih promenljivih su: **1.**uniformna; **2.**eksponencijalna (EXPO); **3.**Puasonova (diskretna); **4.** Erlangova; **5.** normalna raspodela.

1.Uniformna raspodela

KONTINUALNA



Gustina raspodele verovatnoća



Kumulativna verovatnoća

y, sa jednakim verovatnoćama, uzima vrednost iz intervala $[A,B]$. Za vrednosti y koje su $A\le y\le B$ važi:

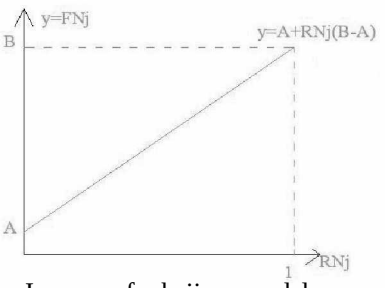
$$f(y)=\frac{1}{B-A}$$

$$F(y)=\int_A^y f(y)dy=\int_A^y \frac{1}{B-A}dy=\frac{1}{B-A}\int_A^y dy=\frac{y-A}{B-A}$$

$$F(y)=\begin{cases}0 & \text{za } y<A\\ \dfrac{y-A}{B-A} & \text{za } A\le y\le B\\ 1 & \text{za } y>B\end{cases}$$

Srednja vrednost $\mu=(B-A)/2$ a standardna devijacija $\sigma=(B-A)/12$

U intervalu $[A,B]$ za koji je $F(y)=\dfrac{y-A}{B-A}\Rightarrow y=A+(B-A)F(y)$.



Inverzna funkcija raspodele verovatnoća

Kako $F(y)$ uzima vrednosti iz intevala $[0,1]$ to znači da umesto nje se može uzeti generator slučajnih brojeva pa se u programu piše: $y=A+RNj(B-A)$

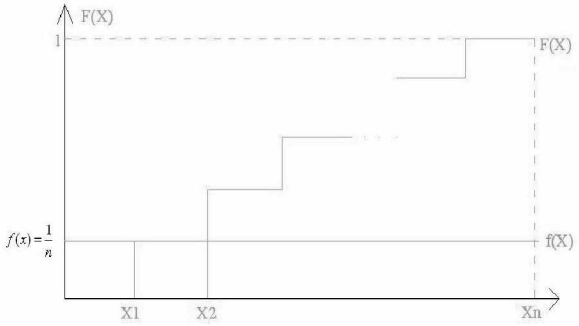
Na taj način GPSS dobija uzorke koji imaju uniformnu raspodelu.

NPR: Ako nam je $A=20$, $B=40$ tada je $y=20+RNj(40-20)$. U programu se ova raspodela definiše na sledeći načina:

```
RAVNOM FUNCTION RN3,C2
       0,20/1,41*
```

*Kako **RNj** uzima vrednosti od 0 do 0,999 broj 40 se nikada neće ostvariti. GPSS, takoće operiše samo sa celobrojnim vrednostima koje zaokružuje na celobrojni deo bez razlomnjenog dela pa se umesto 40 mora staviti 41 (NPR: $41\cdot 0{,}999=40{,}959$ odnosno GPSS će ovu vrednost zaokružiti na 40).

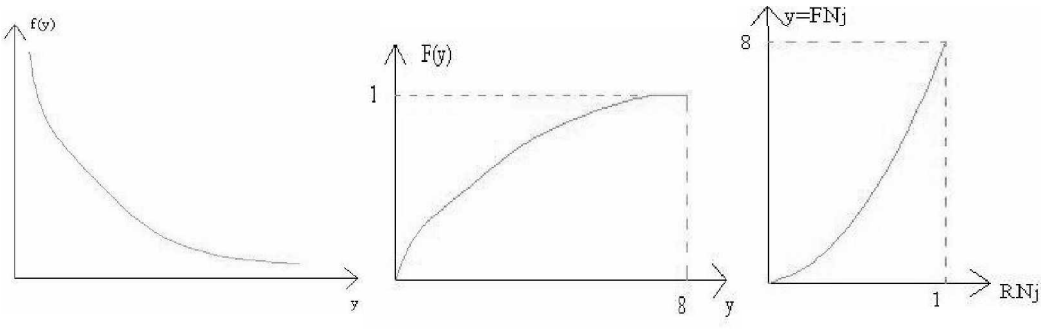
DISKRETNA



Slučajna promenljiva x uzima vrednosti iz diskretnog skupa $\{x_1, x_2, ..., x_n\}$ sa jednakim verovatnoćama. Na slici levo prikazane su funkcije gustine raspodele verovatnoća $f(x)=\frac{1}{n}$ i funkcija raspodele verovatnoća (kumulativna) $F(x)$ koja je stepeničasta.

2.**Eksponencijana raspodela** (EXPO)

Ova raspodela slučajne promenljive je kontinualna i uzima vrednosti veće od nule. Eksponencijalna slučajna promenljiva se koristi za modeliranje vremena između dolazaka kada ne postoji kontrola nad procesom dolazaka (dolasci klijenata u poštu). Eksponencijalna raspodela je potpuno definisana njenom očekivanom vrednošću $\mu = 1/\lambda$.



Na slikama su prikazane: funkcija gustine verovatnoće, kumulativna funkcija verovatnoće i inverzna kumulativna funkcija verovatnoće.

$$0 \le y < \infty$$

$$f(y)=\begin{cases}\lambda e^{-\lambda y} & \text{za } y \ge 0 \\ 0 & \text{za } y < 0\end{cases}$$

$$F(y)=\int_0^y f(y)dy=\int_0^y \lambda e^{-\lambda y}dy=1-e^{-\lambda y},\ 0 \le F(y) \le 1$$

Iz ovoga sledi da je $y=-\frac{1}{\lambda}\ln(1-F(y))$. $F(y)$ možemo zameniti sa generatorom slučajnih brojeva pa će biti:

$y=-\frac{1}{\lambda}\ln(1-RNj)$ a kako je $(1-RNj) \sim RNj$ onda važi sledeća formula za generisanje sučajnih promenljivih u GPSS-u koje podležu eksponencijalnoj raspodeli:

$$y=-\frac{1}{\lambda}\ln RNj$$

Za eksponencijalnu raspodelu važe pretpostavke: trenutak nastupanja događaja ne zavisi od prethodnog događaja i verovatnoća nastupanja događaja na malom intervalu je proporcionalno dužini tog intervala.

Inverzna funkcija raspodele, kako piše u programu GPSS predstavlja ovu kontinualnu krivu, tako što je podeljena na segmente i aproksimira se sa 24 tačke koje se spajaju pravim linijama, dok se ostale tačke dobijaju linearnom interpolacijom. Eksponencijalna funkcija u GPSS-u se definiše na sledeći način:

```
EXPO FUNCTION RN2,C24
     0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
     .75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
     .95,2.99/.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/0.995,5.3/
     0.998,6.2/0.999,7/0.9998,8
```

PR: `GENERATE 50,FN$EXPO` – generišu se transakcije sa srednjim vremenom od 50 v.j. a srednje vreme između dolazaka je eksponencijalno raspodeljeno.

Eksponencijalna raspodela ima visok stepen varijabilnosti. Ovo je konzistentno sa relativno nekontrolabilnim procesom, kao što je dolazak klijenata u poštu. Varijansa ove raspodele je jednaka kvadratu srednje vrednosti.

3.Puasonova raspodela

Kada su vremena između dolazaka eksponencijalno raspodeljena, tada iznos dolazaka povezan sa njim sledi Puasonovu raspodelu ali moraju biti zadovoljene pretpostavke:

- verovatnoća da se dolazak dešava za vreme malog vremenskog intervala proporcionalna je veličini intervala;
- verovatnoća dva ili više dolazaka koji se dešavaju za vreme dovoljno malog vremenskog intervala je neznatno mala (posledica ove pretpostavke je da se simulacioni dolasci ne dešavaju po Puasonovom procesu dolaska. Drugim rečima, vreme između dolazaka nikada nije jednako nuli);
- vremena između dolazaka su nezavisna od svih drugih.

Ova raspodela se koristi ako nam je potreban tačan broj dešavanja. Slučajna promenljiva ima Puasonovu raspodelu ako može uzeti vrednost k iz niza celih brojeva sa verovatnoćom:

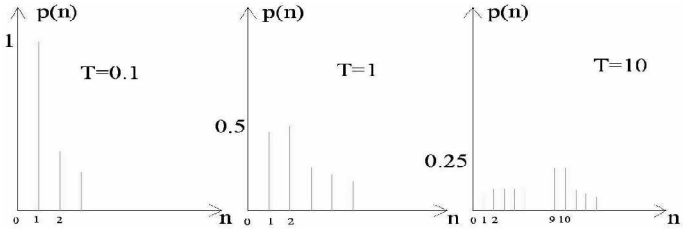
$$P(k)=P(x=k)=\frac{\lambda^k}{k!}e^{-\lambda}\text{, gde je}$$

k - stalan broj dešavanja

λ - očekivan broj dešavanja ili dolazaka (parametar raspodele, veći od 0)

Ako su intervali između dva susedna događaja eksponencijalno raspodeljeni, onda je verovatnoća da se n događaja desi na intervalu T:

$$P(\tilde{n}=n)=\frac{(\lambda\cdot T)^n\cdot e^{-\lambda T}}{n!}$$



4.Erlangova raspodela

Uzorak iz Erlangove raspodele se može dobiti sabiranjem k nezavisnih eksponencijalnih raspodela.Erlangova raspodela se može koristiti za modeliranje vremena opsluge. Ona je definisana sa dva parametra: redom k (parametrom oblika) i srednjim vremenom opsluživanja. Eksponencijalna raspodela je Erlangova raspodela prvog reda. U praksi se najčešće sreću Erlangove raspodele drugog reda. Jedan od načina da uzmemo uzorak iz Erlangove raspodele k-tog reda sa očekivanom vrednošću μ je da formiramo sumu od k uzoraka iz eksponencijalne raspodele sa očekivanom vrednošću μ/k.

$$f(x)=k\cdot Q\cdot e^{-k\cdot Q\cdot x}\frac{(k\cdot Q\cdot x)^{k-1}}{(k-1)!},\ F(x)=1-\frac{\sum_{i=1}^{k-1}e^{-k\cdot Q\cdot x}(k\cdot Q\cdot x)^i}{i!}$$

$$f(x)=\lambda e^{-\lambda x}\frac{(\lambda x)^{k-1}}{(k-1)!},\ F(x)=1-\sum_{j=1}^{k-1}e^{-\lambda x}\frac{(\lambda x)^j}{j!}$$

$$E(x)=E(x_1)+E(x_2)+\ldots+E(x_k)$$

$$x_j\to\frac{1}{k\cdot Q},\ E(x)=\frac{1}{k\cdot Q}+\frac{1}{k\cdot Q}+\ldots+\frac{1}{k\cdot Q}=k\frac{1}{k\cdot Q}=\frac{1}{Q}$$

$$x=\frac{1}{k\cdot Q}(FN\$EXPO+FN\$EXPO+\ldots+FN\$EXPO)$$

PR: Vreme opsluge ima Erlangovu raspodelu drugog reda (dve faze rada) i očekivanu vrednost od 4 minuta $\left(\frac{1}{Q}=4\,\text{min}\right)$. Vremenska jedinica je 1 sekund. Srednje vreme opsluge za obe faze je $\frac{1}{k\cdot Q}=\frac{4}{2}=2\,\text{min}=120\,\text{sec}$. Ovo se u GPSS programu piše na sledeći način:

```
ERLAN VARIABLE 120*(FN$EXPO+FN$EXPO)
```

Ovaj vremenski period možemo simulirati sa dva bloka ADVANCE.

```
ADVANCE 120,FN$EXPO
ADVANCE 120,FN$EXPO
```

5.Normalna (Gausova) raspodela

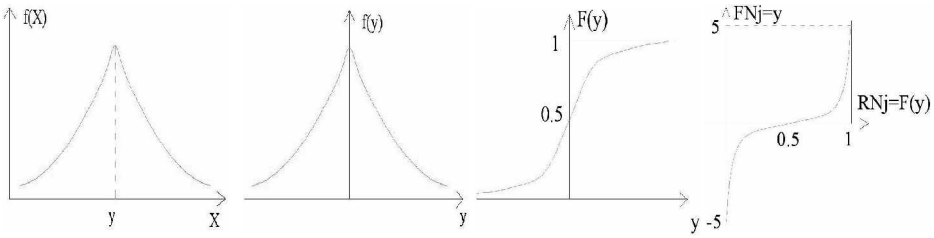
Normalna slučajna promenljiva, koja je kontinualna i simetrička, uzima vrednosti u opsegu od $-\infty$ do $+\infty$. Ova raspodela je u potpunosti definisana sa očekivanom vrednošću μ i standardnom devijacijom σ. Funkcija gustine raspodele verovatnoća normalne raspodele je:

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normalnoj raspodeli podvrgavaju se vremena opsluge, kao što je vreme potrebno za izradu nekog proizvoda ili vreme potrebno za prenos poruke u komunikacionom sistemu.

Normalna slučajna promenljiva sa $\mu=0$ i $\sigma=1$ se naziva ***standardizovana normalna raspodela***.

Ako je x vrednost iz normalne raspodele sa μ i σ a y vrednost iz standardizovane normalne raspodele onda je $y=\frac{x-\mu}{\sigma}$ odnosno $x=\mu+\sigma y$, dok je funkcija gustine standardizovane normalne raspodele: $f(y)=\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}$. Funkcija raspodele verovatnoća $F(y)=\int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}dy$ se ne može rešiti pa se vrši aproksimacija sa 25 tačaka.



Da bi dobili uzorak iz opšte normalne raspodele, potrebno je dobiti uzorak iz standardne normalne raspodele a tada korišćenjem formule $x=\mu+\sigma y$ dobijamo traženi uzorak.

PR: Za $\mu=12$ i $\sigma=2$

```
SNORM FUNCTION RN5,C25                Standardizovana normalna raspodela
      0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2/.06681,-1.5/
      .11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6/.34458,-.4/.42074,-.2/
      .5,0/.57926,.2/.65542,.4/.72575,.6/.78814,.8/.84134,1/.88493,1.2/
      .93319,1.5/.97725,2/.99379,2.5/.99865,3/.9997,4/1,5
GNORM VARIABLE 2*FN$SNORM+12          Normalna slučajna promenljiva
```

Na osnovu **centralne granične teoreme**, raspodela sume (srednjih vrednosti) niza nezavisnih identično raspodeljenih slučajnih promenljivih iz bilo koje raspodele prilazi normalnoj raspodeli za veliki broj članova u sumi. U većini slučajeva, centralna granična teorema je zadovoljena kada je obuhvaćena suma od 30 opservacija.

2.Varijable – izrazi

Služe za specifikaciju odgovarajućih logika ili matematičkih relacija između promenljivih sistema. Oni mogu biti:

- **aritmetički** izraz je sredstvo da se na jednom mestu definiše izraz a da se na drugom mestu u modelu koristi;
- **logički** izrazi se koriste kod logičkih testova (rezultat je 0 – tačno ili 1 – netačno).

Izrazi se formiraju od

- ***operanada*** – SNA ili neka druga promenljiva kao i konstanta;
- ***operatora*** + sabiranje,
  - oduzimanje,
  * množenje,
  / deljenje
  @ modulo deljenje (ostatak deljenja)

Varijable (izrazi) se pozivaju na sledeći način:

V*j* – *j* je broj varijable (NPR: ADVANCE V10)

V$ime – ovako se poziva izraz koji ima ime (NPR: ADVANCE V$KASA)

Aritmetički izrazi

mogu biti:

- sa **fiksnom** tačkom (blok naredba **j VARIABLE <aritmetički izraz>**). To su izrazi koji koriste celobrojnu aritmetiku. Od svakog operanda se uzme celobrojni deo i od rezultata se uzima takođe celobrojni deo.
  aritmetički izraz – izraz tipa: P1*(FN3+5)*V$KASA+5*P10. Ukoliko se u izrazu u ovom slučaju navede konstanta 6,6 u izrazu će se uzeti vrednost 6. Isto važi i za funkcije kao i za rezultat.
  j – u ovom polju definiše se ime ili broj varijable čijim se pozivanjem u programu realizuje vrednost aritmetičkog izraza.
  PR: 100*(FN5/5), za FN5=12,6 biće: 12,6:5=2,52 zaokružuje na 2 pa množi sa 100 i rezultat je 200.
- sa **pokretnom** tačkom (blok naredba **j FVARIABLE <aritmetički izraz>**). Ova blok naredba koristi i razlomnjene delove operanada a rešenju odseca razlomnjeni deo. U odnosu na VARIABLE pravi manju grešku.
  PR: 100*(FN5/5), za FN5=12,6 biće: 12,6:5=2,52 koji se množi sa 100 i rezultat je 252.

Logički izrazi

se koriste kod logičkih testova. Ako su uslovi ispunjeni rezultat je 1 a ako nisu rezulata je 0. Blok naredba ima sledeći oblik: **j BVARIABLE <logički izraz>**

j – broj ili ime logičkog izraza

OPERATORI:

Logički operatori za

UREĐAJE: FIn – uređaj n je prijemtiran; FNin – uređaj n nije prijemtiran; FUn – uređaj n je u upotrebi; FNUn – uređaj n nije u upotrebi; Fn – uređaj n je ili u upotrebi ili je prijemtiran.

SKLADIŠTA: SFn – skladište n je puno; SNFn – skladište n nije puno; SEn – skladište n je prazno; SNEn – skladište n nije prazno.

LOGIČKE PREKIDAČE: LSn – logički prekidač n je uključen; LRn – logički prekidač n je isključen

Uslovni operatori (za poređenje 2 operanda): G – veće od >; L – manje od <; E – jednako =; GE – ne manje od ≥; LE – ne veće od ≤; NE – nije jednako ≠.

Logički (bulovi) operatori: + je ***ili*** – testira da li je bar jedan uslov ispunjen; * je ***i*** – testira da li su oba uslova ispunjena (PR: 5 BVARIABLE FNI2*SF3)

Generisanje slučajnih promenljivih

Većina sistema koji se simulraju pomoću GPSS-a sadrže jednu ili više slučajnih aktivnosti. Uglavnom, ove aktivnosti se opisuju u vidu statističkih raspodela. Na primer, mnogi sistemi masovnog opsluživanja pretpostavljaju proces Puasonovih dolazaka. Da bismo simulirali takve aktivnosti, model mora da generiše slučajne uzorke iz odgovarajućih statističkih raspodela.

Vremena između dolazaka i vremena opsluge mogu da budu deterministička, uniformno raspodeljena i mogu da budu modelirana sa neuniformnim raspodelama verovatnoća. Većina raspodela treba da bude definisana u tabelarnom obliku. Funkcije koje se zadaju u GPSS-u imaju samo jedan argument i jedini način njihovog zadavanja je tabelarni.

Uzimanje uzoraka iz populacije

Generisanje slučajnih uzoraka se generalno izvršava u dva koraka. Najpre se vrednost izvlači iz generatora slučajnih brojeva koji daju brojeve uniformno raspodeljene u otvorenom intervalu (0,1). Zatim se uniformno raspodeljeni slučajni uzorak preslikava u ekvivalentnu vrednost iz ciljne raspodele.

Kontinualne slučajne promenljive mogu da uzimaju bilo koju vrednost u nekom intervalu. Vremena između dolazaka i vremena opsluge obično su kontinualne slučajne promenljive.

Kao podršku dvokoračnom procesu za uzimanje uzoraka potrebno je da imamao izvor slučajnih brojeva uniformno raspodeljenih na intervalu (0,1). Takođe, neophodno je da definišemo način kako da preslikamo slučajnu vrednost iz izvorne uniformne raspodele (0,1) u ekvivalentnu vrednost iz ciljne populacije.

Generatori uniformnih slučajnih brojeva

Osobine generatora slučajnih brojeva su ponovljivost i periodičnost. ***Ponovljivost*** – svaki put kad startujemo program komandom START dobijamo iste sekvence slučajnih brojeva koji se uvek generišu od istog broja (semena). Korišćenjem komande RESET na kraju programa, generisanje se ne vraća na početak već se ponavlja. ***Periodičnost*** – moguće je da se u toku istog izvršavanja programa nakon izvesnog vremena jave isti brojevi jer se najčešće sledeći generiše na osnovu prethodnog. ***Seme*** – slučajni broj od kog slučajni brojevi počinju da se generišu.

U GPSS-u postoji osam identičnih tokova generatora slučajnih brojeva (RN1, RN2,…, RN8). Oni daju vrednosti iz opsega od 0 do 999, osim kada se koriste kao argument funkcije i tada daju vrednosti iz populacije uniformno raspodeljene na otvorenom intervalu [0,1), odnosno od 0 do 0,999 (otvoreni interval je onaj koji ne uključuje krajnje tačke).

Svaki od ovih generatora ima neidentičnu podrazumevanu početnu tačku (seme). U GPSS-u se koristi jedan algoritam (procedura) za dobijanje uniformnih slučajnih brojeva na intervalu 0-1 (najčešće Lehmer-ov multiplikativni kongurentni algoritam). Matematički rečeno, takav generator je definisan kao rekurentna relacija. Algoritam je deterministički, tako da slučajni brojevi koje on stvara nisu stvarno slučajni, već su pseudoslučajni, što znači da su ponovljivi. Pitanje koje se često postavlja od strane novih korisnika simulacije je: zašto ne generišemo stvarne slučajne brojeve, korišćenjem hardverskog uređaja pre nego tehniku determinističkog generisanja pseudoslučajnih brojeva? Razlog za korišćenje tehnike generisanja pseudoslučajnih brojeva je to što ona stvara ponovljive sekvence "slučajnih" uzoraka. Ponovljivost slučajnih brojeva može se smatrati kao prednost pri planiranju statističkih eksperimenata. Interno predstavljanje slučajnih brojeva zavisi od računara na kom se radi.

Transparentno uzimanje uzoraka u blokovima GENERATE A,B; ADVANCE A,B i TRANSFER

Najjednostavniji oblik generisanja slučajnih veličina jeste korišćenje modifikatora u vidu intervala, tj. "proširenja". Operandi **A** i **B** kod blokova GENERATE i ADVANCE mogu se koristiti za specificiranje modifikatora proširenja u vidu poluintervala uniformno raspodeljenih vremena između dolazaka (GENERATE) i vremena opsluge (ADVANCE). Kod ovakvog korišćenja blokova GENERATE i ADVANCE proces uzimanja uzoraka je transparentan. U opštem obliku blok GENERATE se može koristiti kao GENERATE A,B. U tom slučaju, vremena između dolazaka transakcija su iz uniformne raspodele A±B. GPSS uzima uzorak transparentno iz A±B populacije korišćenjem generatora RN1 kao izvorne slučajne brojeve, preslikavajući vrednost RN1 u ekvivalentno vreme između dolazaka t_d prema sledećoj formuli (T.J.Schriber 1991):

$$t_d=(A-B)+RN1*(2*B)$$

U tom slučaju vrednost t_d se kreće u granicama od A-B do A+B, što je konzistentno sa činjenicom da su vremena između dolazaka raspodeljena na otvorenom intervalu A±B. Ako se ova formula definiše kao aritmetički izraz, tada se može koristiti bilo koji od generatora slučajnih brojeva.

Vidimo da u slučaju transparentnog uzimanja uzoraka u blokovima GENERATE i ADVANCE, u svim takvim blokovima u modelu koristi se generator RN1. Kada se vrši transparentno uzimanje uzoraka u dva ili više takva bloka, svaki blok koristi samo podskup vrednosti generisanih od strane RN1 u posmatranom vremenskom opsegu.

Naredba TRANSFER, takođe, koristi tok slučajnih brojeva RN1. Format statističkog bloka TRANSFER je: TRANSFER .A,B,C. Operandi A i B su obeležja blokova u modelu. Operand A ukazuje na verovatnoću sa kojom se transakcije usmeravaju ka bloku čije je obeležje definisano u operandu C. U cilju izvršavanja ovog procesa, GPSS izvlači uzorak iz RN1 i onda koristi uzimanje vrednosti uzorka u cilju određivanja sledećeg bloka u koji će biti upućena transakcija. Na primer, pretpostavimo da transakcija ulazi u sledeći blok TRANSFER:

TRANSFER .250,BLOK1, BLOK 2

U tom slučaju, poziva se generator RN1 i dobijena vrednost (to je trocifreni ceo broj u zatvorenom intervalu od 0 do 999) poredi se sa 250. Ako je izvučeni broj manji od 250, transakcija se upućuje u blok čije je obeležje BLOK2, u suprotnom biće upućena u blok čije je obeležje BLOK1.

Kao što vidimo, u statističkim blokovima TRANSFER postoji transparentna upotreba generatora RN1, kao i za transparentno uzimanje uzoraka u blokovima GENERATE i ADVANCE.

NPR: TRANSFER .250,BLOK1,BLOK2 transparentno se može predstaviti preko komande TEST:
TEST L RN3,.250,BLOK1 – na ovaj način može bilo koji RNj da se pozove. Ako imamo više blokova transfer, na ovaj način uzorci su statistički nezavisni.

Statistička nezavisnost u simulaciji

Ukoliko promene koje se dešavaju u jednom delu modela ne utiču na ponašanje drugih delova modela, tada izvori slučajnosti u modelu moraju da budu statistički nezavisni.

NPR: Posmatraćemo jednu javnu telefonsku govornicu. Dolasci klijenata su puasonovski sa srednjim vremenom između dolazaka od 120s. Vreme opsluge (trajanje razgovora) je ekspnencijalno raspodeljeno sa srednjom vrednošću od 100s. Kada klijent dođe, ukoliko je govornica zauzeta, on staje u red. Potrebno je dobiti statistike vezane za red, kao i za telefonsku govornicu. Simulirati vremenski period od 18 časova, a za vremensku jedinicu uzeti 1s.

Najpre ćemo napisati program korišćenjem istog toka pseudoslučajnih brojeva RN3.

```
* SIMULACIJA RADA JEDNE JAVNE TELEFONSKE GOVORNICE
      SIMULATE
*
EXPO FUNCTION RN3,C24          Eksponencijalna raspodela
      0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
      .75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
      .95,2.99/.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/0.995,5.3/
      0.998,6.2/0.999,7/0.9998,8
*
      GENERATE 120,FN$EXPO     Dolazak klijenata
      QUEUE 1                  Klijent pristupa redu
      SEIZE 1                  Klijent ulazi u govornicu
      DEPART 1                 Klijet napušta red
      ADVANCE 100,FN$EXPO      Trajanje razgovora
      RELEASE 1                Klijent oslobađa govornicu
      TERMINATE                Klijent napušta sistem
*
* SEGMENT  TAJMERA
*
      GENERATE 64800           Generisdanje Tajmera
      TERMINATE 1              Umanji TB za 1
      START 1                  Početak simulacije
      END                      Kraj simulacije
```

Dobijeni su sledeći rezultati simulacije:

```
GPSS/FON Ver. 2.0, Simulating results
Relative clock       64800  Absolute clock       64800
Block counts
Block Current Total
  1    0    517
  2    0    517
  3    0    517
  4    0    517
  5    0    517
  6    0    517
  7    0    517
  8    0      1
  9    0      1
Queue Maximum   Average   Total    Zero    Percent   Average   Current
          contents    contents  entries  entries   zeros   time/trans contents
    1     9        1.470      517   157   30.368    183.888    0
Facility   Average    Number  Average   Seizing   Preempting
           utilisation    entries   time/tran  transact. transaction
    1    0.734    517     91.890    0         0
```

Ovde su dolasci klijenata i trajanje razgovora slučajne promenljive. Eksperimentisanje sa modelom najverovatnije će uključiti ispitivanje efekata promenljivog iznosa dolazaka, vremena opsluge, ili oba. Ako su obe slučajne promenljive generisane korišćenjem istog toka pseudoslučajnih brojeva, tada će promena bilo koje od njih uneti promenu kod obe slučajne promenljive. Zbog toga što se isti tok slučajnih brojeva koristi za uzimanje dva tipa uzoraka, promena kod iznosa dolazaka ili kod vremena opsluge uticaće na proces uzimanja

uzoraka za obe slučajne promenljive. U datom primeru, upotreba dva toka nezavisnih pseudoslučajnih brojeva (npr. RN3 i RN7) daće verodostojnije rezultate. U tom slučaju, iznos dolazaka i vreme opsluge mogu se menjati nezavisno jedno od drugog. Time se omogućava eksperimentisanje sa sistemom koji se modelira na adekvatniji način. U ovom slučaju program bi izgledao ovako:

```
* SIMULACIJA RADA JEDNE JAVNE TELEFONSKE GOVORNICE
          SIMULATE
*
EXPO1     FUNCTION RN3,C24         Eksponencijalna raspodela
          0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
          .75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
          .95,2.99/.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/0.995,5.3/
          0.998,6.2/0.999,7/0.9998,8
*
EXPO2     FUNCTION RN7,C24         Eksponencijalna raspodela
          0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
          .75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
          .95,2.99/.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/0.995,5.3/
          0.998,6.2/0.999,7/0.9998,8
*
          GENERATE 120,FN$EXPO1    Dolazak klijenata
          QUEUE 1                  Klijent pristupa redu
          SEIZE 1                  Klijent ulazi u govornicu
          DEPART 1                 Klijent napušta red
          ADVANCE 100,FN$EXPO2     Trajanje razgovora
          RELEASE 1                Klijent oslobađa govornicu
          TERMINATE                Klijent napušta sistem
*
* SEGMENT TAJMERA
*
          GENERATE 64800           Generisanje tajmera
          TERMINATE 1              Umanji TB za 1
          START 1                  Kraj simulacije
          END
```

Dobijeni su sledeći rezultati simulacije:

```
GPSS/FON Ver. 2.0, Simulating results
Relative clock       64800  Absolute clock       64800
Block counts
Block Current Total
  1    0    540
  2    4    540
  3    0    536
  4    0    536
  5    1    536
  6    0    535
  7    0    535
  8    0      1
  9    0      1
Queue Maximum  Average   Total    Zero   Percent    Average   Current
           contents  contents  entries  entries  zeros    time/trans contents
    1      15        3.031    540   104   19.259      364.284     4
Facility   Average    Number  Average   Seizing   Preempting
 utilisation   entries   time/tran  transact. transaction
    1     0.825        536     99.845       1     0
```

Upoređivanjem rezultata u oba slučaja mogu se uočiti znatne razlike, što ukazuje na značajnu statističku zavisnost kod uzimanja uzoraka u prvom slučaju.

Takođe, i kod transparentnog uzimanja uzoraka nastaje statistička zavisnost.

NPR: Vozila dolaze na tehničku kontrolu. Poslednja provera odnosi se na rad motora. Ukoliko je motor neispravan, šalje se u odeljenje za popravku. Posle popravke vozilo se vraća ponovo na kontrolu. Nakon uspešne provere vozilo ide na parkiranje. Na kontrolnom mestu, na kom se vrši kontrola rada motora, vozila pristižu svakih 10±5 minuta. Kontrolu vozila vrše dva radnika. Vreme provere vozila je 20±5 minuta. Od svih pregledanih vozila njih 70% je ispravno i odlaze na parkiranje. Ostalih 30% vozila idu u odeljenje za popravku, gde radi jedan rdnik. Vreme potrebno za popravku vozila iznosi 45±15 minuta. Potrebno je dobiti statistike redova vozila na mestu kontrole kao i na mestu popravke, pa je potrebno tabelirati: zadržavanje vozila na kontrolnom mestu, pri čemu je gornja granica prvog intervala 5 minuta, širina intervala je 5 minuta, a broj klasa je 5; zadržavanje vozila na popravci, pri čemu je gornja granica prvog intervala 5 minuta, širina intervala je 10 minuta, a broj klasa je 6. Simulirati vremenski period od 8 časova, a za vremensku jedinicu uzeti 0,1 minut.

```
* PROGRAM ZA TEHNIČKU KONTROLU VOZILA
*
RAD  STORAGE 2           Definisanje kapaciteta skladišta
TAB1 TABLE M1,100,50,5   Definisanje tabele
TAB2 TABLE M1,300,100,5  Definisanje tabele
*
     SIMULATE
     GENERATE 100,50     Vozila pristižu svakih 100±50 v.j.
KONT QUEUE RED1          Vozilo staje u red
     ENTER RAD           Radnik vrši kontrolu vozila
     DEPART RED1         Vozilo napušta red
     MARK                Resetovanje tranzitnog vremena
     ADVANCE 200,50      Vreme kontrole 200±50 v.j.
     LEAVE RAD           Radnik postaje slobodan
     TABULATE TAB1       Tabeliranje vremena kontrole
     TRANSFER .300,,OPR  Preusmeravanje vozila
     TERMINATE           Vozilo napušta sistem
OPR  QUEUE RED2          Vozilo staje u red za popravku
     SEIZE PERA          Radnik počinje sa popravkom
     DEPART RED2         Vozilo napušta red
     MARK                Resetovanje tranzitnog vremena
     ADVANCE 450,150     Vreme popravke 450±150 v.j.
     RELEASE PERA        Radnik postaje slobodan
     TABULATE TAB2       Tabeliranje vremena popravke
     TRANSFER ,KONT      Uputi vozilo na ponovnu kontrolu
*
* SEGMENT TAJMERA
*
     GENERATE 4800       Generisanje tajmera transakcije
     TERMINATE 1         Umanji TB za 1
     START 1             Početak simulacije
     END                 Kraj simulacije
```

Posmatrajući ovaj primer, uočićemo da se naredbe GENERATE i ADVANCE eksplicitno pozivaju na isti tok pseudoslučajnih brojeva RN1. Ako je potrebna statistička nezavisnost, upotreba ove osobine u više od jedne tačke u modelu mora da se izbegne, jer se u datom slučaju sve naredbe implicitno pozivaju na tok pseudoslučajnih brojeva RN1.

Da bismo izbegli statističku zavisnost u datom primeru, potrebno je da koristimo sledeći program, u kom se dobijaju verodostojniji simulacioni rezultati.

```
* PROGRAM ZA TEHNIČKU KONTROLU VOZILA
*
          SIMULATE
DOLAZ     VARIABLE (100-50)+(RN7*(2*50))/1000     Dolasci
USLUG1    VARIABLE (200-50)+(RN4*(2*50))/1000     Kontrola
USLUG2    VARIABLE (450-150)+(RN3*(2*150))/1000   Popravka
RAD       STORAGE 2                               Def.kapaciteta sklad.
TAB1      TABLE M1,100,50,5                       Definisanje tabele
TAB2      TABLE M1,300,100,5                      Definisanje tabele
*
          GENERATE V$DOLAZ                        Pristizanje vozila
KONT      QUEUE RED1                              Vozilo staje u red
          ENTER RAD                               Radnik vrši kontrolu
          DEPART RED1                             Vozilo napušta  red
          MARK                                    Reset. tranz. vremena
          ADVANCE V$USLUG1                        Vreme kontrole
          LEAVE RAD                               Radnik postaje slobodan
          TABULATE TAB1                           Tab. vr. kontrole
          TRANSFER .300,,OPR                      Preusmeravanje vozila
          TERMINATE                               Vozilo napušta sistem
OPR       QUEUE RED2                              red za popravku
          SEIZE PERA                              Radnik počinje popravku
          DEPART RED2                             Vozilo napušta red
          MARK                                    Reset. tranz. vremena
          ADVANCE V$USLUG2                        Vreme popravke
          RELEASE PERA                            Radnik postaje slobodan
          TABULATE TAB2                           Tab. vremena popravke
          TRANSFER ,KONT                          Uputi vozilo na pregled
*
* SEGMENT TAJMERA
*
          GENERATE 4800                           Gen. tajmer transakcije
          TERMINATE 1                             Umanji TB za 1
          START 1                                 Početak simulacije
          END                                     Kraj simulacije
```

Pitanje statističke nezavisnosti je od važnosti čak i u ovom prostom modelu, kada su u pitanju uniformne raspodele. Uticaj statističke nezavisnosti je znatno uočljiviji kada su u pitanju drugi oblici raspodela (npr. eksponencijalna).

LITERATURA

1. B. Radenković, M. Stanojević, A. Marković, „Računarska simulacija“, Fakultet organizacionih nauka, Saobraćajni fakultet, CIP – katalogizacija u publikaciji – narodna biblioteka Srbije, Beograd, 2004
2. M. Stanojević, „Beleške sa predavanja“, Saobraćajni fakultet, Beograd, 2005 – 2007
3. M. Stanojević, M. Đogatović, „Beleške sa vežbi“, Saobraćajni fakultet, Beograd, 2005 – 2007
4. M. Đogatović, „Beleške sa računarskih vežbi“, Saobraćajni fakultet, Beograd, 2005 – 2006
5. B. Radenković, „Slajdovi sa vežbi: simulacija na jeziku GPSS“, Fakultet organizacionih nauka, Beograd 2006.