

Urednik
ŽELIMIR ČOLIĆ, dipl. ing.

Recenzenti
Prof. dr. ŽELJKO PANIAN
Prof. dr. GABRO SMILJANIĆ
Prof. dr. VILKO ŽILJAK

Grafički urednik
ŽELJKO BRNETIĆ

Lektorica
ILIJANA MILENKOVIĆ, prof.

Korektor
SLAVKO BRNČIĆ

Tekst, crteži, fotolit
FotoSoft

Prof. dr. VLATKO ČERIĆ
izvanredni profesor Ekonomskog fakulteta
Sveučilišta u Zagrebu

SIMULACIJSKO MODELIRANJE

I. IZDANJE

Odbor za znanstveno-nastavnu literaturu Sveučilišta u Zagrebu odobrio je ovaj udžbenik rješenjem
broj 02 - 592 / 1 - 1991. od 29. svibnja 1991. godine.

CIP - Katalogizacija u publikaciji
Nacionalna i sveučilišna biblioteka, Zagreb
519.876 (075.8)
ČERIĆ, Vlatko
Simulacijsko modeliranje / Vlatko
Čerić. - Zagreb : Školska knjiga, 1993.
- 328 str. : ilustr. ; 24 cm. -
(Udžbenici Sveučilišta u Zagrebu =
Manualia Universitatis studiorum
Zagabiensis)
Bibliografija: str. 319-321. - Kazalo.
930526112

1. izdanje 1993.

Tisk: „Prosvjeta“ d.d. Bjelovar



ŠKOLSKA KNJIGA, ZAGREB, 1993.

SADRŽAJ

PREDGOVOR	13
-----------------	----

I. DIO	
OSNOVE SIMULACIJE	
1. Osnovne ideje simulacije	19
1.1. Modeliranje	19
1.2. Simulacijsko modeliranje	22
1.3. Modeliranje i računala	22
1.4. Metode modeliranja i rješavanja problema	24
1.5. Komponente simulacijskog modeliranja	24
1.6. Potreba za simulacijom	26
1.7. Korištenje simulacije u donošenju odluka	26
1.8. Simulacijski proces	27
1.9. Prednosti i nedostaci simulacije	30
2. Pristupi simulacijskome modeliranju	33
2.1. Podjele simulacijskih modela	33
2.1.1. Deterministički i stohastički modeli	33
2.1.2. Diskretni i kontinuirani modeli	34
2.2. Tipovi simulacijskih modela	36
2.2.1. Monte Carlo simulacija	36
2.2.2. Kontinuirana simulacija	38
2.2.3. Simulacija diskretnih događaja	41
2.2.4. Kombinirana diskretno-kontinuirana simulacija	44
2.3. Tipovi sistema i izbor tipa simulacijskog modela	45
2.3.1. Tipovi sistema	45
2.3.2. Izbor tipa simulacijskog modela	47
II. DIO	
SIMULACIJA DISKRETNIH DOGAĐAJA	
3. Osnovne ideje simulacije diskretnih događaja	51
3.1. Osnovni pojmovi simulacije diskretnih događaja	51
3.2. Specifični problemi koje mora rješavati mehanizam simulacije diskretnih događaja	54
3.3. Struktura alata za simulaciju diskretnih događaja	54
3.4. Tipični problemi koji se mogu rješavati simulacijom diskretnih događaja	56
3.5. Nekoliko primjena simulacije diskretnih događaja	59

4. Izgradnja konceptualnih simulacijskih modela	65	6.4.1. Vizualna interaktivna simulacija	115
4.1. Konceptualni simulacijski modeli	65	6.4.2. Objektno orijentirani pristup simulaciji	115
4.2. Dijagrami ciklusa aktivnosti	70	6.4.3. Korištenje metoda i alata umjetne inteligencije	115
4.2.1. Osnovni koncepti	70	6.4.4. Simulacija na paralelnim računalima	116
4.2.2. Način izgradnje i funkcioniranja dijagrama ciklusa aktivnosti	71		
4.3. Primjer dijagrama ciklusa aktivnosti	73	7. Simulacijski jezik GPSS	117
4.4. Korištenje varijabli u dijagramima ciklusa aktivnosti	77	7.1. Uvod	117
4.5. Potpunost prikaza simulacijskih modela pomoću dijagrama ciklusa aktivnosti	78	7.2. Osnovni koncepti jezika GPSS	117
5. Strategije izvođenja simulacije	81	7.2.1. Pregled karakteristika jezika GPSS	117
5.1. Mehanizmi pomaka vremena	81	7.2.2. Stalni entiteti	118
5.1.1. Pomak vremena za konstantni prirast	81	7.2.3. Privremeni entiteti	120
5.1.2. Pomak vremena na sljedeći događaj	82	7.2.4. GPSS događaji	120
5.2. Simulacijske strategije	82	7.2.5. Moguća stanja transakcija	121
5.2.1. Pojam simulacijske strategije	82	7.2.6. GPSS upravljački mehanizam	121
5.2.2. Strategija planiranja događaja	84	7.3. Osnovne naredbe jezika GPSS	125
5.2.3. Strategija prelaženja aktivnosti	84	7.3.1. Naredbe upravljanja	125
5.2.4. Strategija međudjelovanja procesa	87	7.3.2. Naredbe za definiranje blokova	126
5.3. Trofazna strategija simulacije	87	7.3.3. Definicije entiteta	133
5.4. Ručna simulacija pomoću trofazne metode	91	7.3.4. Atributi	136
5.5. Ručna simulacija jednog problema	92	7.3.5. Indirektno adresiranje	136
5.6. Skupljanje statističkih podataka o izvođenju simulacijskih eksperimenata	97	7.3.6. Tip izlaznih rezultata	138
5.6.1. Osnovni podaci od interesa za analizu rezultata simulacije	97	7.4. Modeliranje i simulacija nekoliko jednostavnih problema jezikom GPSS	139
5.6.2. Primjer skupljanja statističkih podataka simulacije	100		
5.7. Planiranje budućih događaja	103	8. Stvaranje povjerenja u simulacijske modele	149
5.7.1. Važnost planiranja budućih događaja u simulaciji diskretnih događaja	103	8.1. Provjera ispravnosti simulacijskih modela	149
5.7.2. Metode planiranja budućih događaja	105	8.2. Osnovne pretpostavke procesa stvaranja povjerenja u simulacijski model	150
5.7.3. Usporedba metoda za planiranje budućih događaja	106	8.3. Verifikacija računarskog modela	151
5.7.4. Korištenje metoda za planiranje budućih događaja u simulacijskim jezicima	107	8.4. Vrednovanje konceptualnog modela	152
6. Izgradnja simulacijskih programa	109	8.4.1. Replikativno vrednovanje modela	153
6.1. Osnovne klasifikacije softvera za simulaciju	109	8.4.2. Strukturno vrednovanje modela	154
6.1.1. Prevodioci i interpretatori	109	8.4.3. Prediktivno vrednovanje modela	154
6.1.2. Opći i specijalni jezici	110		
6.1.3. Korištenje simulacijskih strategija	110	9. Osnovni elementi vjerojatnosti i statistike	155
6.1.4. Softver za mikroračunala	110	9.1. Slučajne varijable i njihova svojstva	155
6.2. Pristupi izgradnji simulacijskih programa	111	9.1.1. Diskretne slučajne varijable	156
6.2.1. Korištenje općih programskih jezika	111	9.1.2. Kontinuirane slučajne varijable	158
6.2.2. Biblioteke procedura	111	9.2. Uzorci i statističko zaključivanje o svojstvima populacije	160
6.2.3. Simulacijski jezici s opisom naredbi	112	9.2.1. Stvaranje slučajnog uzorka	160
6.2.4. Simulacijski jezici za izgradnju scenarija	112	9.2.2. Statističko zaključivanje o populaciji na temelju uzorka	160
6.2.5. Interaktivni generatori simulacijskih programa	113	9.3. Statističko zaključivanje o sredini i varijanci	162
6.2.6. Generički simulacijski programi pokretani podacima	114	9.3.1. Točkaste procjene sredine i varijance	162
6.3. Kriteriji izbora simulacijskog softvera	114	9.3.2. Intervali pouzdanosti za sredinu	163
6.4. Razvojne tendencije simulacijskog softvera	115	9.3.3. Testovi hipoteza za sredinu	165
10. Generiranje uzoraka	167		
10.1. Korištenje slučajnih brojeva u simulacijskim eksperimentima	167		
10.2. Što su slučajni brojevi	167		

10.3. Kako se mogu generirati slučajni brojevi	168	12.3.2. Antitetske varijable	226
10.3.1. Fizički pribori	168	12.3.3. Selektivni uzorci	227
10.3.2. Korištenje iracionalnih brojeva	168		
10.3.3. Aritmetički procesi	169		
10.4. Linearni kongruentni generatori	170		
10.5. Testiranje generatora slučajnih brojeva	171	13. Analiza izlaza simulacijskih eksperimenata	229
10.5.1. Test frekvencija	172	13.1. Osnovne ideje analize izlaza simulacijskih eksperimenata	229
10.5.2. Test serija	172	13.1.1. Uvod	229
10.5.3. Test porasta	172	13.1.2. Tipovi simulacijskih eksperimenata	231
10.5.4. Testovi autokorelacije	172	13.1.3. Mjere performansi sistema	233
10.6. Slučajne varijable i njihovo generiranje	175	13.2. Analiza izlaznih podataka jednog sistema	234
10.7. Jednostavne transformacije slučajne varijable	175	13.2.1. Intervali pouzdanosti za terminirajuće simulacije	235
10.8. Metoda inverzne transformacije	176	13.2.2. Intervali pouzdanosti za stacionarne simulacije	237
10.8.1. Kontinuirane slučajne varijable	176	13.2.3. Određivanje ostalih mjera performansi	239
10.8.2. Diskretne slučajne varijable	179	13.2.4. Višestruke mjere performansi sistema	239
10.9. Metoda prihvaćanja i odbijanja	182	13.3. Usporedba alternativnih sistema	240
10.10. Generiranje nekih značajnih razdioba	183	13.3.1. Usporedba dvaju sistema	240
		13.3.2. Usporedba većeg broja sistema	241
		13.4. Modeliranje veze između ulaza i izlaza simulacije	242
11. Analiza ulaznih podataka	185		
11.1. Tretman ulaznih podataka simulacijskog modela	185	III. DIO	
11.2. Korisne razdiobe vjerojatnosti	186	SISTEMSKA DINAMIKA	
11.2.1. Teorijske razdiobe	186		
11.2.2. Empirijske razdiobe	193	14. Osnovne ideje sistemske dinamike	245
11.3. Skupljanje ulaznih podataka	195	14.1. Sistemi s povratnom vezom	245
11.4. Postavljanje hipoteze o porodici razdioba	196	14.2. Osnovne karakteristike sistema s povratnom vezom	247
11.4.1. Kontinuirane razdiobe	196	14.2.1. Nivoi	247
11.4.2. Diskretne razdiobe	198	14.2.2. Brzine	247
11.5. Procjena parametara razdioba	199	14.2.3. Kašnjenja	248
11.6. Testovi slaganja	201	14.2.4. Konstrukcija modela	248
11.6.1. Neformalna vizualna procjena	201	14.3. Pristup sistemske dinamike	248
11.6.2. Hi-kvadrat test	201	14.4. Područja primjene i ciljevi upotrebe sistemske dinamike	250
11.6.3. Kolmogorov-Smirnov test	202		
11.7. Izbor razdiobe u nedostatku podataka	203		
11.8. Analiza veza među podacima	204	15. Konceptualni modeli sistemske dinamike	251
11.8.1. Jednostavna linearna regresija	205	15.1. Dijagrami uzročnih petlji	251
11.8.2. Uključivanje međudjelovanja nezavisnih varijabli u model ..	208	15.1.1. Način opisa i osnovni pojmovi	251
11.8.3. Korištenje regresijskih modela ulaznih podataka u simulaciji	208	15.1.2. Određivanje tipa petlje	251
12. Planiranje simulacijskih eksperimenata	211	15.1.3. Pozitivne povratne petlje	253
12.1. Osnovne ideje planiranja simulacijskih eksperimenata	211	15.1.4. Negativne povratne petlje	254
12.2. Dizajn simulacijskih eksperimenata	212	15.1.5. Povezivanje pozitivnih i negativnih povratnih petlji	256
12.2.1. Potpuni 2^k faktorski dizajn	213	15.2. Dijagрами toka	257
12.2.2. Djelomični 2^{k-p} faktorski dizajn	217	15.2.1. Način opisa i osnovni pojmovi	257
12.2.3. Traženje optimalnih kombinacija faktora metodologijom površine odziva	218	15.2.2. Primjeri dijagrama toka	259
12.2.4. Faktorski dizajn sa slučajnim varijablama	221	15.2.3. Razvoj dijagrama toka na osnovi dijagrama uzročnih petlji	260
12.3. Tehnike redukcije varijance	224		
12.3.1. Zajednički slučajni brojevi	224	16. Simulacijske jednadžbe, programi i jezik DYNAMO	263
		16.1. Računarski modeli sistemske dinamike	263
		16.2. Opisivanje promjene vremena u jednadžbama modela	264

16.3. Tipovi jednadžbi modela	265
16.4. Modeliranje kašnjenja materijala	266
16.4.1. Eksponencijalna kašnjenja prvog reda	267
16.4.2. Eksponencijalna kašnjenja višeg reda	269
16.4.3. Način označavanja eksponencijalnih kašnjenja	271
16.4.4. Izbor reda eksponencijalnog kašnjenja	271
16.5. Modeliranje kašnjenja informacija	272
16.6. Primjeri DYNAMO-programa	273
16.7. Sintaksa i dodatne naredbe jezika DYNAMO	274
16.7.1. Sintaksa jezika DYNAMO	274
16.7.2. Naredbe TABLE, STEP i PULSE	275
16.7.3. Naredbe za izlaz i specifikaciju trajanja simulacije	277
16.7.4. Naredbe za izvođenje simulacije	278
16.7.5. Primjer potpunog DYNAMO programa	278
16.8. Izbor veličine vremenskog koraka modela	278
16.9. Ostali programski jezici sistemske dinamike	279
16.10. Matematičke osnove sistemske dinamike	280
16.10.1. Diferencijalne i diferencijske jednadžbe	280
16.10.2. Numeričke metode rješavanja diferencijalnih jednadžbi	283
16.10.3. Rješavanje sistema diferencijalnih jednadžbi	285
16.10.4. Rješavanje diferencijalnih jednadžbi višeg reda	286
16.10.5. Pogreške numeričkog rješavanja diferencijalnih jednadžbi	287
16.10.6. Izbor optimalnog koraka numeričkog proračuna	288
16.10.7. Stabilnost numeričkih rješenja diferencijalnih jednadžbi	288
16.10.8. Matematičko modeliranje i sistemsko modeliranje	289
16.11. Stohastička sistemsko modeliranje	291
16.12. Optimizacija u sistemskoj dinamici	291
16.13. Simulacijski proces sistemsko modeliranje	292
16.14. Sistemsko modeliranje i diskretna simulacija: sličnosti, razlike i veze	294
17. Neke klase problema koje se mogu modelirati sistemskom dinamikom	297
17.1. Dinamika populacija	297
17.1.1. Jedna jedinstvena populacija	297
17.1.2. Stratificirana populacija	299
17.1.3. Međusobna ovisnost više populacija	301
17.2. Dinamika zatvorenog ciklusa života	303
17.3. Modeli procesa odlučivanja	305
17.3.1. Upravljanje vodostajem u branama	305
17.3.2. Upravljanje nivoom zaliha u skladištu	308
17.4. Zagadivanje okoline	309
Bibliografija	311
Najznačajnije knjige	311
Knjige objavljene kod nas	316
Časopisi, društva i kongresi	317
Literatura	319
Indeks pojmova	323

PREDGOVOR

Simulacijsko modeliranje jedna je od vodećih suvremenih metoda modeliranja uz pomoć računala. Ova metoda omoguće opis, razumijevanje i kvantitativnu analizu složenih dinamičkih sistema u područjima proizvodnje, transporta, ekonomije, masovnog posluživanja, računarstva itd.

Svojim značajkama i mogućnostima primjene simulacijsko modeliranje pripada području *operacijskog istraživanja* (analiza alternativa i pomoći odlučivanju), *statistike* (generiranje slučajnih brojeva i varijabli, planiranje i analiza eksperimenta) i *računarske znanosti* (programske aspekti, korištenje resursa računala). Metodološke sličnosti s prirodnim znanostima ogledaju se u eksperimentalnom karakteru provjere hipoteza, te korištenju ove metode za razumijevanje uzroka počinjanja sistema koji se ispituju. Posljednjih godina simulacijsko modeliranje doživljava buran razvoj zahvaljujući dostupnosti mikroračunala, računarske grafike, korištenju metoda umjetne inteligencije, te primjeni i korištenju objektno orijentiranog programiranja i paralelnih računala.

Cilj teksta

Cilj je ovog teksta da jasno i pregledno prikaže osnovne ideje i mogućnosti simulacijskog modeliranja, i to u oba osnovna tipa simulacijskog modeliranja: *simulaciji diskretnih događaja* te *kontinuiranoj simulaciji*, koja je ovdje prikazana pomoću modela *sistemsko modeliranje*. Odabran je pristup u kojem se za većinu komponenata simulacijskog procesa ukratko opisuju raspoložive metode, a zatim se odabire jedna od metoda koja se potanki razrađuje i opisuje njezinu funkcioniranje u karakterističnom primjeru. Da bi se što više olakšalo razumijevanje opsežne, složene i raznovrsne materije, tekst je popraćen mnogobrojnim primjerima te grafičkim prikazima koji ilustriraju metode i primjere rješavanja problema. Za poticanje praktičnog rada na računalu u tekstu su opisani simulacijski jezici GPSS (diskretna simulacija) i DYNAMO (sistemska dinamika).

Vjerujemo da će ovaj tekst omogućiti da se u razumnom vremenu dostigne razumijevanje prilično složene i raznovrsne metodologije simulacijskog modeliranja i njezina korištenja u modeliranju i analizi složenih dinamičkih problema. Čitalac će također shvatiti kakve su opasnosti *ad hoc* pisanja vlastitih simulacijskih programa, korištenja simulacijskog modeliranja samo na osnovi priručnika za neki od simulacijskih jezika, odnosno tretiranja simulacijskog modeliranja kao isključivo programske zadatke bez poznавanja ostalih komponenata simulacijskog procesa.

Objašnjavajući osnovne ideje simulacijskog modeliranja i primjenjujući ih na prikladnim primjerima, ovaj tekst ujedno omoguće prijelaz na detaljnije studiranje različitih mehanizama simulacije, simulacijskih jezika, primjene statističkih metoda u si-

mulacijskom procesu i povezivanje simulacijskog modeliranja sa suvremenim metodama i alatima računarske znanosti. U tome mu pomaže i popratna bibliografija s popisom i sažetkom sadržaja i ocjenom kvalitete dvadesetak odabralih knjiga iz tog područja, te popis časopisa koji su specijalizirani za simulacijsko modeliranje.

Kome je tekst namijenjen

Tekst je namijenjen studentima, stručnjacima koji izvode simulacijske studije te istraživačima iz ovog područja. Za studente posebnu važnost ima sistematičnost izlaganja, korištenje velikog broja primjera i ilustracija, te opis i demonstracija najviše korištenih programskih jezika u diskretnoj simulaciji i sistemskoj dinamici. Stručnjaci koji izvode simulacijske studije u ovom će tekstu naći sve elemente simulacijske studije te pregled i opis alternativnih metoda i alata koji se primjenjuju u studijama.

Za istraživače može biti posebno važna sistematizacija različitih pristupa simulacijskom modeliranju, prikaz obju glavnih metoda simulacijskog modeliranja (diskretne simulacije i sistemsko dinamike), opis nekih otvorenih problema i suvremenih pristupa simulaciji, te pregled sadržaja i ocjena odabralih knjiga publiciranih u ovom području. Ako im osim toga pruži i neke nove metodološke spoznaje, nepoznate referencije i, najvažnije od svega, hranu za razmišljanje, autor ovog teksta bit će prezadovoljan.

Upotreba u nastavi

Ovaj tekst nastao je za potrebe i kao rezultat autorovih predavanja na kolegiju simulacijskog modeliranja na Ekonomskom fakultetu u Zagrebu. On može biti osnova jednosemstarskog ili dvosemstarskog *dodiplomskog kolegija* iz simulacijskog modeliranja za studente informatičke, organizacijske, poslovne i inženjerske orientacije, odnosno za studente računarskih znanosti.

Tekst također može biti osnova jednosemstarskog *postdiplomskog kolegija* iz naprednoga simulacijskog modeliranja, u kojem su posebno važni dijelovi poglavlja 2, 5, 6, 7, 10, 11, 12, 13, 16 i 17. Tekst se može upotrijebiti kao dio kolegija operacijskih istraživanja.

Praktični rad na računalu pomaže opis dvaju ključnih jezika simulacijskog modeliranja, GPSS (diskretna simulacija) i DYNAMO (sistemska dinamika). Potanko su opisani osnovni elementi tih jezika i popraćeni većim brojem primjera. Za oba ta jezika postoje verzije za mikroračunala.

O tekstu

Tekst je podijeljen na tri dijela. U prvom dijelu prikazane su *osnovne ideje simulacije*, proces rješavanja problema pomoću simulacijskog modeliranja i osnovni tipovi simulacijskih modela. U drugom dijelu prikazana je *simulacija diskretnih dogadaja*. Taj dio uključuje opis osnovnih ideja diskretnе simulacije, tipova problema koji se ovom metodom rješavaju, strategija izvođenja simulacije te svih faza simulacijskog procesa i metoda koje se u njima primjenjuju. Modeliranje i simulacija uz pomoć računala de-

monstrirana je simulacijskim jezikom GPSS. U trećem dijelu prikazana je *sistemska dinamika*. Taj dio uključuje opis osnovnih ideja sistemskе dinamike, sistema s povratnom vezom, konceptualnih modela sistemskе dinamike, matematičkih aspekata sistemskе dinamike, te opis nekoliko klase problema od šireg interesa koji se mogu modelirati sistemskom dinamikom. Modeliranje i simulacija uz pomoć računala demonstrirana je simulacijskim jezikom DYNAMO.

Simulacijsko modeliranje je preširoko područje a da bi jedan čovjek mogao u jednakoj mjeri poznavati sve metode i tehnike koje se u njemu koriste. Stoga je za ovakav tekst bilo nužno konzultirati širok spektar literature. Doprinos autora je u kompoziciji cijele materije, određivanju strukture njezinih dijelova i izboru metoda i tehnika koje autor smatra metodološki ili s praktične strane najznačajnijima. Sistematisacija nekih dijelova teksta (modeliranje, tipovi simulacijskog modela, izbor tipa simulacijskog modela, konceptualni modeli sistemskе dinamike, matematičke osnove sistemskе dinamike, klase problema koje se mogu modelirati pomoću sistemskе dinamike), opis nekih metoda (konceptualni modeli u diskretnoj simulaciji, koncepti i način izgradnje dijagrama ciklusa aktivnosti, usporedba procesa sistemskе dinamike i diskretne simulacije), prikaz simulacijskih studija na kojima je autor radio, te niz primjera i grafičkih ilustracija su originalni.

Zahvale

Za pojavljivanje teksta u ovom obliku zahvalan sam i svojim kolegama i suradnicima koji su uložili znatan napor te pregledali tekst i dali niz izvanredno korisnih sugestija i primjedaba. Mr. Vesna Hljuz Dobrić iz Sveučilišnog računskog centra u Zagrebu pregledala je poglavlja vezana za statistički tretman simulacije, a mr. Jadranka Božikov iz Škole narodnog zdravlja "Andrija Štampar" u Zagrebu poglavlja iz dijela o sistemskоj dinamici. Prof. dr. Mladen Alić s Prirodoslovno-matematičkog fakulteta u Zagrebu pregledao je dio teksta o sistemskоj dinamici povezan uz diferencijalne i diferencijske jednadžbe. Zahvaljujući tome korigiran je veći broj grešaka, tekst je terminološki usklađen i dodani su novi dijelovi o matematičkom modeliranju i numeričkom rješavanju diferencijalnih jednadžbi. Mr. Vlatka Hlupić s Ekonomskog fakulteta u Zagrebu unijela je u računalo velik dio prve verzije ovog teksta i pažljivo pretražila greške u cijelom tekstu.

Odgovornost za sve preostale semantičke i sintaktičke greške u tekstu ostaje naravno na autoru, koji će biti zahvalan čitateljima koji ga upozore na bilo koje nedostatke.

Ovaj tekst ne bi bio moguć da nije bilo strpljivosti moje obitelji tijekom premnogih sati rada na njemu. Posebno sam zahvalan supruzi Lidiji i sinu Ivanu za velik trud uložen u pripremi crteža.

Zagreb, u veljači 1991.

I. DIO

OSNOVE SIMULACIJE

1. OSNOVNE IDEJE SIMULACIJE

U ovom poglavlju prikazane su osnovne ideje modeliranja i simulacije te značenje korištenja računala u modeliranju. Opisani su različiti oblici modela i razlozi njihova korištenja u prirodnim znanostima te inženjerstvu i ekonomiji. Naglašene su specifičnosti simulacijskog modeliranja vezane za potrebu prikaza modela koji se mijenjaju u vremenu te uz eksperimentalnu prirodu simulacije. Simulacija je stavljena i u kontekst različitih metoda modeliranja i rješavanja problema, te je pokazana veza među osnovnim komponentama simulacijskog modeliranja.

Istaknuti su razlozi zbog kojih se pojavljuje potreba za korištenjem simulacijskog modeliranja i prikazano značenje ove metode u donošenju odluka. Opisan je simulacijski proces, tj. niz koraka od kojih se sastoji proces rješavanja problema uz pomoć simulacijskog modeliranja. Na kraju su sažete dobre i loše strane simulacijskog modeliranja kao metode rješavanja složenih dinamičkih problema.

1.1. MODELIRANJE

Modeliranje je proces blisko vezan za način ljudskog razmišljanja i rješavanja problema. Mentalni modeli su strukture koje ljudski mozak neprekidno konstruira kako bi mogao povezati niz činjenica s kojima se čovjek susreće, te kako bi mogao na temelju toga djelovati. Takvi modeli omogućuju na primjer razumijevanje fizičkog svijeta, komunikaciju među ljudima i planiranje akcija.

Istraživanje prirode i razvoj tehnologije doveli su do stvaranja opipljivijih i formalnijih vrsta modela, i to zbog potrebe da se omogući opisivanje složenih fenomena da se oni mogu preciznije opisivati i rješavati, da se omogući duže trajanje modela i da modeli mogu postati sredstvo za komunikaciju većeg broja ljudi koji se bavi nekim fenomenom.

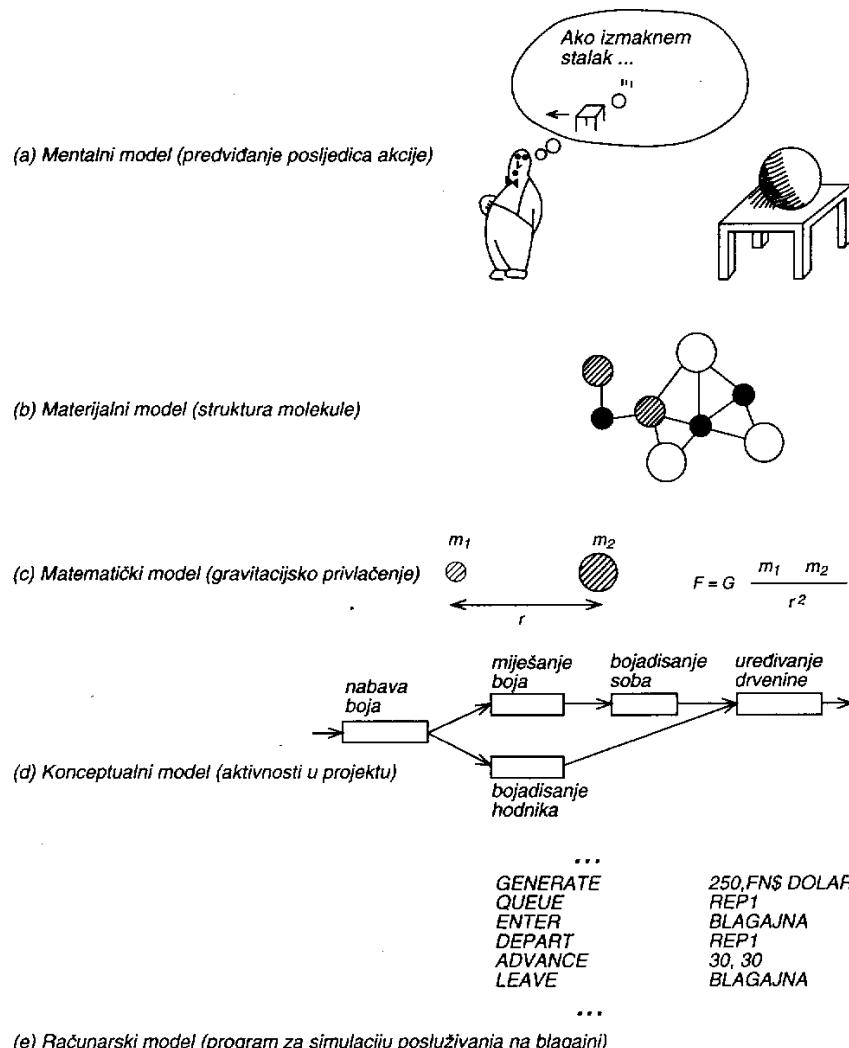
U istraživanju prirode modeli trebaju omogućiti razumijevanje strukture i funkcionaliranja prirode, te su oni oruđe za postavljanje i dokazivanje hipoteza. Tako je, na primjer, Newtonov zakon gravitacije model za razumijevanje gravitacijskog međudjelovanja tijela u svemiru, a ujedno omogućuje i točno predviđanje gibanja tijela u gravitacijskom polju.

U inženjerstvu i ekonomiji modeli imaju posebnu važnost u oblikovanju i ispitivanju obilježja novih rješenja koja se često i ne mogu drukčije ispitivati. Tako model složenoga proizvodnog pogona može pomoći određivanju broja i vrste strojeva koji omogućuju određen proizvodni kapacitet pogona uz najmanja ulaganja.

Jedna od osnovnih uloga modela danas je formuliranje novih ideja. Možda najpoznatija primjena modeliranja kao sredstva za formuliranje i ispitivanje znanstvenih

hipoteza jest otkrivanje strukture DNA molekule, ključne molekule u genetici, tijekom kojeg su se James Watson i Francis Crick intenzivno koristili različitim oblicima modela. Za svoje su otkriće 1962. godine dobili Nobelovu nagradu (Watson, 1968).

Modeli mogu biti *materijalni* (npr. model kemijske strukture molekule) ili *simbolički*. Simbolički modeli su: *matematički* (npr. diferencijalna jednadžba njihala), *koncepcionalni* (npr. dijagram logike povezanosti aktivnosti u projektu) ili *računarski* (npr. program za računalno koji opisuje proračun protoka kroz prometnu mrežu). Na slici 1.1. prikazani su primjeri različitih vrsta modela.



Slika 1.1. Različite vrste modela

Modeli koji će nas najviše zanimati u ovom tekstu jesu konceptualni i računarski modeli. *Konceptualni modeli* se stvaraju na temelju predodžbe o strukturi i logici rada sistema ili problema koji se modelira, i prikazuju se u obliku čije je značenje precizno definirano, npr. kao dijagram koji se koristi točno definiranim simbolima. Takav prikaz omogućuje da se modeli vizualiziraju, da se mogu prikazati složeni modeli, da se njima može baratati te da služe kao sredstvo za komunikaciju među ljudima koji njima rade. Već i sama izgradnja konceptualnog modela znači važan korak prema boljem razumijevanju problema koji rješavamo. Napokon, konceptualni su modeli osnova za izradu računarskih modela, a danas se oni (npr. u obliku dijagrama) već mogu izravno razvijati i upotrebljavati na računalima.

Računarski modeli su prikaz konceptualnih modela u obliku programa za računalo. U tom obliku modeli postaju sredstvo kojim se može analizirati rad modela u različitim vanjskim uvjetima te s različitim unutrašnjim parametrima, i tako dobiti uvid u razumijevanje sistema koji model opisuje te mogućnost predviđanja njegova ponašanja. Računarski se modeli koriste programskim jezicima kao svojim sredstvom izražavanja, te su stoga blisko vezani za razvoj računarskih znanosti.

Treba naglasiti da za proces izrade modela nema striktnih pravila, već veliko značenje ima zdrav razum, sposobnost apstrakcije, sistematičnost i iskustvo. Stoga je modeliranje prije svega umijeće, a ne znanost. Ono se ne može automatizirati, i vrlo je važno da se radi pažljivo te da se što više i temeljiti provjerava. Dugo iskustvo velikog broja ljudi koji su se time bavili dovelo je do nekih *opcih preporuka pri izradi modela* (Gordon, 1969):

- (1) granica sistema s okolinom mora biti odabrana tako da sistem, odnosno njegov model, obuhvaća samo fenomene od interesa. Okolina sistema modelira se tako da se ne opisuju detalji fenomena i uzročna veza među njima, već se daje samo njihov sažeti prikaz (npr. slučajna razdioba dolazaka u sistem);
- (2) modeli ne smiju biti suviše složeni ni detaljni, već trebaju sadržati samo relevantne elemente sistema – suviše složene i detaljne modele gotovo nije moguće vrednovati ni razumjeti, što znači da su i njihov razvoj i korištenje teški i neizvjesne kvalitete;
- (3) model ne smije ni suviše pojednostavljiti problem, npr. izbacivanjem važnih varijabli potrebnih za adekvatan opis sistema ili suviše velikim stupnjem agregiranja komponenti sistema;
- (4) model je razumno rastaviti na više dobro definiranih i jednostavnih modula s točno određenom funkcijom, koje je lakše i izgraditi i provjeriti;
- (5) u razvoju modela preporučuje se korištenje neke od provjerenih metoda za razvoj algoritama i programa. Tako se npr. u metodi profinjenja u koracima (tzv. pristup "odozgo prema dolje") polazi od prve verzije modela koja se sastoji od nekoliko modula s više funkcija, zatim se u sljedećem koraku ti moduli rastavljaju na nekoliko jednostavnijih modula itd., sve dok se ne dosegne željeni stupanj detaljnosti modela. Pri tome je moguće razumjeti model i njegove module u svim fazama razvoja modela;
- (6) potrebna je provjera logičke i kvantitativne ispravnosti modela, i to kako pojedinačnih modula, tako i cijelog modela. Kod modela koji uključuju slučajne varijable to znači i primjenu odgovarajućih statističkih tehniki.

U ovom ćemo tekstu opisati, na primjeru simulacijskih modela, kako se izgrađuju konceptualni i računarski modeli, kavim se alatima izgrađuju, kako se ispituje njihova ispravnost te kako se planiraju, izvode i analiziraju eksperimenti s računarskim modelima.

1.2. SIMULACIJSKO MODELIRANJE

Simulacijski modeli su modeli dinamičkih sistema, tj. sistema koji se mijenjaju u vremenu. Mi ćemo se uglavnom baviti modelima koji se ne mogu opisati ni rješavati matematičkim sredstvima, i koji su u obliku konceptualnih i računarskih modela. Njihove tipične primjene su u područjima inženjerstva i ekonomije. To su, na primjer, modeli repova čekanja, proizvodnih procesa, skladištenja i prometa.

Simulacijski modeli moraju ponajprije omogućiti ispravan prikaz i efikasno izvođenje *pomaka vremena*. Također je bitno omogućavanje *istovremenog* odvijanja aktivnosti, te opisivanje procesa koji *konkuriraju* za iste resurse (npr. strojeve, blagajne ili finansijska sredstva). Ti su zahtjevi znatan problem za modeliranje, zbog čega su se simulacijski modeli razvili u posebnu kategoriju modela. Razvijeni su i specifični alati za konceptualno modeliranje i specifični programski jezici kao adekvatna sredstva za posebne zahtjeve simulacijskog modeliranja.

Kod primjene simulacijskog modeliranja ne može se dobiti rješenje u analitičkom obliku, u kojem su zavisne varijable funkcije nezavisnih varijabli (slika 1.2a), već se rješenje problema dobiva eksperimentiranjem modelom sistema. Pri tome simulacijski eksperimenti daju kao rezultat *skup točaka*, tj. vrijednosti zavisnih varijabli za pojedine vrijednosti nezavisnih varijabli (slika 1.2b). Zbog *slučajnog* karaktera varijabli modela dobiva se čak i više različitih vrijednosti zavisnih varijabli za istu vrijednost nezavisnih varijabli, tj. eksperimenti daju određeni *uzorak* vrijednosti zavisnih varijabli (slika 1.2c). Pri tome planiranje i analiza simulacijskih eksperimenata zahtijevaju *statistički* pristup.

1.3. MODELIRANJE I RAČUNALA

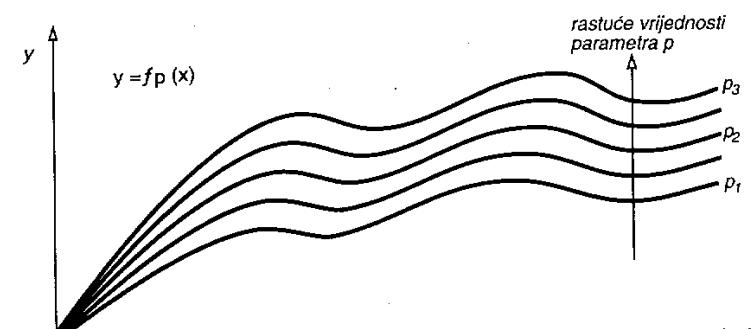
Suvremeno modeliranje nezamislivo je bez računala. Računala, uz različite metode i programske alate računarske znanosti, omogućuju dobar ambijent za stvaranje složenih modela i efikasan rad s njima.

Računala se u modeliranju koriste u dvije različite svrhe: u razvoju modela i u izvođenju proračuna na temelju stvorenog modela (npr. proračuna promjene modela u vremenu kod simulacijskih modela). Kod *razvoja modela* koristi se računalo kao stroj koji manipulira simbolima — tako je moguće manipulirati i grafičkim objektima, memorirati ih pa čak i transformirati u druge oblike. Na temelju razvijenih konceptualnih modela moguće je automatski generirati odgovarajuće računarske modele (programe).

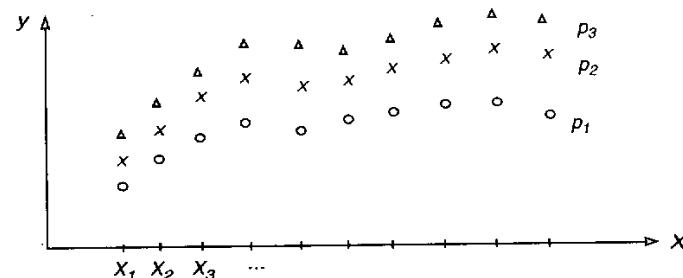
Izvođenje proračuna razvijenim računarskim modelom (programom) koristi se brzinom rada računala kao odlučujućim faktorom koji omogućuje da se u razumnom razdoblju izvedu raznovrsne analize alternativnih struktura modela i promjenljivih vanjskih uvjeta.

Modeliranje korištenjem računala tako postaje disciplina koja može adekvatno i efikasno prikazivati složene sisteme, te oblikovati i ispitivati njihovo ponašanje.

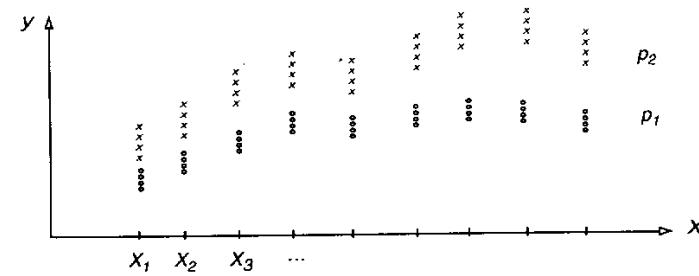
Specifičnosti *simulacijskog modeliranja* računalo također veoma dobro podržava. Grafički prikaz sistema koji se modelira (npr. tlocrt proizvodnog procesa) i animacija rada sistema tijekom izvođenja simulacijskih eksperimenata omogućuju lakše vrednovanje logike i dinamike rada simulacijskih modela te lakše praćenje razvoja modela u vremenu, što je posebno zanimljivo za korisnike simulacijskog modeliranja. Automatsko generiranje programa dovodi do značajnog skraćivanja vremenskog ciklusa



(a) Analitičko rješenje (funkcijska ovisnost)



(b) Numeričko rješenje determinističkog problema



(c) Numeričko rješenje sistema sa slučajnim varijablama

Slika 1. 2. Tipovi rješenja dinamičkih problema

razvoja modela i uvođenja izmjena u model, pa stoga i omogućuje duže ukupno vrijeme za analizu rada modela i ispitivanje većeg broja alternativa u usporedbi s klasičnim pisanjem programa. Velika brzina rada računala ujedno omogućuje da se kod modela sa slučajnim varijablama izvede veći broj eksperimenata kako bi se stvorili uzorci potrebne veličine.

Osim toga, simulacijsko modeliranje se sve više povezuje za baze podataka te suvremene koncepte i alate umjetne inteligencije: baze znanja, ekspertne sisteme, jezike Prolog i Lisp, obradom prirodnog jezika i slično. Kao posljedica toga, simulacijsko modeliranje pruža sve veće mogućnosti u formuliranju i rješavanju složenih problema i sve je pristupačnije za korištenje.

1.4. METODE MODELIRANJA I RJEŠAVANJA PROBLEMA

Prije razrade osnova simulacijskog modeliranja, napraviti ćemo kratak pregled različitih tipova modela i metoda rješavanja problema, kako bi se vidjele specifičnosti simulacijskog modeliranja u usporedbi s ostalim metodama modeliranja.

Prema rastućoj složenosti sistema, odnosno problema koji modeliramo, metode modeliranja i rješavanja problema jesu:

(1) Analitičko modeliranje

Modeli su u *analitičkom obliku* (npr. sistemi algebarskih ili diferencijalnih jednadžbi), pa su i rješenja u *analitičkom obliku* (funkcijske veze zavisnih o nezavisnim varijablama).

To su modeli problema koji se svode na matematički tretman pomoću metoda algebre, matematičke analize, teorije vjerojatnosti i sl. Primjer takvih problema su jednostavni problemi njihala i repova čekanja.

(2) Numeričke metode

Modeli su u *analitičkom obliku*, ali se zbog nemogućnosti nalaženja analitičkog rješenja rješavaju *numeričkim postupcima* (tj. nalaženjem parova vrijednosti nezavisnih i zavisnih varijabli koji zadovoljavaju zadane jednadžbe modela).

Primjer problema koji se rješavaju numeričkim metodama (najčešće različitim iterativnim postupcima) složeniji su problemi repova čekanja, difuzije, vremenske prognoze itd.

(3) Simulacijski modeli

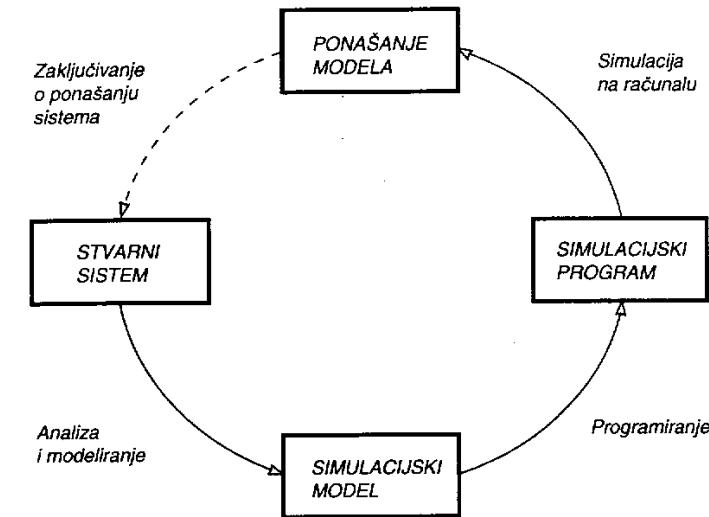
Zbog nemogućnosti prikaza složenih dinamičkih sistema u analitičkom obliku, modeli su zadani u *proceduralnom obliku* kojim se prikazuje način rada sistema. Problem se rješava numerički, provodenjem *eksperimentata* modelom koji *oponašaju razvoj sistema u vremenu*.

Primjer sistema koji se modeliraju i analiziraju simulacijskim modeliranjem su *diskretni sistemi*, npr. proizvodni procesi i transportni sistemi, te *kontinuirani sistemi* s povratnom vezom, npr. iskorištavanje resursa na Zemlji i dinamika promjene populacija biljaka i životinja.

Vidimo da simulacijski modeli značajno proširuju klasu problema dostupnih za rješavanje. Cijena toga je povećanje složenosti izgradnje modela i rješavanja problema.

1.5. KOMPONENTE SIMULACIJSKOG MODELIRANJA

Opisati ćemo osnovne komponente simulacijskog modeliranja i dati njihove kratke radne definicije (Zeigler, 1976). Osnovne komponente simulacijskog modeliranja i njihove relacije prikazane su na slici 1.3.



Slika 1.3. Sistemi, modeli, simulacija (prema Zeigler, 1976)

Osnovne komponente:

Sistem, odnosno skup komponenata koje djeluju zajednički kako bi ostvarile zadani cilj ili funkciju. Zbog međudjelovanja komponenata sistema nastaje evolucija poнаšanja sistema u vremenu.

Model (konceptualni model), odnosno formalni apstraktни prikaz sistema. Model prikazuje strukturu sistema, komponente sistema i njihovo međudjelovanje.

Program (računarski model), odnosno detaljan opis strukture i načina rada modela.

Računalo, odnosno računski proces (čovjek ili stroj) koji na temelju instrukcija programa i ulaznih podataka generira razvoj modela u vremenu.

Operacije na komponentama:

Analiza i modeliranje, odnosno analiza strukture i načina rada sistema te predstavljanje sistema u formalnom apstraktnom obliku.

Programiranje, odnosno detaljan prikaz modela u obliku pogodnom za rad na računalu.

Simulacija, odnosno izvođenje instrukcija programa na računalu, čime se oponaša razvoj sistema u vremenu.

Kao što smo vidjeli, simulacijsko modeliranje je veoma vezano za metode i alate *računarske znanosti*. No simulacijsko modeliranje je takođe orijentirano i na rješavanje problema, pa je tako blisko vezano za ideje i alate *operacijskog istraživanja*. Zbog postojanja slučajnih varijabli u simulacijskim modelima, koriste se pristupi i metode *teorije vjerojatnosti i statistike*. Osim toga, eksperimentalna priroda metode rješavanja problema u simulacijskom modeliranju ima analogije s empiričkom prirodnom istraživanjem u *prirodnim znanostima*, budući da se u oba područja koristi planiranje, izvođenje i analiza eksperimentata radi razumijevanja rada sistema koji se ispituje, odnosno provjere postavljenih hipoteza.

1.6. POTREBA ZA SIMULACIJOM

Ima više razloga za korištenje simulacijskog modeliranja u rješavanju problema (Naylor i dr., 1966):

- (1) eksperiment s realnim sistemom može biti skup ili čak nemoguć (npr. u kemijskim postrojenjima ili ekonomskim sistemima),
- (2) analitički model nema analitičkog rješenja (npr. modeli složenijih sistema repova čekanja),
- (3) sistem može biti suviše složen da bi se mogao opisati analitički (npr. sa sistemom diferencijalnih jednadžbi).

Simulacijsko modeliranje je osobito zanimljiva metoda za modeliranje i analizu suvremenih tehnoloških i ekonomskih sistema, i to zbog:

- (1) složene strukture i načina rada sistema,
- (2) dinamike i nestacionarnosti rada sistema (npr. promjene gustoće prometa na cestama tijekom dana),
- (3) postojanja slučajnih varijabli (npr. vrijeme posluživanja na šalterima),
- (4) upravljanja radom sistema (npr. u proizvodnim procesima).

1.7. KORIŠTENJE SIMULACIJE U DONOŠENJU ODLUKA

Simulacijsko modeliranje koristi se za rješavanje više različitih tipova problema važnih za donošenje odluka:

- (1) razumijevanje sistema ili problema, odnosno nalaženje glavnih faktora utjecaja u sistemu,
- (2) oblikovanje sistema, odnosno sintetiziranje i vrednovanje predloženih rješenja (npr. usporedbu alternativnih struktura i tehnologija rada sistema),
- (3) analizu rada sistema, odnosno određivanje radnih karakteristika sistema, uskih grla sistema, kapaciteta sistema i sl.,
- (4) sinkronizaciju i fino podešavanje rada sistema, analizu utjecaja kvarova na rad sistema te traženje odgovarajuće organizacije rada sistema,
- (5) predviđanje ponašanja sistema u budućnosti, odnosno prognoziranje efekata promjene vanjskih uvjeta i unutrašnjih parametara sistema,
- (6) trening donosioca odluka (tzv. igre za rukovodioce), koji uključuje razvoj simulacijskih modela u kojima odluke donose "igrači" koji zastupaju suprotstavljene ekipе (npr. konkurentna poduzeća u modelu). Model proračunava posljedice tih odluka, koje tako utječu na daljnji razvoj modelirane situacije. Takve igre rukovodicima daju iskustvo u usporedbi strategija rada i procjeni efekata pogrešnih odluka u dinamičnom i kompetitivnom ambijentu sličnom stvarnom svijetu, i to bez gubitaka koji bi se pokazali da se iskustvo stječe u stvarnom svijetu. Posebna kategorija modela su modeli ratnih igara koji simuliraju posljedice odluka u strategiji vođenja bitaka.

Simulacijsko modeliranje igra značajnu ulogu u procesu donošenja odluka zbog složenosti problema koje može opisati i rješavati, raznovrsnih načina upotrebe modela te lakoće i pogodnosti rada koje pružaju mikroračunala, animacija simulacije, suvremeni način rada pomoću menija, prozora i sl. Prednost simulacijskog modeliranja prema

ostalim načinima modeliranja je i sličnost među strukturom i načinom rada problema i modela, što daje dodatno povjerenje korisnicima simulacije.

Tri su osnovna tipa simulacijskih modela koji se koriste u donošenju odluka (Davies i O'Keefe, 1989):

(1) Modeli za jednokratnu upotrebu

Modeli za jednokratnu upotrebu koriste se samo jedanput i mogu npr. služiti za donošenje odluke o izboru opreme koja se kupuje, oblikovanju novog sistema i sl. Podaci se skupljaju najčešće također samo za jednu upotrebu.

(2) Modeli za dugoročnu upotrebu

Modeli za dugoročnu upotrebu pomažu pri donošenju odluka periodičnog karaktera, npr. u rukovođenju prometom, financijskom odlučivanju itd. Ti modeli traže točne podatke koji se stalno ažuriraju. Simulacijski modeli često su integrirani s cjelokupnim procesom odlučivanja, od prikupljanja podataka, njihove analize i spremanja do različitih vrsta analiza s rezultatima dobivenih simulacijom i drugim oblicima modeliranja. Takvi integrirani sistemi zovu se i *sistemi za podršku odlučivanju* (slika 1.4).

(3) Modeli troškova

Niz ulaznih i izlaznih varijabli može se povezati za troškove. To su npr. troškovi nabave opreme, radne snage ili gubitaka neispunjene narudžabe. Model troškova može se ili uključiti u strukturu simulacijskog modela ili se pojavljuje kao poseban model koji služi za analizu rezultata simulacije.

1.8. SIMULACIJSKI PROCES

Simulacijski je proces *struktura rješavanja stvarnih problema* pomoću simulacijskog modeliranja. On se može prikazati u obliku niza koraka koji opisuju pojedine faze rješavanja problema ovom metodom. Struktura simulacijskog procesa nije strogo sekvenčijalna, jer je moguć i povratak na prethodne korake procesa, zavisno od rezultata dobivenih u pojedinim fazama procesa.

Valja imati na umu i to da prije donošenja odluke o rješavanju nekog problema pomoću simulacijskog modeliranja treba ispitati mogućnost korištenja jednostavnijih i jeftinijih metoda modeliranja (kao teorije repova, matematičkog programiranja i sl.).

Osnovni su koraci simulacijskog procesa (Law i Kelton, 1982) ovi (slika 1.5):

(1) Definicija cilja simulacijske studije

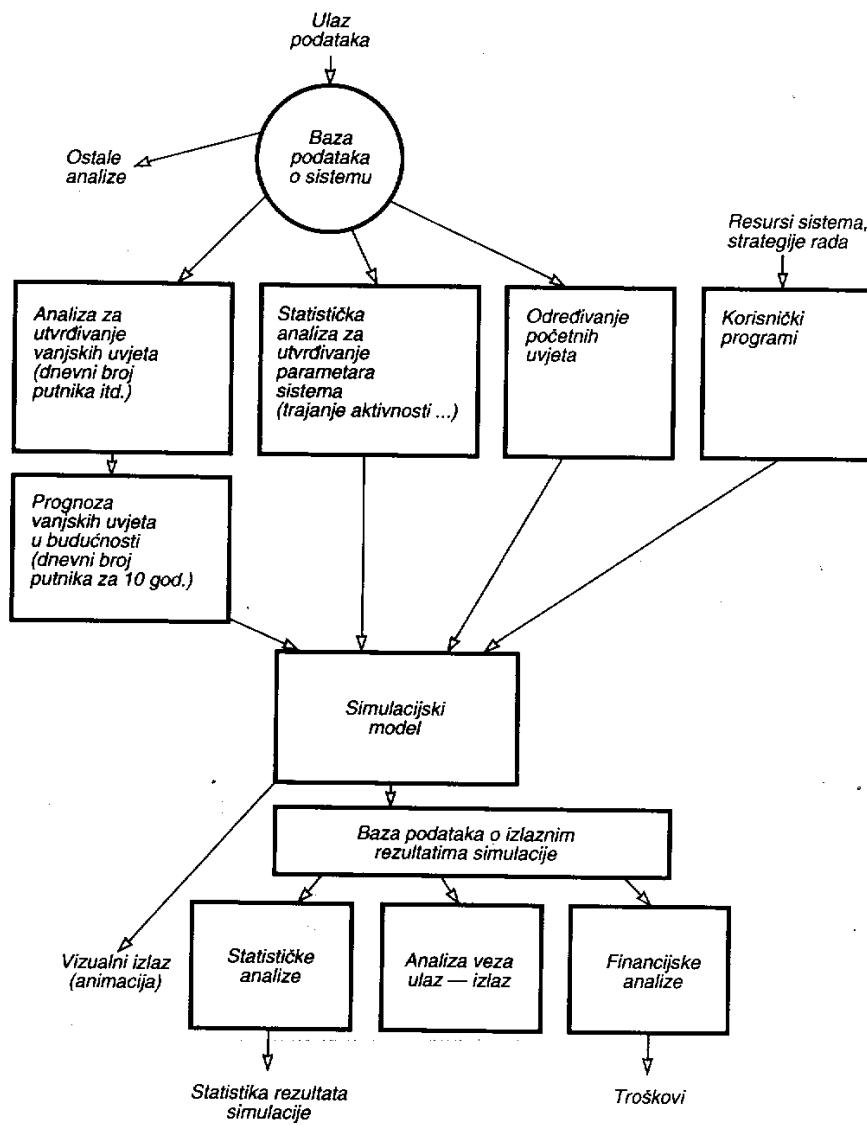
Definicija željenog cilja i svrhe studije: problem koji treba riješiti (oblikovanje sistema, analiza sistema i sl.), granice sistem/okolina, nivo detaljnosti.

(2) Identifikacija sistema

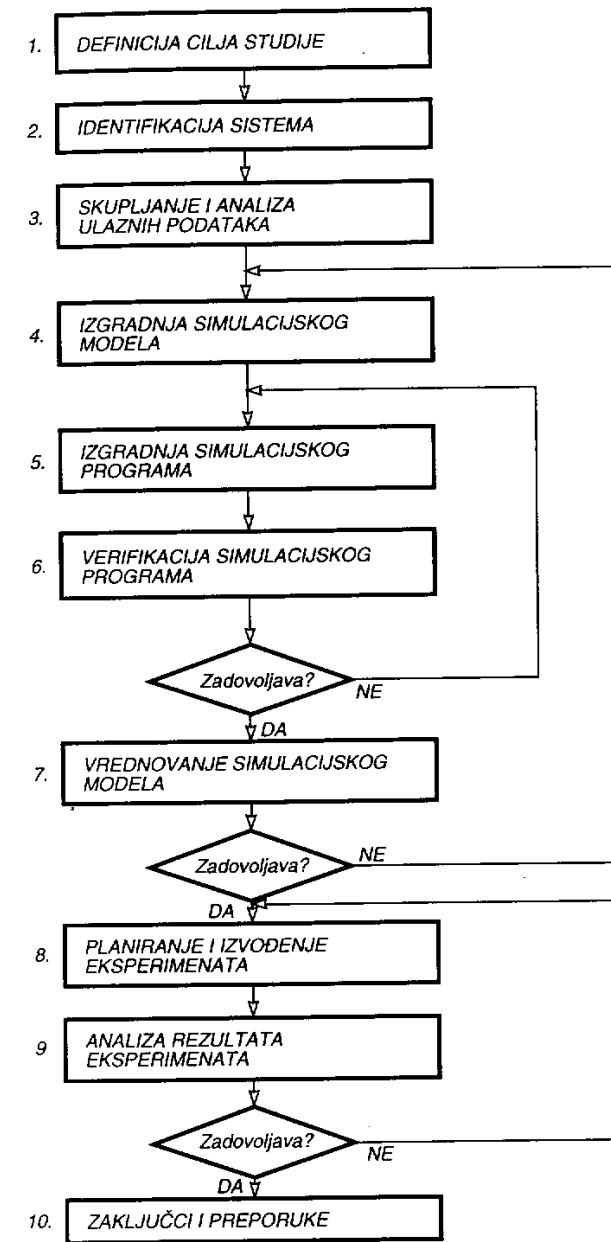
Opis komponenata sistema, interakcija komponenata, način rada, veze s okolinom, formalni prikaz sistema.

(3) Skupljanje podataka o sistemu i njihova analiza

Skupljanje i mjerjenje relevantnih podataka o sistemu, analiza tih podataka (izbor razdioba nezavisnih slučajnih varijabli, ocjena vrijednosti parametara razdioba).



Slika 1. 4. Simulacijski model kao dio sistema za podršku odlučivanju
(prema Davies i O'Keefe, 1989)



Slika 1. 5. Simulacijski proces

(4) Izgradnja simulacijskog modela

Stvaranje konceptualnog modela koji adekvatno opisuje sistem i omogućuje rješavanje zadalog problema.

(5) Izgradnja simulacijskog programa

Izbor programske jezike ili paketa, te stvaranje simulacijskog programa (računarskog modela) bilo pisanjem programa, bilo automatskim generiranjem programa na osnovi konceptualnog modela.

Tijekom rada na fazama 4 i 5 može se pokazati potreba za dopunom koraka 3 (skupljanje i analiza dodatnih podataka o sistemu).

(6) Verificiranje simulacijskog programa

Testiranje simulacijskog programa u odnosu prema postavkama simulacijskog modela (pojedinačne procedure, generiranje slučajnih varijabli, povezivanje procedura).

Ako verifikacija programa nije zadovoljila, potreban je povratak na točku 5 (izmjene u simulacijskom programu).

(7) Vrednovanje simulacijskog modela

Ispitivanje da li simulacijski model adekvatno predstavlja stvarni sistem (ispitivanjem podudaranja izlaza modela i stvarnog sistema, analizom rezultata od eksperata, analizom osjetljivosti).

Ako vrednovanje modela nije zadovoljilo, potreban je povratak na točku 4 (izmjene u simulacijskom modelu).

(8) Planiranje simulacijskih eksperimenata i njihovo izvođenje

Planiranje simulacijskih eksperimenata koji omogućuju ispunjenje cilja studije (plan mijenjanja parametara modela te ponavljanje eksperimenata zbog analize utjecaja slučajnih varijabli). Izvođenje simulacijskih eksperimenata prema odabranom planu eksperimenata.

(9) Analiza rezultata eksperimenata

Statistička analiza rezultata simulacijskih eksperimenata (izbor tipa razdiobe slučajnih zavisnih varijabli, ocjene parametara razdioba).

Tijekom analize rezultata eksperimenata može se pokazati potreba za dopunom koraka 8 (izvođenje dodatnih eksperimenata).

(10) Zaključci i preporuke

Prezentacija relevantnih rezultata na temelju kojih se mogu donijeti odgovarajuće odluke (izbor konfiguracije sistema, izmjene sistema, organizacija rada).

U dalnjem će tekstu detaljno biti opisani svi koraci simulacijskog procesa.

1.9. PREDNOSTI I NEDOSTACI SIMULACIJE

Na kraju ovog poglavlja sažet ćemo prednosti i nedostatke primjene simulacijskog modeliranja u rješavanju stvarnih problema (Law i Kelton, 1982; Schmidt i Taylor, 1970).

Prednosti simulacijskog modeliranja:

(1) moguće je opisati i rješavati složene dinamičke probleme sa slučajnim varijablama koji su nedostupni matematičkom modeliranju,

(2) moguće je riješiti raznovrsne probleme (oblikovanje, analiza rada, predviđanja itd.),

(3) uvjeti eksperimentiranja su pod potpunom kontrolom, za razliku od eksperimenata sa stvarnim sistemom gdje nije moguće utjecati npr. na dinamiku stizanja narudžaba ili brzinu rada šaltera,

(4) vrednovanje i analiza logike i dinamike rada sistema veoma su olakšani animacijom rada modela.

Nedostaci simulacijskog modeliranja:

(1) razvoj modela je i dug i skup (iako ga dosta ubrzava automatsko generiranje programa kada je dostupno),

(2) zbog statističkog karaktera simulacije potrebno je izvođenje većeg broja simulacijskih eksperimenata kako bi se dobio odgovarajući uzorak rezultata simulacije, a već i pojedinačno izvođenje eksperimenta može zahtijevati dosta vremena i memorije računala,

(3) ne dobivaju se zavisnosti izlaznih varijabli o ulaznim varijablama modela ni optimalna rješenja,

(4) za ispravno korištenje simulacijskog modeliranja potrebno je poznавanje više različitih metoda i alata (iako razvoj ekspertnih sistema povezanih simulacijskim modelima obećava da će nadomjestiti bar dio tih znanja),

(5) vrednovanje modela je dosta složeno i zahtijeva dodatne eksperimente.

2. PRISTUPI SIMULACIJSKOME MODELIRANJU

Simulacijsko modeliranje je područje bogato raznovrsnim pristupima modeliranju sistema. Prikladnost pojedinih opisa ovisi o vrsti sistema koji modeliramo, podrobnosti opisa sistema te o vrsti problema koje želimo riješiti.

Osnovne podjele tipa modela su na (1) determinističke ili stohastičke modele, zavisno o predvidljivosti zbivanja u modelu, te na (2) kontinuirane, diskrette ili mješovite modele, zavisno o načinu na koji se stanje modela mijenja u vremenu. Osim toga, u ovom poglavlju opisujemo osnovne tipove simulacijskih modela: Monte Carlo simulaciju, kontinuiranu simulaciju, simulaciju diskretnih događaja i miješanu kontinuirano-diskretnu simulaciju, te područja njihove primjene. Napokon, opisana je veza tipa sistema i tipa modela.

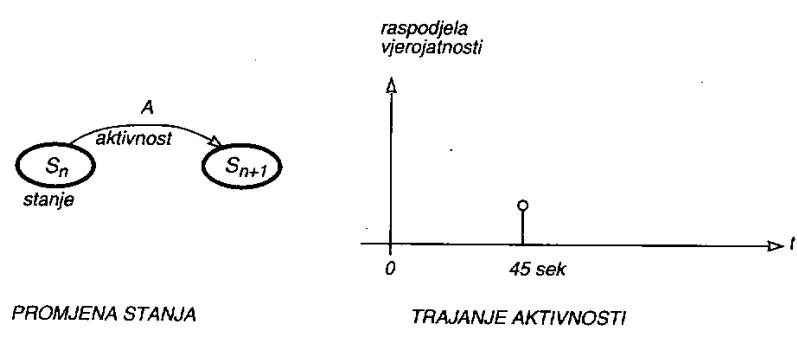
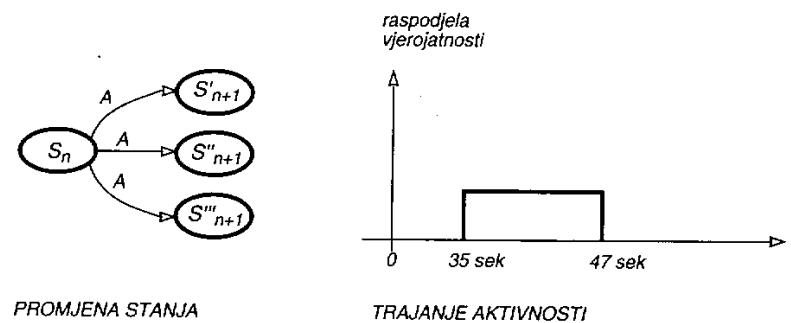
2.1. PODJELE SIMULACIJSKIH MODELA

Dva su osnovna tipa podjele simulacijskih modela, jedan prema vrsti varijabli u modelu a drugi prema načinu na koji se stanje modela mijenja u vremenu.

2.1.1. Deterministički i stohastički modeli

Deterministički modeli su oni čije je ponašanje potpuno predvidivo, tj. u kojima je novo stanje sistema koji je modeliran u potpunosti određeno prethodnim stanjem. Na slici 2.1a prikazan je deterministički model u kojem se stanje sistema S_n promjenilo pod utjecajem aktivnosti A u stanje S_{n+1} (Graybeal i Pooch, 1980). Prikazana je i aktivnost A s determinističkim trajanjem od 42 sekunde, npr. trajanje obradbe na automatiziranom stroju.

Stohastički modeli su oni čije se ponašanje ne može unaprijed predvidjeti, ali se mogu odrediti vjerojatnosti promjena stanja sistema. Dakle, stohastičke modele karakterizira slučajno ponašanje, odnosno postojanje slučajnih varijabli u sistemu. Na slici 2.1b prikazan je stohastički model u kojem se stanje sistema S_n može promjeniti u jedno od stanja S_{n+1} , S_{n+1}^1 ili S_{n+1}^2 pod utjecajem aktivnosti A. Tako neki stroj može nakon izvođenja operacije ostati ili u ispravnom stanju ili se pokvariti, pri čemu svako od tih stanja ima neku vjerojatnost (zavisnu o vrsti i starosti stroja).

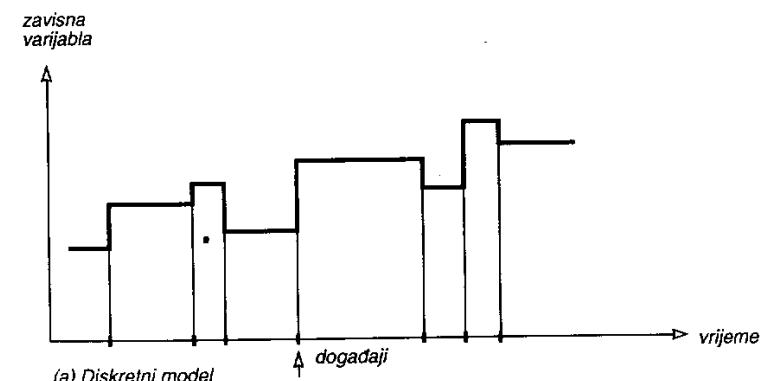
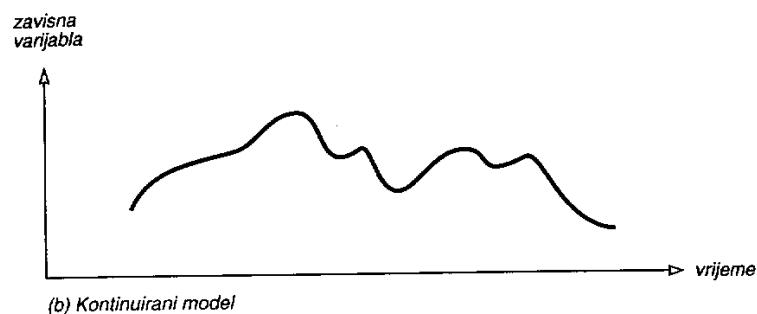
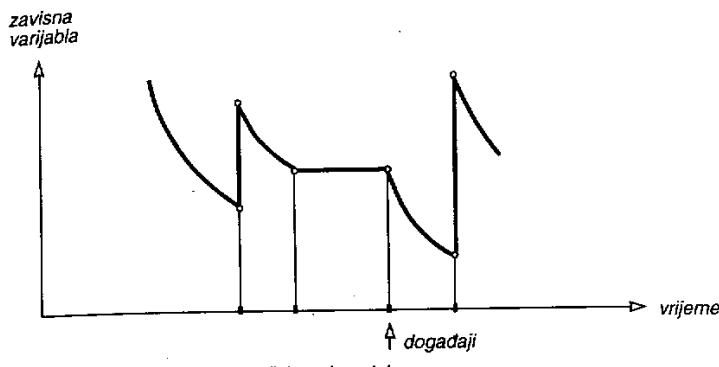


Slika 2.1. Deterministički i stohastički modeli (prema Graybeal i Pooch, 1980)

Drugi je izvor slučajnosti, prikazan na istoj slici, aktivnost sa slučajnim trajanjem, tj. s trajanjem koje se određuje iz uniformne razdiobe vjerojatnosti. Primjer takve aktivnosti je točenje goriva na benzinskoj crpki, gdje se mjerenjem određenog uzorka točenja goriva vozilima dobivaju podaci iz kojih se može odrediti razdioba vjerojatnosti trajanja te aktivnosti.

2.1.2. Diskretni i kontinuirani modeli

U diskretnim modelima stanje sistema mijenja se samo u nekim vremenskim točkama. Tako promjene zovu se dogadaji. Diskretna (tj. diskontinuirana) promjena stanja prikazana je na slici 2.2a. Tako model prodavaonice sa samoposluživanjem može sadržati varijable koje opisuju broj posjetilaca u repovima pred blagajnama, a taj broj može se mijenjati samo u času dolaska posjetilaca u rep i u času početka posluživanja na blagajni. Isto tako, cijelo posluživanje na blagajni karakterizira se samo časom početka i završetka posluživanja.



Slika 2. 2. Diskretni, kontinuirani i mješoviti simulacijski modeli (prema Pritsker, 1984)

U *kontinuiranim modelima* varijable stanja mijenjaju se kontinuirano u vremenu, kao što je prikazano na slici 2.2b. Primjer kontinuirane promjene je let aviona čiji se položaj i brzina mijenjaju kontinuirano u vremenu. Treba imati na umu da se na digitalnim računalima ne mogu izvoditi kontinuirane promjene veličina, već se one moraju aproksimirati brojevima od konačno mnogo znamenki. Već i zbog ograničenja u ukupnom vremenu izvođenja simulacije mora se kontinuirani tok vremena zamjeniti pomakom vremena u malim odsjećima.

Mogući su i *miješani kontinuirano-diskretni modeli* koji sadrže i kontinuirane i diskrette varijable. Promjena stanja takvog modela prikazana je na slici 2.2c. Primjer takvog modela je model tankera s naftom koji dolaze u luku gdje se njihov sadržaj pretače u rezervoar. Kontinuirane varijable su nivoi nafte u tankerima i rezervoaru, a diskretni događaji su npr. dolazak tankera u luku i dnevno vrijeme otvaranja i zatvaranja procesa pretakanja nafte u rezervoar.

2.2. TIPOVI SIMULACIJSKIH MODELA

Prikazane podjele simulacijskih modela dovele su do formuliranja četiriju osnovnih tipova simulacijskih modela, koji se razlikuju kako pristupom modeliranju i tipu problema koji se njima rješavaju, tako i tehnikama modeliranja i simulacije koje su za njih razvijene. To su Monte Carlo simulacija, kontinuirana simulacija, simulacija diskretnih događaja i miješana kontinuirano-diskretna simulacija (Law i Kelton, 1982; Kreutzer, 1986; Evans, 1988). Svi su ti tipovi simulacije, osim Monte Carlo simulacije, dinamički. Monte Carlo simulacija, iako je statička tehnika simulacije, spomenuta je ovdje kako zbog toga što su u njoj tretman slučajnih događaja i generiranje slučajnih vrijednosti bliski onom iz simulacije diskretnih događaja, tako i stoga da čitalac bude svjestan razlike ovog tipa simulacije u odnosu prema ostalim tipovima.

2.2.1. Monte Carlo simulacija

Monte Carlo simulacija (statistička simulacija), kao što joj i ime kaže, povezana je slučajnim fenomenima. Zanimljivo je da je to jedna od prvih primjena programiranja računala. Metoda je razvijena u toku drugoga svjetskog rata u Los Alamosu za rješavanje složenih problema vezanih za proizvodnju atomske bombe, poput proračuna raspršenja neutrona na atomskoj jezgri. Nema potpune suglasnosti o korištenju tog termina. Neki autori Monte Carlo simulacijama zovu bilo koju vrstu programa što se koriste slučajnim brojevima. Kao u većini literature o simulacijskom modeliranju, i u ovom tekstu će se taj termin upotrijebiti samo za *statičke* tipove simulacije kod kojih se u rješavanju problema koristi stvaranje uzoraka iz razdioba slučajnih varijabli. Pri tome problemi mogu biti bilo determinističkoga, bilo stohastičkoga karaktera.

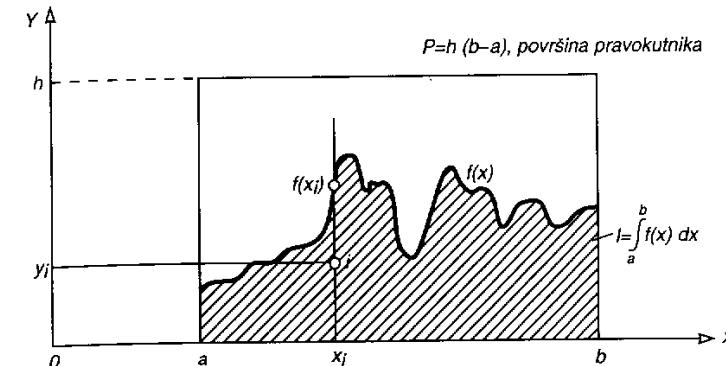
Razlikujemo ove tipove primjene Monte Carlo simulacije (Kleijnen, 1974).

(1) Deterministički problemi koje je teško ili skupo rješavati

Tipičan primjer ovog tipa je računanje vrijednosti određenih integrala koji se ne mogu rješiti analitički, tj. čija je podintegralna funkcija takva da se ne može naći rješenje u obliku matematičkog izraza. Na slici 2.3. prikazan je primjer računanja jednostrukog integrala

$$I = \int_a^b f(x) dx$$

2.2. TIPOVI SIMULACIJSKIH MODELA



Slika 2.3. Računanje određenog integrala Monte Carlo simulacijom

Monte Carlo simulacija pristupa proračunu integrala tako da se generira niz slučajnih točaka jednakе vjerojatnosti unutar pravokutnika (slika 2.3), te da se za svaku točku ispita da li je pala unutar površine koja odgovara vrijednosti integrala. Točke se generiraju tako da se generira dvojka vrijednosti (x_i, y_i) , time da se vrijednost x_i generira uniformnim generatorom slučajnih brojeva koji s jednakim vjerojatnostima daje svaku vrijednost u intervalu (a, b) a vrijednost y_i uniformnim generatorom za interval $(0, h)$. Zatim se ispita da li je točka generirana unutar površine koja odgovara integralu, tj. da li je $y_i \leq f(x_i)$. Ako je generirano n točaka od kojih je m palo unutar podintegralne površine, tada je približna vrijednost integrala jednaka

$$I = \frac{m}{n} P = \frac{m}{n} h(b - a)$$

Najčešće se ovom metodom rješavaju višestruki integrali tj. integrali s podintegralnom funkcijom koja se 'slabo ponaša', odnosno ima diskontinuitete i sl., i koja se klasičnim numeričkim metodama teško rješava ili zahtjeva korištenje previše vremena računala.

Kao što ćemo vidjeti, pristup pomoću generiranja slučajnih brojeva analogan je onom koji se koristi kod simulacije diskretnih događaja.

(2) Složeni fenomeni koji nisu dovoljno poznati

Druga klasa problema koji se rješavaju Monte Carlo simulacijom su fenomeni koji nisu dovoljno poznati da bi se mogli precizno opisati. Umjesto poznавanja načina međudjelovanja elemenata poznate su samo vjerojatnosti ishoda međudjelovanja, koje se u Monte Carlo simulaciji koriste za izvođenje niza eksperimenta što daju uzorce mogućih stanja zavisnih varijabli. Statističkom analizom takvih uzoraka dobiva se razdiob vjerojatnosti zavisnih varijabli od interesa. Ovim se pristupom najčešće analiziraju društveni ili ekonomski fenomeni poput rasta populacija, ekonomskih predviđanja ili analize rizika odlučivanja.

(3) Statistički problemi koji nemaju analitičkog rješenja

Statistički problemi bez analitičkog rješenja jedna su klasa od širokih klasa problema kod koje se koristi Monte Carlo simulacija. Njima npr. pripadaju procjene kritičnih vrijednosti ili snaga testiranja novih hipoteza. U rješavanju problema također se koristi generiranje slučajnih brojeva i varijabli.

Kod usporedbe različitih metoda regresije Monte Carlo simulacija služi za generiranje ulaznih podataka, koji se zatim analiziraju različitim regresijskim metodama što daju procjene parametara regresije tih podataka. Budući da su ulazni podaci generirani s nekim unaprijed odabranim parametrima, moguće je uspoređivati kvalitetu različitih metoda regresije po točnosti procjena parametara regresije koje one daju s poznatim parametrima pomoću kojih su podaci generirani.

2.2.2. Kontinuirana simulacija

Kontinuirana simulacija koristi se za dinamičke probleme kod kojih se varijable stanja mijenjaju kontinuirano u vremenu. Dvije su osnovne klase problema koji se rješavaju ovom metodom.

U prvoj su klasi *razmjerno jednostavnii* problemi koji su opisani *vrlo detaljno*, i kod kojih su promjene 'glatke' i prirodno se opisuju diferencijalnim jednadžbama. To su tipično problemi u području fizike, biologije i inženjerstva. U drugoj klasi su problemi što nastaju opisom *veoma složenih sistema u agregiranom obliku*, u kojem se niz elemenata sistema reducira na manji broj komponenti te u kojima se promjene u sistemu aproksimiraju *konstantnim brzinama promjene*. To su najčešće problemi iz područja ekonomije i društvenih znanosti.

Tri su osnovna tipa kontinuiranih simulacijskih modela (Aburdene, 1988).

(1) Sistemi običnih diferencijalnih jednadžbi

To su jednadžbe s jednom nezavisnom varijablom (x) po kojoj se deriviraju zavisne varijable (y_i) čija je brzina promjene (dy_i/dx) opisana u odnosu prema nezavisnoj varijabli.

Primjer problema koji dovodi do običnih diferencijalnih jednadžbi je automobilski kotač prikazan na slici 2.4a (Gordon, 1969). Na kotač mase M djeluje sila $F(t)$ koja se mijenja u vremenu. Kotač je izvješen na opruzi čija je sila proporcionalna njegovom produženju ili skraćenju, a udarce sile F apsorbira pneumatski mehanizam čija je sila prigušivanja proporcionalna ubrzaju mase kotača M . Jednadžba gibanja kotača opisana je linearom diferencijalnom jednadžbom:

$$M = \frac{d^2x}{dt^2} + D \frac{dx}{dt} + Kx = KF(t)$$

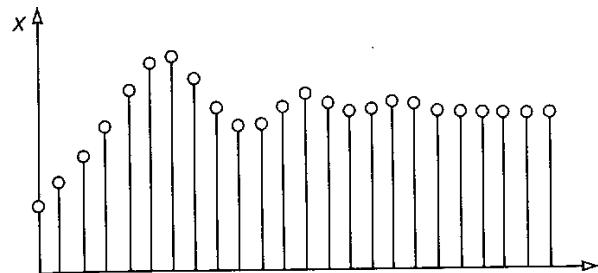
Ovakav model može se rješiti analitički, ali je to dosta rijetko kod stvarnih problema. Tako se uvođenjem realnih karakteristika u model automobilskog kotača, npr. modeliranjem ograničenja kod gibanja opruge, dovodi do nelinearnih diferencijalnih jednadžbi koje se u pravilu ne mogu rješiti analitički. U tom slučaju preostaju numenički postupci rješavanja: numeričke metode koje određuju razvoj varijabli u vremenu su metode kontinuirane simulacije.

Rješavanje problema kontinuiranom simulacijom izvodi se pomakom vremena za zadani vremenski interval, i rješavanja statičkog sistema jednadžbi u točkama pomaka vremena. Za to je razvijeno više simulacijskih jezika, a najpoznatiji su CSMP, CSSL i ESL. Na slici 2.4b prikazan je primjer rezultata simulacije opisanog modela automobilskog kotača.

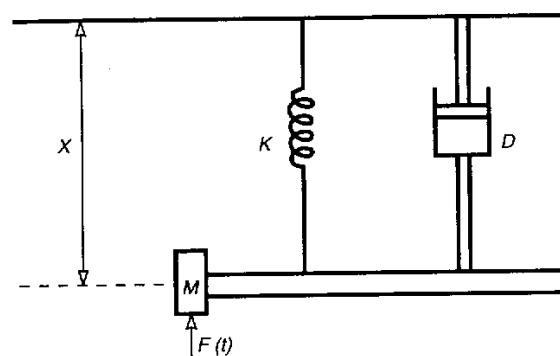
(2) Sistemi parcijalnih diferencijalnih jednadžbi

Parcijalne diferencijalne jednadžbe sadrže više od jedne nezavisne varijable (x_j) po kojima se deriviraju zavisne varijable. Tipični primjeri problema koji se mogu opisati

2.2. TIPOVI SIMULACIJSKOG MODELA



(b) Rješenje dinamike gibanja kotača



(a) Model kotača

Slika 2.4. Kontinuirana simulacija suspenzije automobilskog kotača
(prema Gordon, 1969)

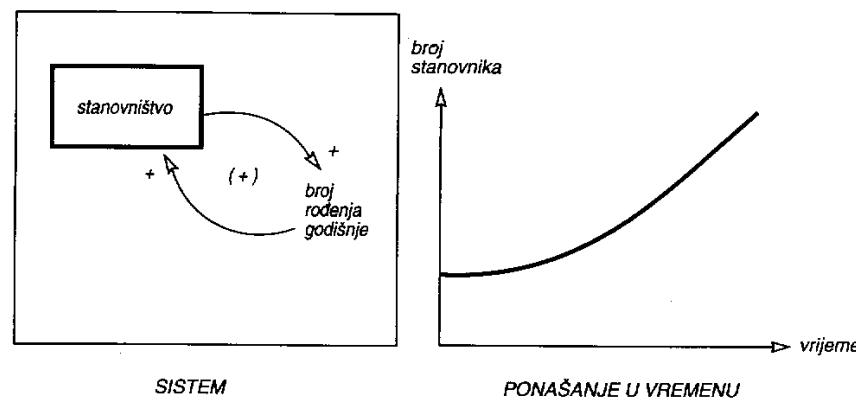
parcijalnim diferencijalnim jednadžbama su problemi aerodinamike, hidrodinamike i meteorologije. U meteorologiji se, na primjer, opisuju promjene pritiska i temperature (zavisne varijable) u tri prostorne dimenzije i vremenu (nezavisne varijable). U tom području razvijen je niz specifičnih postupaka rješavanja problema i odgovarajućih paketa programa (PDEL, PDELAN itd.). U ovom se tekstu tim područjem nećemo baviti.

Sistemi parcijalnih diferencijalnih jednadžbi zovu se još i *sistemi s distribuiranim parametrima*, za razliku od sistema običnih diferencijalnih i diferencijskih (s konačnim razlikama) jednadžbi koje se zovu *sistemi s grupiranim parametrima* (engleski 'lumped-parameter systems').

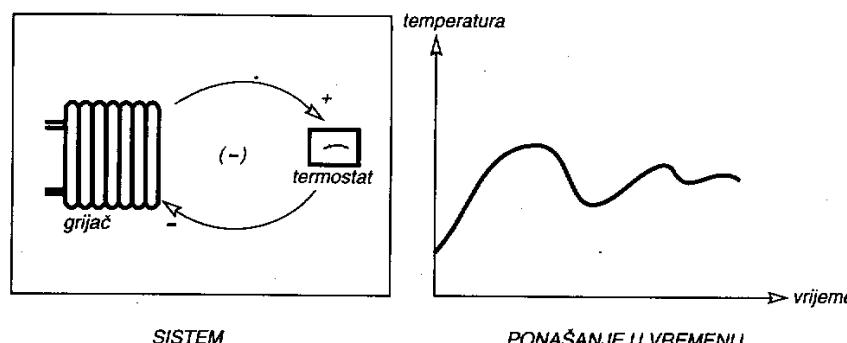
(3) Sistemska dinamika

Sistemska je dinamika simulacija *sistema s povratnom vezom*, tj. sistema u kojima postoji veza između izlaza i ulaza sistema (Wolstenholme, 1990). Povratna veza može biti *pozitivna*, kada se pojačava rad sistema, ili *negativna*, kada se rad sistema zagušuje ili stabilizira.

Primjer negativne povratne veze je sistem za grijanje prostorija upravljan termostatom. Kada temperatura u sobi (izlazna varijabla) poraste preko neke granice, tada će termostat isključiti grijanje (ulazna varijabla), a kada padne ispod neke granice, tada će ga ponovno uključiti (slika 2.5a). U sistemu skladištenja analogna se situacija pojavljuje kada razina zaliha u skladištu padne ispod neke željene granice, jer se tada naručuje nova roba za skladište koja će povećati zalihe. Primjer pozitivne povratne veze je porast broja stanovnika zbog rada (slika 2.5b). Kada ne bi bilo smrtnosti, broj stanovnika bi brzo rastao (ovdje se ne uzimaju u obzir nikakva druga ograničenja za rast broja stanovnika).



(b) Pozitivna povratna veza



(a) Negativna povratna veza

Slika 2. 5. Sistemi s povratnom vezom

Modeli s povratnom vezom koriste se za modeliranje inženjerskih sistema te bioloških, ekonomskih i društvenih fenomena. Sistemska dinamika prikazuje sisteme kao povezane upravljačke petlje. Pri tome su pojedinačni dogadaji jako *agregirani*, a

posljedica toga jest da se mogu opisivati kao *kontinuirani tokovi* opisani *diferencijskim jednadžbama* (tj. konačnim razlikama a ne beskonačno malim veličinama). Novo stanje sistema u sljedećem času računa se na temelju promjena koje nastupaju u stanju iz prethodnog vremenskog časa, odnosno iz više prethodnih časova.

Takav agregirani prikaz sistema s kontinuiranim tokovima omogućuje da se istražuje vremenski razvoj sistema kao cjeline, te da se ispita stabilnost sistema i reakcija ponašanja sistema na vanjske perturbacije i promjene parametara sistema. Sistemska dinamika koristi se za modeliranje problema rada industrijskih sistema, rasta gradova, ekosistema itd.

Jedan od najpoznatijih primjera primjene sistemske dinamike bila je prva studija graniča rasta na Zemlji objavljena u knjizi *Granice rasta* (Meadows i dr., 1974). Model rasta obuhvatio je i povezao utjecaje osnovnih faktora važnih za život na Zemlji: porast broja stanovnika, proizvodnju hrane, crpljenje zaliha ruda, potrošnju energije, zagadživanje, industrijski i uslužni kapital i sl. Studija je omogućila predviđanje trendova promjena osnovnih veličina važnih za Zemlju i ljudsku vrstu, te mogućnost utjecaja na njih različitim strategijama razvoja ljudskog društva. Značenje modela je to što je pokazao smjer i brzinu razvoja osnovnih veličina na Zemlji u zavisnosti o strategiji pristupa razvoju (npr. ograničavanje rasta pučanstva, industrijske proizvodnje i zagadživanja), te nužnost planiranja razvoja. Kašnjenje u počinjanju planiranja može biti nenađeknadio i može uzrokovati drastičan pad količine raspoloživih sirovina i hrane te porast zagadenja, a posljedice bi bili povećanje smrtnosti stanovništva i smanjenje kvalitete života ljudi u samo nekoliko desetaka godina.

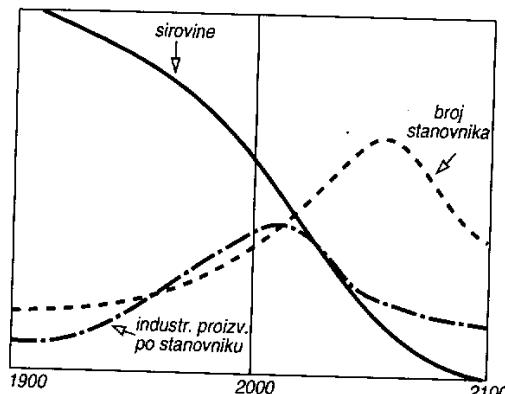
Na slici 2.6 prikazan je dio modela sistemske dinamike razvoja Zemlje. Pozitivna i negativna povratna veza su naravno rođenja i smrtnost. Industrijska proizvodnja omogućuje ulaganje kapitala u uslužne djelatnosti (stanove, škole, bolnice i sl.) a veličina ulaganja kapitala podijeljena brojem stanovnika daju usluge po stanovniku. Povećani broj stanovnika uzrokuje sniženje razine zdravstvenih usluga, a time i povećanje smrtnosti pučanstva (negativna povratna veza). Slabije planiranje obitelji uzrokovano smanjenim uslugama dovodi do povećanja priraštaja stanovnika (pozitivna povratna veza), isto kao i povećana industrijska proizvodnja po stanovniku. Neki elementi jednog od tipova razvoja svjetskog modela dobivenog simulacijom prikazani su na slici 2.6b.

Najpoznatiji programski jezik za sistemsку dinamiku je DYNAMO, a noviji razvijeni jezici su STELLA i DYNSMAP.

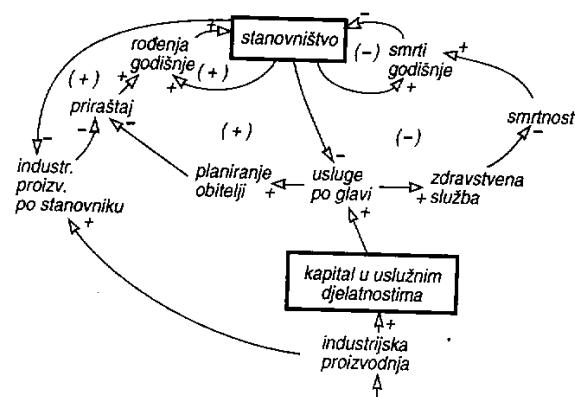
2.2.3. Simulacija diskretnih događaja

Simulacija diskretnih događaja ima mjesto među veoma detaljnim modelima kontinuiranih simulacija jednostavnih sistema i kao agregiranim modelima sistemske dinamike složenih sistema. Ta metoda omogućuje kombinaciju opisa detalja sistema i složenih međudjelovanja u njemu (Pidd, 1984).

Simulacija diskretnih događaja uključuje modele koji su strukturirane kolekcije *objekata*. Međudjelovanje objekata u aktivnostima sistema uzrokuje promjene stanja sistema. Te promjene stanja, zvane *događaji*, dešavaju se u diskretnim (tj. diskontinuiranim) vremenskim trenucima. Ovom se metodom najčešće opisuju sistemi s repovima, tj. sistemi u kojima dinamički objekti sistema traže posluživanje od ograničene kolekcije *resursa*, tj. statickih objekata. Posljedica ograničenja raspoložive količine *resursa* je stvaranje repova čekanja pred resursima.



(b) Jeden tip razvoja modela Zemlje, bez promjene u fizičkim, ekonomskim i društvenim odnosima

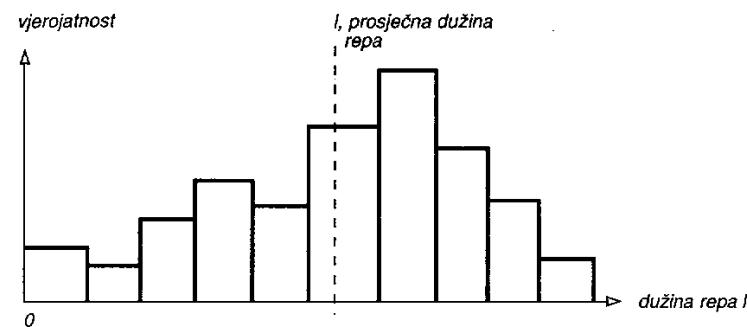
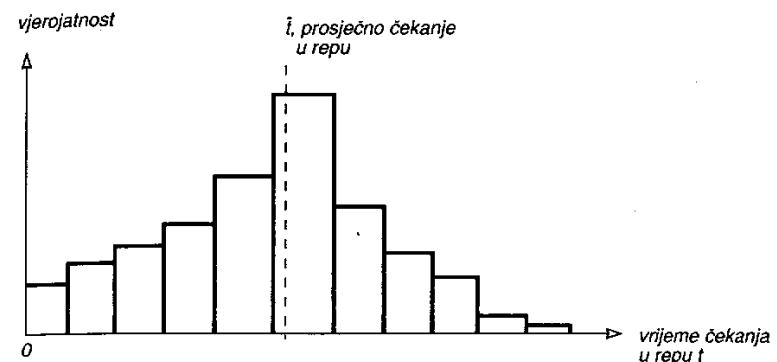


(a) Dio modela vezan uz stanovništvo

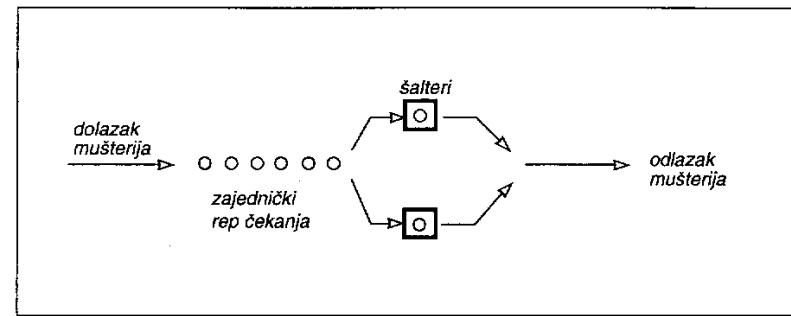
Slika 2.6. Sistemska dinamika razvoja Zemlje (prema Meadows i dr., 1974)

Osnovne komponente simulacije diskretnih događaja su izgradnja modela, rukovanje vremenom, baratanje slučajnim procesima, statistička analiza podataka i mehanizam izvođenja pomaka vremena u simulacijskim eksperimentima. Više je pristupa simulaciji diskretnih događaja: planiranje događaja, prelaženje aktivnosti, međudjelovanje procesa i trofazna simulacija. O simulaciji diskretnih događaja više će biti riječi u sljedećim poglavljima. Neki od poznatih programskih jezika u ovom području su GPSS, SIMSCRIPT i SIMULA.

Primjeri primjena simulacije diskretnih događaja su modeli prodavaonica sa samoposluživanjem, aerodromskih zgrada, računarskih mreža, transportnih sistema i proizvodnih procesa. Na slici 2.7a prikazan je primjer jednostavnog sistema sa zajedničkim repom pred dva šaltera u banci, a na slici 2.7b primjer vrste izlaznih rezultata simulacije.



(b) Oblik nekih rezultata simulacija repa u baci



(a) Rep posjetitelja u baci

Slika 2.7. Simulacija diskretnih događaja sistema s repovima

2.2.4. Kombinirana diskretno-kontinuirana simulacija

Za neke vrste sistema ni kontinuirana simulacija ni simulacija diskretnih događaja ne mogu u potpunosti opisati način rada sistema. To su sistemi koji sadrže i procese koji teku kontinuirano i događaje koji dovode do diskontinuiteta u razvoju sistema. Da bi se takvi sistemi modelirali i simulirali razvijena je kombinirana simulacija koja omogućuje integriranje kontinuiranih i diskretnih elemenata sistema (Pritsker, 1984; Kreutzer, 1986).

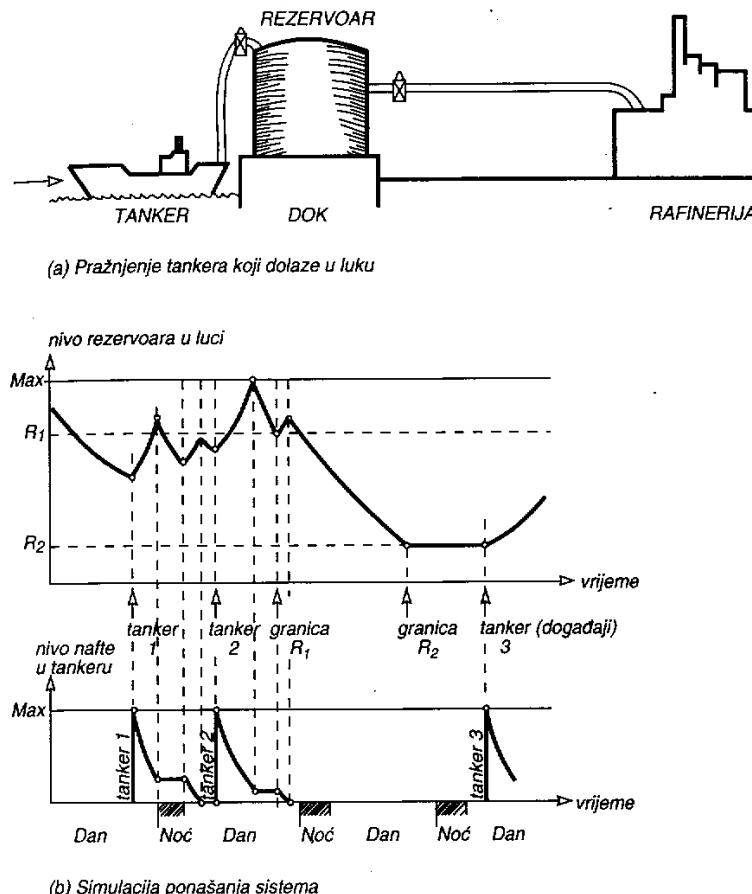
Veza među diskretnim i kontinuiranim pristupom postiže se dvama tipovima događaja. Vremenski događaji su događaji koje planira upravljanje događajima, kakvo postoji i u simulaciji diskretnih događaja. Oni mogu trenutačno promijeniti stanje kontinuirane varijable. Događaji stanja su događaji koji aktivira upravljanje pomoćom malih vremenskih intervala karakteristično za kontinuirane simulacije. Ti se događaji mogu aktivirati kada kontinuirane varijable zadovolje neke uvjete, npr. kada vrijednost kontinuirane varijable prijeđe neki granični nivo ili kada vrijednost jedne kontinuirane varijable premaši vrijednost druge kontinuirane varijable. Ti događaji mogu aktivirati događaje diskretnog dijela modela.

Diskrete i kontinuirane varijable mogu međudjelovati na nekoliko načina.

- (1) Diskretni događaj može aktivirati promjenu stanja kontinuirane varijable. Na primjer posipanje nekog područja sredstvima protiv komaraca smanjuje populaciju komaraca (koja se inače kontinuirano mijenja) praktično trenutačno.
- (2) Diskretni događaj može uzrokovati promjenu načina razvoja kontinuirane varijable. Na primjer zagadenje ekosistema nekim sredstvom može promijeniti relacije među populacijama različitih vrsta.
- (3) Ako vrijednost kontinuirane varijable prijeđe neki prag, to može uzrokovati dešavanje ili planiranje diskretnih događaja. Kada, npr., u kemijskom procesu koncentracija neke komponente padne ispod nekog nivoa kemijski proces završava i pokreće se proces čišćenja i održavanja postrojenja.

Simulacijski jezici za kombiniranu simulaciju su GASP, SLAM i jedna verzija jezika SIMSCRIPT.

Na slici 2.8. prikazan je primjer kombinirane simulacije iskrčavanja tankera s naftom u luci, pražnjenje nafte s tankera u rezervoar u luci i pražnjenje rezervoara te slanje nafte u rafineriju (Pritsker, 1984). Dolazak tankera je diskretan događaj, dok su količine pretočene nafte iz tankera u rezervoar i iz rezervoara u rafineriju kontinuirane varijable. Tanker se može prazniti samo u jednom periodu dana, za što su vezani događaji otvaranja i zatvaranja procesa pretakanja. Kada se rezervoar s naftom napuni, pražnjenje tankera se zaustavlja sve dok količina nafte u rezervoaru ne padne ispod neke granice (R_1). Kada se rezervoar isprazi do neke granice (R_2), odvod nafte u rafineriju se zaustavlja dok nivo nafte u rezervoaru ne prijeđe neku zadanu granicu. Na slici 2.8b vidimo rezultate jedne simulacije rada tog sistema. Vide se promjene nivoa nafte u rezervoaru zbog njegova pražnjenja i punjenja, te dva tipa zastojeva pražnjenja tankera: zbog obustave pretakanja u rezervoar noću i zbog prijelaza nivoa nafte u rezervoaru preko neke granice.



Slika 2. 8. Kombinirana diskretno-kontinuirana simulacija (prema Pritsker, 1984)

2.3. TIPOVI SISTEMA I IZBOR TIPOA SIMULACIJSKOG MODELA

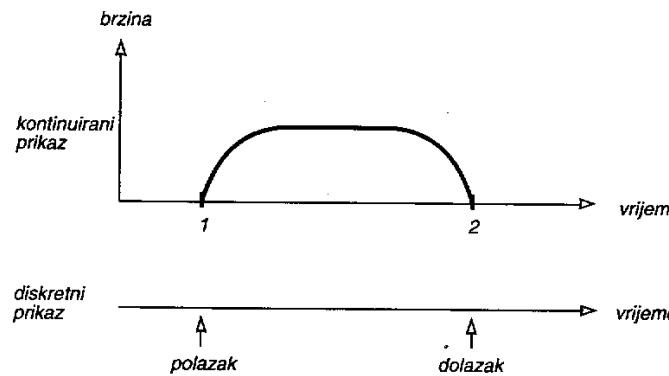
2.3.1. Tipovi sistema

Osnovna podjela sistema slična je kao i podjela simulacijskih modela, budući da su raznovrsni tipovi simulacijskih modela upravo i razvijeni stoga da omoguće prikaz i proračun vremenskog razvoja odgovarajućih tipova sistema. Kao što ćemo vidjeti, u klasifikaciji sistema je izvesna proizvoljnost jer sisteme klasificiramo ne samo prema njihovim osnovnim karakteristikama već i prema tome koliko ih poznajemo, odnosno što o njima želimo doznati da bismo riješili problem koji nas zanima.

Ukratko ćemo analizirati dvije osnovne vrste podjele sistema, i to na diskretne i kontinuirane sisteme, odnosno na determinističke i stohastičke sisteme.

Diskretni sistemi su sistemi kod kojih se varijable stanja mijenjaju samo u diskretnim vremenskim točkama. Tako se broj putnika koji čekaju u repu pred šalterom registracije aerodroma mijenja samo onda kada neki putnik uđe u taj rep ili kada završi svoju registraciju. *Kontinuirani sistemi* su oni kod kojih se varijable stanja mijenjaju kontinuirano u vremenu. Tako kompozicija u metrou kontinuirano mijenja položaj i brzinu između dviju stanica metroa, i to od brzine nula u času kretanja preko ubrzanja, postizanja maksimalne brzine, usporavanja i konačnog zaustavljanja u određenoj stanici.

Pogledajmo zašto je podjela sistema na diskrete i kontinuirane donekle proizvoljna, odnosno kako isti sistem možemo često opisati i kao diskretni i kao kontinuirani, zavisno o točnosti i svrsi opisivanja sistema. Na primjer, čini nam se prirodno da opišemo putovanje kompozicije između stanica metroa kao kontinuiranu varijablu. Ako međutim promatramo mrežu linija metroa i zanima nas samo vozni red te mreže, tada nas ne zanimaju detalji promjene brzine kompozicije u toku vožnje. Umjesto toga možemo izračunati vrijeme putovanja između stanica i sistem promatrati kao diskretni sistem koji promjene stanja svodi na događaje odlazaka i dolazaka kompozicija u stanicu. Cijelo putovanje kompozicije između dviju stanica izraženo je samo vremenom putovanja (slika 2.9).



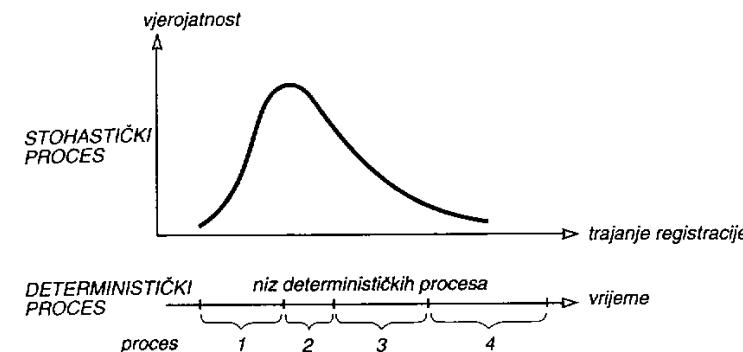
Slika 2. 9. Kontinuirani i diskretni prikaz vožnje između dviju stanica

Općenito, kontinuirani prikaz se koristi ili kada fenomen želimo opisati veoma detaljno ili, u suprotnom slučaju, kada agregiramo niz detalja složenih sistema i povezujemo ih u kontinuirane tokove.

Deterministički sistemi su oni koji ne sadrže slučajne varijable. Primjer determinističkog sistema je trajanje operacije obrade na automatiziranim strojevima gdje vrijeme obrade veoma malo fluktira. *Stohastički sistemi* sadrže bar jednu slučajnu varijablu. Tako su u banci dolasci stranaka na šaltere slučajne veličine s nekom razdiobom vjerojatnosti vremena između dvaju uzastopnih dolazaka.

I ovaj tip podjele sistema je donekle proizvoljan. Tako se aktivnost zadržavanja vozila na benzinskoj crpki u pravilu tretira kao slučajna varijabla jer vrijeme posluživanja vozila fluktira. Ako bismo međutim aktivnost posluživanja na crpki detaljno analizirali, mogli bismo je prikazati kao niz uzastopnih kraćih operacija (aktivnosti) – otvaranje rezervoara, točenje goriva, zatvaranje rezervoara, eventualna nabava ulja, plaćanje

i sl. Isto tako bismo mogli ustanoviti različite faktore utjecaja na trajanje pojedine operacije – vrijeme točenja goriva npr. ovisi o tipu vozila (motocikl, mali automobil, veći automobil, kombi, kamion i sl). Dakle, kada bi nas prvenstveno zanimali detalji rada bezinskih crpki, mogli bismo napraviti složeniji model koji bi obuhvatio detaljnije operacije na crpki i faktore utjecaja na vremena tih operacija. Takav model mogao bi se tretirati kao deterministički model (slika 2.10).



Slika 2.10. Deterministički i stohastički prikaz registracije na aerodromu

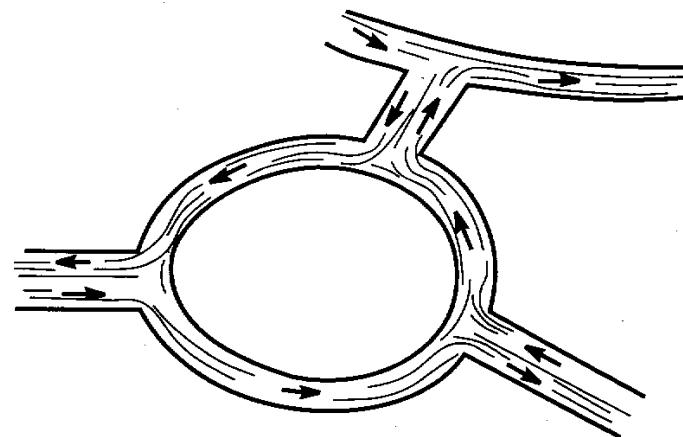
Općenito, stohastički prikaz sistema je pogodan kada dijelove sistema ne želimo veoma detaljno analizirati već pomoći opisa vjerojatnosti njihova ponašanja obuhvaćamo mnoge u modelu nevidljive faktore utjecaja.

2.3.2. Izbor tipa simulacijskog modela

Tip simulacijskog modela najčešće se odabire tako da bude jednak tipu sistema, odnosno 'prirodnim' tipu sistema. To ipak ne znači da uvjek moramo diskretni sistem prikazati diskretnim modelom, odnosno kontinuirani sistem kontinuiranim modelom. Izbor tipa simulacijskog modela ovisi prije svega o cilju simulacijske studije. Osim toga, važnu ulogu u izboru tipa simulacijskog modela igra i sposobnost onoga tko model razvija da pronađe adekvatnu apstrakciju sistema za rješavanje određenog problema. Pri tome može biti važno i prethodno iskustvo i intuicija u razvoju simulacijskih modela, a često i poznavanje te pristupačnosti određenog simulacijskog jezika.

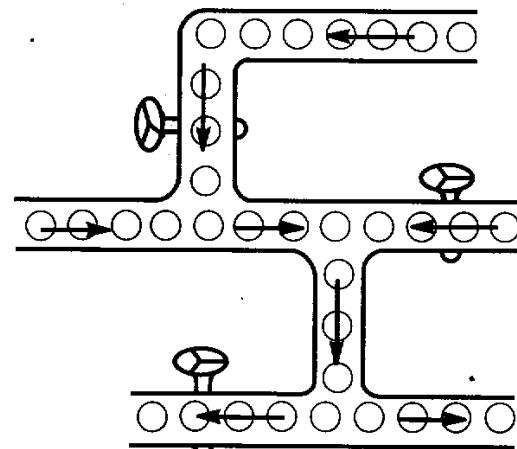
Najvažnije pri izboru tipa simulacijskog modela je da model bude što jednostavniji i razumljiviji, kako zbog olakšanja njegova razvoja (i razvoja njegovih modifikacija) tako i zbog potrebe da ga korisnik što lakše razumije. Model također mora biti dovoljno efikasan u pogledu utroška resursa računala (vrijeme, memorija) kako bi se mogao efikasno koristiti u rješavanju problema.

Tako se može model prometnog toka na auto-cesti prikazati 'prirodnim' diskretnim modelom toka pojedinačnih vozila, ako je značajno da se uvaže karakteristike i gibanja pojedinačnih vozila. No ako to nije važno, moguće je agregirati vozila u kontinuirani model toka koji se opisuje diferencijalnim jednadžbama (slika 2.11), i koji može dati dovoljno točan opis tokova vozila.



Slika 2.11. Kontinuirani model toka vozila na mreži auto-cesta

Drugi primjer alternativne mogućnosti modeliranja istog sistema je kontinuirani tok tekućine u kemijskom postrojenju koji se 'prirodno' opisuje kontinuiranim simulacijskim modelom diferencijalnih jednadžbi tokova tekućine. Tok tekućine može se međutim diskretizirati i prikazati kao gibanje zasebnih elemenata volumena (možemo zamisliti da se umjesto tekućine gibaju kuglice koje jedna drugu guraju), kao što je prikazano na slici 2.12. Time se može postići još uvijek zadovoljavajuća točnost prikaza modela uz mnogo veću efikasnost izvođenja simulacije.



Slika 2.12. Diskretni model toka tekućine u kemijskom postrojenju

II. DIO

SIMULACIJA DISKRETNIH DOGAĐAJA

3. OSNOVNE IDEJE SIMULACIJE DISKRETNIH DOGAĐAJA

Ovim poglavljem počinjemo detaljan opis metode simulacije diskretnih dogadaja. Opisat ćemo osnovne pojmove koji se upotrebljavaju u simulaciji diskretnih dogadaja, kao i osnovne probleme koji se moraju riješiti u sklopu ove metode simulacijskog modeliranja. Zatim ćemo opisati sastav alata za simulaciju diskretnih dogadaja potrebnih za modeliranje, izvođenje i analizu simulacijskih eksperimenata. Poglavlje ćemo zaokružiti opisom tipičnih problema koji se mogu modelirati i rješavati ovom metodom, te opisom osnovnih obilježja nekoliko simulacijskih studija složenih realnih sistema.

3.1. OSNOVNI POJMOVI SIMULACIJE DISKRETNIH DOGAĐAJA

Simulacija diskretnih dogadaja je metoda za simulacijsko modeliranje koja opisuje promjene stanja što se dogadaju diskontinuirano u vremenu, tj. samo u nekim vremenskim trenucima. Modeli sadrže objekte određenih svojstava, koji svojim međudjelovanjem uzrokuju promjene stanja sistema u vremenu.

Ukratko ćemo opisati osnovne pojmove simulacije diskretnih dogadaja kojima ćemo se koristiti u daljem prikazu ove metode i njezine primjene (Pidd,1984).

Model je apstraktni prikaz sistema koji opisuje objekte sistema i njihovo međudjelovanje, a obično sadrži matematičke (npr. proračun trajanja,aktivnosti) i logičke relacije (uvjeti početka aktivnosti, pravila za izbor entiteta) koje odgovaraju strukturi i načinu rada sistema.

Entiteti (objekti) komponente su sistema koji modeliramo. Entitete možemo individualno identificirati i njima baratati. *Stalni entiteti (resursi)* oni su koji ostaju u modelu u toku sveg vremena trajanja simulacije, a *privremenih entiteta* su oni koji prolaze kroz sistem.

Atributi entiteta opisuju svojstva entiteta. Svaki entitet može imati veći broj atributa. *Klase entiteta* su grupe entiteta istog tipa.

Skupovi entiteta su grupe entiteta pojedine klase koji imaju neka zajednička obilježja (npr. jednake vrijednosti pojedinog atributa). S obzirom na mogućnost dinamičke promjene svojstava entiteta, oni se u toku simulacije mogu premještati iz skupa u skup. Posebni tip entiteta su *repovi čekanja* koji prikazuju grupu privremenih entiteta što čeka da se osloboди neki resurs. Vrijeme čekanja entiteta u repu ne može se unaprijed odrediti, tj. ono ovisi o razvoju sistema u vremenu.

Stanje sistema (modela) skup je svih informacija nužnih za opis sistema. Stanje sistema ovisi o entitetima koji su u njemu prisutni u tom času, i o vrijednostima njihovih atributa.

Dogadaj je promjena stanja sistema u jednom vremenskom času. On može nastupiti zbog:

- (1) ulaska/izlaska privremenog entiteta iz sistema ili
- (2) promjena atributa pojedinih entiteta sistema
(zbog početka/završetka međudjelovanja entiteta).

Između dvaju uzastopnih događaja stanje sistema se ne mijenja.

Uvjetni dogadaji su oni dogadaji koji se mogu dogoditi tek pošto je ispunjen neki uvjet. Oni su obično povezani za dostupnost nekog resursa, tj. početak aktivnosti.

Bezuvjetni (planirani) dogadaji su oni čiji je jedini uvjet da bi se mogli dogoditi povezan za prolaz vremena. Obično su povezani uz oslobađanje resursa, tj. završetak aktivnosti.

Aktivnost je međudjelovanje određenih entiteta koje traje određeno vrijeme. Aktivnost dovodi do promjene stanja entiteta koji u njoj sudjeluju. U simulaciji diskretnih događaja promjena stanja se prikazuje u početnom i završnom događaju aktivnosti, a u toku trajanja aktivnosti stanje entiteta uključenih u aktivnost se ne mijenja. Dakle, kontinuirano odvijanje aktivnosti u stvarnom sistemu prikazuje se kao diskretna promjena stanja u času početka i završetka aktivnosti. Početak aktivnosti je obično vezan za neke uvjete, a aktivnost završava nakon protoka određenog vremena *trajanja aktivnosti*.

Proces je niz logički povezanih uzastopnih događaja kroz koje prolazi neki privremeni entitet. Proces može obuhvaćati dio ili cijelinu "života" privremenog entiteta u simulaciji.

Koristeći se ovim pojmovima možemo karakterizirati simulaciju diskretnih događaja ovako (Shannon, 1975):

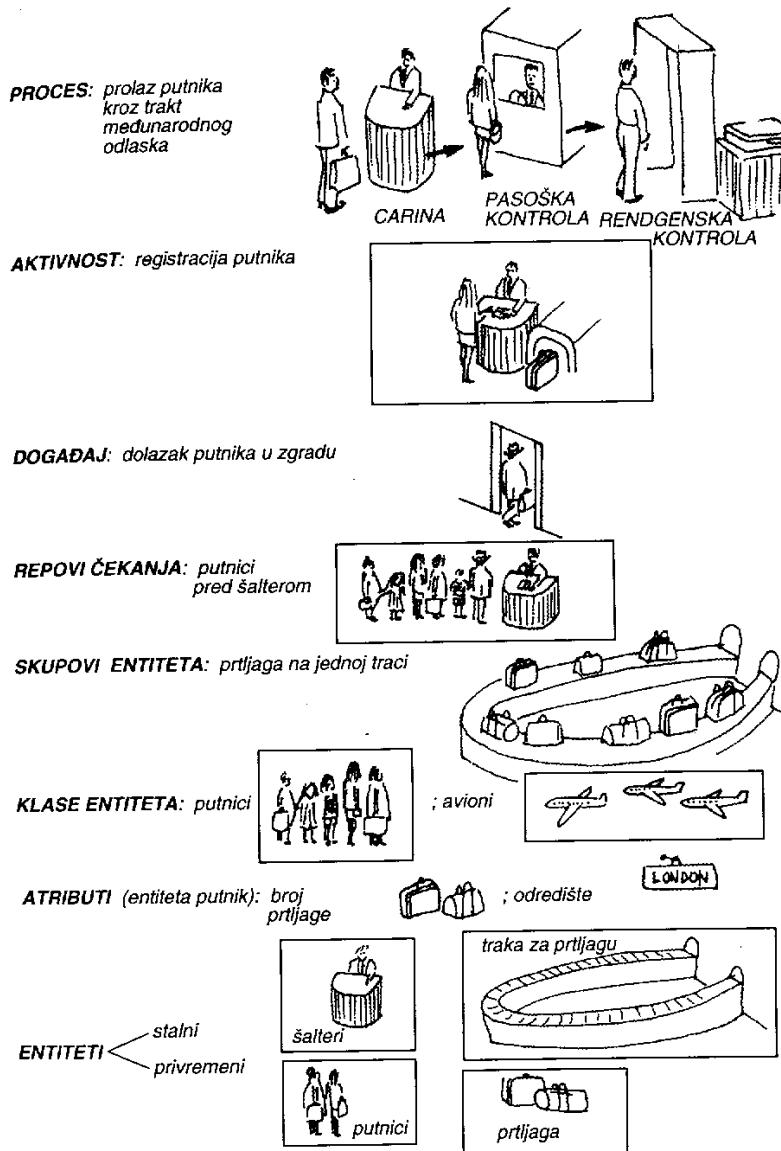
Entiteti

koji imaju **attribute**
međudjeluju u **aktivnostima**
uz izvjesne **uvjete**
stvarajući **događaje**
koji mijenjaju **stanje** sistema.

Da bi se osnovni pojmovi simulacije diskretnih događaja mogli što bolje razumjeti, prikazat ćemo njihovo značenje na primjeru simulacije putničke zgrade aerodroma (slika 3.1). *Model* sistema opisuje njegovu strukturu i način rada. *Stalni entiteti* sistema (resursi) jesu npr. šalteri za registraciju, rendgen za kontrolu putnika i traka za izdavanje prtljage, a *privremeni entiteti* su avioni, putnici i prtljaga. *Atributi* putnika su npr. odredište i broj prtljage. *Klase entiteta* su npr. svi šalteri registracije i svi putnici. *Skupovi entiteta* su npr. svi putnici koji putuju na isto odredište ili sva prtljaga koja je na istoj traci za prtljagu. *Repozi čekanja* su npr. skupovi putnika koji čekaju pred šalterima za registraciju za polijetanje. *Stanje sistema* opisano je svim resursima i privremenim entitetima s vrijednostima njihovih atributa.

Planirani dogadaj je npr. dolazak novog putnika sa slijetanjem (povećanje broja entiteta u sistemu) ili završetak posluživanja na šalteru registracije (promjena broja putnika pred šalterom i promjena atributa šaltera — u tom času šalter postaje slobodan). *Uvjetni dogadaj* je npr. čas uzimanja prtljage s trake za prtljagu (ispunjeno je uvjet da je putnik koji čeka pred trakom za prtljagu došla njegova prtljaga). *Aktivnost* je npr. registracija putnika u kojoj međudjeluju putnik, njegova prtljaga i šalter registracije (uključujući osobu koja na njemu radi). Sve što se o toj aktivnosti zna u modelu jest

njezin početak, vrijeme trajanja i završetak. *Proces* je npr. prolazak putnika kroz trakt međunarodnog odlaska gdje putnik uzastopno čeka pred različitim šalterima (carina, pasoš, rendgen), poslužuje se na njima i hoda između njih.



Slika 3.1. Prikaz osnovnih pojmoveva simulacije diskretnih događaja na modelu putničke zgrade aerodroma

3.2. SPECIFIČNI PROBLEMI KOJE MORA RJEŠAVATI MEHANIZAM SIMULACIJE DISKRETNIH DOGAĐAJA

Osnovni problemi koje mora rješavati simulacija diskretnih događaja, i koji ovu metodu čine specifičnom metodom modeliranja, jesu:

- (1) Izvođenje *promjene stanja sistema u vremenu*, s međudjelovanjem entiteta sistema.
- (2) Simulacija *istovremenih (paralelnih) akcija* na sekvencijalnom računalu.
- (3) Simulacija *konkurentnih* procesa na sekvencijalnom računalu. U konkurentnim procesima entiteti se natječu za posluživanje od ograničenih resursa. Posljedica toga je *čekanje u repovima* koje može biti različito organizirano (posluživanje po slijedu dolaska u rep, po prioritetu, odustajanje od čekanja).
- (4) Rješavanje *statističkih aspekata* koji se odnose na opisivanje slučajnih varijabli i izvođenje operacija s njima.

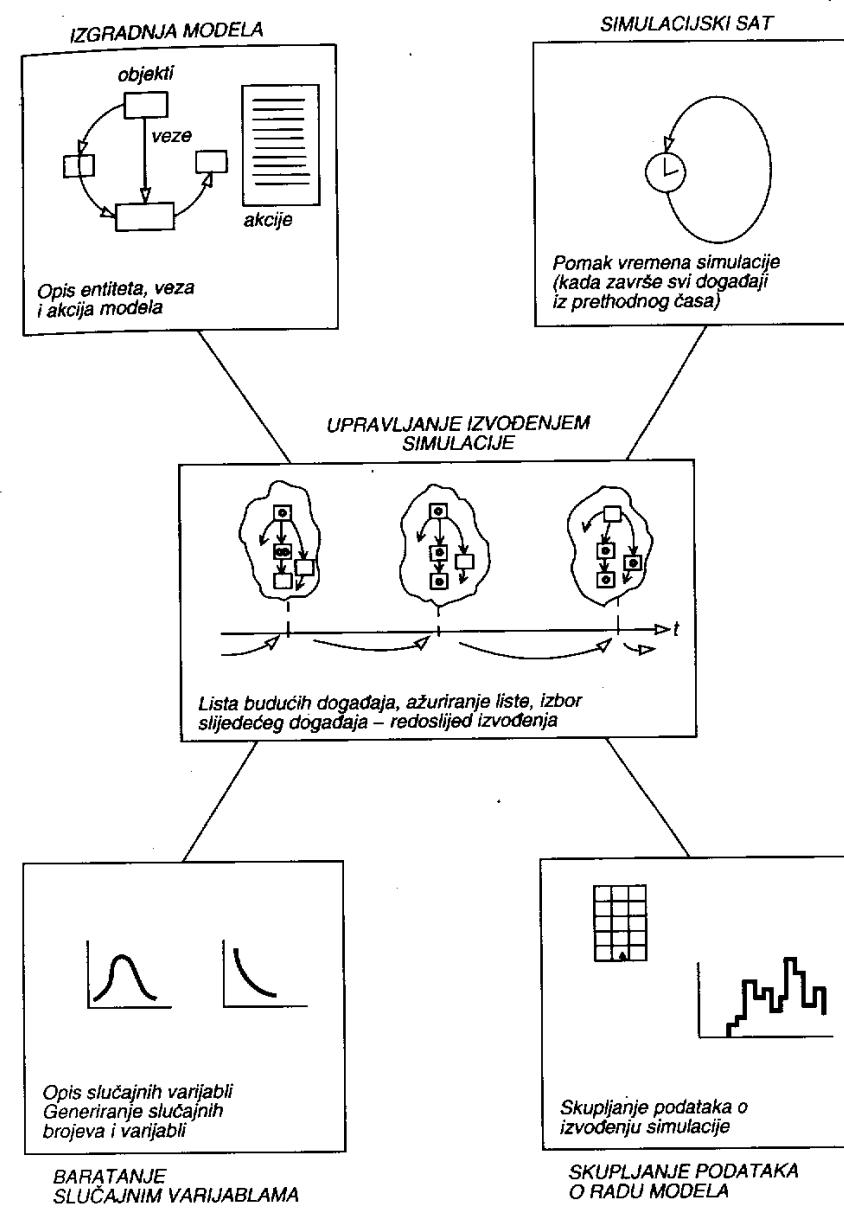
Dio navedenih problema susreće se već pri konceptualnom modeliranju, dio pri programiranju a svi pri izvođenju simulacijskih eksperimenta. Konceptualni modeli moraju osigurati adekvatan prikaz ovakvih modela, a programski jezici ili paketi za simulaciju diskretnih događaja moraju uključivati mogućnost prikaza i korektnog izvođenja simulacijskih programa. Efikasnost u rješavanju problema je, s gledišta potrošenih resursa računa (vremena i memorije), također značajna za omogućavanje sveobuhvatne analize i rješavanja složenih dinamičkih problema.

Navedene specifične aspekte treba poznavati svatko tko razvija model, kako bi se osiguralo stvaranje ispravnih i efikasnih simulacijskih modela. Osobito treba poznavati probleme u vezi s *paralelizmom* odvijanja simulacije diskretnih događaja, budući da ništa slično ne postoji u klasičnom programiranju. Stoga su i greške u formuliranju konceptualnog simulacijskog modela i simulacijskog programa prilično česte.

3.3. STRUKTURA ALATA ZA SIMULACIJU DISKRETNIH DOGAĐAJA

Iako su brojni alati za različito rješavanje pojedinih problema simulacije diskretnih događaja, svi oni imaju neke zajedničke komponente. Na slici 3.2. prikazana je osnovna struktura alata za simulaciju diskretnih događaja (Kreutzer, 1986).

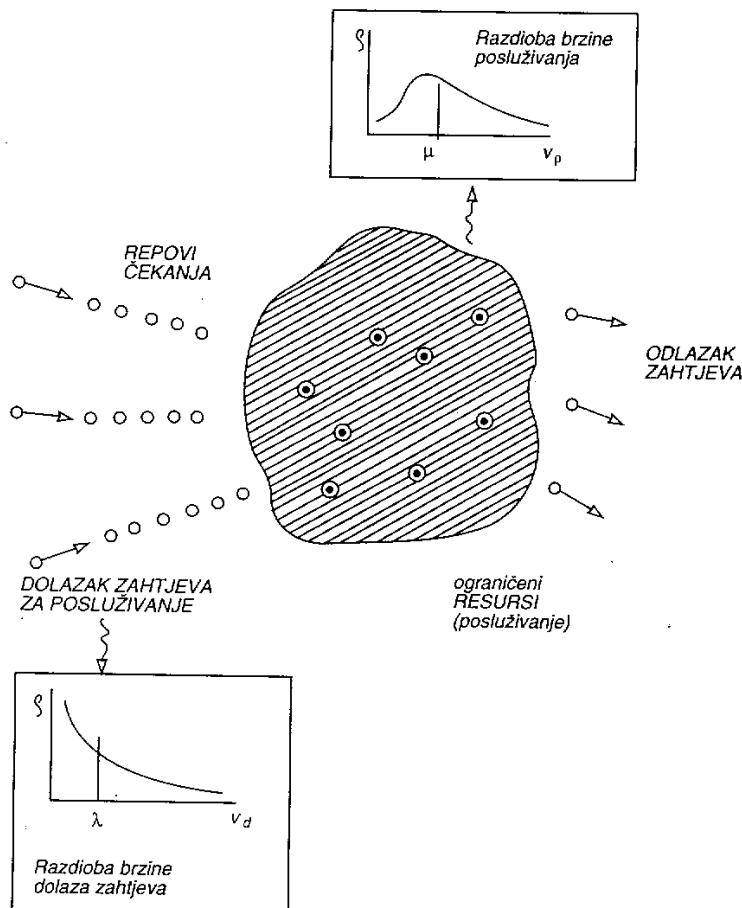
- (1) Alati za *izgradnju modela* omogućuju prikaz strukture i načina rada modela. Prikaz uključuje entitete modela i njihove atribute, vezu među entitetima i opis akcija modela.
- (2) *Sat modela* pokazuje trenutni čas rada modela. Vrijeme sata se ne mijenja dok se ne završe svi događaji koji se u tom času mogu dogoditi. Nakon toga, sat će biti pomaknut na čas izvođenja sljedećeg događaja.
- (3) Opis *slučajnih varijabli* u sistemu i generiranje slučajnih brojeva i varijabli.
- (4) *Skupljanje statističkih podataka* o ponašanju modela tijekom izvođenja simulacijskog eksperimenta, za potrebe detaljne analize rada modela.
- (5) *Upravljanje izvođenjem simulacije* koje osigurava pravilan izbor slijeda izvođenja simulacije, te adekvatno strukturiranje i baratanje podacima o događajima, budući da se oni često mijenjaju.



Slika 3. 2. Osnovne komponente alata za simulaciju diskretnih događaja (prema Kreutzer, 1986)

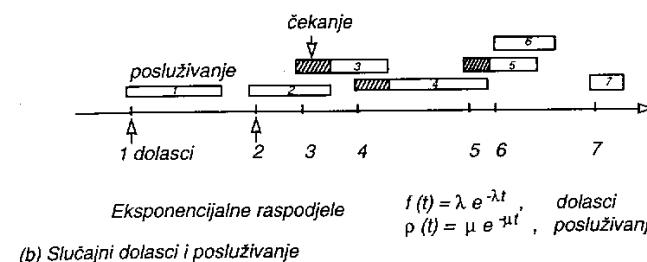
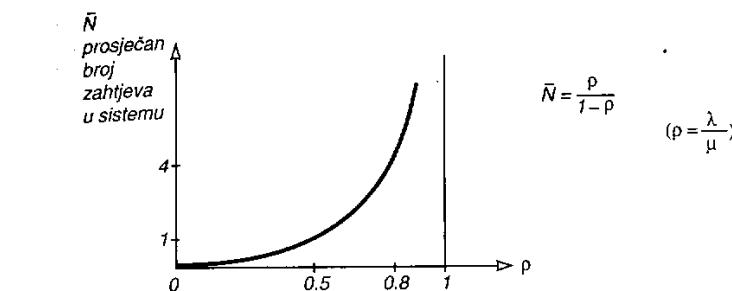
3.4. TIPIČNI PROBLEMI KOJI SE MOGU RJEŠAVATI SIMULACIJOM DISKRETNIH DOGAĐAJA

Sistemi masovnog posluživanja (sistemi s repovima) najčešći su tip sistema koji se modeliraju i analiziraju simulacijom diskretnih događaja. To su sistemi u kojima zahtjevi za posluživanjem dolaze od resursa ograničenog kapaciteta. Struktura takvih sistema prikazana je na slici 3.3.

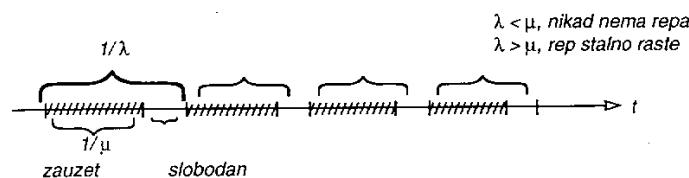


Slika 3.3. Sistemi masovnog posluživanja

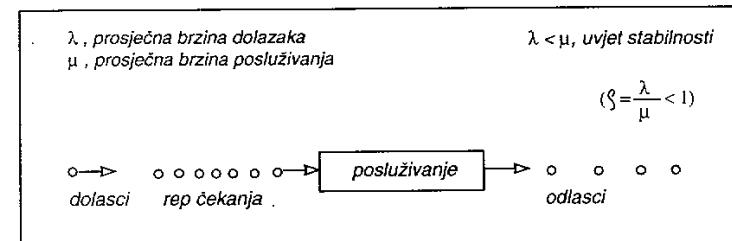
Ako su kod ovakvih sistema vrijeme dolazaka, veličine zahtjeva i brzine posluživanja predvidivi (*deterministički*) tada ili *nema čekanja* (ako je potražnja za resursima manja od kapaciteta resursa) ili *repovi čekanja neprekidno rastu* (ako je potražnja veća od kapaciteta resursa), slika 3.4a.



(b) Slučajni dolasci i posluživanje



(a) Deterministički dolasci i posluživanje



Sistem s jednim repom čekanja

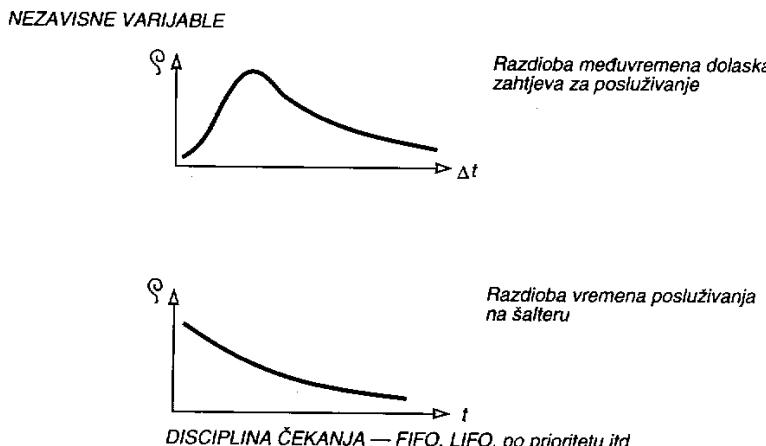
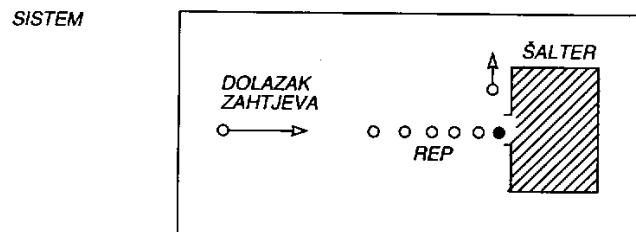
Slika 3.4. Ponašanje sistema s repom čekanja

Ako su vrijeme dolazaka, veličine zahtjeva ili vrijeme posluživanja nepredvidivi (*slučajne veličine*), tada povremeno nastaju konflikti pri korištenju resursa i stvaraju

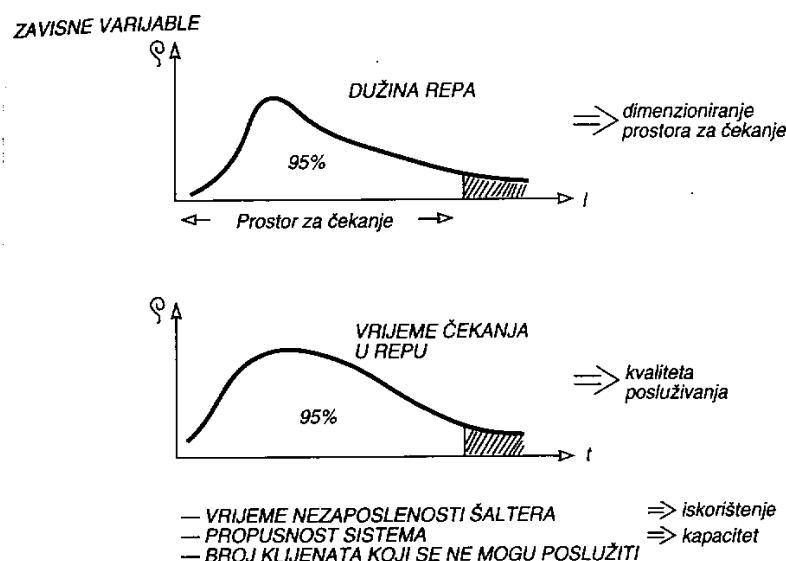
se *repovi čekanja* dolaznih zahtjeva čak i kada je potražnja manja od kapaciteta resursa. Dužine repova ovise kako o odnosu *prosječnih* brzina dolazaka i posluživanja, tako i o *statičkim* fluktuacijama tih veličina (opisanih razdiobama vjerojatnosti), slika 3.4b.

Primjeri sistema masovnog posluživanja su šalteri u bankama, prometni sistemi, proizvodni sistemi, ekonomski sistemi, računarski sistemi itd.

Na primjeru šaltera za posluživanje (u bankama, aerodromima i sl.), prikazanom na slici 3.5, pokazat ćemo osnovne tipove varijabli i karakteristike sistema koji se pojavljuju u sistemima masovnog posluživanja. Tako su prikazane *ulazne (nezavisne) varijable*: funkcije vjerojatnosti međuvremena dolazaka i vremena posluživanja te discipline čekanja dolaznih zahtjeva u repu, i *izlazne (zavisne) varijable*: funkcije vjerojatnosti dužine repova i vremena čekanja, vrijeme nezaposlenosti sistema, broj zahtjeva koji se ne mogu poslužiti itd. Simulacijom dobivene karakteristike zavisnih varijabli mogu poslužiti kao podloga za *dimenzioniranje prostora* pred šalterima, ocjenu *kvalitete posluživanja* s obzirom na čekanje, zaposlenost šaltera, propusnost sistema i sl.



3.5. NEKOLIKO PRIMJENA SIMULACIJE DISKRETNIH DOGAĐAJA



Slika 3. 5. Tipovi varijabli i karakteristike sistema: primjer šaltera za posluživanje

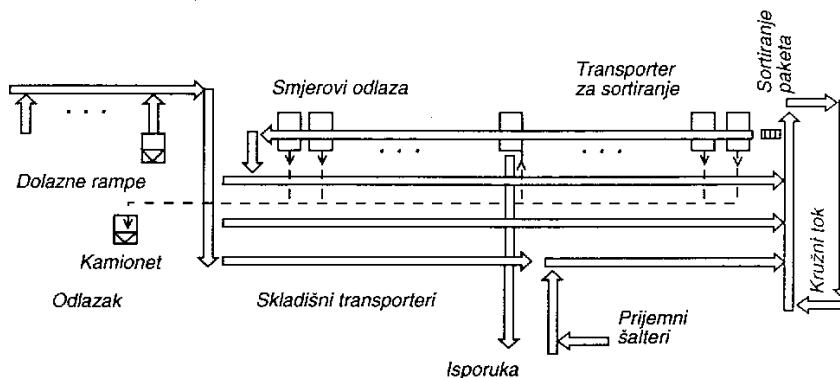
3.5. NEKOLIKO PRIMJENA SIMULACIJE DISKRETNIH DOGAĐAJA

Da bismo ilustrirali probleme koji se mogu rješavati simulacijom diskretnih dogadaja, prikazat ćemo nekoliko vlastitih primjena te metode u rješavanju složenih realnih problema. Pri tome ćemo ukratko dati osnovne podatke o sistemu koji smo modelirali, simulacijskom jeziku koji je korišten te vrsti dobivenih rezultata.

(1) Simulacija paketnog poštanskog centra

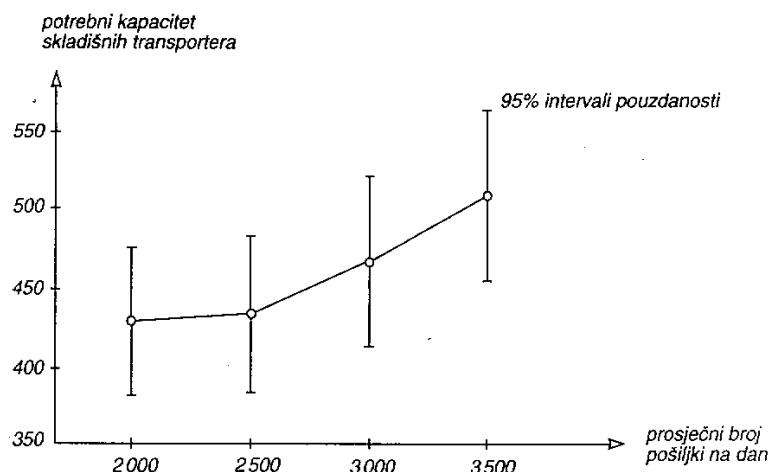
U simulacijskoj studiji paketnog poštanskog centra (Vrgoč i Čerić, 1988) modeliran je paketni poštanski centar Maribor, s protokom oko 2 000 paketa dnevno. Poštanski centar je čvor u poštanskoj servisnoj mreži sa zadatkom da sortira i pošalje na prava odredišta i u odgovarajuće vrijeme pošiljke koje u njega dolaze. Pošiljke dolaze poštanskim kamionima iz različitih smjerova, te s poštanskih šaltera u gradu. Obrada pošiljki sastoji se od iskrcanja i primanja, skladištenja i transporta pokretnim trakama (konvejerima), transporta preko kružnog transportnog toka do mjesta sortiranja i, nakon toga, do izlaznih smjerova odvoza pošiljki. Struktura paketnog poštanskog centra prikazana je na slici 3.6. Sistem ima više slučajnih varijabli: dolazne količine i vrijeme, vrijeme obradbe, količine paketa po odlaznim smjerovima itd. Paketi se u poštanskom centru obrađuju u tri faze sortiranja unutar 24-satnog radnog ciklusa.

Simulacijski model napisan je GPSS jezikom i implementiran na računalu UNIVAC 1100. Cilj simulacijskih eksperimenata bio je da omoguće dobivanje performansi sistema



Slika 3. 6. Struktura modeliranog poštanskog paketnog centra
(prema Vrgoč i Čerić, 1988)

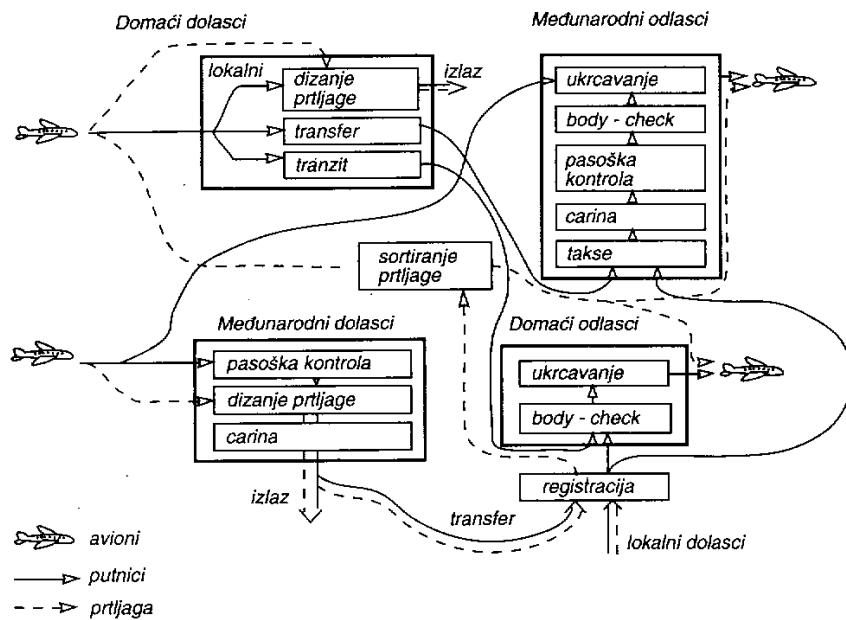
u uvjetima rastućeg opterećenja sistema, i to od 2 000 do 3 500 paketa na dan (srednjoročni horizont planiranja). Među ispitivanim performansama su npr. popunjavanje skladišnih transporterera (slika 3.7) te dio paketa koji nije obraden na vrijeme da bi stigao na kamionete u odlasku. Dobiveni rezultati omogućili su procjenu rada poštanskog centra u uvjetima očekivanog porasta broja posiljki u periodu planiranja. U cjelini, simulacijska studija je omogućila analizu rada poštanskog centra s rastućim brojem posiljki u periodu od tekućeg do srednjoročnog planskog razdoblja.



Slika 3. 7. Potrebni kapacitet skladišnih transporterera poštanskog centra dobiven simulacijskim eksperimentima (prema Vrgoč i Čerić, 1988)

(2) Simulacija putničke zgrade aerodroma

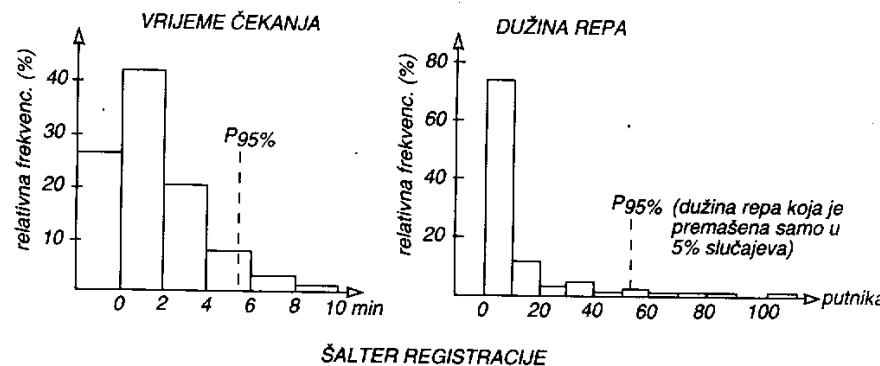
U simulacijskoj studiji (Čerić, 1988) putničke zgrade aerodroma modelirana je putnička zgrada aerodroma "Zagreb" površine oko 11 000 kvadratnih metara i kapaciteta od 1.4 do 1.5 milijuna putnika godišnje, te planiranog proširenja na oko 36 000 kvadratnih metara uz očekivani kapacitet od 3.75 do 4.5 milijuna putnika godišnje. Sistem se sastoji od složenog toka putnika i prtljage između domaćih i međunarodnih dolazaka i odlazaka, uz različite elemente posluživanja putnika (šalteri registracije putnika, pasoške kontrole, carine itd.) i prtljage (sortiranje, trake za prtljagu itd.), što se vidi na slici 3.8. Broj putnika po pojedinim smjerovima dolazaka i odlazaka povezan je za red letenja (tj. planirano je vrijeme dolazaka i odlazaka aviona). Za potrebe simulacijske studije izvršeno je i opsežno mjerjenje i analiza ulaznih podataka. Mjerilo se vrijeme posluživanja po šalterima, dinamika dolazaka lokalnih putnika, grupiranje putnika i prtljage (npr. putovanje obitelji) itd.



Slika 3. 8. Struktura modelirane putničke zgrade aerodroma (prema Čerić, 1988)

Simulacijski model razvijen je u jeziku GPSS i implementiran na računalu UNIVAC 1100. Program ima oko 1 800 linija koda (uključujući komentare). Rezultati izvođenja simulacije uključuju analizu kašnjenja polijetanja, određivanje potrebne površine prostora u putničkoj zgradbi te detaljniju sliku čekanja u svim dijelovima sistema, a posebno pred šalterima (slika 3.9). Simulacijska studija pokazala je da se, u usporedbi s prethodnim inženjerskim proračunom, može znatno smanjiti potrebna oprema. Samo zbog redukcije broja traka za prtljagu ušteđeno je oko 1.5 % od cijene konstrukcije terminalske zgrade (a to je bilo oko 15 puta više od cijene simulacijske studije). Osim toga, simulacija je

omogućila određivanje potrebne količine opreme (različiti šalteri i trake za prtljagu) koja će osigurati protok kroz sistem bez stvaranja velikih gužvi i čekanja.



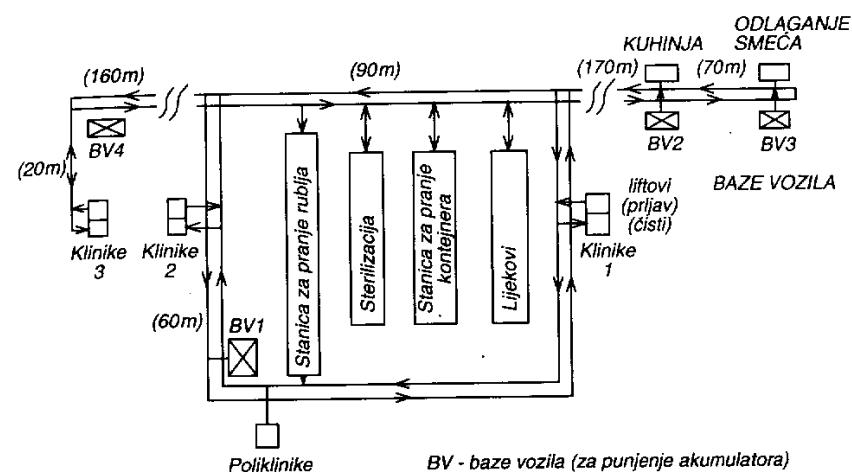
Slika 3.9. Vremena čekanja i dužine repova pred šalterom registracije putničke zgrade aerodroma — rezultati simulacijskih eksperimenata (prema Čerić, 1988)

Cijela studija trajala je 10 do 11 mjeseci, a od toga mjerjenje i analiza ulaznih podataka 4 do 5 mjeseci, razvoj modela i njegovo vrednovanje 4 do 5 mjeseci, te izvođenje i analiza simulacijskih eksperimenata 2 do 3 mjeseca. U cjelini, simulacijska studija je dala podatke za oblikovanje putničke zgrade aerodroma, omogućila je znatne uštede, te ostvarila usklađivanje kapaciteta koji osiguravaju ravnomjeran tok putnika i prtljage kroz sistem.

(3) Simulacija automatski vođenih vozila u bolnici

Simulacijska studija automatski vođenih vozila u bolnici (Čerić, 1990) opisala je interni transport u Sveučilišnoj bolnici u Zagrebu, koja će imati oko 1100 kreveta a sagrađena je na površini oko 140 000 kvadratnih metara. Sistem automatskog transporta jedna je od najznačajnijih funkcija podrške radu bolnice koja je simulirana. Uz pomoć ovog sistema bit će potrebno prevoziti oko 35 tona različitog materijala dnevno (hrana, posteljinu, sterilni materijal, smeće itd.), i to na udaljenosti i većoj od 400 metara. U transportnoj mreži prevoze se kontejneri koje vuku kompjutorski upravljana i programirana vozila, a za vertikalni se transport koriste automatska dizala. Svaki dan prevezе se oko 350 pošiljaka, i to prema planiranom dnevnom rasporedu. Sistem ima složeni način rada i upravljanja vozilima u mreži. Struktura sistema prikazana je na slici 3.10.

Simulacijski model napravljen je najprije u jeziku GPSS, pri čemu je zbog vrlo složenog rada sistema bilo gotovo nemoguće adekvatno opisati strukturu i način rada sistema. Druga verzija programa napisana je u jeziku SIMSCRIPT II.5, koji se pokazao mnogo fleksibilniji i prikladniji za opis takvog sistema. Program za testiranje korektnosti logike ulaznih podataka ima oko 1700 linija koda, a program za simulaciju oko 3 200 linija koda (oba uključujući komentare). Model je determinističkog karaktera, budući da sistem ima fiksna vremena početaka transporta ('dolasci'), dok brzina transporta nema velikih fluktuacija pa su transportna vremena praktično fiksna.



Slika 3.10. Struktura modeliranog automatskog transporta u bolnici (Čerić, 1990)

Izveden je složen niz simulacijskih eksperimenata. Određeni su: broj potrebnih vozila (veoma skupih) koja osiguravaju ispunjavanje transportnog plana, položaj i veličina baza vozila (u kojima se pune akumulatori automatskih vozila), čekanja u mreži, realizacija transportnog plana i maksimalna vremena transporta posijki (posebno važno zbog transporta hrane pacijentima). Na slici 3.11. prikazana je realizacija transportnog plana za nekoliko tipova pošiljki. U cjelini, simulacijska studija je pomogla oblikovanju i analizi rada automatski vođenog transporta u bolnici koji osigurava ostvarivanje planiranog rada sistema, te omogućila određivanje minimuma troškova za skupu transportnu opremu.

Tip pošiljke	Vrijeme u danu													
	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Hrana za pacijente	→	↔	↔	↔	↔	↔	↔	↔	↔	↔	↔	↔	↔	↔
Publike					↑	↓								
Sterilni materijali					↑	↓								
Smeće					↑	↓								

Slika 3.11. Realizacija transportnog plana automatskog transporta u bolnici dobivena simulacijskim eksperimentima (Čerić, 1990)

4. IZGRADNJA KONCEPTUALNIH SIMULACIJSKIH MODELA

Brojne su metode za izgradnju i prikaz konceptualnih modela simulacije diskretnih događaja. U ovom su poglavlju opisani svrha i osnovne karakteristike konceptualnog simulacijskog modeliranja te tipovi konceptualnih modela. Zatim su podrobno opisani dijagrami ciklusa aktivnosti kao veoma jednostavna i prikladna grafička metoda za izgradnju konceptualnih simulacijskih modela. Dani su njezini osnovni koncepti, način izgradnje i funkcionalanja, a opisan je i jedan primjer primjene metode. Na kraju je komentirana potpunost prikaza modela pomoću dijagrama ciklusa aktivnosti.

4.1. KONCEPTUALNI SIMULACIJSKI MODELI

Izgradnja konceptualnih simulacijskih modela je prvi korak simulacijskog modeliranja. Važnost konceptualnih simulacijskih modela (Paul i Čerić, 1993) je da:

- (1) izdvoje najvažnije karakteristike sistema,
- (2) opišu elemente sistema i njihovo međudjelovanje,
- (3) pomognu u komunikaciji onih koji razvijaju model i onih koji se koriste njime,
- (4) pomognu u razvijanju računarskog modela (programa).

Osnovna svrha konceptualnih modela je da omoguće strukturiranje problema, i tako služe kao alat za razmišljanje o problemu i za njegovo bolje razumijevanje. Kao što je i ljudski jezik sredstvo za razmišljanje tako su i konceptualni modeli sa simbolima kojima se koriste alat koji omogućuje opažanje, grupiranje i povezivanje složenih fenomena. Dakle, oni pružaju odgovarajuću okolinu za razvoj mentalnih aktivnosti povezanih za rješavanje problema.

Mogućnosti čovjeka da istovremeno obradi velike količine povezanih informacija vrlo su ograničene, i to najviše zbog ograničenog kapaciteta njegove kratkotrajne memorije. Stoga se složene strukture i procesi moraju prikazivati kao mali konceptualno zatvoreni segmenti, a oni se zatim hijerarhijski povezuju. Za prikazivanje modela razumno se koristiti sistemima simbola na visokom semantičkom nivou, koji opisuju grupe povezanih konceptata (Kreutzer, 1986).

Konceptualni modeli sadrže grubi opis sistema i njegovu razradu u module. Oni su most između ideja o strukturi i načinu rada sistema te rigoroznih programa za računalo koji omogućuju simulaciju ponašanja sistema. Teško je, naime, očekivati da

se odjedanput može napraviti prijelaz između identifikacije sistema i detaljnog opisa simulacijskog programa. U simulacijskom modeliranju to je posebno otežano zbog složenosti uzrokovane vremenskim razvojem sistema i paralelizmom akcija u sistemu. Stoga se ni konceptualni simulacijski modeli ne smiju promatrati kao statički objekti već kao objekti s dinamičkim paralelnim međudjelovanjem.

Od kvalitetnog konceptualnog modela se očekuje:

- (1) jednostavan, prirodan, lako razumljiv i nedvosmislen prikaz elemenata sistema,
- (2) velike izražajne mogućnosti modeliranja,
- (3) modularan i fleksibilan prilaz koji omogućuje jednostavne i sigurne izmjene modela.

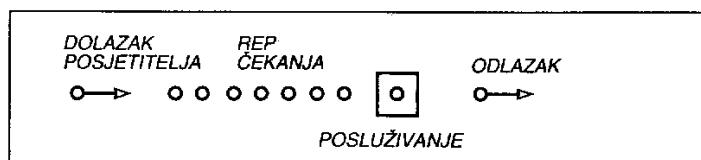
Konceptualno simulacijsko modeliranje obuhvaća niz različitih metoda, od kojih su neke razvijene kao sasvim manuelne metode, a neke se koriste računalom. Osnovne grupe metoda za konceptualno simulacijsko modeliranje (Paul i Čerić, 1993) jesu:

(1) Grafičke metode

Grafičke metode koriste se za prikaz sistema simbola povezanih u dijagrame. Te su metode osobito prikladne zato što omogućuju da konceptualno blisko povezane elemente prikazuju fizički blizu. Za razliku od sekvenčnog prikaza kod proceduralnih metoda, ovdje se međudjelovanje elemenata sistema prikazuje mnogo lakše u dvije dimenzije. Paralelizam čovjekovog vizualnog sistema pri tom olakšava obuhvaćanje i razumijevanje grafičkog prikaza modela.

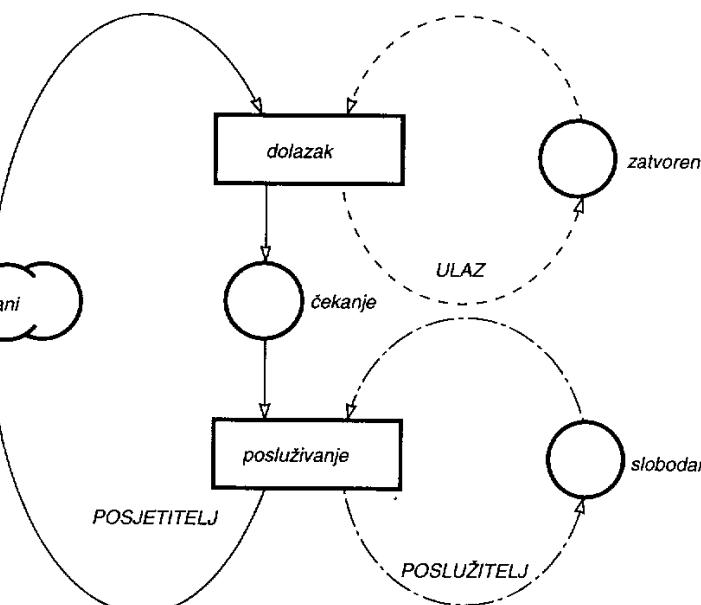
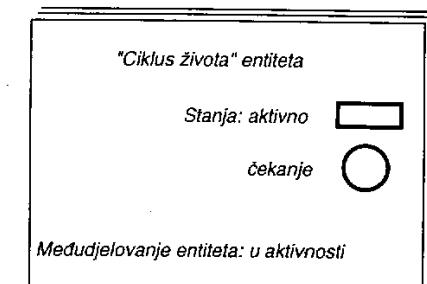
Među grafičkim metodama najpoznatija su različita proširenja Petrijevih mreža, dijagrami ciklusa aktivnosti, grafovi dogadaja i dijagrami simulacijskih jezika za međudjelovanje procesa (GPSS, SLAM i sl.).

Zbog važnosti grafičkih metoda konceptualnog modeliranja (Evans, 1988; Čerić i Paul, 1992) za ilustraciju ćemo navesti četiri različita prikaza modela jednostavnog sistema s repom sa jednim mjestom posluživanja (slika 4.1). Na slici 4.2. prikazan je *dijagram ciklusa aktivnosti* sistema koji sadrži cikluse života entiteta sistema koji međudjeluju u aktivnostima. Na slici 4.3. prikazana je *proširena Petrijeva mreža* sistema sa procesom posjetioca i ciklusom posluživanja. Osim aktivnosti i repova čekanja, u njoj su opisani i događaji u sistemu. Na slici 4.4. prikazan je *GPSS blok dijagrami* sistema koji sadrži proces privremenog entiteta posjetilac, dok su stalni entiteti prikazani kao parametri određenih blokova modela. Na slici 4.5. vidi se *graf dogadaja* sistema koji se sastoji od svih tipova događaja u sistemu te veza među njima (protok vremena između događaja, i uvjeta za početak događaja).



Slika 4.1. Sistem s repom i jednim mjestom posluživanja

4.1. KONCEPTUALNI SIMULACIJSKI MODELI



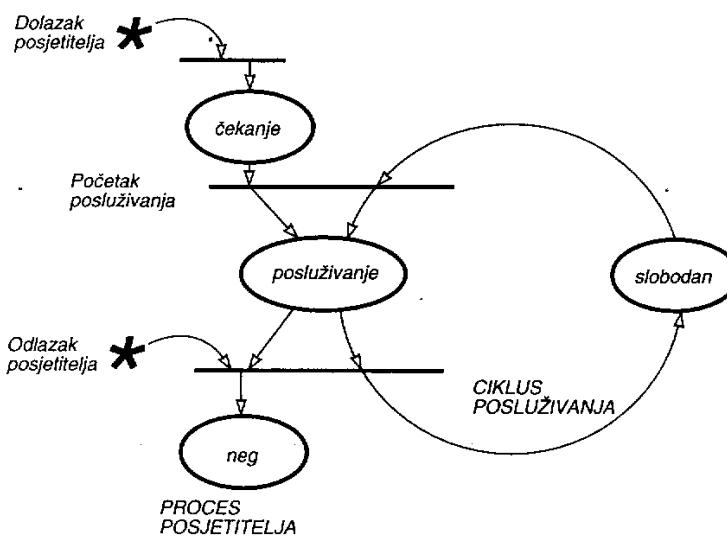
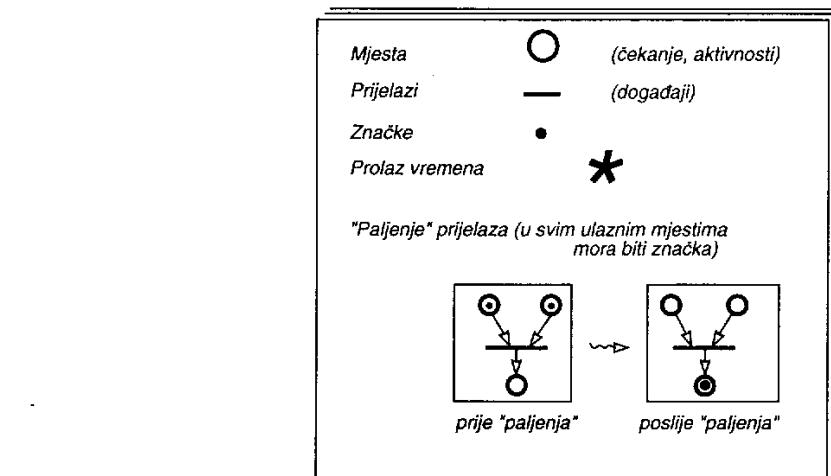
Slika 4.2. Dijagram ciklusa aktivnosti sistema

(2) Proceduralne metode

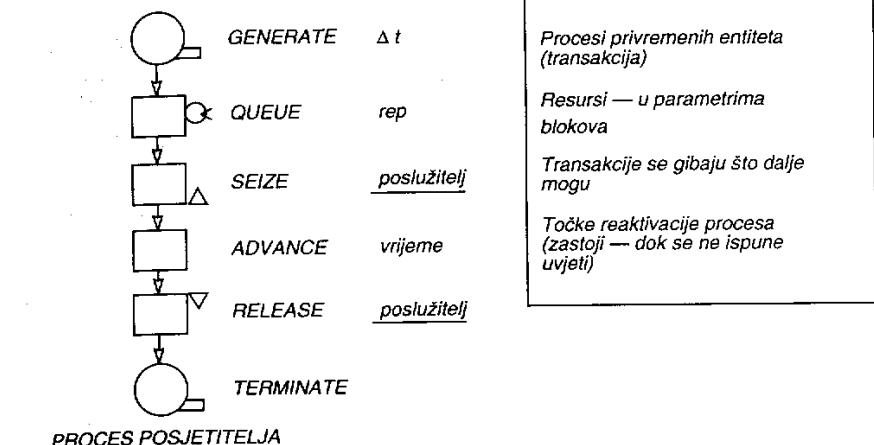
Proceduralne su metode jedna vrsta pseudo-koda koji služi za sekvenčnalni opis procedura što opisuju rad sistema. Najpoznatije proceduralne metode su procesno orijentirani specifikacijski jezik modela, jezik za specifikaciju uvjeta i jezik za prikaz shema.

(3) Kombinirane grafičke i proceduralne metode

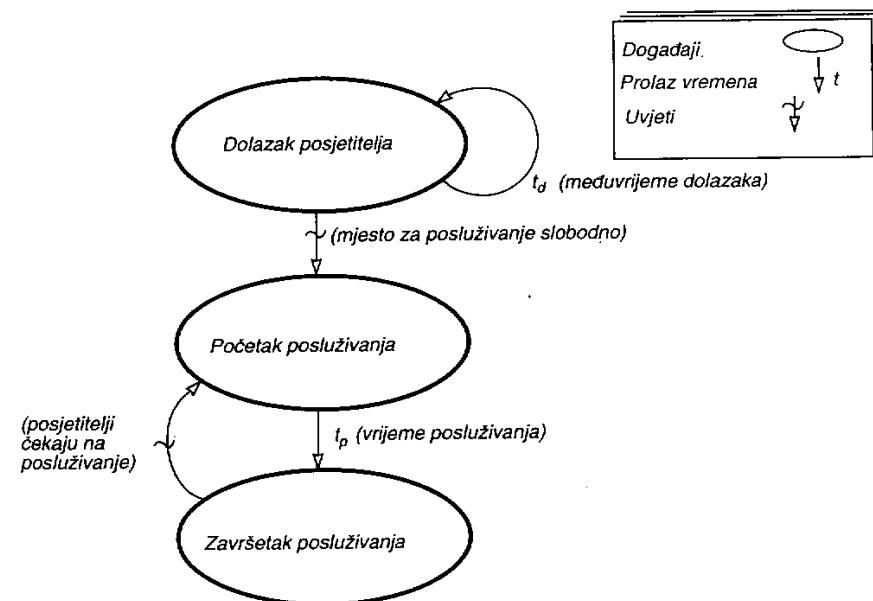
Kombinirane grafičke i proceduralne metode kombiniraju preglednost grafičkih metoda i preciznost opisa proceduralnih metoda koje detaljnije opisuju grafičke module. Najpoznatije su metode hijerarhijske dekompozicije i opis pomoću teorije sistema.



Slika 4. 3. Proširena Petrijeva mreža sistema



Slika 4.4. GPSS blok dijagram sistema



Slika 4.5. Graf dogadaja sistema

(4) Metode interaktivnih upitnika

Metode interaktivnih upitnika su softverski sistemi koji posjeduju osnovna znanja o simulaciji. Pomoću interaktivnog pristupa izvodi se sistematski i strukturirani intervju korisnika koji davanjem odgovora opisuje karakteristike simulacijskog modela. Ovakav pristup dovodi do baze podataka o modelu na temelju koje se u pravilu stvara simulacijski program pomoću automatskih generatora programa.

Najpoznatije metode ovog tipa su metode temeljene na dijagramima ciklusa aktivnosti, mješani prilazi s korištenjem različitih simulacijskih strategija i metode temeljene na tehnikama umjetne inteligencije (korištenje ekspertnih sistema ili upita u prirodnom jeziku).

4.2. DIJAGRAMI CIKLUSA AKTIVNOSTI

Podrobno ćemo prikazati opis konceptualnog modela dijagramom ciklusa aktivnosti (Pidd, 1984). To je grafička metoda koju odlikuje mali broj veoma jednostavnih simbola za opis modela i vrlo jednostavno prikazivanje strukture modela. Ova metoda korištena je za razvoj modela raznovrsnih tipova sistema, a pokazala se prikladnom i za automatsko generiranje simulacijskih programa.

Dijagrami ciklusa aktivnosti (DCA) osobito su prikladni za probleme koji se mogu prikazati kao strukture *repova čekanja*, odnosno kao sistemi masovnog posluživanja. Iako se koriste veoma jednostavnim simbolima, oni omogućuju izgradnju konfiguracija koje prikazuju prilično složene sisteme.

4.2.1. Osnovni koncepti

Dijagrami ciklusa aktivnosti opisuju *životni ciklus* objekata koji postoje u sistemu. Ciklus se u DCA prikazuje samo dvama osnovnim simbolima: aktivnosti i repa čekanja.

Načet ćemo osnovne koncepte DCA.

Entitet

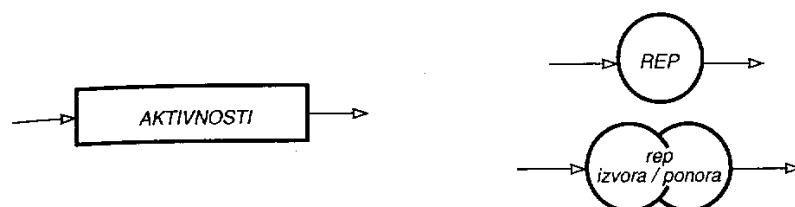
Entitet (objekt) *komponenta je sistema* koju trebamo modelirati i koja zadržava svoj identitet u toku vremena u kojem je u simulacijskome modelu. U svakom času entitet može biti u jednom od dva moguća stanja: aktivnom stanju ili repu čekanja (pasivnom stanju). Entitet u aktivnom stanju angažiran je s drugim entitetima istog ili različitog tipa u izvođenju aktivnosti koja traje neko vrijeme. Entitet u repu čekanja ne sudjeluje u aktivnosti, već čeka da se stvore uvjeti za početak neke aktivnosti.

U DCA se aktivnosti prikazuju pravokutnikom, a repovi čekanja kružnicom (izuzetno se rep izvora/ponora privremenih entiteta pokazuje dvjema kružnicama koje se preklapaju), slika 4.6.

Aktivnost

Aktivno stanje (aktivnost) realizira se *angažiranjem entiteta* (najčešće različitih vrsta) koji međusobno *kooperiraju* (suraduju). Tako se aktivnost iskrcaja tereta iz broda može odvijati samo ako su prisutni i raspoloživi (tj. nisu zauzeti) ovi entiteti: brod sa teretom, dizalica, dok.

4.2. DIJAGRAM CIKLUSA AKTIVNOSTI



Slika 4.6. Simboli koji se koriste u DCA

Trajanje aktivnosti se u simulaciji izračunava *prije početka odvijanja same aktivnosti*. Čak i onda ako je trajanje aktivnosti slučajna veličina, vrijednosti trajanja mogu se unaprijed izračunati generiranjem slučajnih varijabli (tj. izvlačenjem vrijednosti trajanja aktivnosti u skladu s njihovim vjerojatnostima iz odgovarajuće razdiobe vjerojatnosti). To je dakle uzimanje uzorka iz poznate razdiobe vjerojatnosti, a ne analiza razloga trajanja aktivnosti u specifičnim okolnostima. Dakle, pretpostavlja se da je cijela raspoloživa informacija o trajanju aktivnosti komprimirana i sadržana u odgovarajućoj razdiobi vjerojatnosti trajanja aktivnosti.

Zbrajanjem poznatog vremena početka aktivnosti i njezina izračunanoj trajanju dobiva se planirano vrijeme završetka aktivnosti. Isti tip aktivnosti može nanovo biti aktiviran i dok traje prethodna aktivacija, odnosno u istom času može biti aktiviran proizvoljan broj puta (s kojim god početnim časovima aktivacije).

Rep čekanja

Rep čekanja, ili pasivno stanje, jest stanje u kojem entitet nije uključen *ni u kakvu aktivnost* te čeka na sudjelovanje u dalnjim aktivnostima. U svakom repu može čekati proizvoljni broj entiteta istog tipa. Vrijeme koje će entitet provesti čekajući u repu ne može se unaprijed odrediti jer ono ovisi o ostvarenju uvjeta za početak aktivnosti, tj. oslobođanju i stavljanju na raspolaganje za ovu aktivnost svih entiteta potrebnih za početak aktivnosti. Vrijeme čekanja u repu, dakle, posljedica je dinamike razvoja sistema koju dobivamo samom simulacijom rada sistema.

Primjer repa za čekanje je čekanje brodova na iskrcaj dok se ne oslobode i stave brodu na raspolaganje svi entiteti potrebni za iskrcaj. (Nije dovoljno samo da ti entiteti budu slobodni, jer možda će oni prethodno biti stavljeni na raspolaganje drugim brodovima koji duže čekaju ili imaju veći prioritet.)

Osim u stvarnim (fizičkim) repovima čekanja, entiteti mogu biti i u zamišljenim repovima čekanja. Na primjer, korisnici na raznim terminalima računarskog sistema mogu čekati na određene resurse sistema (centralnu procesnu jedinicu, eksternu memoriju). Korisnici, naravno, nisu u fizički jedinstvenom repu čekanja, već su to samo zahtjevi koje su oni poslali u sistem. Isto vrijedi npr. i za strojeve koji čekaju na određeni alat. Takva stanja čekanja mogu se, u apstraktnom smislu, također obuhvatiti pojmom repa čekanja.

4.2.2. Način izgradnje i funkcioniranja dijagrama ciklusa aktivnosti

Osnovni principi i konvencije za izgradnju DCA jesu:

(1) CIKLUSI ŽIVOTA ENTITETA

U DCA se za svaku klasu entiteta koja u modelu postoji izgrađuje poseban ciklus života (ciklus aktivnosti). Kroz ciklus života prolaze pojedinačni entiteti te klase u različitim fazama. U isti čas u ciklusu života može biti proizvoljan broj entiteta u bilo kojim fazama ciklusa.

* Ciklus života entiteta crta se uvijek kao zatvoren kružni ciklus.

* Zbog konvencije u svakom se ciklusu života entiteta repovi čekanja i aktivnosti u koje je entitet uključen striktno alterniraju (izmjenjuju). Ako je potrebno, stvaraju se pomoćni (fiktivni) repovi čekanja da bi se to postiglo (npr. u više uzastopnih aktivnosti u kojima su uključeni isti entiteti), pa takvi repovi mogu biti i trajno prazni (fiktivni repovi).

* Posebno se uvodi entitet dolaska zahtjeva za posluživanje u model (npr. dolazak brodova u luku). Taj entitet omogućuje održavanje želenog ritma (dinamike) dolaska zahtjeva u model. Entitet dolaska je "logički" tip entiteta, za koje postoji samo jedan entitet. Taj entitet je na početku simulacije u odgovarajućem repu čekanja na početak aktivnosti dolaska. Kada aktivnost dolaska počne, rep čekanja tog entiteta se isprazni, jer entitet dolaska sudjeluje u aktivnosti dolaska. Rep čekanja se napuni tek kada je aktivnost dolaska završena (tj. upravo pošto prođe vrijeme između dvaju uzastopnih dolazaka entiteta u sistemu), i time se omogućuje novi dolazak entiteta u sistem. Drugi entitet potreban za početak aktivnosti dolaska je privremeni entitet čiji jedinici ima proizvoljno mnogo u repu dolaska, tako da aktivnost dolaska može početi čim je logički element dolaska u svom repu. Prema tome, upravo trajanje aktivnosti dolaska daje ritam uzastopnim dolascima zahtjeva za posluživanje (bez entiteta dolaska zahtjevi bi neprekidno dolazili u model).

* U ciklusu života privremenih entiteta (zahtjeva za posluživanje) obavezno je i rep dolaska i odlaska tih entiteta iz sistema. Ciklus života trajnih entiteta (resursa) ne sadrži dolaske ni odlaske tih entiteta iz sistema, jer su entiteti u njemu trajno prisutni.

(2) DCA CIJELOG MODELA

Potpuni DCA sastoji se od kombinacije svih individualnih ciklusa života klase entiteta koji u modelu postoje. Mjesta međudjelovanja pojedinačnih ciklusa života entiteta su aktivnosti u kojima entiteti tih klasa kooperiraju. To su ujedno i mesta natjecanja pojedinačnih entiteta iste klase za dobivanje trajnih entiteta (resursa) potrebnih za obavljanje aktivnosti.

* Konvencija je da se u DCA tokovi entiteta svake klase crtaju drukčije (punom linijom, crtkano, u boji i sl.) zbog lakše vizualne identifikacije pojedinačnih ciklusa života.

* Zbog što veće čitkosti DCA, dogovoreno je da se sve ulazne strelice u aktivnosti (tj. uvjeti početka aktivnosti) crtaju s iste strane pravokutnika koji označava tu aktivnost. Isto vrijedi i za izlazne strelice aktivnosti.

(3) MEĐUDJELOVANJE ENTITETA U AKTIVNOSTIMA

Aktivnosti u modelu odvijaju se u točkama međudjelovanja različitih klase entiteta, odnosno točkama kontakta njihovih ciklusa života. Aktivnosti ovako funkcioniраju.

* Da bi aktivnost mogla početi, nužno je da svaki rep čekanja na početak te aktivnosti sadrži bar po jedan entitet, odnosno onoliko entiteta koliko je za tu aktivnost potrebno. Vrijedi i obrat, tj. ako je u svakom repu čekanja na početak neke aktivnosti prisutan bar po jedan entitet, tada će aktivnost i početi.

4.3. PRIMJER DIJAGRAMA CIKLUSA AKTIVNOSTI

* Kada aktivnost završi, svi entiteti uključeni u nju oslobađaju se i šalju u odgovarajuće repove čekanja koji slijede poslije ove aktivnosti u ciklusu života entiteta.

Tako DCA preko točaka međudjelovanja ciklusa života pojedinačnih klase entiteta sinkronizira i upravlja gibanjem entiteta u modelu, tj. dinamikom odvijanja simulacije.

Ovakav način funkcioniranja aktivnosti omogućuje da se odvijaju i paralelne (istovremene) aktivnosti ne samo različitih tipova već i istog tipa, budući da entiteti mogu pokrenuti odvijanje iste bez obzira na to je li ona već u toku (višestruko aktiviranje iste aktivnosti od različitih pojedinačnih entiteta).

Samo kada se to želi spriječiti, npr. pri aktivnosti dolaska, uvođe se specijalne "logičke" klase entiteta (npr. entitet "dolaska") u kojima postoji samo jedan entitet. Čim je taj entitet angažiran u nekoj aktivnosti ona se ne može paralelno odvijati za druge entitete jer je jedan od uvjeta i to da na nju mora čekati i taj "logički" entitet (koji je međutim zauzet do kraja trajanja upravo te aktivnosti).

(4) ATRIBUTI ENTITETA

Za odvijanje nekog stvarnog procesa, odnosno za simulaciju rada odgovarajućeg modela, mogu imati značenje neke karakteristike (atributi) pojedinačnih entiteta (npr. količina tereta na pojedinom brodu). Vrijednosti atributa pojedinačnih entiteta u klasi mogu se postavljati odmah pri pojavi entiteta u simulaciji, a mogu se i mijenjati u toku odvijanja simulacije. Isto tako, one mogu upravljati dalnjim tokom pojedinačnih entiteta (npr. kada količina tereta na brodu pri iskrcaju padne na nulu, iskrcaj je gotov, pa brod i dizalica prelaze na daljnje operacije u svojim ciklusima života). Kao i kod vremena trajanja aktivnosti, vrijednost atributa entiteta koji su slučajne variable određuje se uzimanjem uzorka iz odgovarajuće razdiobe vjerojatnosti, bez analize razloga za vrijednost atributa u pojedinačnim slučajevima.

(5) PRIORITETI AKTIVNOSTI

Ako poslije čekanja u nekom repu čekanja ima više mogućih aktivnosti u koje se entitet nakon završetka čekanja može uključiti, tada se mogu definirati različiti prioriteti za uključivanje entiteta u pojedine aktivnosti (npr. konobar u baru može ići ili na posluživanje gosta koji čeka piće ili na pranje čaša). Prioriteti razrješavaju i slijed izvođenja istovremenih aktivnosti koje nemaju zajedničke entitete.

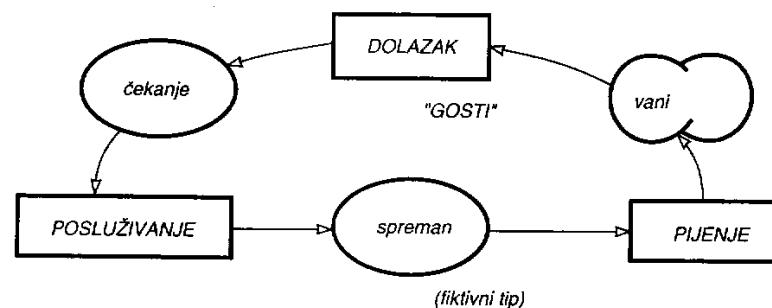
4.3. PRIMJER DIJAGRAMA CIKLUSA AKTIVNOSTI

Osnovne principe i konvencije za izgradnju DCA te način njegova funkcioniranja demonstrirat ćemo DCA modelom bara za izдавanje pića. Ovim se primjerom koristilo više autora (npr. Paul i Balmer, 1991) i izabran je zato što je poznat, lako razumljiv i dovoljno jednostavan, tako da se možemo koncentrirati na njegov DCA model.

Jednostavan model bara

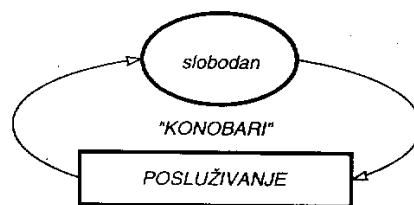
U baru poslužuje određeni fiksni broj konobara kojima je na raspolaganju neograničena količina pića. U bar dolazi proizvoljni broj gostiju. Gosti koji dolaze popiju po jedno piće i odlaze iz bara.

Ciklus života gosta u baru prikazan je na slici 4.7. Taj ciklus vrijedi za cijelu klasu entiteta "gosti", tj. za sve pojedine goste koji dolaze u bar. Ciklus je prikazan u zatvorenom kružnom obliku, i u njemu se uzastopno izmjenjuje rep čekanja gostiju



Slika 4.7. Ciklus života klase entiteta "gosti"

izvan bara na dolazak u bar (to nije čekanje na ulazak u bar, već je tako riješen pravilan uzastopni dolazak gostiju), aktivnost dolaska gostiju, rep čekanja na posluživanje, aktivnost posluživanja, fiktivni rep gostiju spremnih na pijenje pića, aktivnost pijenja i rep odlaska iz bara. Gosti su privremeni entiteti (zahtjevi za posluživanje) kojih može biti proizvoljno mnogo. Odgovarajući ciklus života konobara prikazan je na slici 4.8. Konobari su trajni entiteti (resursi) i ima ih ograničen broj.

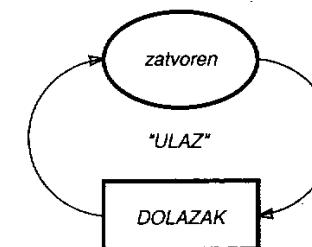


Slika 4.8. Ciklus života klase entiteta "konobari"

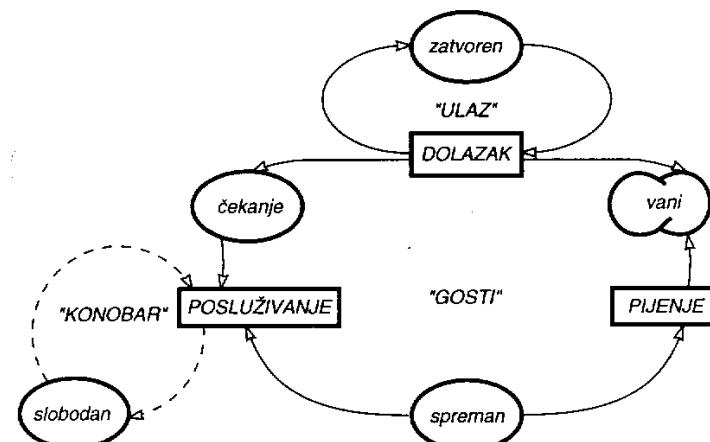
Ciklus života specijalnog logičkog entiteta "ulaz", jedinog entiteta te klase, prikazan je na slici 4.9. Bez postojanja ciklusa života entiteta "ulaz", gosti bi neprekidno dolazili iz repa "vani" u aktivnost dolaska. Ovako je, međutim, uvjet za početak aktivnosti dolaska to da je bar jedan gost u repu "vani" te da je entitet "ulaz" u repu "zatvoren", a on može biti u tom repu tek pošto završi aktivnost dolaska. To osigurava da se u aktivnosti dolaska ne može paralelno (istovremeno) odvijati više od jednog dolaska gosta.

Potpuni DCA ovog jednostavnog modela bara za piće prikazan je na slici 4.10, uz poštovanje konvencija o različitim oznakama toka u ciklusu života različitih klasa entiteta, te zajedničkim mjestima ulaska i izlaska tokova u aktivnosti i iz njih. Mesta međudjelovanja pojedinih ciklusa života entiteta su aktivnosti DOLAZAK i POSLUŽIVANJE. U aktivnosti DOLAZAK kooperira entitet klase "gost" s jedinstvenim entitetom "ulaz", dok u aktivnosti POSLUŽIVANJE kooperira entitet klase "konobar" s entitetom klase "gost". Aktivnost POSLUŽIVANJE je i mjesto natjecanja različitih entiteta iz klase "gost" (zahtjevi za posluživanje) za entitete (resurse) klase "konobar" koji omogućuju ispunjenje njihovih zahtjeva za posluživanjem.

4.3. PRIMJER DIJAGRAMA CIKLUSA AKTIVNOSTI



Slika 4.9. Ciklus života logičkog entiteta "ulaz"



Slika 4.10. DCA jednostavnog bara

DCA bara za posluživanje piće funkcioniра tako da aktivnost dolaska gostiju počinje čim je u repu "vani" gost (a taj rep je uvijek pun jer gostiju ima proizvoljno mnogo) i kada je entitet "ulaz" u repu "zatvoren" (što je na početku simulacije ispunjeno). To je ujedno jedina aktivnost koja na početku simulacije ima ispunjene sve uvjete za aktivaciju. Sljedeći dolazak gosta u bar dogada se tek kada je entitet "ulaz" završio prethodnu aktivnost dolaska i vratio se u rep "zatvoren" (što slijedi automatski).

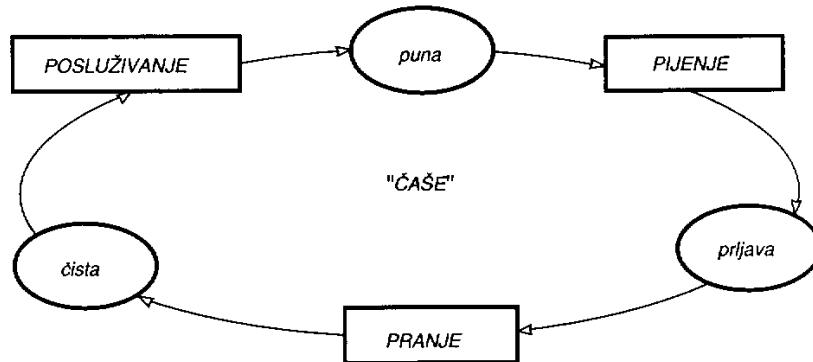
Nakon završetka aktivnosti DOLAZAK gost ide u rep "čekanje" gdje čeka na posluživanje. Čim je neki konobar slobodan (a na početku simulacije su svi slobodni), on preuzima posluživanje gosta koji čeka. Na završetku posluživanja oslobađaju se i gost i konobar. Pri tome konobar ostaje slobodan za sljedeće posluživanje, a gost ide dalje u fiktivni rep "spreman" iz kojeg odmah ide u aktivnost PIJENJE i nakon njezina završetka ide u rep "vani" (tj. odlazi izvan sistema).

Složeniji model bara

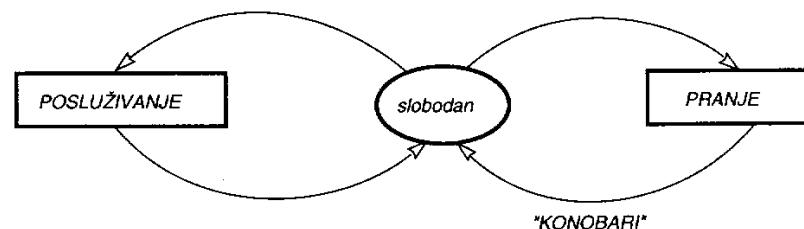
U baru je i određeni fiksni broj čaša. Za posluživanje gosta potreban je slobodan konobar i čista čaša. Kada je neki konobar slobodan (tj. ne poslužuje goste), a

ima prljavih čaša, konobar pere čaše. Gosti koji dolaze mogu popiti i više od jednog pića (ali za svako piće posebno čekaju nakon što su popili prethodno piće), nakon čega napuštaju bar.

Da bi se napravio model takva bara, treba prije svega uvesti dodatnu klasu entiteta "čaše" te napraviti njihov ciklus života (slika 4.11). Osim toga, u ciklus života konobara uvodi se nova aktivnost PRANJE (slika 4.12). Kompletan DCA složenijeg modela bara prikazan je na slici 4.13. U sklopu DCA vidi se i promjena ciklusa života klase entiteta "gosti", koju ćemo sada detaljnije opisati.



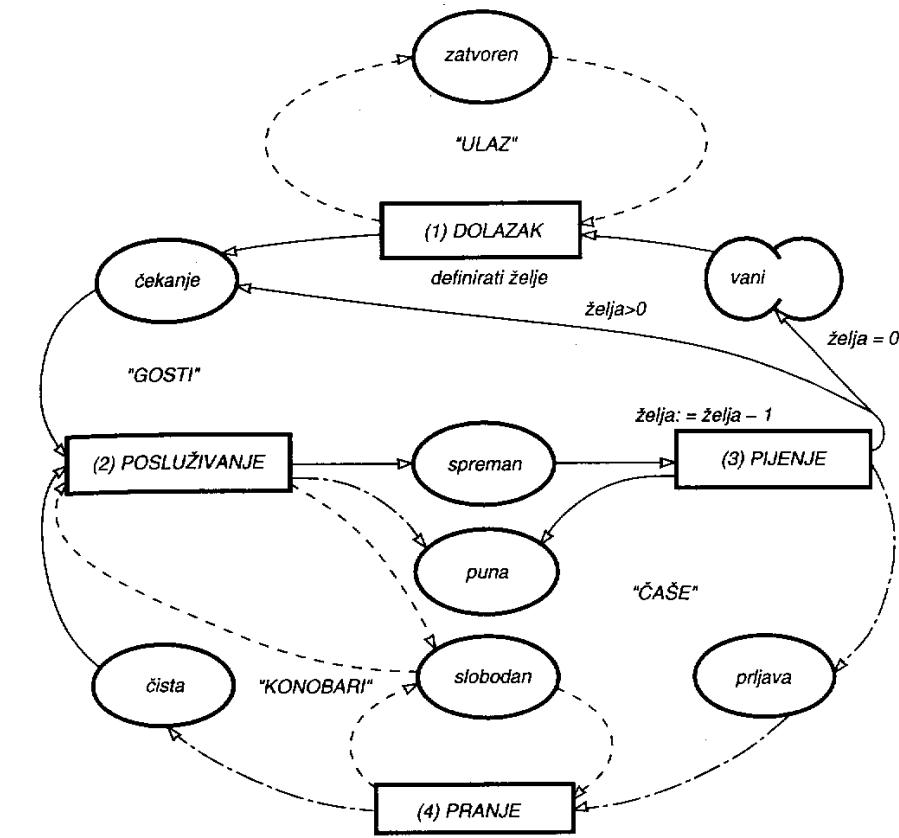
Slika 4.11. Ciklus života klase entiteta "čaše"



Slika 4.12. Izmjenjeni ciklus života klase entiteta "konobari"

Svakom pojedinačnom entitetu "gost" pridružujemo atribut "želja", koji označava broj pića što će ih gost popiti u baru. Čim se pojedini entitet "gost" pojavi u aktivnosti DOLAZAK, pridružuje mu se vrijednost atributa "želja" generiranjem vrijednosti slučajne varijable iz odgovarajuće razdiobe vjerojatnosti (uzimanje uzorka). Svaki put kada gost počinje aktivnost PIJENJE, njegova želja se smanjuje za jedan. Ako nakon pijenja želja još nije nestala (veća je od nule), gost se vraća u rep "čekanje" u sklopu svojeg ciklusa posluživanja. Ako je, međutim, želja za pijenjem nestala (= 0), gost izlazi iz bara.

Primjer pokazuje mogućnost pojave alternativnih stanja entiteta (različiti mogući putovi za entitete iste klase) nakon završetka aktivnosti, te upravljanja tokovima entiteta u DCA pomoću vrijednosti atributa entiteta.



Slika 4.13. DCA složenijeg bara

U ciklusu života konobara vidi se da slobodan konobar može ići bilo na posluživanje gosta, bilo na pranje čaša (ako su svi uvjeti za početak obje te aktivnosti ispunjeni). Dodjeljivanjem prioriteta tim aktivnostima može se osigurati da slijed izbora sljedeće aktivnosti uvijek bude jednak (npr. da konobar uvijek prvo posluži gosta). Na slici 2.8. najveći prioritet, 2 — manji itd.). Primjer ujedno demonstrira snagu DCA za otkrivanje i razrješenje mesta potencijalnih alternativa ili konflikata.

4.4. KORIŠTENJE VARIJABLJI U DIJAGRAMIMA CIKLUSA AKTIVNOSTI

U dijagramima ciklusa aktivnosti varijable se mogu koristiti za prikaz veličina koje se dinamički mijenjaju, i na temelju čijih se vrijednosti donose odluke u različitim fazama

života jednog ili više entiteta (to su globalne varijable dostupne u cijelom DCA, tj. u svim njegovim ciklusima života). Varijable se često koriste i za prikaz trajnih entiteta (resursa) sistema koje je neprikladno ili prekomplificirano prikazati ciklusom života entiteta (Pidd, 1984).

Korištenje varijabli u DCA ilustrirat ćeemo primjerom rada skladišta koje se puni robom što je dovoze kamioni, a prazni se kada prodavaonice traže robu. Ako u skladištu nema mjesta u času kada stigne kamion s robom, tada on mora čekati dok se skladište djelomično ne isprazni. Isto tako, ako je skladište prazno kada stigne zahtjev prodavaonce, tada zahtjev mora čekati dok se skladište ne popuni dovoljno da se roba može izdavati. Konstanta "max" će označavati popunjenošću skladišta iznad koje se skladište više ne može puniti, a konstanta "min" minimalnu popunjenošću skladišta da bi se roba mogla početi izdavati. U času dolaska kamiona generirat će se vrijednost atributa kamiona "dovoz", koja označava količinu robe koju kamion dovozi. Analogno, u času zahtjeva prodavaonice za robom iz skladišta generirat će se iznos tražene količine robe atributom zahtjeva prodavaonice "potražnja".

Najprikladnije rješenje za opis logike rada ovog sistema jest da se uvede varijabla SKLAD koja označava trenutačnu količinu robe na skladištu. Sistem se opisuje DCA-om koji se sastoji od dvaju odvojenih ciklusa života, ciklusa "kamiona" i "prodavaonice" (slika 4.14). Ta dva ciklusa života povezana su varijablom SKLAD koja se u oba ciklusa koristi za ispitivanje uvjeta za početak aktivnosti (tj. završetka boravka u repu), te promjene vrijednosti popunjenošću skladišta (u odgovarajućim aktivnostima).

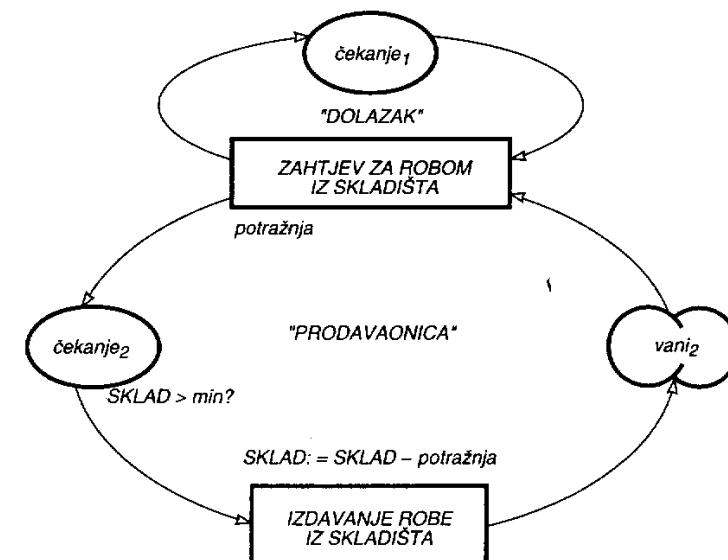
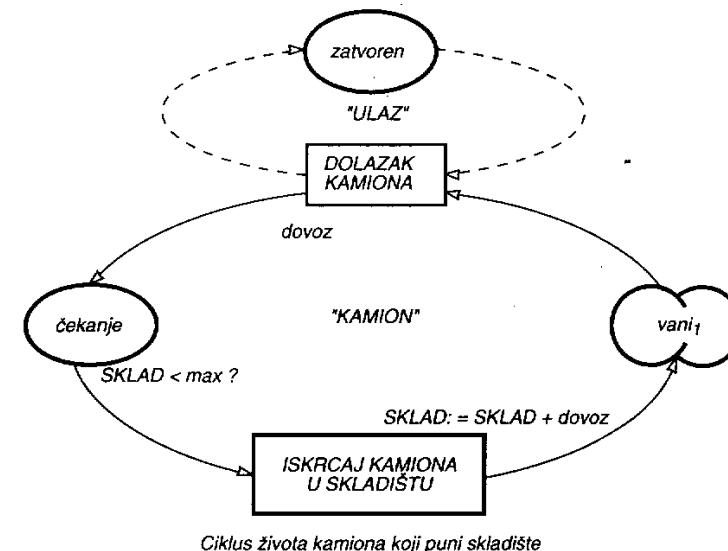
U navedenom primjeru demonstriran je i novi način aktiviranja početka odvijanja aktivnosti, i to preko *uvjeta* za napuštanje repa. Dakle, nije dovoljno da je entitet u određenom repu, već je potrebno da on i ispunjava određene uvjete da napusti rep i time omogući počinjanje aktivnosti koja se logički nastavlja na boravak entiteta u tom repu.

4.5. POTPUNOST PRIKAZA SIMULACIJSKIH MODELA POMOĆU DCA

DCA je osnovna struktura i alat za razvoj konceptualnih simulacijskih modela. Svrlja DCA je da se olakša izgradnja ispravnih simulacijskih modela koji prikazuju *osnovne karakteristike* problema koji modeliramo. Pri tome vizualni prikaz, jednostavnost pravila DCA te njegova modularnost olakšavaju postupnu izgradnju modela i razumijevanje interakcija entiteta modela.

Neke karakteristike sistema koji modeliramo nije lako prikazati DCA-om, iako ih je relativno lako ugraditi u odgovarajući simulacijski program. Tako ujedno čuvamo i čitost i razumljivost DCA modela kao provjerene osnove za daljnje profinjenje programa. Na primjer, u primjeru bara može biti više vrsta čaša za različite vrste pića, a pojedini gosti mogu imati afinitet prema nekim vrstama pića. Te karakteristike možemo dodati u program uvođeci odgovarajuće atribute gostima i čašama i sručavajući te atribute prije početka aktivnosti posluživanja.

Kao što smo vidjeli, DCA je snažan alat za formalni opis osnovnih karakteristika logike rada sistema te za analizu njegova rada. Već se to može pokazati kao vrlo korisno za razumijevanje rada sistema koji modeliramo, pa je tako i djelomično rješenje problema zbog kojeg se simulacijsko modeliranje provodi.



Slika 4.14. DCA skladišta — demonstracija interakcije ciklusa života pomoći varijabli

5. STRATEGIJE IZVOĐENJA SIMULACIJE

Razvoj simulacijskog modela u vremenu jedan je od ključnih problema simulacije. O načinu njegova rješenja ovisi i ispravnost i efikasnost izvođenja simulacije. Ovdje opisujemo mehanizme pomaka vremena simulacije te različite strategije izvođenja simulacije koje dovode do različitih prikaza modela i efikasnosti njihova razvoja u vremenu. Detaljno je prikazana strategija trofazne simulacije, koja ujedinjuje jednostavnost prikaza modela i vrlo jasan pristup izvođenju razvoja simulacije u vremenu. Demonstrirano je i ručno izvođenje simulacije modela prikazanog dijagramom ciklusa aktivnosti pomoću strategije trofazne simulacije. Opisane su i demonstrirane osnove skupljanja statistike simulacije. Na kraju je ukratko prikazan problem planiranja budućih događaja u simulaciji, te izbor odgovarajućih struktura podataka i algoritama za rješavanje ovog problema bitnog za efikasno izvođenje simulacije diskretnih događaja.

5.1. MEHANIZMI POMAKA VREMENA

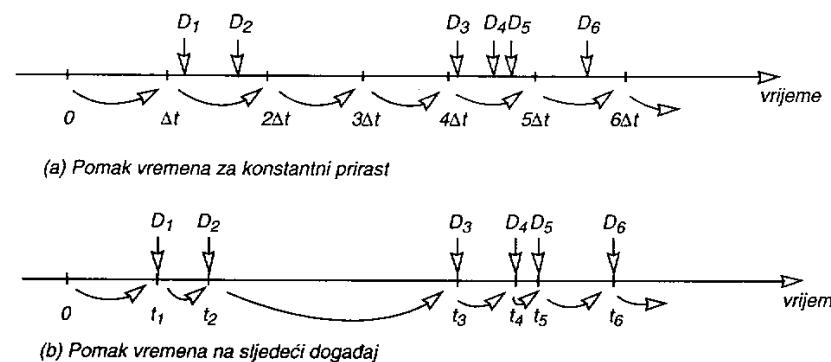
U simulaciji diskretnih događaja koriste se dva osnovna mehanizma pomaka vremena: pomak vremena za konstantni prirast i pomak vremena na sljedeći događaj (Law i Kelton, 1982).

5.1.1. Pomak vremena za konstantni prirast

Vrijeme se pomiče tako da se uvijek dodaje isti prirast vremena. Nakon svakog pomaka vremena (tj. vrijednosti simulacijskog sata) ispituje se da li su se u prethodnom vremenskom intervalu trebali dogoditi neki događaji. Ako je takvih događaja trebalo biti, tada se oni planiraju za kraj intervala.

Pristup je prikazan na slici 5.1a. Vidimo da u prvom intervalu nema događaja, dok su u drugom intervalu dva događaja D_1 i D_2 , a oba se izvode u času $2\Delta t$. Potrebno je dakle odrediti i način izbora slijeda izvođenja događaja D_1 i D_2 .

Opisani pristup je najjednostavniji pristup za pomak simulacijskog vremena, ali on ima određene nedostatke. Pomakom događaja na kraj vremenskog intervala u kojem bi se oni trebali dogoditi uvodi se greška u izvođenje simulacije. Osim toga, događaji koji u stvarnosti nisu istovremeni u ovom se pristupu prikazuju kao istovremeni, i zatim se određuje slijed njihova izvođenja (koji se može razlikovati od slijeda izvođenja u stvarnosti). Smanjenjem vremenskog prirasta te se greške smanjuju, ali se zato povećava vrijeme koje se troši na izvođenje simulacije. Pri tome je sve veći broj vremenskih intervala u kojima nema događaja, pa oni uzalud prelaze u simulaciju.



Slika 5.1. Mehanizmi pomaka vremena u simulaciji diskretnih događaja

Ovaj mehanizam simulacije ima najviše smisla primjeniti u situacijama kad se događaji zaista nižu jedan po jedan u konstantnim vremenskim intervalima, npr. u ekonomskim sistemima gdje se u konstantnim vremenskim periodima mijenjaju stanja (npr. periodičko izvođenje kontrole popunjenoštiti skladišta, te stizanje popunjena u skladište).

5.1.2. Pomak vremena na sljedeći događaj

Simulacijski sat se pomiče na vrijeme u kojem će se dogoditi prvi sljedeći događaj (ili više njih). U tom času se događaj izvede i napravi se odgovarajuća promjena stanja sistema; zatim se ponovo ispituje koji će se događaj sljedeći dogoditi itd. Simulacija završava kada više nema dalnjih događaja ili kada je zadovoljen neki unaprijed zadani uvjet završetka simulacije. Takvim se načinom pomaka vremena izbjegava greška u vremenu izvođenja događaja, a ujedno se preskaču i intervali u kojima nema događaja.

Ovaj pristup demonstriran je na slici 5.1b. Vidimo da simulacijski sat redom skače s časa događaja D_1 na čas događaja D_2 itd. Prikazani pristup je doduše komplikiraniji od prethodnoga, ali je zato mnogo efikasniji i ne unosi greške u vremenu izvođenja događaja. On se stoga koristi u svim ključnim simulacijskim jezicima i paketima.

5.2. SIMULACIJSKE STRATEGIJE

5.2.1. Pojam simulacijske strategije

Simulacijske strategije su različiti pristupi prikazivanja dinamike simulacije i načina upravljanja izvođenjem simulacije (Evans, 1988). Za razliku od *ad hoc* pristupa kojim se često služe programeri za izradu vlastitog simulacijskog softvera, simulacijske strategije su sistematičan, dobro definiran i pouzdani pristup razvoju simulacijskih jezika i paketa.

Sve simulacijske strategije temeljene su na osnovnim pojmovima simulacije diskretnih događaja – događajima, aktivnostima i procesima. Četiri su osnovne strategije simulacije: planiranje događaja, prelaženje aktivnosti, međudjelovanje procesa i trofazni pristup. Svi simulacijski jezici i paketi temelje se na nekoj od tih strategija ili na njihovim mješavinama.

Prije opisa simulacijskih strategija nešto ćemo detaljnije opisati simulacijski sat i dva osnovna tipa događaja, što će pomoći razumijevanju simulacijskih strategija.

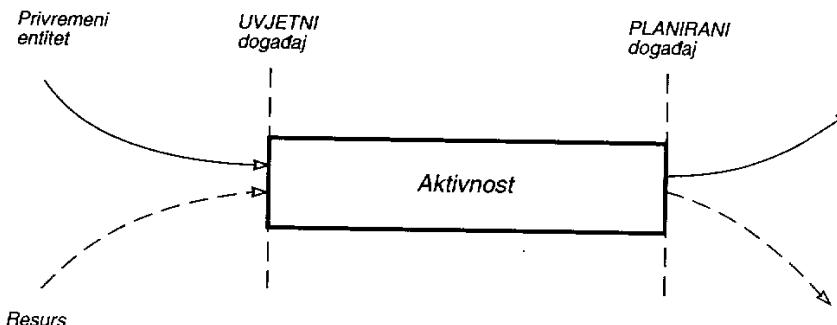
Simulacijski sat pokazuje vrijeme odvijanja simulacijskog eksperimenta prikazanog u vremenskim jedinicama koje modelar slobodno odabire. Vremenska jedinica odabire se prema dinamici odvijanja akcija u simulaciji i željenoj točnosti opisa vremenskog razvoja simulacije. Simulacijski sat u pravilu nema nikakve veze sa stvarnim satom koji mjeri naše vrijeme. Tako se recimo u simulacijskom eksperimentu može u desetak minuta odvijati više godina ili možda samo nekoliko milisekundi razvoja sistema koji model opisuje.

Događaji su promjene stanja sistema koje se dešavaju u jednom času simulacije. Oni mogu biti bezuvjetni ili uvjetni.

Bezuvjetni (planirani) događaji su oni kod kojih se jedini uvjet izvođenja događaja može izraziti samo kao funkcija vremena. To je u pravilu planirano vrijeme završetka neke aktivnosti. Bezuvjetni događaji obično su povezani za oslobadanje resursa koji su bili aktivirani u trajanju aktivnosti. To su, na primjer, događaji dolaska mušterije ili završetka posluživanja.

Uvjetni događaji mogu se dogoditi samo ako je zadovoljen neki uvjet koji nije vremenskog tipa. Obično su povezani za zauzimanje resursa, tj. početak aktivnosti. To je, na primjer, događaj početka posluživanja koji se može dogoditi samo ako postoji zahtjev za posluživanjem a resurs za posluživanje je slobodan.

Na slici 5.2. prikazani su uvjetni i planirani događaji povezani za aktivnosti iz dijagrama ciklusa aktivnosti.



Slika 5.2. Uvjetni i planirani događaji

Ukratko ćemo opisati prve tri spomenute simulacijske strategije, a zatim ćemo detaljno razraditi opis trofaznog pristupa simulaciji. Strategije simulacije demonstrirat ćemo na klasičnom primjeru jednostavnog repa čekanja pred jednim mjestom posluživanja gdje se posjetioци primaju onim redom kojim su došli. Ovaj je primjer detaljnije razrađen i demonstriran u knjizi (Pidd, 1984).

5.2.2. Strategija planiranja događaja

(a) Prikaz dinamike sistema

Dinamika sistema prikazana je pomoću odvijanja niza *bezuvjetnih događaja* (koji se zovu i *bilješke događaja*) različitih tipova u vremenu. Svaki tip događaja prikazan je odgovarajućom *rutinom događaja* koja opisuje niz operacija nad entitetima sistema (tj. promjena stanja) što slijede nakon događaja tog tipa. U ovom pristupu svi uvjetni događaji moraju biti uključeni u rutine događaja.

U primjeru jednostavnog repa čekanja bezuvjetni su događaji (1) 'dolazak posjetilaca' (koji se može planirati nakon prolaska međuvremena dolaska od časa prethodnog dolaska) i (2) 'završetak posluživanja' (koji se može planirati nakon prolaska vremena posluživanja od časa početka posluživanja). Uvjetni događaji su (1) 'posluživalac postaje slobodan' (što se dešava ako je posluživanje završeno a nema sljedećeg posjetioca koji čeka na posluživanje) i (2) 'posluživalac postaje zauzet' (ako posjetilac dolazi a posluživalac je slobodan). Na slici 5.3. prikazana je rutina događaja 'dolazak posjetilaca' koja demonstrira uključivanje uvjetnih događaja ('Da li je posluživalac slobodan', 'zauzmi mjesto posluživanja') i planiranje bezuvjetnih događaja ('dolazak posjetilaca', 'završetak posluživanja').

(b) Upravljanje tokom simulacije

Za upravljanje tokom simulacije koristi se *lista događaja* koja prikazuje dinamički promjenljivu kolekciju bilješki budućih događaja. Svaka bilješka događaja sadrži vrijeme u kojem bi se taj događaj trebao dogoditi i identifikaciju događaja.

Upravljanje tokom simulacije provodi se cikličkim izvođenjem ovih faza:

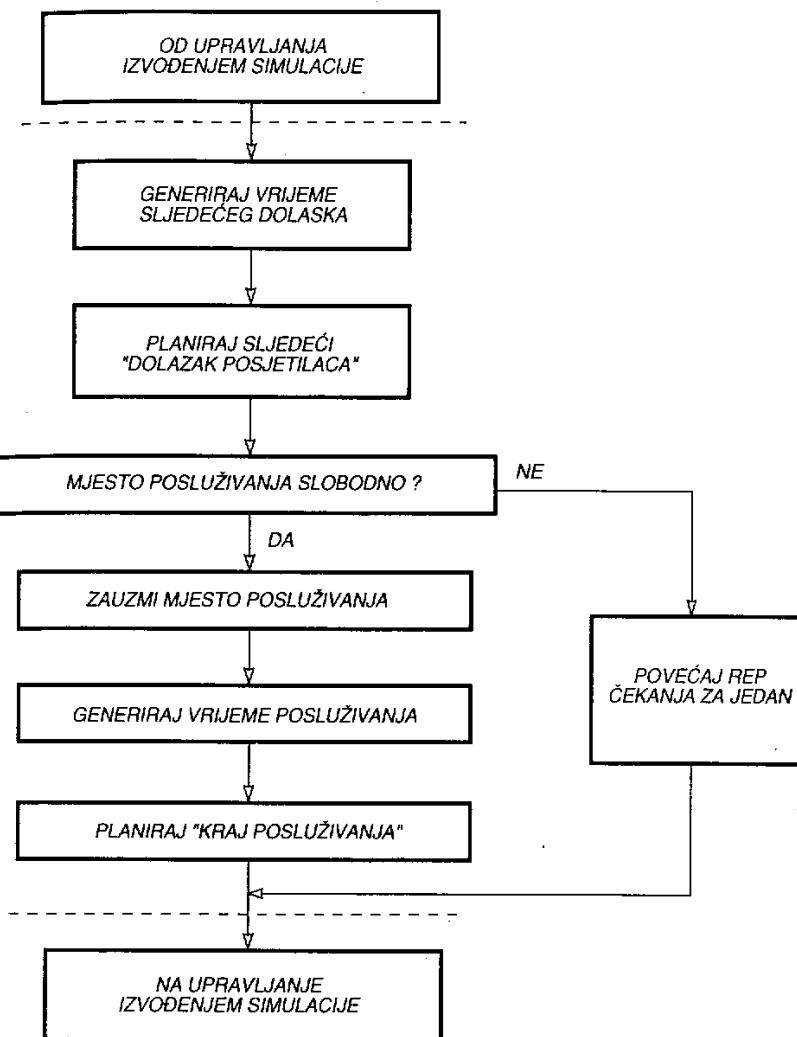
- faza *prelaska vremena*, u kojoj se određuje vrijeme sljedećeg događaja prelaskom ('scan') liste događaja, nakon čega se simulacijski sat postavlja na to vrijeme
- stvaranje *liste sadašnjih događaja*, koja sadrži događaje što se trebaju dogoditi sada
- faza *izvođenja događaja*, koja osigurava izvođenje svih događaja iz liste sadašnjih događaja te izbacivanje njihovih bilješki događaja iz liste sadašnjih događaja nakon izvođenja.

5.2.3. Strategija prelaženja aktivnosti

(a) Prikaz dinamike sistema

Prikaz dinamike sistema temelji se na međudjelovanju različitih tipova entiteta u *aktivnostima* podvrgnutim specificiranim uvjetima. Dinamika sistema prikazana je nizom različitih nezavisnih tipova aktivnosti koje se sastoje od *glave za testiranje* (uvjeti koji moraju biti ispunjeni da bi se aktivnost mogla izvesti) i *akcija* (operacija koje čine samu aktivnost). Poslije svakog događaja, umjesto opisivanja svih operacija koje mu slijede (kao u pristupu planiranja događaja), redom se pokušava izvesti *svaku od aktivnosti* modela. Time ovaj pristup omogućava lakše modeliranje, uz cijenu manje efikasnosti u usporedbi s pristupom planiranja događaja.

U primjeru jednostavnog repa čekanja dinamika sistema je, umjesto sa dva bezuvjetna događaja, prikazana pomoću aktivnosti (1) 'dolazak posjetitelja', (2) 'početak



Slika 5.3. Rutina događaja *DOLAZAK POSJETILACA* za sistem s jednostavnim repom čekanja u strategiji planiranja događaja (prema Pidd, 1984)

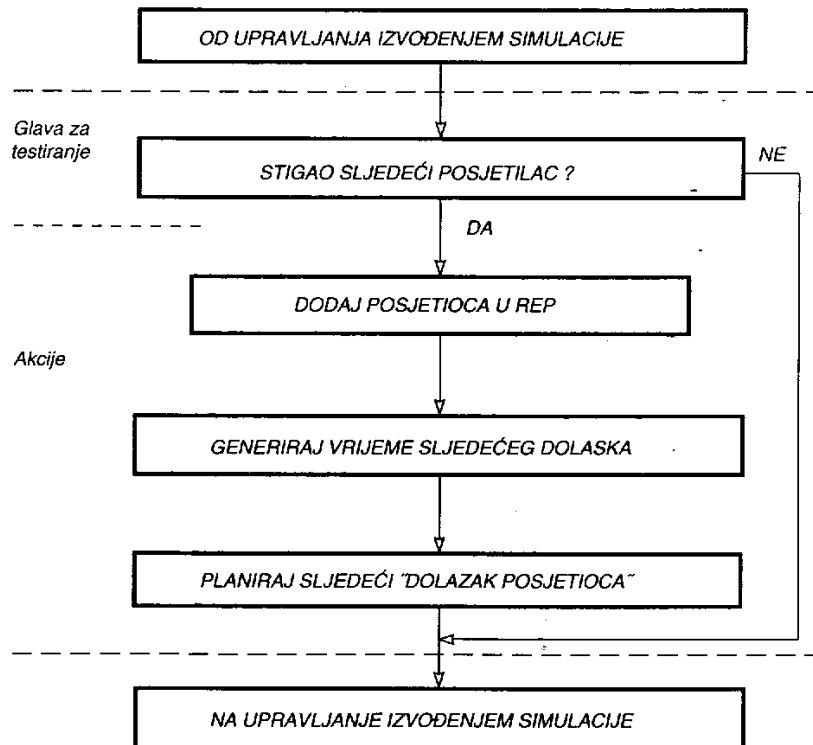
posluživanja' i (3) 'završetak posluživanja'. Na slici 5.4. prikazana je aktivnost 'dolazak posjetitelja', koja ilustrira jednostavnost strukture opisa aktivnosti.

(b) Upravljanje tokom simulacije

Upravljanje tokom simulacije provodi se cikličkim izvođenjem ovih faza:

- faze *prelaženja vremena*, koja je odgovorna za identifikaciju vremena sljedećeg događaja (promjene stanja). Za to se koriste atributi stalnih entiteta, tzv. *vremenske*

ćelije, koji pokazuju vrijeme kada će entitet promijeniti stanje. Tako vremenska ćelija entiteta poslužitelj u primjeru jednostavnog repa čekanja pokazuje vrijeme kada će posluživalac završiti tekuću aktivnost posluživanja. Takve se vremenske ćelije mogu memorirati u obliku liste koja je uređena po vrijednostima.



Slika 5.4. Aktivnost DOLAZAK POSJETILACA za sistem s jednostavnim repom čekanja u strategiji planiranja događaja (prema Pidd, 1984)

– faza prelaženja aktivnosti, koja nastoji naći i izvesti sve aktivnosti što se mogu aktivirati u sadašnjem trenutku, a koje je odabrala faza prelaženja vremena. Aktivnosti za pretraživanje moraju biti uređene po padajućem prioritetu kako bi se dobio ispravan niz operacija u slučaju istovremenih događaja (npr. u slučaju jednostavnog repa posluživanja s praznim repom čekanja, novi posjetilac koji dolazi u istom času kada završava posluživanje mora zapravo doći prije izvođenja završetka posluživanja kako bi već čekao na posluživanje kada posluživalac postane slobodan).

5.2.4. Strategija međudjelovanja procesa

(a) Prikaz dinamike sistema

U ovom pristupu *proces* je gibanje entiteta kroz sistem, od događaja njegova dolaska do događaja njegova odlaska. Za razliku od drugih pristupa, u procesu može prolaziti vrijeme, što omogućuje da se u proces uključi veći broj događaja i aktivnosti. Privremeni entitet prolazi kroz operacije procesa sve dok ne bude blokiran ili zadržan. U slučaju (1) *bezuvjetnog zadržavanja* (npr. čekanja da prođe vrijeme posluživanja) entitet čeka dok ne prođe unaprijed određeno vrijeme, a u slučaju (2) *uvjetnog zadržavanja* (npr. čekanja u repu dok poslužitelj ne postane slobodan) entitet čeka dok mu uvjeti ne dopuste da se giba dalje. Točke u procesu gdje može doći do zadržavanja zovu se *točke reaktivacije*. One također pokazuju odakle će se entitet nastaviti gibati kada njegovo sadašnje zadržavanje bude završeno.

U primjeru jednostavnog repa čekanja može se definirati jedinstven proces 'posjetitelj' koji prikazuje potpuni tok posjetitelja kroz sistem. Taj proces je prikazan na slici 5.5, na kojoj su zvjezdicom označene točke reaktivacije. Ovaj primjer demonstrira uključivanje svih događaja i aktivnosti u jedan jedini proces.

(b) Upravljanje tokom simulacije

Upravljanje tokom simulacije nastoji pomaknuti entitet kroz što je moguće više koraka procesa, a pri tome se koristi dvjema listama *slogova* koje sadrže informacije o identitetu entiteta i njegovoj točki reaktivacije. *Lista sadašnjih događaja* sadrži slogove entiteta koji su prethodno bili bezuvjetno zadržani i planirani za sadašnje vrijeme simulacije, te entiteta koje je zaustavio neki uvjet. *Lista budućih događaja* sadrži slogove bezuvjetno zadržanih entiteta planiranih za neko vrijeme poslije sadašnjeg vremena simulacijskog sata.

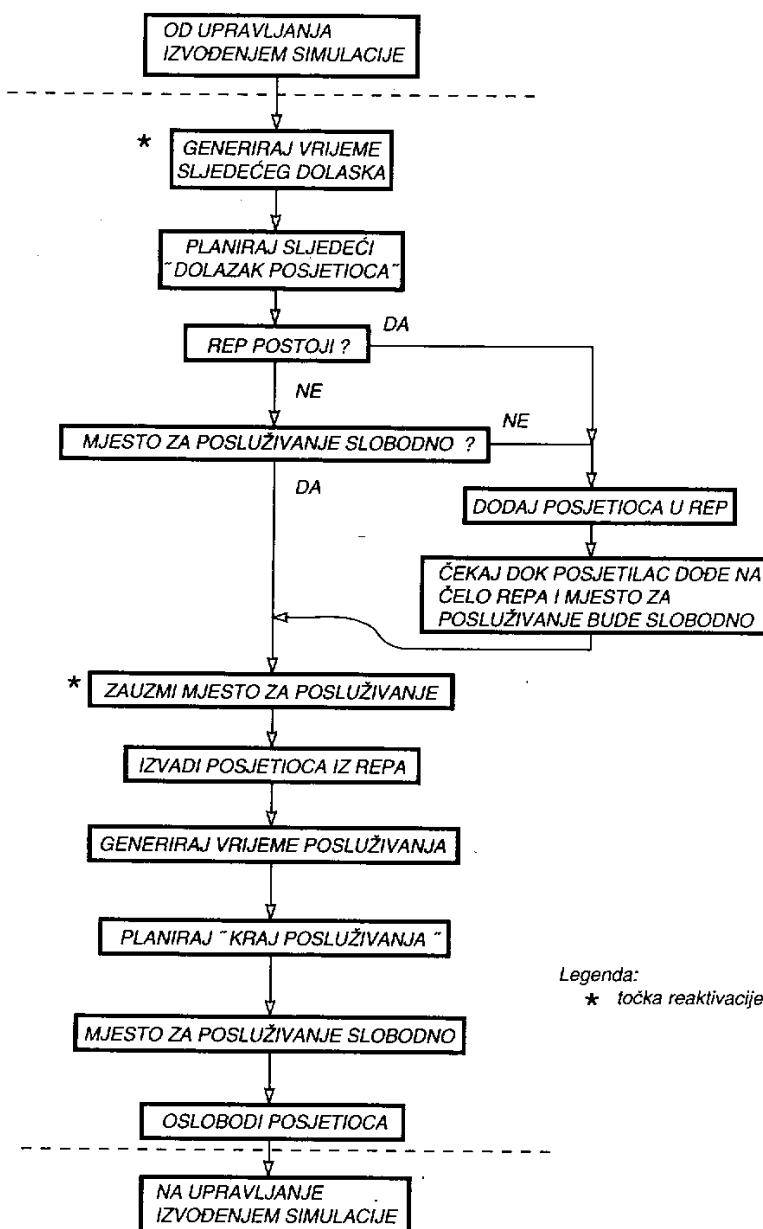
Upravljanje izvođenjem simulacije provodi se cikličkim izvođenjem ovih faza:

- prelaženje budućih događaja, tijekom kojega se u listi budućih događaja traži vrijeme najranije reaktivacije procesa
- pomicanje slogova sadašnjih događaja iz liste budućih događaja u listu sadašnjih događaja
- prelaženje liste sadašnjih događaja, pri čemu se nastoji pomaknuti svaki entitet iz te liste kroz njegov proces sve dok ne dođe do njegova blokiranja. Blokirani entiteti se pri tom stavljaju u odgovarajuće liste. Ovaj postupak provodi se sve dok više nema entiteta iz liste sadašnjih događaja koji se mogu gibati.

Kao što vidimo, proces je prilično prirodan način prikaza rada sistema sa stajališta modelara, ali je za njegovo ostvarenje potreban prilično složen mehanizam upravljanja u toku simulacije.

5.3. TROFAZNA STRATEGIJA SIMULACIJE

Trofazna je strategija (metoda) simulacije profinjenje strategije prelaženja aktivnosti. Ona pri tome čuva jednostavnost prikaza strategije uz povećanu efikasnost izvođenja simulacije. Ova metoda ima jednostavnu, dobro definiranu i modularnu strukturu, pa



Slika 5.5. Proces POSJETILAC za sistem s jednostavnim repom čekanja u strategiji međudjelovanja procesa (prema Pidd, 1984)

je pogodna za učenje simulacije. Budući da omogućuje stvaranje dobro strukturiranog programskog koda, prikladna je i za programiranje simulacije.

Trofazna metoda (Pidd, 1984; Paul i Balmer, 1991) cikličko je izvođenje ovih triju faza:

FAZA A : POMAK NA SLJEDEĆI DOGAĐAJ

(A – ‘advance’ : pomak)

Ispitaju se vremena završetaka svih aktivnosti koje su u toku. Pronalazi se najranije vrijeme završetka aktivnosti, te sve aktivnosti koje u taj čas trebaju završiti. Simulacijski sat se pomakne na to vrijeme.

FAZA B : IZVOĐENJE PLANIRANIH DOGAĐAJA

(B – ‘bound events’ : vezani/planirani događaji)

Izvode se redom svi planirani događaji završetka aktivnosti predviđeni u tom najranijem vremenu i identificirani u fazi A. Pri tome se oslobođe svi entiteti uključeni u te aktivnosti i pomiču se u repove koji u njihovim ciklusima života u DCA slijede nakon završetka aktivnosti.

Dakle, u tom se času oslobođe svi resursi (stalni entiteti) koji se u sistemu mogu oslobođiti (pa stoga slijed izvođenja završetaka aktivnosti nije važan, jer se ne prelazi na sljedeću fazu dok se ne izvedu svi završeci aktivnosti planirani za taj čas).

FAZA C : IZVOĐENJE UVJETNIH DOGAĐAJA

(C – ‘conditional events’ : uvjetni događaji)

Redom se pokušava izvođenje svakog tipa uvjetnog događaja početka aktivnosti (aktivnosti se pretražuju prema prioritetu, tj. prema rastućem rednom broju tipa aktivnosti). Unutar svakog od tipova aktivnosti ta se aktivnost počinje toliko puta dok su ispunjeni (ako jesu), tj. dok se ne iscrpe mogućnost pokretanja aktivnosti tog tipa. Nakon tогa se prelazi na sljedeći tip aktivnosti.

Pri izvođenju početka aktivnosti u aktivnost se pomiču svi potrebni entiteti iz repova čekanja koji su u njihovim ciklusima života u DCA ispred te aktivnosti (izbor između više entiteta u istom repu koji mogu pokrenuti aktivnosti može biti po slijedu dolaska, po prioritetu i sl.). Zatim se izračuna vrijeme trajanja aktivnosti te na temelju toga i vrijeme završetka aktivnosti. Za čas završetka aktivnosti incira se odgovarajući planirani događaj.

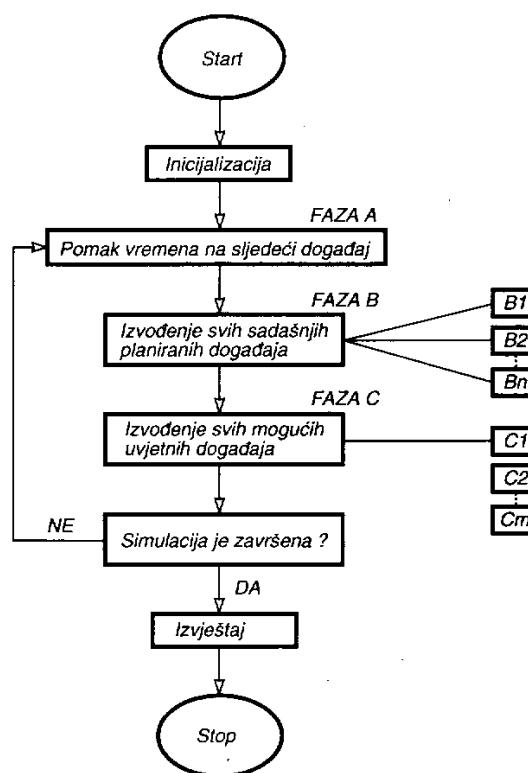
Cijeli ovaj ciklus faza A–B–C–A–B–C ... ponavlja se dok nisu ili (a) iscrpljene sve aktivnosti u simulaciji ili (b) ispunjeni uvjeti za planirani prekid simulacije (specijalni događaj).

Dijagram izvršenja trofazne metode prikazan je na slici 5.6.

Kao što smo vidjeli, osnovni element trofazne metode je *aktivnost* koju opisuju *dva događaja*: događaj završetka aktivnosti i događaj početka aktivnosti.

- Događaji završetka aktivnosti (*B-događaji*, tj. ‘bound’: vezani događaji) izvode se kada je dosegnuto njihovo planirano vrijeme. Za razliku od B-aktivnosti u strategiji planiranja aktivnosti (npr. aktivnosti završetka posluživanja), ovdje B-događaji nemaju postavljen uvjet aktiviranja već se planiraju kao bezuvjetni događaji.

- Događaji početka aktivnosti (*C-događaji*, tj. ‘conditional’: uvjetni događaji) izvode se kada su se oslobođili svi entiteti potrebeni za početak aktivnosti, odnosno kada su zadovoljeni eventualni drugi specifični uvjeti.



Slika 5.6. Dijagram izvršenja trofazne metode

B i C događaji za svaki tip aktivnosti *programiraju* se kao *nezavisne* procedure. U programskoj realizaciji trofazne metode za svaku aktivnost izgrađuje se jedna procedura C-događaja (početka aktivnosti) i toliko procedura B-događaja (završetka aktivnosti) koliko je tipova entiteta angažirano u aktivnosti. U proceduri za opis C-događaja ispituje se jesu li svi uvjeti za početak aktivnosti zadovoljeni. Kada su uvjeti zadovoljeni, angažiraju se potrebni entiteti iz repova te se planiraju odgovarajući B-događaji u izračunanim časima završetka aktivnosti. Svaka od procedura za opis B-događaja opisuje oslobođanje odgovarajućeg tipa entiteta i njihovo pomicanje u odgovarajući rep.

S programske strane trofazni pristup ima ove prednosti:

- jasnoću i preglednost strukture programa
- moduarnost programa
- mogućnost davanja prioriteta aktivnostima.

Sve to olakšava postizanje adekvatnog opisa problema, stvaranje pouzdanih programa, otkrivanje grešaka te unošenje izmjena u programe (lokalizacija mjesta izmjene).

5.4. RUČNA SIMULACIJA POMOĆU TROFAZNE METODE

Ručna se simulacija izvodi da bi se produbilo i učvrstilo razumijevanje DCA tehnikе modeliranja, te da pokaže sve detalje razvoja modela u vremenu. To ujedno pridonosi i razumijevanju funkcioniranja trofazne metode simulacije.

Da bi se ručna simulacija što lakše izvela učiniti ćemo ovo:

- prikazati simulacijski model DCA tehnikom (i to dovoljno velikih dimenzija)
- izabrati prioritete izvođenja aktivnosti (ako treba birati među izvođenjem više mogućih aktivnosti u jednom času), te ćemo odgovarajući prioritet upisati uz naziv aktivnosti u DCA (1 – najveći prioritet, 2 – manji prioritet itd.)
- za svaku klasu entiteta izabrati predmete (npr. novčice ili sl.) pomoću kojih ćemo prikazati gibanje entiteta u DCA dijagramu, ili ćemo, umjesto toga, crtati i brisati odabранe znakove za entitete
- u inicijalnom (početnom) stanju modela staviti (ili nacrtati) u odgovarajuće repove ukupan broj predstavnika entiteta koji u modelu postoje
- u skladu s opisanim načinom rada trofazne metode simulacije ručno ćemo izvoditi cikluse A-B-C, A-B-C– itd. faza simulacije
 - svaki korak u razvoju simulacije prati se:
 - (1) odgovarajućim *pomicanjem predmeta* (ili brisanjem simbola i crtanjem na drugome mjestu) koji predstavljaju entitete modela u DCA dijagramu
 - (2) *upisivanjem* vremena i tipa događaja-koji se u tom koraku ostvaruju ili planiraju u tablicu izvođenja simulacije (sa tri stupca : A, B i C, kao što se vidi u tablici 5.1); pri tome se prilikom ostvarivanja svakog planiranog događaja (iz stupca B) stavi oznaka izvršenja na mjesto prethodnog planiranja tog događaja (u stupcu C)

Tablica 5.1. Sadržaj tablice za praćenje ručnog izvođenja simulacije

Faza A	Faza B		Faza C		
Sljedeći čas simulacije	Planirani događaj završetka aktivnosti	Nove vrijednosti atributa entiteta	Uvjetni događaji početaka sljedećih aktivnosti	Vremena završetaka tih aktivnosti	Nove vrijednosti broja slobodnih resursa (po tipu)
...

- kod početka svake aktivnosti (C događaj) vrijeme trajanja aktivnosti i eventualno postavljanje i izmjenu vrijednosti atributa entiteta dobit ćemo uzimanjem uzorka iz odgovarajuće razdiobe vjerojatnosti (korištenjem tablica s generiranim vrijednostima slučajnih varijabli iz odgovarajućih razdioba)

- vrijednosti eventualnih atributa entiteta modela upisivat ćemo uz predmet (znak) koji prikazuje taj specifični entitet.

5.5. RUČNA SIMULACIJA JEDNOG PROBLEMA

Prikazat ćemo određen broj koraka ručne simulacije bara opisanog u prethodnom poglavlju.

Prepostavit ćemo da imamo ove ukupne količine stalnih entiteta (resursa) u baru: 2 slobodna konobara i 4 čiste čaše. Osim toga, imamo 1 entitet ulaza i proizvoljni broj gostiju u vanjskom repu.

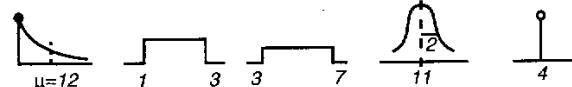
Odabrane razdiobe vremena trajanja aktivnosti i vrijednosti atributa prikazane su u tablici 5.2a. Uzorak od prvih desetak generiranih vrijednosti iz ovih razdioba prikazan je u tablici 5.2b (svejedno je da li je uzorak generiran unaprijed, ili se vrijednosti slučajnih varijabli generiraju u času kada su tražene tijekom simulacije).

Tablica 5.2. Podaci za simulacijski model bara

2 konobara
4 čiste čaše

(a) Razdiobe vremena trajanja aktivnosti i vrijednosti atributa

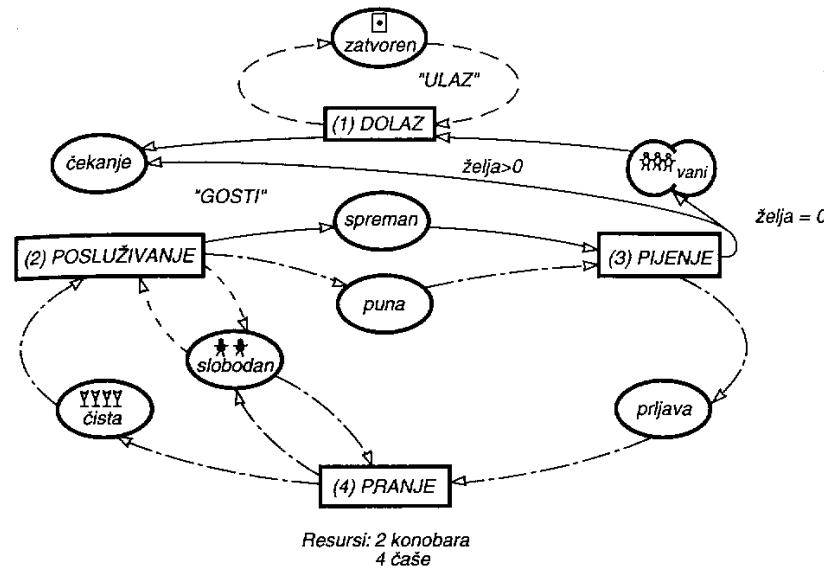
Aktivnosti i atributi	Dolazak	Želja	Posluživanje	Pijenje	Pranje čaše
Razdiobe	Neg. exp (12)	Uniformna (1,3)	Uniformna (3,7)	Normalna (11,2)	4



(b) Uzorak iz razdioba vremena trajanja aktivnosti i vrijednosti atributa
(slijed generiranih vrijednosti)

R. br. generiranja	Dolazak	Želja	Posluživanje	Pijenje	Pranje čaše
1	17	3	3	10	4
2	8	3	6	13	4
3	7	2	5	10	4
4	0	2	3	13	4
5	10	3	7	11	4
6	17	2	4	13	4
7	8	3	5	11	4
8	15	1	4	11	4
9	14	2	4	11	4

Izbor entiteta iz repova je po slijedu dolaska u rep (FCFS – ‘first-come-first-served’, tj. prvi-došao-prvi-poslužen).



Slika 5.7. DCA početnog stanja modela

DCA početnog stanja modela prikazan je na slici 5.7. Uz nazive aktivnosti dani su i njihovi odabrani prioriteti.

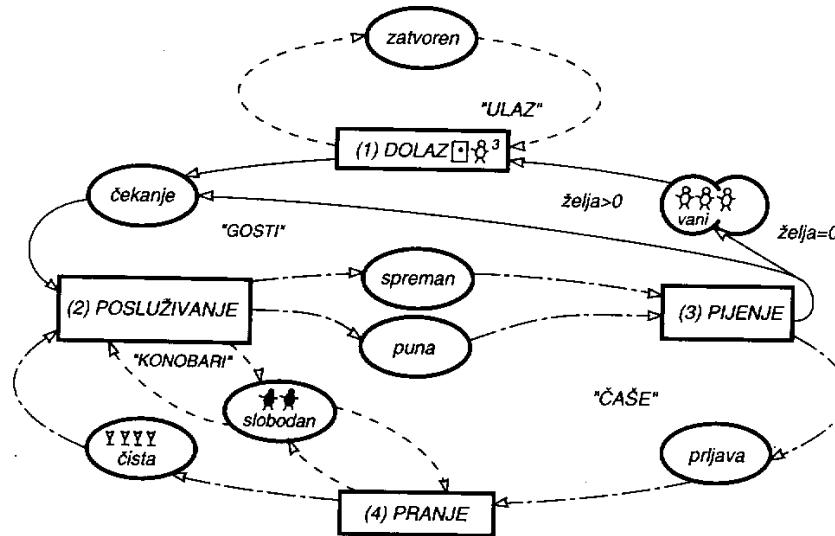
Tablica 5.3. Početna tablica izvođenja simulacije

Faza A	Faza B		Faza C			
	Završetak aktivnosti	Nova vrijednost atributa "želja"	Aktivnosti koje mogu početi	Čas završetka tih aktivnosti	Broj slobodnih resursa	
Čas					KONOBAR	ČAŠE
0	—		DOLAZAK ₁	17	2	4

/
r. br. entiteta
GOST

U tablici 5.3. prikazana je početna tablica izvođenja simulacije. U času 0 sistem počinje rad bez ijedne aktivnosti u toku (tj. bez planiranih događaja za faze A i B). Iz DCA se vidi

da u času 0 može početi aktivnost DOLAZAK (za gosta br. 1), jer je u repu "zatvoren" entitet "ulaz", dok u repu "vani" ima entiteta "gosti". Stoga počinje aktivnost DOLAZAK (faza C) za gosta br. 1, kao što se vidi na slici 5.8. Iz tablice 5.2b dobivamo trajanje aktivnosti DOLAZAK od 17 vremenskih jedinica (tj. aktivnost DOLAZAK ima planirani završetak u času $0 + 17 = 17$), i želju za 3 pića. U času 0 više se ništa ne događa.



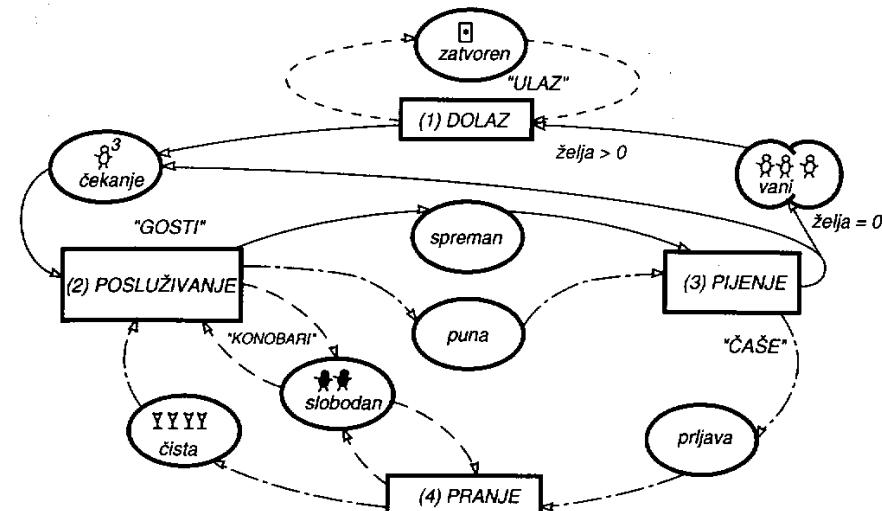
Slika 5.8. DCA modela u času 0 (faza C)

Tablica 5.4. Tablica izvođenja simulacije u času 17

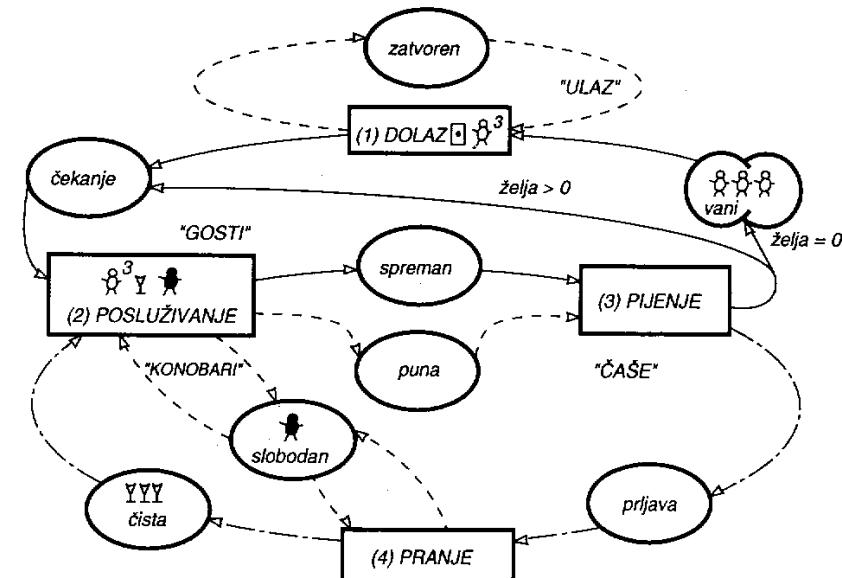
Faza A	Faza B		Faza C				
Čas	Završetak aktivnosti	Nova vrijednost atributa "želja"	Aktivnost koje mogu započeti	Čas završetka tih aktivnosti	Broj slobodnih resursa		
					KONOBAR	ČAŠE	(u repu "slobodan")
0	—		DOLAZ ₁	17✓	2	4	0
17	DOLAZ ₁	3	DOLAZ ₂	25			
			POSLUŽIVANJE ₁	20	1	3	

Tablica 5.4. prikazuje izvođenje simulacije u sljedećem dogadaju, koji nastupa u času 17 (faza A), i to je dolazak gosta br. 2 (tj. završetak aktivnosti DOLAZAK). On se ostvaruje (faza B), a pri tome se entitet "ulaz" premješta u rep "zatvoren" a entitet "gost" u rep "čekanje".

(slika 5.9a). Budući da nema drugih B događaja za čas 17, prelazimo u fazu C. U fazi C se po slijedu prioriteta pretraže aktivnosti da se vidi da li neka od njih može početi. Vidimo da



Slika 5.9. (a) DCA modela u času 17 (faza B)



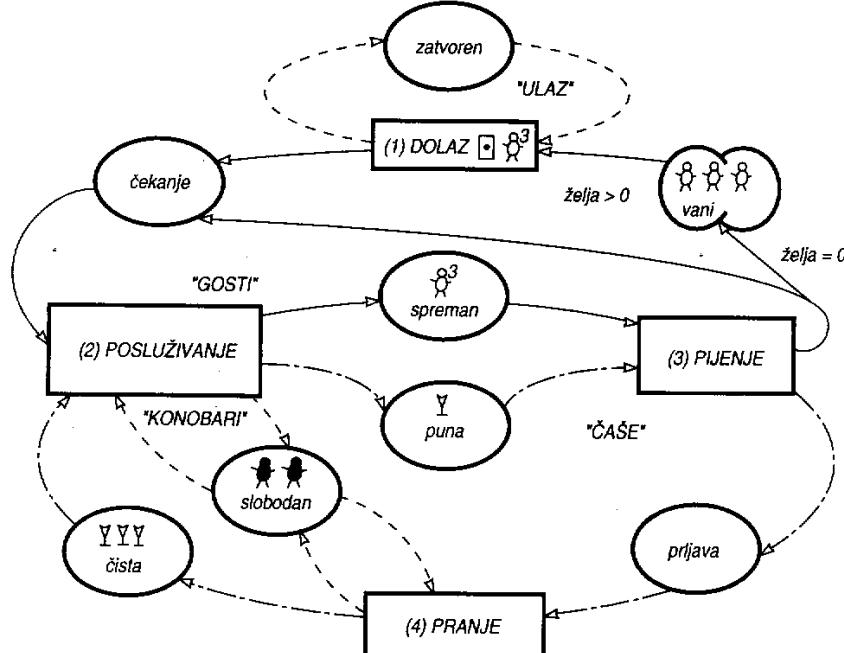
Slika 5.9. (b) DCA modela u času 17 (faza C)

mogu početi (1) aktivnost DOLAZAK, koja se pokreće uzimanjem entiteta "gost" i "ulaz" iz odgovarajućih repova, i generiranjem vrijednosti za njezino vrijeme trajanja (ona će trajati 8 vremenskih jedinica, tj. završit će u $17 + 8 = 25$ vremenskoj jedinici); i (2) aktivnost POSLUŽIVANJE za koju se uzimaju entiteti "gost", "konobar" i "čaša" iz odgovarajućih repova (slika 5.9b). Time se broj slobodnih konobara smanjuje na 1, a broj čistih čaša na 3.

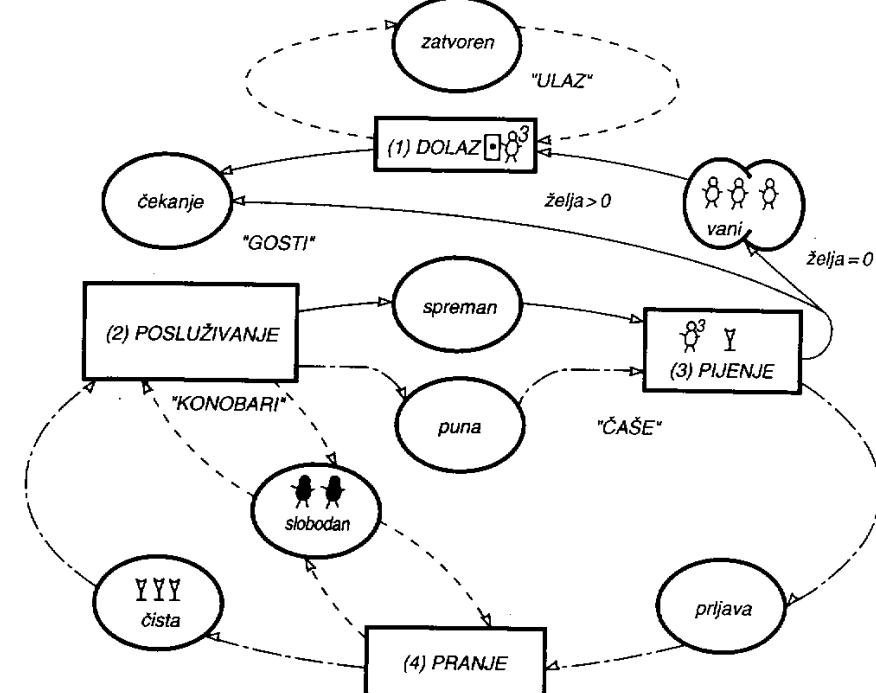
Kao što se vidi u tablici 5.5. sljedeći je događaj u času 20 (faza A), i to završetak aktivnosti POSLUŽIVANJE (događaj B). Nakon izvršenja tog događaja (slika 5.10a) počinje aktivnost PIJENJE koja traje 10 vremenskih jedinica (tj. završava u času $20 + 10 = 30$), kao što se vidi na slici 5.10b.

Tablica 5.5. Tablica izvođenja simulacije u času 20

Faza A	Faza B	Faza C			
Čas	Završetak aktivnosti	Nova vrijednost atributa "želja"	Aktivnosti koje mogu početi	Čas završetka tih aktivnosti	Broj slobodnih resursa
			KONOBAR ČAŠE	u repu "slobodan" u repu "čista" u repu "prijava"	
0	—		DOLAZ 1	17✓	2 4 0
17	DOLAZ 1	3	DOLAZ 2 POSLUŽIVANJE 1	25 20✓	1 3
20	POSLUŽIVANJE 1		PIJENJE 1	30	2



Slika 5.10. DCA modela u času 20
(a) Faza B



Slika 5.10. DCA modela u času 20
(b) Faza C

Ovdje nećemo opisati detalje daljeg izvršenja simulacije. Zbog provjere razumijevanja rada trofazne strategije prikazali smo tablicu izvođenja simulacije do časa 32 (tablica 5.6). Stanje modela nakon izvršenja B i C faze u času 32 prikazano je na slici 5.11.

5.6. SKUPLJANJE STATISTIČKIH PODATAKA O IZVOĐENJU SIMULACIJSKIH EKSPERIMENTA

5.6.1. Osnovni podaci od interesa za analizu rezultata simulacije

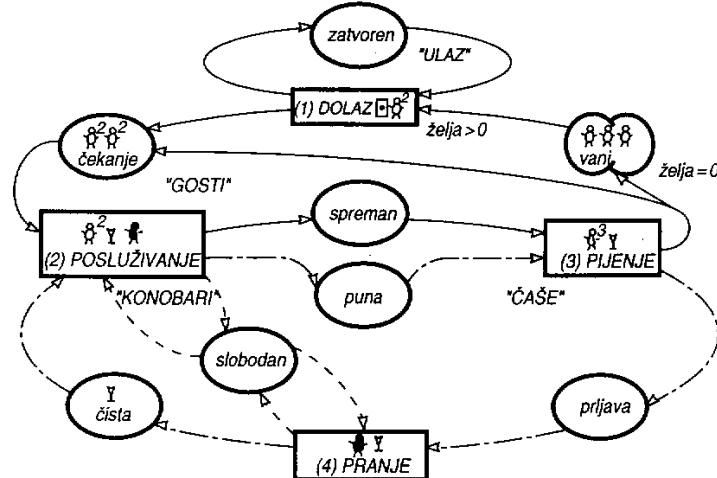
Tijekom izvođenja simulacije potrebno je skupiti podatke o zavisnim varijablama simulacije (performanse sistema) koje pokazuju ponašanje modela i omogućuju donošenje odluka u vezi sa sistemom (Banks i Carson, 1984).

Među osnovnim podacima od interesa su:

- (1) Brojači entiteta i izvođenja događaja

Tablica 5.6. Tablica izvođenja simulacije u času 32

Čas	Završetak aktivnosti	Nova vrijednost atributa "želja"	Aktivnosti koje mogu početi	Čas završetka tih aktivnosti	Broj slobodnih resursa		
					KONOBAR		ČAŠE
					u repu "slobodan"	u repu "čista"	u repu "prijava"
0	—		DOLAZ 1	17✓	2	4	0
17	DOLAZ 1	3	DOLAZ 2 POSLUŽIVANJE 1	25✓ 20✓	1	3	
20	POSLUŽIVANJE 1		PIJENJE 1	30✓	2		
25	DOLAZ 2	3	DOLAZ 3 POSLUŽIVANJE 2	32✓ 31✓	1	2	
30	PIJENJE 1	2	POSLUŽIVANJE 1 (PRANJE čeka na slobodnog konobara)	35	0	1	1
31	POSLUŽIVANJE 2		PIJENJE 2 PRANJE	44 35	1 0		0
32	DOLAZ 3	2	DOLAZ 4 (POSLUŽIVANJE 3 čeka slobodnog konobara)	32✓			
	DOLAZ 4	2	DOLAZ 5 (POSLUŽIVANJE 4 čeka slobodnog konobara)	42			



Slika 5.11. DCA modela u času 32 (nakon izvršenja B i C faze)

(2) *Sumarna statistika* varijabli, koja obuhvaća prosjek, standardne devijacije, ekstremne vrijednosti i sl.

(3) *Iskorištenje resursa*, tj. nalaženje dijela vremena u kojem je resurs angažiran

(4) *Zauzetost grupe resursa*, tj. nalaženje dijela vremena u kojem se resursi iz grupe u prosjeku koriste (npr. grupe šaltera, telefona i sl.)

(5) *Razdioba vjerojatnosti* značajnih varijabli

(6) *Prijelazna vremena* gibanja privremenih entiteta od jednoga do drugoga dijela sistema.

Postojanje slučajnih efekata nezavisnih (ulaznih) varijabli modela dovodi do fluktuacije zavisnih (izlaznih) varijabli modela, tako da su vrijednosti dobivene simulacijom zapravo statističke procjene pravih vrijednosti koje se traže. Ovaj aspekt će biti detaljnije razrađen pri opisu analize izlaznih rezultata simulacijskih eksperimenta, a ovdje želimo samo pokazati kako izgleda proces skupljanja podataka o izvođenju simulacijskih eksperimenta.

Statistika ekstremnih (minimalnih i maksimalnih) vrijednosti dobiva se tako da se svaka nova vrijednost varijable usporedi s tekućom minimalnom i maksimalnom vrijednošću te varijable; ekstremne vrijednosti se promijene ako je nova vrijednost varijable manja (odnosno veća) od sadašnje ekstremne vrijednosti.

Srednja vrijednost S i standardna devijacija D dobiju se iz skupa promatranja

X_i ($i = 1, 2, \dots, n$) na uobičajeni način :

$$S = \frac{1}{n} \sum_{i=1}^n X_i$$

$$D = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (S - X_i)^2}$$

Pojedinačno iskorištenje resursa (U) izračunava se kao dio od ukupnog vremena simulacije (T) tijekom kojeg je taj resurs korišten. Ako je t_{korist}^j j-ti interval vremena korištenja resursa tijekom simulacije (N je ukupan broj tih intervala u vremenu T), tada je korištenje resursa :

$$U = \frac{1}{T} \sum_{j=1}^N t_{korist}^j$$

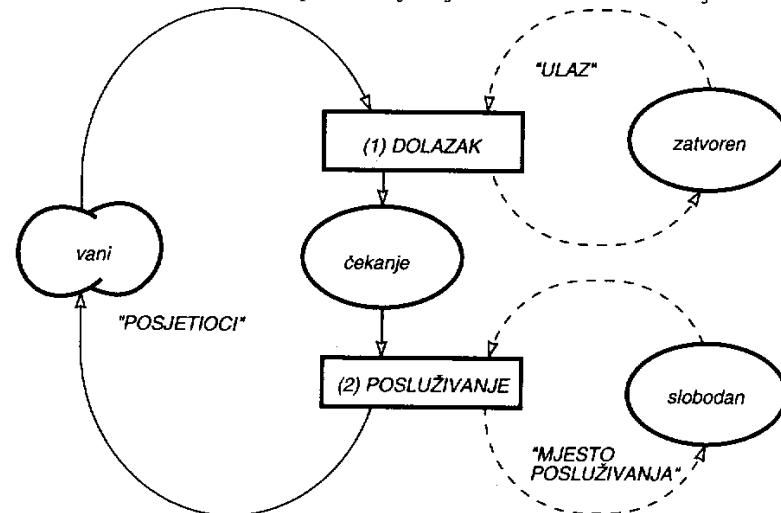
Zauzetost grupe resursa dobiva se na analogno, uzimanjem u obzir dijela od ukupnog broja resursa angažiranih po vremenskim intervalima.

Razdioba neke veličine dobiva se brojanjem koliko puta je vrijednost te veličine pala unutar odabralih intervala, tj. za svaki interval akumulira se vrijednost brojača za tu veličinu. Kod svakog promatranja varijable ispituje se u koji interval pripada vrijednost promatranja, te se na temelju toga povećava brojač promatranja za taj interval. Vremena čekanja upisuju se u tablicu intervala u času kada je entitet završio čekanje (odnosno kada je počeo posluživanje). Pri računanju razdiobe broja entiteta u repu čekanja promatranja se izvode u jednakim vremenskim intervalima.

Prijelazna vremena privremenih entiteta mijere se tako da se prilikom prolaza entiteta kroz početnu točku zapamti čas prolaza, a prilikom prolaza kroz završnu točku izračunava se razlika vremena prolaza entiteta između tih dviju točaka.

5.6.2. Primjer skupljanja statističkih podataka simulacije

Skupljanje statističkih podataka simulacije demonstrirat ćemo na primjeru izvođenja simulacije jednostavnog repa čekanja s jednim mjestom posluživanja. Dijagram ciklusa aktivnosti ovog sistema prikazan je na slici 5.12, a u tablici 5.7. prikazan je niz vremena međudolazaka posjetilaca i vremena posluživanja koji će se koristiti za izvođenje simulacije.



Slika 5.12. Dijagram ciklusa aktivnosti jednostavnog repa čekanja sa jednim mjestom posluživanja

Tablica 5.7. Generirani uzorak ulaznih podataka simulacije za jednostavni rep čekanja

Posjetilac	Vrijeme međudolaska	Vrijeme posluživanja
1	2	3
2	4	2
3	3	4
4	1	4
5	2	4
6	2	1
7	4	2
8	5	1
9	3	2
10	3	4

Izvođenje trofazne simulacije do časa simulacije 25 prikazano je u tablici 5.8. Na osnovi toga pripremljeni su podaci po posjetiocima (t_1 , t_2 i t_3) u tablici 5.9. Pomoću tih podataka izračunana su vremena čekanja u repu, ukupna vremena provedena u sistemu po posjetiocima, te vremena tijekom kojih su posluživaoci slobodni (razlika vremena između početka posluživanja posjetilaca i završetka posluživanja prethodnog posjetioca).

Na slici 5.13a prikazan je vremenski razvoj svakog od posjetilaca u sistemu, na slici 5.13b vremenski razvoj broja posjetilaca u repu, i na slici 5.13c razvoj korištenja mesta za posluživanje u vremenu.

Tablica 5.8. Tablica izvođenja trofazne simulacije za jednostavni rep čekanja

Faza A (sljedeći čas simulacije)	Faza B (planirani događaj završetka aktivnosti)	Faza C	
		uvjetni događaj početka aktivnosti	vrijeme završetka aktivnosti
0	—	start DOLAZ 1	2 ✓
2	end DOLAZ 1	start DOLAZ 2	6 ✓
		start POSLUŽ 1	5 ✓
5	end POSLUŽ 1	—	—
6	end DOLAZ 2	start DOLAZ 3	9 ✓
		start POSLUŽ 2	8 ✓
8	end POSLUŽ 2	—	—
9	end DOLAZ 3	start DOLAZ 4	10 ✓
		start POSLUŽ 3	13 ✓
10	end DOLAZ 4	start DOLAZ 5	12 ✓
12	end DOLAZ 5	start DOLAZ 6	14 ✓
13	end POSLUŽ 3	start POSLUŽ 4	17 ✓
14	end DOLAZ 6	start DOLAZ 7	18 ✓
17	end POSLUŽ 4	start POSLUŽ 5	21 ✓
18	end DOLAZ 7	start DOLAZ 8	23 ✓
21	end POSLUŽ 5	start POSLUŽ 6	22 ✓
22	end POSLUŽ 6	start POSLUŽ 7	24 ✓
23	end DOLAZ 8	start DOLAZ 9	26
24	end POSLUŽ 7	start POSLUŽ 8	25 ✓
25	end POSLUŽ 8	—	—

Na temelju podataka dobivenih izvođenjem simulacije izračunat ćemo neke od statističkih pokazatelja rada sistema:

(1) Prosječno čekanje u repu po posjetiocu =

$$= \frac{\text{ukupno vrijeme čekanja posjetilaca u repu}}{\text{ukupan broj posjetilaca}} = \\ = \frac{20}{8} = 2.5 \text{ vremenskih jedinica.}$$

(2) Prosječno iskorištenje mesta posluživanja =

$$= \frac{\text{ukupno vrijeme zaposlenosti poslužioca}}{\text{trajanje simulacije}} = \\ = \frac{25 - 4}{25} = \frac{21}{25} = 0.84$$

(3) Prosječno vrijeme posluživanja po posjetiocu =

$$= \frac{\text{ukupno vrijeme posluživanja}}{\text{ukupan broj posjetilaca}} = \\ = \frac{21}{8} = 2.625 \text{ vremenskih jedinica po posjetiocu.}$$

(4) Prosječno vrijeme među dolascima =

$$= \frac{\text{suma svih vremena među dolascima (osim prvog)}}{\text{ukupan broj dolazaka} - 1} = \\ = \frac{21}{8-1} = \frac{21}{7} = 3.0 \text{ vremenskih jedinica}$$

(5) Dio posjetilaca koji čekaju u repu =

$$= \frac{\text{broj posjetilaca koji čekaju}}{\text{ukupan broj posjetilaca}} = \frac{5}{8} = 0.625.$$

(6) Prosječno vrijeme čekanja onih posjetilaca koji čekaju =

$$= \frac{\text{ukupno vrijeme čekanja u repu}}{\text{ukupan broj posjetilaca koji čekaju}} = \\ = \frac{20}{5} = 4.0 \text{ vremenskih jedinica.}$$

(7) Prosječno vrijeme posjetilaca u sistemu =

$$= \frac{\text{ukupno vrijeme posjetilaca u sistemu}}{\text{ukupan broj posjetilaca u sistemu}} = \\ = \frac{41}{8} = 5.125 \text{ vremenskih jedinica.}$$

Prikazat ćemo i primjer razdioba vremena čekanja u repu, pri čemu ćemo odabrati vremena čekanja u intervalima 0–2, 2–4, 4–6, 6–8, pri čemu ćemo donju granicu uključiti u interval a gornju nećemo (jer ona pripada sljedećem intervalu). Uz pomoć podataka iz tablice 5.9. dobit ćemo razdiobu koja je prikazana na slici 5.13d.

Tablica 5.9. Skupljanje podataka o posjetiocima za jednostavni rep čekanja

R. br. posjetioca	Vrijeme dolaska (kraj DOLAZ)	Vrijeme početka POSLU- ŽIVANJA	Vrijeme zavšetka POSLU- ŽIVANJA	Vrijeme čekanja u repu	Ukupno vrijeme u sistemu	Slobodno vrijeme posluživaoca
	t_1	t_2	t_3	$(t_2 - t_1)$	$(t_3 - t_1)$	$(t_{\text{pot. posluž}}^i - t_{\text{kraj posl.}}^{i-1})$
1	2	2	5	0	3	2
2	6	6	8	0	2	1
3	9	9	13	0	4	1
4	10	13	17	3	7	0
5	12	17	21	5	9	0
6	14	21	22	7	8	0
7	18	22	24	4	6	0
8	23	24	25	1	2	0

5.7. PLANIRANJE BUDUĆIH DOGAĐAJA

5.7.1. Važnost planiranja budućih događaja u simulaciji diskretnih događaja

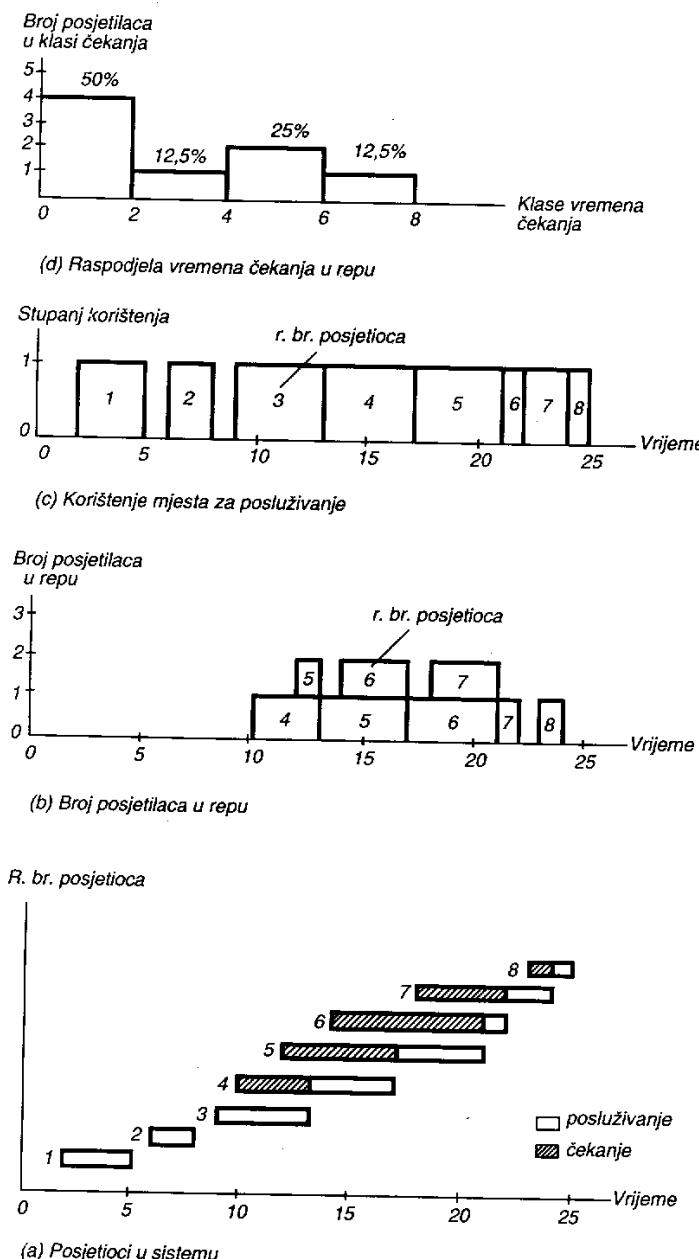
Ključ za izvođenje vremenskog razvoja u simulaciji diskretnih događaja su operacije s događajima, i to posebno s onima iz skupa bilješki o budućim događajima (Evans, 1988). Događaji iz tog skupa izvode se onda kada na njih dođe red, a kada su završeni, tada više nisu potrebni u ovom skupu. Skup budućih događaja pretražuje se u rastućem vremenskom slijedu, a nove bilješke o budućim događajima, nastale kao posljedica izvođenja akcija koje slijede nakon dešavanja prethodnih događaja, mogu se ubaciti u skup budućih događaja. Izbacuju se izvedene bilješke o sadašnjim događajima i o mogućim neželjenim događajima u budućnosti.

Bilješke o događaju sadrže vrijeme događaja i pokazivač na akcije koje slijede poslije događaja, čiji oblik ovisi o upotrijebljenoj simulacijskoj strategiji. Planiranje diskretnih događaja sastoji se od dviju osnovnih operacija:

- (1) ubacivanje bilješke u skup bilježaka o novim događajima,
- (2) izbacivanje najranijih bilježaka o događajima.

Pitanje je, naravno, koji je najbolji algoritam i struktura podataka za izvođenje operacija planiranja budućih događaja. Operacije planiranja rade se s objektima (bilješkama o budućim događajima) što sa sobom nose podatak o vremenu dešavanja, koji je prioritet za pomicanje iz skupa. Stoga se skup bilježaka o budućim događajima može promatrati kao rep s prioritetom.

Planiranje budućih događaja nezavisno je od simulacijske strategije, koja opisuje način izvođenja promjena stanja sistema, pa je ono, prema tome, bitno za efikasno odvijanje razvoja simulacije u vremenu.



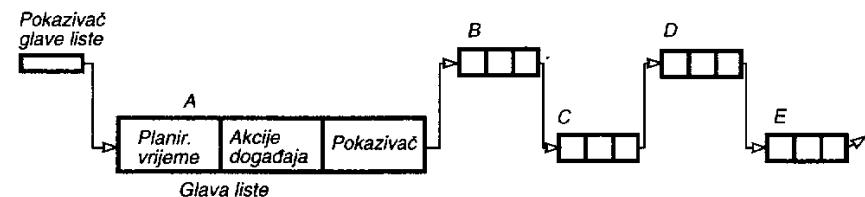
Slika 5.13. Prikaz skupljenih podataka o simulaciji za jednostavni rep čekanja.

5.7.2. Metode planiranja budućih događaja

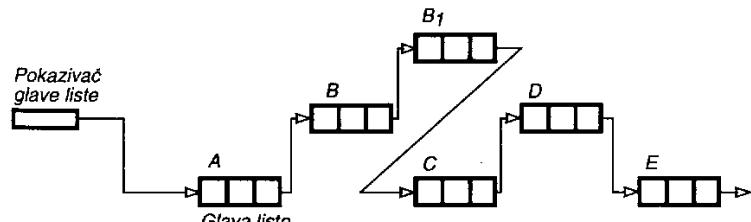
Radi planiranja budućih događaja koristi se niz različitih struktura podataka i algoritama. Od strukture podataka to su npr. jednostavne linearne liste, indeksirane linearne liste, 'heap' strukture (jedna vrst balansirane strukture stabala) i binarna stabla pretraživanja. Te strukture moraju omogućiti takvo strukturiranje podataka koje čuva podatke u uređenom obliku, s tim da se broj podataka skupa može dinamički mijenjati.

Koristi se niz različitih algoritama koji nastoje što efikasnije riješiti problem ubacivanja i izbacivanja bilješki iz skupa budućih događaja. Najjednostavniji mogući pristup je *pretraživanje svih bilježaka* o budućim događajima kako bi se našla bilješka s minimalnim planiranim vremenom. Međutim, vrijeme pretraživanja raste s porastom veličine skupa budućih događaja, tako da kod velikih skupova budućih događaja ova metoda postaje krajnje neefikasna.

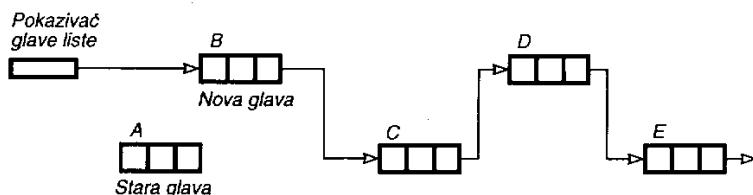
Opisat ćemo još ukratko jedan jednostavan pristup, koji je mnogo korišten u prvim simulacijskim jezicima, a i danas se koriste njegove neke varijacije. To je organizacija bilješki o budućim događajima u *jednostavne linearne liste*, gdje su elementi liste logički poredani po rastućem planiranom vremenu aktivacije. Elementi liste su smješteni na slučajni način u memoriji, a međusobna logička veza među elementima uspostavlja se pokazivačima. Na slici 5.14. prikazana je *jednostruko vezana linearna lista*. Pretraživanje od početka liste, tj. od najranije bilješke o budućim događajima, ubrzava ubacivanje nove bilješke jer se bilješka smješta iza prve nađene bilješke u skupu od koje ima veće planirano vrijeme izvođenja. Na slici 5.15. prikazano je ubacivanje nove bilješke B1 u listu, jednostavnom promjenom adresa u pokazivačima bilježaka B i B1. Traženje sljedećeg događaja je sasvim jednostavno —glava liste A je sadašnji događaj koji se nakon izvođenja izbacuje iz liste, a događaj B, na koji je glava pokazivala automatski je nova glava liste i novi sadašnji događaj (slika 5.16).



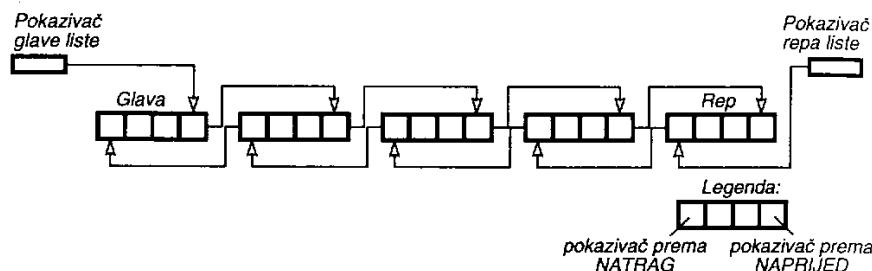
Slika 5.14. Jednostruko vezana linearna lista



Slika 5.15. Ubacivanje nove bilješke u listu



Slika 5.16. Pretraživanje novog događaja i izbacivanje sadašnjeg događaja



Slika 5.17. Dvostruko vezane liste

Da bi se ubrzalo i izbacivanje bilješki čije se izvođenje zbog nekog razloga otkazuje, koriste se *dvostrukе vezane linearne liste*, prikazane na slici 5.17. Pretraživanje unatrag koristi se i u nekim metodama ubacivanja bilježaka u liste.

5.7.3. Usporedba metoda za planiranje budućih događaja

Usporede efikasnosti različitih struktura podataka i algoritama za planiranje budućih događaja izvedene su različitim empirijskim i teorijskim pristupima.

Od *empirijskih metoda* najviše je korišten tzv. 'hold model', nazvan prema 'hold' operaciji u simulacijskom jeziku SIMULA. Taj model kao tipičnu akciju planiranja uzima: nalaženje minimalne bilješke skupa budućih događaja, pomak sata modela na vrijeme te bilješke i planiranje jedne bilješke iz skupa budućih događaja dobivene slučajnim izborom iz dane razdiobe vjerojatnosti za planiranje bilježaka budućih događaja. Rezultati primjene 'hold modela' pokazali su da su najbolje performanse imali tzv. Henriksonov algoritam (to je jedna vrsta indeksirane liste koja se binarno pretražuje), binarno pretraživano stablo s prekrivanjem pri pretraživanju i tzv. 'splay stablo' (koje se koristi heurističkim pretraživanjem s pomakom čvorova stabla).

Teorijske metode ispituju ponašanje algoritama neovisno o arhitekturi računala, uvjetima računanja i korištenom programskom jeziku. One tipično ispituju efikasnost algoritama u ovisnosti o veličini skupa budućih događaja te analizi najgorih slučajeva. Pоказало се да резултати теоријских испитивања премало разликују pojedine алгоритме, па нпр. не могу предвидети ни слабу ефикасност 'heap' структуре ни балансираних stabala.

5.7.4. Korištenje metoda za planiranje budućih događaja u simulacijskim jezicima

Poznato je da se jezik SIMSCRIPT i starije verzije jezika GPSS koriste vezanim listama. Jezik SIMULA koristi se tzv. p-stablim. Jezici GPSS/H i SLAM koriste se Henriksonovim algoritmom (indeksiranim listama s linearnim pretraživanjem).

Detaljna analiza problema planiranja budućih događaja može se naći u knjizi (Evans, 1988), koja je korištena i u ovom tekstu.

6. IZGRADNJA SIMULACIJSKIH PROGRAMA

U ovom su poglavlju opisane osnovne karakteristike softvera za simulaciju diskretnih događaja. Prikazane su klasifikacije simulacijskog softvera na prevodioce i interpretatore, opće i specijalne simulacijske jezike itd., te osnovni tipovi alata za simulaciju, koji na različite načine ostvaruju izgradnju simulacijskih programa. Pri nabavi softvera za simulaciju potrebno je poznavati kriterije za izbor odgovarajućeg softvera ovdje opisanih. Na kraju poglavlja kratak je prikaz razvojnih tendencija simulacijskog softvera.

6.1. OSNOVNE KLASIFIKACIJE SOFTVERA ZA SIMULACIJU

Simulacija diskretnih događaja koristi se već tridesetak godina a zanimanje za nju i dalje se povećava, kako zbog sve složenijih tehnoloških i ekonomskih sistema koje je teško ili čak nemoguće zadovoljavajuće modelirati i analizirati drugim metodama, tako i zbog razvoja što su ga doživjeli softverski alati za simulaciju. Tako su pojavom mikroračunala softverski alati postali mnogo dostupniji; novi pristupi razvoju korisniku prijateljskih ('user friendly') softvera (meniji, prozori, korištenje miša, ikona i sl.) olakšali su rad sa simulacijskim softverom; simulacijska grafika je omogućila grafički prikaz statističkih pokazatelja rada sistema; animacija izvođenja simulacije omogućila je vizualiziranje dinamike rada sistema koji ne mora još ni postojati; automatski generatori simulacijskih programa značajno su skratili izradu simulacijskih programa i povećali njihovu pouzdanost; a korištenje metoda umjetne inteligencije (baze znanja, ekspertni sistemi, korištenje prirodnih jezika itd.) postupno idu prema olakšavanju korištenja složene metodologije simulacije i komunikacije čovjeka sa simulacijskim softverom.

Posljedica toga posljednjih je godina buran razvoj različitih tipova simulacijskog softvera, tako da danas postoji niz različitih simulacijskih alata. Dio tih alata je zastario, a dio se modernizirao i prilagodio novim zahtjevima modelara i korisnika simulacije. Navest ćemo osnovne kategorije podjele simulacijskog softvera (Kreutzer, 1986; Pidd, 1984).

6.1.1. Prevodioci i interpretatori

Prevodioci su programi koji pretvaraju cijeli simulacijski program iz izvornog oblika u strojni kod koji računalo može izvoditi. Prevođenje je uspješno završeno tek kada su ispravljene sve greške u izvornom programu. Tada se dobiveni strojni kod može izvoditi proizvoljni broj puta bez prevodenja. *Interpretatori* prevode jednu po jednu liniju (ili po nekoliko linija) izvornog koda i odmah ih izvode. Interpretator ne traži

potpuno ispravan cijeli izvorni program i ne stvara strojni kod koji možemo izvoditi bez prevođenja, već ujvek mora interpretirati izvorni program. Oni su dakle fleksibilniji, i posebno pogodni za razvoj programa, praćenje izvođenja i traženje grešaka, ali su mnogo sporiji za izvođenje. Dakle, za višestruko izvođenje već istestiranih velikih programa prevodioci su mnogo efikasniji.

Većina softverskih alata za simulaciju diskretnih događaja su prevodioci, a nekih od njih ima u objema verzijama. Primjer prevodioca su jezici SIMSCRIPT i SIMULA, a npr. jezik GPSS postoji u obje verzije.

6.1.2. Opći i specijalni jezici

Opći simulacijski jezici mogu opisati različite vrste sistema, a specijalni jezici opisuju samo jednu klasu sistema. Opći simulacijski jezici imaju dakle šire područje primjene, a specijalni su jezici prikladniji i efikasniji za modeliranje pojedinih klasa sistema.

Opći simulacijski jezici su npr. GPSS, SIMSCRIPT, SIMULA I SLAM. Specijalni jezici su posljednjih godina posebno usmjereni na područje proizvodnih sistema, npr. SIMAN I SIMFACTORY. Jezik SIMNETWORK primjenjuje se za modeliranje i analizu mreža računala.

6.1.3. Korištenje simulacijskih strategija

Većina simulacijskih jezika implementira jednu od simulacijskih strategija, dok neki od jezika kombiniraju elemente različitih strategija. U nekim jezicima implementirane su i po dvije različite strategije simulacije (npr. SIMSCRIPT). Primjeri jezika prema korištenoj strategiji simulacije su:

- * Planiranje događaja
SIMSCRIPT, GASP, SIMPAS, SLAM
- * Međudjelovanje procesa
GPSS, SIMULA, SLAM, SIMPL/I,
novije verzije jezika SIMSCRIPT i GASP
- * Prelaženje aktivnosti
CSL
- * Trofazna simulacija
SIMON, ECSL, VS6 (VS7)

Međudjelovanje procesa je, kao što vidimo, najčešće implementirana strategija simulacije.

6.1.4. Softver za mikroračunala

Pojavom i širenjem mikroračunala uslijedila je gotovo opća preorientacija simulacijskog softvera na korištenje pomoću mikroračunala. Pri tome su uvedene i pogodnosti razvijene u radu orijentiranom prema korisniku (meniji, prozori, korištenje miša i sl.). Tako je većina 'klasičnih' simulacijskih jezika dobila svoje verzije za mikroračunala (npr. GPSS, SIMSCRIPT i HOCUS), a neki jezici su specijalno razvijeni i dizajnirani samo za rad na mikroračunalima (SEE-WHY, SIMFACTORY, VS6).

Jaka tendencija u razvoju simulacijskog softvera za mikroračunala je *vizualna animacija* izvođenja simulacije. Većina simulacijskih jezika razvijenih za mikroračunala ima mogućnost animacije, a i mnogi jezici preuređeni za mikroračunala također uključuju animaciju (neke verzije jezika GPSS, dodatni modul SIMANIMATION jezika SIMSCRIPT, dodatni modul CINEMA jezika SIMAN itd.).

6.2. PRISTUPI IZGRADNJI SIMULACIJSKIH PROGRAMA

Pošto je razvijen konceptualni simulacijski model, potrebno je razviti računarski model, tj. simulacijski program, čije će nam izvođenje dati podatke o ponašanju sistema u vremenu. Da bi se izgradio simulacijski program, potreban je, dakako, odgovarajući simulacijski jezik ili paket koji predstavlja 'građevni materijal' iz kojega stvaramo simulacijski program.

Danas imamo više različitih pristupa (Kreutzer,1986; Pidd,1984) koji nas mogu dovesti do istoga cilja, simulacijskog programa, na različite načine.

6.2.1. Korištenje općih programske jezike

Simulacijski program može se napisati, u principu, u bilo kojem općem programskom jeziku. Mnogo simulacijskih programa napisano u jezicima FORTRAN i PASCAL, a neki i u jezicima PL/I i BASIC. Osnovni je problem kod ovog pristupa to što se moraju ponovno razviti svi osnovni mehanizmi koji omogućuju opis modela i izvođenje simulacijskih eksperimenata. To međutim znatno produžuje vrijeme razvoja simulacijskih programa i povećava mogućnost grešaka, a ujedno smanjuje i fleksibilnost i efikasnost programa.

Stoga je nerazumno i riskantno upuštati se u takav posao, i on ima smisla samo (1) ako nema nikakve druge mogućnosti za nabavu prikladnijeg simulacijskog alata, (2) ako se to radi zbog potreba edukacije i (3) ako se razvija novi simulacijski jezik ili paket kod kojeg se žele postići neka posebna svojstva.

6.2.2. Biblioteke procedura

Da bi se ipak iskoristile nesumnjive prednosti općih programskih jezika, npr. njihova široka rasprostranjenost, prenosivost, širok krug korisnika i dosta visoka efikasnost, razvijeni su programski paketi koji sadrže biblioteke procedura pisanih u općim programskim jezicima. Te procedure namijenjene su opisu elemenata modela, rješavaju probleme baratanja vremenom, generiranja slučajnih brojeva i varijabli itd. Izgradnja simulacijskih programa znači pisanje dijelova programa u općem programskom jeziku, i to obično glavnog programa i nekih procedura (npr. opis akcija koje slijede poslije pojedinih tipova događaja), koji osnovne probleme simulacije diskretnih događaja rješavaju pozivom odgovarajućih procedura iz biblioteke.

Prednost je ovakva pristupa, osim što se koristi pristupačan i dobro poznat jezik, to što omogućuje razvoj efikasnih simulacijskih programa, odnosno programa koji točno odgovaraju zahtjevu korisnika. Osim toga, korisnik ima potpuni uvid u sve detalje simulacijskog koda, koji su inače u simulacijskim jezicima sakriveni od korisnika. *Slaba strana* ovog pristupa je to što se dio simulacijskog koda mora napisati u općem

programskom jeziku, što može biti izvor grešaka. Osim toga, u ovakvu pristupu poruke o greškama pri prevodenju i izvođenju uglavnom se odnose na greške u naredbama općeg programskog jezika, što otežava ionako složen problem verifikacije simulacijskih programa.

Neki od poznatih primjera ovakva pristupa su simulacijski jezici/paketi GASP, SLAM, SIMON i SEE-WHY, napisani u općem programskom jeziku FORTRAN. Drugi dosta korišten jezik za ove svrhe je PASCAL, u kojem je npr. napisan simulacijski jezik PASSIM i niz drugih simulacijskih paketa najčešće razvijenih na sveučilištima za potrebe edukacije.

Iako je jezik FORTRAN osnova većine programskih paketa ovog tipa, on ima dosta nedostataka u usporedbi s potrebama simulacijskog modeliranja. Na primjer, imena varijabli mogu imati samo šest znakova, što znatno otežava prirođan opis entiteta, a ne mogu se opisati ni liste ni sloganovi koji su značajna pomoć za strukturiranje entiteta simulacijskih modela i baratanja njima. Takvi nedostaci otežavaju razvoj, razumijevanje i testiranje simulacijskih programa, pa se može tvrditi da je danas FORTRAN zastario i da nije prikladan za ovakve potrebe. Jezik PASCAL je u tom pogledu mnogo prikladniji, a ujedno je izvanredno popularan, posebno na sveučilištima razvijenih zemalja. Međutim, ni PASCAL nije pogodan npr. za razvoj procesno ili objektno orijentiranog pristupa simulaciji. Neka od rješenja nastojala su proširiti jezik PASCAL, ali je nevolja što se time smanjuje prenosivost programa. Zato je posljednjih godina bilo razvijeno nekoliko simulacijskih paketa u jezicima SIMULA, MODULA, ADA I C, koji imaju veće izražajne mogućnosti (strukture podataka i jezične strukture) i veću efikasnost (npr. dinamički tretman memorije).

6.2.3. Simulacijski jezici s opisom naredbi

Simulacijski jezici s opisom naredbi strukturirani su kao opći programski jezici s orijentacijom na rješavanje problema u području simulacije diskretnih događaja. Većina tih jezika ima osnovni dio koji se može koristiti poput bilo kojeg općeg programskog jezika, proširenog konstrukcijama što podupiru simulacijsko modeliranje. Primjer simulacijskih jezika s opisom naredbi su SIMSCRIPT i SIMULA. Oni zahtijevaju dobro poznavanje programiranja. Prednost im je velika fleksibilnost pri opisu problema.

Jezik SIMSCRIPT podržava pristup simulacijskom modeliranju pomoću entiteta, atributa i skupova. Entiteti opisuju objekte sistema a atributi njihova svojstva. Skupovi opisuju grupiranje entiteta. Entiteti mogu biti stalni i privremeni. Privremeni entiteti mogu se dinamički stvarati i uništavati. Entiteti mogu posjedovati bilo koji broj skupova, a mogu biti i elementi bilo kojeg broja skupova. Moguće je opisati način skupljanja statističkih podataka simulacije. SIMSCRIPT je moderan i dobro strukturiran jezik.

Jezik SIMULA je proširenje općeg programskog jezika ALGOL. On podržava objektno-orijentirano programiranje u kojem se koriste tzv. klase, tj. kolekcije aktivnih komponenti koje mogu međudjelovati jedna s drugom. Komponente imaju svoja imena i komuniciraju isključivo porukama koje pretražuju ili mijenjaju varijable, odnosno iniciraju proračune. Klase se koriste i mehanizmom nasleđivanja svojstava i poruka od drugih klasa.

6.2.4. Simulacijski jezici za izgradnju scenarija

Ovaj tip simulacijskih jezika temelji se na modulima (blokovima) koji omogućuju jednostavno i prirodno modeliranje nekog simulacijskog scenarija, odnosno pogleda na

svijet. Tipičan primjer scenarija je simulacija repova čekanja u kojem se koriste koncepti mesta posluživanja, repova i transakcija (entiteta koji 'prolaze' kroz blokove modela) koje se mogu generirati i uništiti. Programiranje se svodi na opis blokova kroz koje transakcije prolaze tijekom svojeg 'života' u modelu. Scenariji su najčešće vezani za neki grafički prikaz konceptualnog modela, npr. dijagrame ciklusa aktivnosti, proširene Petrijeve mreže ili blok-dijagrame specifičnog simulacijskog jezika.

(a) Scenariji toka transakcija

U scenarijima toka transakcija sve akcije modela pokreću transakcije, tj. privremeni entiteti modela, čiji je prolaz kroz grupu blokova modela proces.

Najpoznatiji jezik ovog tipa je GPSS. To je najviše korišteni jezik scenarija uopće. Detaljnije ćemo ga opisati u sljedećem poglavljju. Jezik GPSS omogućuje prikaz repova, mesta posluživanja i različitih grupa objekata.

(b) Scenariji ciklusa aktivnosti

Scenariji ciklusa aktivnosti, o kojima smo detaljno pisali kod opisa konceptualnog modeliranja dijagramima ciklusa aktivnosti (DCA), prikazuju cikluse života različitih klasa entiteta pomoću uzastopnih aktivnosti te čekanja na njihov početak. DCA se mogu koristiti za programske jezike orijentirane na procese (svaki životni ciklus privremenog entiteta postaje proces), aktivnosti (uvjeti za početak aktivnosti i sadržaj aktivnosti vide se u DCA) i događaje (događaji su početak i kraj aktivnosti, a vidi se i veza među njima te uvjeti njihovog početka).

Primjeri simulacijskih jezika ovog tipa su ECSL, HOCUS, CAPS, DRAFT, VS6 i DEMOS.

6.2.5. Interaktivni generatori simulacijskih programa

Interaktivni generatori simulacijskih programa su softverski alati koji na temelju opisa karakteristika modela automatski generiraju odgovarajući simulacijski program. Ti alati gotovo bez izuzetka rade interaktivno. Svi danas poznati generatori simulacijskih programa temeljeni su na dijagramima ciklusa aktivnosti. Oni omogućuju da se, na osnovi pripremljenog dijagrama ciklusa aktivnosti (DCA), interaktivno unesu karakteristike DCA korištenjem softvera koji postavlja strukturirana pitanja o karakteristikama modela. Na temelju podataka dobivenih iz odgovora na pitanja generira se odgovarajući simulacijski program. Budući da DCA opisuju samo osnovne karakteristike sistema, opis detaljnih karakteristika rada sistema izvodi se unošenjem odgovarajućih promjena ili proširenja generiranog programskega koda.

Dakle, osnovna karakteristika ovog pristupa je mogućnost brzog i relativno jednostavnog stvaranja programa korištenjem prikaza konceptualnog modela, te automatsko generiranje programa na temelju unesenih podataka. Eventualne daljnje intervencije u programskom kodu olakšane su stvaranjem dobro strukturiranog i istestiranog koda, ali ipak zahtijevaju poznavanje i strukture i funkcije tog koda, kao i programskog jezika u kojem je pisan generirani program (PASCAL, FORTRAN, SIMULA i sl.).

Primjeri interaktivnoga generatora simulacijskog programa su CAPS (koji generira programe u simulacijskom jeziku ECSL orijentiranom na događaje), DRAFT (koji generira program u jednom od odabranih jezika ALGOL, FORTRAN ili SIMULA) i VS6 (koji generira program u jeziku PASCAL).

6.2.6. Generički simulacijski programi pokretani podacima

Generički simulacijski modeli namijenjeni su modeliranju i simulaciji određene klase problema (sistema), tako da se detalji za specifični sistem unutar klase mogu opisati podacima o parametrima tog sistema. Ovdje, dakle, nije potrebno poznавanje simulacijskog programa, koji je 'crna kutija' za korisnika. Cijena jednostavnosti korištenja generičkih modela je ograničenje na opis samo sistema određene klase.

Generički modeli obično su realizirani interaktivnim softverskim alatima te podržavaju animaciju izvođenja simulacije. Primjer ovakvih modela u području proizvodnje su paketi SIMFACTORY, WITNESS i MAST. Nabava ili čak razvoj takvih modela isplati se za organizacije koje se dugoročno bave rješavanjem određenog tipa problema. U razvoju novih paketa razumno je krenuti od gotove biblioteke procedura.

6.3. KRITERIJI IZBORA SIMULACIJSKOG SOFTVERA

Kao što smo upravo pokazali, izbor raznovrsnih alata za simulacijsko modeliranje je velik. Svaki od njih ima neke prednosti i nedostatke. Prednosti i nedostaci moraju se promatrati prije svega kroz namjenu za koju je odgovarajući tip softvera stvoren. Nema simulacijskog softvera, kao ni bilo kojeg drugog softvera ili alata uopće, koji je najbolji za sve moguće namjene.

Da bi se izabrao odgovarajući simulacijski softver, potrebno je prije svega analizirati:

1. zašto se nabavlja simulacijski softver (područje primjene i vrsta problema koji se rješavaju),
2. tko će se koristiti softverom (programeri ili korisnici ili obje kategorije),
3. kakva je očekivana korist (profit) od korištenja modela.

Pri tome, naravno, treba razlikovati zahtjeve pri nabavi simulacijskog softvera za organizaciju koja će se njime dugoročno koristiti od izbora simulacijskog softvera za izvođenje samo jedne simulacijske studije.

Analizom navedenih faktora bitno se suzuje krug odgovarajućih softverskih alata. Unutar kruga preostalih alata potrebno je obratiti pažnju i na sljedeće grupe faktora koje omogućuju daljnje razlikovanje softverskih simulacijskih alata te konačan izbor prikladnog alata (Shannon, 1975):

- fleksibilnost korištenja i odgovarajuća softverska okolina; podrška osnovnim operacijama potrebnim za simulaciju; podrška uklanjanju grešaka kod prevodenja i izvođenja programa
- da li je alat poznat korisniku; da li je instaliran na dostupnom računalu
- cijena nabave i održavanja alata; lakoća učenja i korištenja; postojanje i kvaliteta odgovarajuće literature za učenje, priručnika za korištenje i tečajeva; podrška proizvođača u početnoj fazi primjene alata, te podrška u slučaju pojave grešaka
- prenosivost alata; zahtjevi za memorijom i brzina rada (kod prevodenja i izvođenja programa).

Zbog pada cijena hardvera i porasta cijena ljudskog rada, postepeno se smanjuje značaj potrebnih resursa računala (memorijske, grafičke mogućnosti i sl.), a povećava važnost lakoće i brzine razvoja simulacijskih programa te kvalitete dobivenih rezultata simulacijskih eksperimenata.

6.4. RAZVOJNE TENDENCIJE SIMULACIJSKOG SOFTVERA

Ovo poglavlje zaokružit će moć kratkim prikazom razvojnih tendencija simulacijskog softvera. Budući da je simulacijsko modeliranje dio računarskih znanosti, ono također eksperimentira s novim tendencijama razvoja, metodama i alatima računarstva. Tako se ulaze dosta napora za ispitivanje mogućnosti povezivanja simulacijskog modeliranja s interaktivnom vizualnom grafikom, objektno orijentiranim pristupom programiranju, korištenjem jezika umjetne inteligencije te izvođenjem simulacije na paralelnim računalima.

6.4.1. Vizualna interaktivna simulacija

Vizualna interaktivna simulacija (Hurrian, 1989) vezana je za snažni razvoj računarske grafičke tehnologije 1980-tih godina. Posebno je važan bio proboj te tehnologije na područje relativno jeftinih mikroračunala, što je omogućilo njezino uključivanje u simulacijski softver i široku upotrebu takvog tipa softvera. Vizualna interaktivna simulacija koristi se za:

- razvoj simulacijskih modela
- animaciju ponašanja simulacijskih modela u vremenu
- interakciju s modelom tijekom izvođenja simulacijskog eksperimenta (kako bi se što jednostavnije ispitao utjecaj različitih faktora na rad sistema)
- prikaz rezultata simulacije grafički (histogrami, odresci kruga i sl.).

Vizualna interaktivna simulacija vrlo je važna zbog olakšavanja komunikacije među članovima ekipa za modeliranje (modelari, korisnici) te korisničkova lakšeg prihvaćanja tehnike simulacijskog modeliranja i pouzdanja u nju. Uz nove simulacijske jezike koji su od početka razvijani za vizualnu interaktivnu simulaciju (kao SEE-WHY, SIMFACTORY, VS6) neki od 'klasičnih' simulacijskih jezika također su dobili verzije za animaciju (GPSS, modul SIMANIMATION jezika SIMSCRIPT, modul CINEMA jezika SIMAN).

6.4.2. Objektno orijentirani pristup simulaciji

Objektno orijentirano programiranje promatra programe kao kolekcije aktivnih objekata. Računanje se izvodi porukama kojima objekti međusobno komuniciraju (i njima traže izvršenje određenih zadataka). Objektno orijentirano programiranje također obuhvaća mehanizme nasljeđivanja svojstava od klase objekata. Ovakav pristup programiranju prikidan je za direkstan prikaz elemenata realnog sistema u modelu. Pokazalo se također da objektno orijentirano programiranje omogućuje modularno i produktivno programiranje složenih modela i softverskih sistema.

Već mnogo prije nego što je objektno orijentirano programiranje postalo popularno u računarstvu, ono je bilo uključeno u simulacijski jezik SIMULA (Kreutzer, 1986). Neki od novijih simulacijskih paketa razvijeni su u objektno orijentiranim tehnikama umjetne inteligencije. To su npr. paketi SimKit, ROSS i KBS. Kompleksan pristup objektno orijentiranoj simulaciji dan je u knjizi (Zeigler, 1990).

6.4.3. Korištenje metoda i alata umjetne inteligencije

Umjetna inteligencija privukla je veliko zanimanje istraživača iz različitih područja računarstva, pa tako i simulacijskog modeliranja (Paul, 1989). Od posebnog interesa su pri tom metode prikaza znanja, logičko i funkcionalno programiranje i ekspertni sistemi.

Tako su među ostalim razvijene verzije jezika PROLOG (TS-PROLOG, CS-PROLOG) koje omogućuju simulaciju orijentiranu na cilj, tj. umjesto specifikacije slijeda akcija dinamičkih entiteta one proistječu iz cilja koji je tom entitetu zadan. Realizacija cilja postiže se traženjem logičkog lanca uzroka i posljedica, definiranih pravilima. Ako izabrani logički put ne vodi do zadanog cilja, koristi se tzv. 'backtracking', tj. povratak u prethodno stanje (aktivnost) i pokušaj traženja novog logičkog puta do cilja ispitivanjem svih mogućih putova u logičkom stablu.

Nekoliko simulacijskih paketa napravljeno je korištenjem jezika LISP, jezika OPS5 temeljenog na produpcionim pravilima, i jezika koji se koriste prikazom znanja temeljenog na tzv. okvirima.

6.4.4. Simulacija na paralelnim računalima

Simulacije složenih inženjerskih, ekonomskih, vojnih i sl. problema zahtijevaju potrošak velikih iznosa vremena na sekvenčnim računalima. Pojava paralelnih računala stvorila je mogućnost da se izvođenje simulacijskih eksperimentata znatno ubrza. Istraživanja posljednjih godina (Fujimoto, 1990) pokazala su međutim da je problem paralelizacije izvođenja simulacije veoma složen. Osnovni razlog za to jest nepravilna, asinkrona priroda odvijanja simulacije u kojoj se događaji pojavljuju u nepravilnim vremenskim intervalima, bez međusobne sinkronizacije.

Dva osnovna pristupa simulaciji na paralelnim računalima su tzv. konzervativni i optimistički pristup. *Konzervativni pristup* striktno izbjegava mogućnost dešavanja uzročne greške, tj. izvođenja događaja koji ovisi o nekom prethodnom događaju koji međutim još nije izveden. Ovaj se pristup temelji na strategijama određivanja kada je izvođenje događaja sigurno, tj. kada su svi događaji koji mogu utjecati na taj događaj već izvedeni. Pristup je pogodan za neke klase problema, te za situacije u kojima se primjenjuju specifična znanja o sistemu kako bi se maksimalizirala efikasnost simulacijskog mehanizma.

Optimistički pristup naprotiv dopušta nezavisan vremenski razvoj svakog procesa u simulacijskom modelu. Kada se otkrije da je nastala uzročna greška, poziva se mehanizam povratka na prethodna stanja ('roll-back'). Taj se pristup koristi u simulacijama općeg tipa, i njime su postignuti odlični rezultati u različitim primjenama.

7. SIMULACIJSKI JEZIK GPSS

Za demonstraciju simulacijskog softvera odabrali smo simulacijski jezik GPSS zbog njegove jednostavnosti i rasprostranjenosti. Prikazat ćemo osnovne koncepte jezika, tipove entiteta koje podržava i njegov upravljački mehanizam. Zatim ćemo opisati osnovne upravljačke naredbe, naredbe za definiciju blokova i definiciju entiteta. Mogućnosti jezika GPSS demonstrirat ćemo modeliranjem i simulacijom nekoliko jednostavnih sistema.

7.1. UVOD

Simulacijski jezik GPSS (General Purpose Simulation System) jedan je od prvih jezika simulacijskog modeliranja, a ujedno i jedan od prvih simboličkih programskih jezika uopće. Razvio ga je G. Gordon 1961. godine, i od tada je on najpopularniji i najviše korišteni simulacijski jezik. Svoju popularnost ponajviše duguje jednostavnom konceptu toka transakcija, pristupu orijentiranom na procese koji na prirodan način opisuju sisteme s repovima te snažnim naredbama koje omogućuju da se sistem prikaže relativno malim programom.

Jezik GPSS preživio je veći broj inovacija koje su se dogodile u razvoju simulacijskog modeliranja i računarskih znanosti. Pri tom je razvijeno niz različitih verzija jezika (i prevodioca i interpretatora), pojačane su njegove mogućnosti (novi tipovi entiteta i nove naredbe, komunikacija s općim programskim jezicima), uvedeni algoritmi koji značajno ubrzavaju izvođenje simulacije (npr. u verziji jezika GPSS/H), jezik je implementiran na mikroričunalima i omogućena je animacija izvođenja simulacije (npr. verzija GPSS/PC). Verzija jezika koja je ovdje opisana, GPSS/360, praktično je standardna verzija koja je opisana u gotovo svoj literaturi o jeziku GPSS (Bobillier i dr., 1976).

Smatramo da je jezik GPSS pogodan za početno učenje korištenja softverskih alata za simulacijsko modeliranje, i to iz nekoliko razloga: on omogućuje relativno jednostavan i jezgrovit opis jednostavnih sistema prikladnih za učenje, veoma je rasprostranjen i moguće ga je naći kako u verziji za mikroričunala, tako i za velika računala, a i literatura s opisom jezika i njegove primjene je bogata (na žalost ne na našem jeziku).

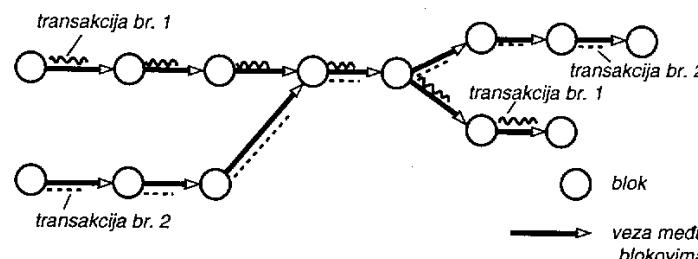
7.2. OSNOVNI KONCEPTI JEZIKA GPSS

7.2.1. Pregled karakteristika jezika GPSS

Osnovne ideje modeliranja jezikom GPSS i njegovim načinom rada su:

- (1) model je prikazan nizom međusobno povezanih blokova,
- (2) ima više različitih tipova blokova, od kojih svaki predstavlja neku specifičnu akciju modela,

(3) dinamički (privremeni) entiteti modela, zvani transakcije, kreću se kroz blokove modela i time pokreću željene akcije modela (slika 7.1).



Slika 7.1. Prolaz transakcija kroz GPSS blokove

Entiteti modela mogu biti:

- stalni entiteti (facility, storage, queue itd.)
- privremeni entiteti (transakcije).

Prikupljanje statistike izvođenja simulacije:

- osnovna statistika prikuplja se *automatski* (brojači blokova, statistika stalnih entiteta)
- može se zadati i dodatno prikupljanje statistike (tablice).

Upravljanje radom modela, tj. slijedom izvođenja akcija simulacije, izvodi se *automatski*.

Spremanje vrijednosti koje se izračunavaju u toku izvođenja simulacije:

- informacije koje se tiču samo transakcija spremaju se u parametrima transakcija
- informacije koje se tiču cijelog sistema spremaju se u tzv. *save* lokacijama (spremnicima vrijednosti).

Napomena: U nalaženju prikladnog prijevoda specifičnih pojmoveva kojima se koristi jezik GPSS prisutne su određene teškoće. Zbog toga, kao i zbog lakšeg razumijevanja informacija koje GPSS procesor daje na izlazu simulacijskog eksperimenta, većinu ćemo pojmoveva navoditi u originalu, a pri prvom spominjanju pojma dati njegov opisni prijevod.

7.2.2. Stalni entiteti

Stalni entiteti jezika GPSS jesu:

FACILITY (mjesto posluživanja)

- *facility* je tip entiteta koji može prihvati samo jednu transakciju u jednom času;
- primjeri: blagajne, šalteri, benzinske crpke
- osnovni GPSS blokovi su SEIZE i RELEASE.

STORAGE (skladište)

- *storage* je tip entiteta koji u jednom času može prihvati više jedinica storagea koje mu donose transakcije, a ima i konačan kapacitet;
- primjeri: parking, hotelske sobe, memorija računala
- osnovni GPSS blokovi su ENTER i LEAVE.

LOGIC SWITCH (logički prekidač)

- *logic switch* je indikator sa dva moguća stanja: 'set' (uključen) i 'reset' (isključen);
- primjeri: semafor, ventil
- osnovni GPSS blokovi su LOGIC i GATE.

QUEUE (rep)

- *queue* je entitet za automatsko prikupljanje statistike o vremenu čekanja i dužini repa transakcija (odnosno jedinica koje transakcije unose u rep) pred ulazom u blok s uvjetnim ulazom (npr. zahtjev za stalnim entitetom, testiranje uvjeta), a ima i neograničen kapacitet;
- primjeri: čekanje pred blagajnom, strojem, semaforom
- osnovni GPSS blokovi su QUEUE i DEPART.

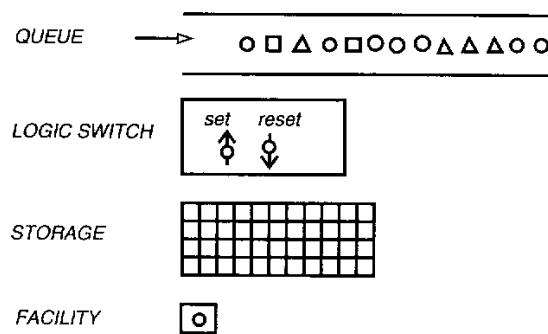
USER CHAIN (korisnički lanac)

- *user chain* je entitet koji može prihvatiti bilo koji broj transakcija što će dezaktiviraju ulaskom u određeni tip bloka. Te transakcije mogu se aktivirati samo od drugih aktivnih transakcija, kada ove aktivne transakcije uđu u blok za aktivaciju. Ovi entiteti služe za ubrzavanje rada modela, mogućnost reorganizacije repova ili promjene slijeda operacija modela;
- osnovni GPSS blokovi: su LINK i UNLINK.

GROUP (grupa)

- *group* je skup čiji članovi mogu biti transakcije ili numeričke vrijednosti, i koji imaju neki zajednički atribut;
- primjeri: slobodne sobe u hotelu, strojevi u kvaru
- osnovni GPSS blokovi su JOIN i LEAVE.

Na slici 7.2. prikazani su simbolički entiteti tipa *facility*, *storage*, *logic switch* i *queue*.



Slika 7.2. Stalni GPSS entiteti

7.2.3. Privremeni entiteti

Privremeni entiteti jezika GPSS zovu se transakcije. Njihove su osnovne karakteristike:

- transakcije se stvaraju u bloku GENERATE, a uništavaju u bloku TERMINATE
- transakcije putuju kroz blokove sekvencijalno, osim kad su usmjerene na skok u neki blok (označen odgovarajućom labelom)
- transakcije provode nula jedinica simulacijskog vremena u bloku, osim kad su zaustavljene zbog jednog od ovih mogućih razloga:
 - a) vrijeme zaustavljanja specificirano u bloku ADVANCE (trajanje posluživanja),
 - b) ulaz u sljedeći blok nije moguć (stalni entitet koji se traži nije slobodan ili zadani uvjet ulaska u blok nije ispunjen),
 - c) transakcija je dezaktivirana (vezanje u korisnički lanac),
 - d) transakcija je uništena (ulaz u TERMINATE blok),
- transakcija se može umnožiti, odnosno podijeliti na više identičnih transakcija, koje se kasnije mogu opet spojiti u jednu transakciju.

7.2.4. GPSS događaji

Osnovne karakteristike događaja u jeziku GPSS jesu:

- Kada se transakcije gibaju kroz blokove modela stanje modela se mijenja, pa su to događaji u radu modela.
- Informacije koje definiraju događaj (atributi transakcija) su: sadašnji blok transakcije, adresa sljedećeg bloka, planirano vrijeme napuštanja bloka i prioritet transakcije.
- Zbog potreba upravljanja tijekom simulacije GPSS smješta transakcije u *uredene liste* zvane *lanci*. GPSS automatski podržava dva tipa lanaca transakcija:

Lanci sadašnjih događaja (LSD)

LSD sadrže transakcije koje su spremne za gibanje čim im to dopuste *uvjeti* koji su ih zaustavili. Kod njih je dakle

$$VNB \leq CI, \text{ gdje je}$$

VNB planirano vrijeme napuštanja bloka
CI sadašnje vrijeme simulacijskog sata.

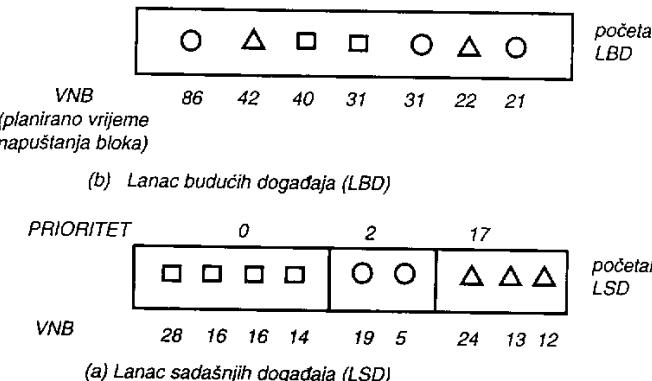
LSD su uređeni po padajućim vrijednostima prioriteta, a unutar grupe transakcija istog prioriteta po rastućim vrijednostima VNB.

Lanci budućih događaja (LBD)

LBD sadrže transakcije planirane za gibanje u neko buduće vrijeme, tj.
 $VNB > CI$

LBD su uređeni po rastućim vrijednostima VNB.

Na slici 7.3. prikazano je uređenje lanaca sadašnjih i budućih događaja.



Slika 7.3. Lanici sadašnjih i budućih događaja

7.2.5. Moguća stanja transakcija

Transakcija tijekom svog 'života' u modelu može biti u jednom od ovih stanja:

(1) Aktivno stanje

Transakcija je u aktivnom stanju kada se giba po modelu, dakle kada je pod kontrolom GPSS upravljačkog programa.

(2) Suspendirano stanje

Transakcija je u suspendiranom stanju kada je u LSD ili u LBD. Ona tada čeka da je aktivira GPSS upravljački program.

(3) Pasivno stanje

Transakcija je u pasivnom stanju kada je u korisničkom lancu. Može je aktivirati samo neka druga aktivna transakcija prilikom ulaza u UNLINK blok.

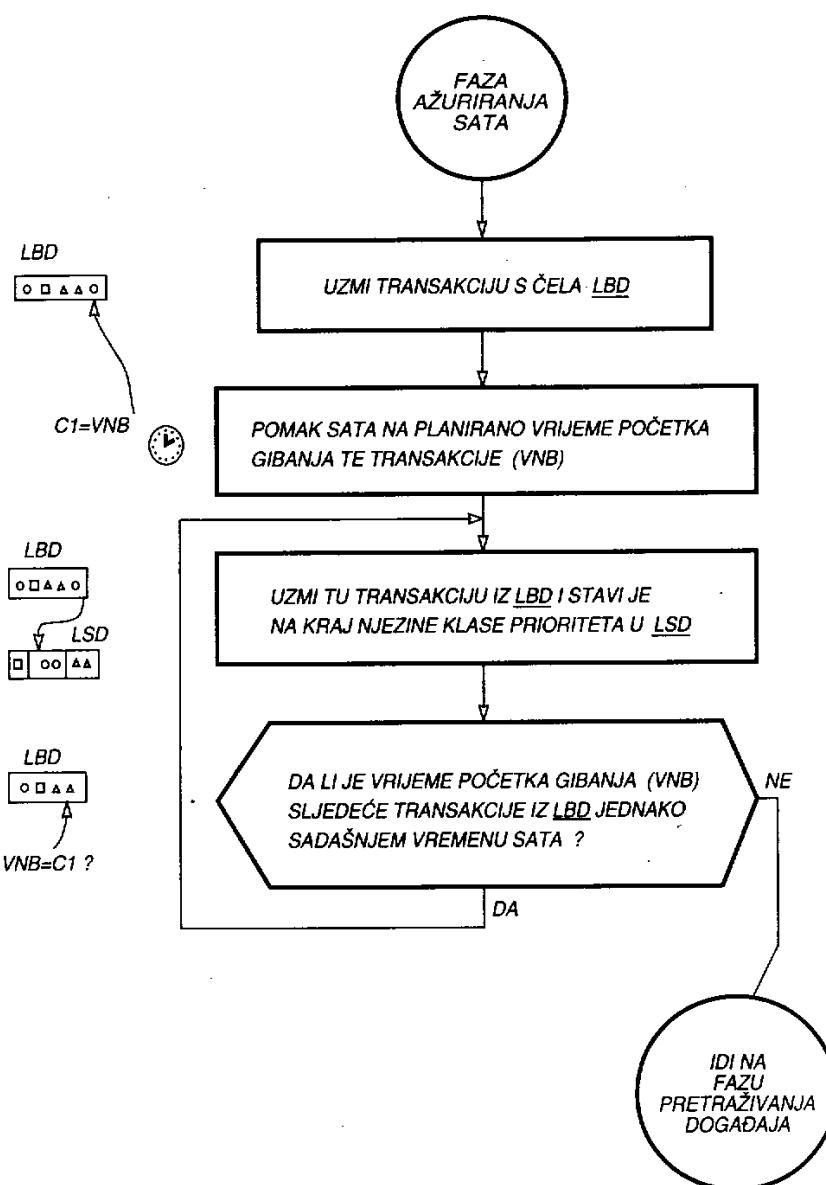
(4) Terminirano stanje

Transakcija je u terminiranom stanju (tj. više ne postoji u modelu) kada je uništena ulaskom u TERMINATE blok.

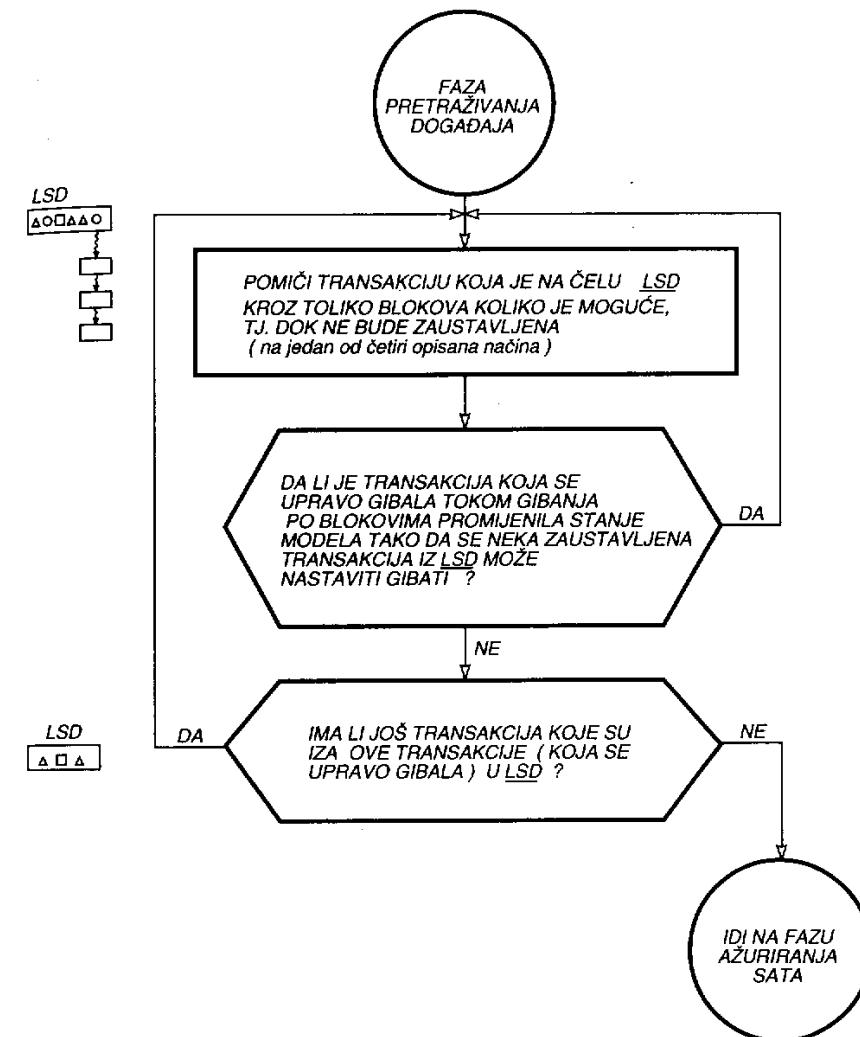
7.2.6. GPSS upravljački mehanizam

Ključ upravljanja slijedom operacija u simulaciji su *ažuriranje simulacijskog sata* i *pretraživanje događaja* (Schriber, 1974). Pri tome se GPSS procesor koristi lancem sadašnjih događaja (LSD) i lancem budućih događaja (LBD). Princip rada tih dviju ključnih faz simualcije prikazan je na slikama 7.4. i 7.5.

Kao što se vidi na slici 7.4., u fazi *ažuriranja simulacijskog sata* sat se pomiče na planirano vrijeme početka gibanja transakcije s čela LBD (koja ima najranije planirano vrijeme napuštanja bloka). Zatim se iz LBD prebacuju u LSD sve transakcije čije je vrijeme planiranog početka pomaka jednak onom koje ima transakcija s čela LBD (transakcije se prebacuju u LSD na kraj njihove klase prioriteta). Kada su sve transakcije prebačene u LSD, prelazi se na fazu pretraživanja događaja koji se mogu izvesti u tom času simulacijskog vremena.



Slika 7.4. Faza ažuriranja sata jezika GPSS (Schriber, 1974)

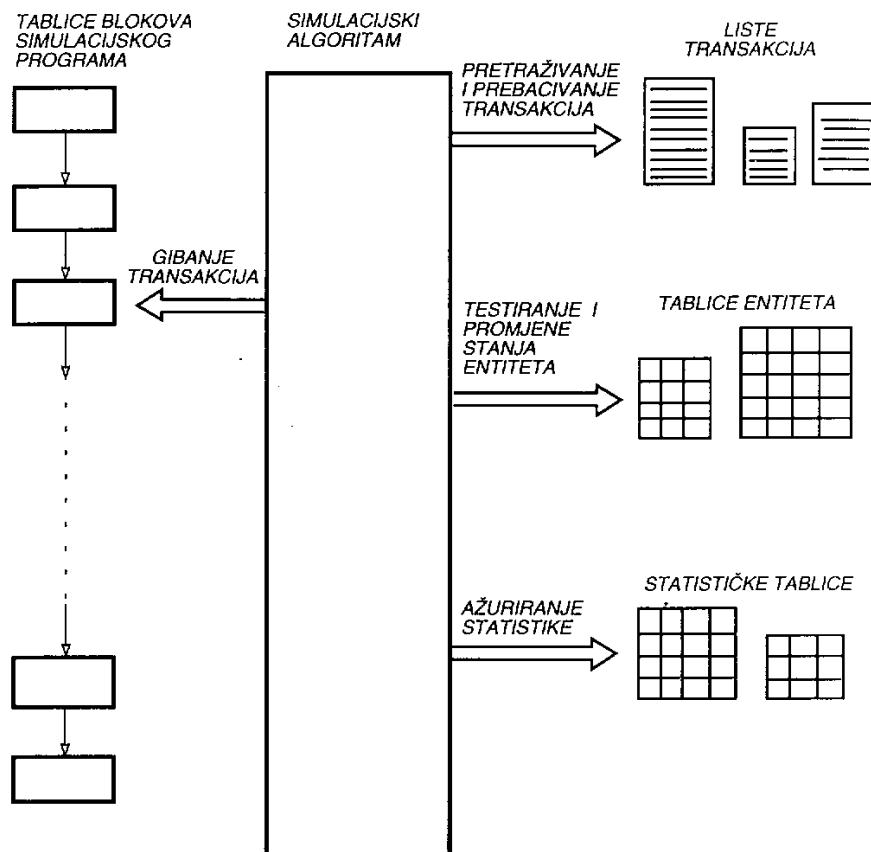


Slika 7.5. Faza pretraživanja događaja jezika GPSS (Schriber, 1974)

Na slici 7.5. vidi se da se u fazi pretraživanja događaja transakcije s čela LSD pomiču toliko blokova u simulacijskom modelu koliko je to moguće, tj. sve dok transakcija ne bude zaustavljena jednim od mogućih mehanizama zaustavljanja. Ako se tijekom gibanja transakcije promijenilo stanje modela koje omogućuje aktiviranje neke od zaustavljenih transakcija iz LSD, tada se i te transakcije pokreću. Ovaj postupak traje sve dok još ima transakcija u LSD koje se mogu gibati. Nakon toga prelazi se na fazu ažuriranja simulacijskog sata, tj. na pomak sata na sljedeći događaj.

Simulacija završava kada više nema novih događaja ili kada je izведен specifičan događaj završetka simulacije.

Pomaci transakcija zapravo ne postoje, jer transakcije nisu stvarni objekti, već strukture podataka. Dakle, rad GPSS algoritma zapravo predstavlja promjenu stanja različitih tablica koje opisuju stanja blokova, entiteta itd. (Gordon, 1969). Promjene koje izvodi simulacijski algoritam simbolički su prikazane na slici 7.6.



Slika 7.6. Korištenje GPSS internih tablica pri izvođenju simulacije (Gordon, 1969)

7.3. OSNOVNE NAREDBE JEZIKA GPSS

Opisat ćemo osnovne GPSS naredbe koje omogućuju konstrukciju jednostavnih simulacijskih programa u jeziku GPSS, te razumijevanje načina funkcioniranja GPSS programa (Bobillier i dr., 1974). Kako bi se olakšalo učenje osnova jezika GPSS, odabran je vrlo ograničen skup potrebnih naredbi, a ujedno su izostavljeni i neki operandi u naredbama, odnosno neke moguće alternative operanada.

7.3.1. Naredbe upravljanja

(1) Zahtjev za simulacijom

SIMULATE

To je prva naredba u simulacijskom programu. Ako nje nema, tada se samo ispituje ispravnost programa u fazi prevodenja programa.

(2) Upravljanje izvođenjem simulacije

START A,B

Ovom naredbom inicira se početak izvođenja simulacije, a ujedno se definiraju uvjeti završetka simulacije. Operandi su:

A: brojač završetka simulacije, koji se smanjuje za vrijednost operanda A u naredbi TERMINATE kada u nju uđe transakcija. Simulacija završava kada ovaj brojač padne na vrijednost koja je ≤ 0 .

B: ako je na njegovu mjestu napisano NP, tada se nakon završetka simulacije ne štampa statistika izvođenja simulacije.

Primjer:

A TERMINATE 5

B TERMINATE

C TERMINATE 100

START 100

Simulacija završava kada ili 20 transakcija uđe u blok A ili jedna transakcija uđe u blok C. Ulazak transakcija u blok B ne utječe na završetak simulacije.

Na kraju simulacije štampa se statistika izvođenja simulacije.

(3) Završetak simulacije

END

To mora biti posljednja naredba u programu.

(4) Pražnjenje modela

CLEAR selektori (save lokacije)

Ova naredba reinicijalizira model na početno stanje, tj. izbacuje transakcije iz modela te inicijalizira na nulu statistiku, save lokacije, vrijeme i logičke prekidače. Ne stavlja na početne vrijednosti jedino generatore slučajnih brojeva i selektirane save lokacije navedene u naredbi. Stvara novu transakciju u svakom GENERATE bloku.

Primjer:

CLEAR X3, X5-X7

Prazni model, ali ne inicijalizira save lokacije 3, 5, 6 i 7.

(5) Brisanje akumulirane statistike

RESET selektori (facility, storage, repovi)

Ova naredba briše akumulirane statističke podatke i stavlja relativni sat modela na nulu. Ne modificira stanje modela, tj. transakcije, save lokacije itd. Selektirane entitete ne briše.

Primjer:

RESET S3, F2-F4

Briše svu statistiku osim one koja je povezana za storage 3 i facilityje 2, 3 i 4.

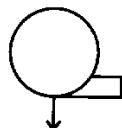
7.3.2. Naredbe za definiranje blokova

Naredbe za definiranje blokova moraju biti napisane u formatu:

2	8	19
labela	operacija	operandi

Blokovi orientirani na transakcije

(1) Stvaranje transakcija



GENERATE A, B, C, D, E

Ovaj blok periodički stvara transakcije koje odlaze u sljedeći blok. Ako transakcija ne može ući u sljedeći blok, vrijeme među uzastopnim stvaranjem transakcija ne počinje se računati sve dok sadašnja transakcija ne uspije ući u sljedeći blok. Time se poremeti zadani ritam stvaranja transakcija. Operandi su:

A: srednje vrijeme koje protekne između stvaranja uzastopnih transakcija.

Primjer:

GENERATE 35

Transakcije se generiraju svakih 35 vremenskih jedinica (s time da se prva generira 35 vremenskih jedinica nakon časa 0, tj. u času 35).

B: modifikator srednjeg vremena stvaranja transakcija.

a) B je konstanta

Tada je vrijeme između dvaju uzastopnih stvaranja transakcija jednolikou raspodijeljeno između vrijednosti A-B do A+B. Uvijek mora biti B ≤ A

Primjer:

GENERATE 60, 15

Transakcije se generiraju svakih 45–75 vremenskih jedinica.

b) B je funkcija

Tada je vrijeme između dvaju uzastopnih generiranja jednako A*funkcija, s tim da se vrijednost funkcije izračunava u času generiranja.

Primjer:

GENERATE 40,FN\$DOLAZ

Transakcije se generiraju svakih 40*FN\$DOLAZ vremenskih jedinica.

C: forsirano vrijeme generiranja prve transakcije.

Primjer:

GENERATE 45, 15, 300

Prva transakcija se generira u času 300, a sljedeće svakih 30–60 vremenskih jedinica.

D: ukupan broj transakcija koje blok može generirati.

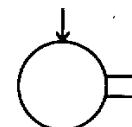
E: prioritet generiranih transakcija (0–127). Ako operand E nije zadan, prioritet je 0 (najniži prioritet).

Primjer:

GENERATE 25,,10, 3

Generiranje ukupno 10 transakcija prioriteta 3 svakih 25 vremenskih jedinica.

(2) Uništavanje transakcija

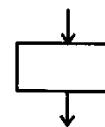


TERMINATE A

Uništavanje transakcija koje uđu u blok. Operand:

A: broj jedinica za koje se smanjuje brojač završetka simulacije (naveden kao operand A naredbe START).

(3) Vremensko zadržavanje transakcije



ADVANCE A, B

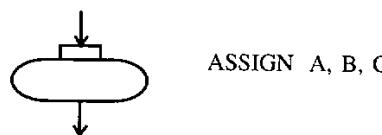
Zadržavanje transakcija u bloku. Operandi A i B imaju isto značenje kao i odgovarajući operandi bloka GENERATE (srednje vrijeme zadržavanja u bloku i njegov modifikator).

Primjer:

ADVANCE P2

Zadržavanje transakcije za P2 vremenskih jedinica (tj. za vrijednost parametra 2 transakcije koja je došla u blok).

(4) Baratanje atributima transakcije



Promjena sadržaja parametra transakcije. Operandi:

A: broj parametra u koji se spremi vrijednost koja slijedi. Ako je iza tog broja + ili -, tada se tom parametru dodaje ili oduzima vrijednost koja slijedi.

B: vrijednost koja se spremi, dodaje ili oduzima

C: ako je zadan, tada je to broj funkcije koja se izračunava i množi operandom B i tako dobivena vrijednost se spremi, dodaje ili oduzima u zadani parametar.

Primjeri:

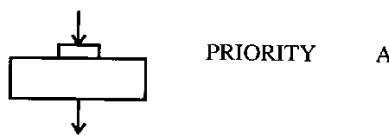
ASSIGN 3, P1, 4

U parametar 3 transakcije spremi se vrijednost parametra 1 pomnožena s vrijednosti funkcije 4.

ASSIGN 1 +, S3

Vrijednost parametra 1 povećava se za tekuće popunjene 'storagea' 3.

(5) Postavljanje prioriteta transakcije



Postavljanje (ili zamjena) vrijednosti prioriteta transakcije. Operand:

A: vrijednost prioriteta

Primjer:

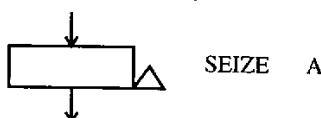
PRIORITY 5

Prioritet transakcije postavlja se na vrijednost 5.

Blokovni orijentirani na opremu

a) Facility

(1) Zauzimanje facilityja



Transakcija ne može ući u blok dok facility A nije slobodan. Čim facility postane slobodan, transakcija ulazi u blok i zauzima facility te nastavlja u sljedeći blok. Nijedna druga transakcija ne može zauzeti facility dok ga transakcija koja ga je zauzela ne oslobodi. Operand:

A: ime ili broj facilityja.

(2) Oslobođenje facilityja



Oslobođenje facilityja A. Nema uvjeta za ulaz u ovaj blok, ali facility može oslobođiti samo transakcija koja ga je zauzela (inace nastaje greška u izvođenju simulacije). Operand:

A: ime ili broj facilityja.

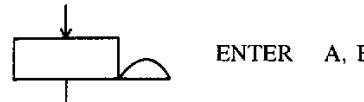
Primjer:

SEIZE KRAN
ADVANCE 25
RELEASE KRAN

Transakcija zauzima facility KRAN kada on postane slobodan, zadržava ga 25 vremenskih jedinica i tada ga oslobađa.

b) Storage

(1) Zauzimanje storagea



Transakcija može ući u blok samo ako u storageu A ima bar B slobodnih jedinica storagea. Čim uđe u blok, transakcija zauzima B jedinica storagea A i nastavlja u sljedeći blok. Operandi:

A: ime ili broj storagea.

B: broj jedinica storagea koji traži transakcija. Ako operand B nije zadan, tada je njegova vrijednost 1.

(2) Oslobođenje storagea



Transakcija oslobađa B jedinica storagea A. Nema uvjeta za ulaz u blok.

(Zato je u složenijim situacijama razumno prije ulaza u ovaj blok testirati da li u storageu ima dovoljno jedinica, kako bi se izbjegla moguća greška u izvođenju simulacije i prekid izvođenja).

Operandi:

A: ime ili broj storagea.

B: broj jedinica storagea koje se oslobađaju iz storagea A. Taj broj mora biti manji ili jednak tekućem popunjenu storagea. Ako operand B nije zadan, tada je njegova vrijednost 1.

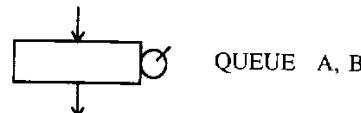
Primjer:

ENTER HOTEL, P1
ADVANCE 15
LEAVE HOTEL P1

Transakcija zauzima P1 jedinica storagea HOTEL (npr. soba), i to tek kada ih toliko bude slobodno (aranžman za grupu), zadržava se 15 dana i tada oslobađa svih P1 soba.

c) Rep

(1) Ulazak transakcije u rep

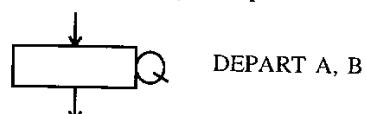


Transakcija postaje član repa A i u njega unosi B jedinica repa, te nakon toga nastavlja u sljedeći blok. Nema uvjeta za ulazak u rep jer rep ima neograničen kapacitet. Operandi:

A: ime ili broj repa.

B: broj jedinica repa koje se dodaju repu. Ako ovaj operand nije zadan, tada je B jednak 1.

(2) Izlazak transakcije iz repa



Izlazak transakcije iz repa A i smanjenje sadržaja repa za B jedinica. Operandi:
A: ime ili broj repa.

B: broj jedinica koje se iznose iz repa. Ako nije zadan, operand B ima vrijednost 1.

Primjer:

```

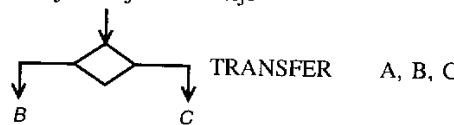
QUEUE CEKA, P1
SEIZE BLAG
DEPART CEKA, P1
ADVANCE 15, 5
RELEASE BLAG
  
```

Rep CEKA mjeri čekanje pred facilityjem BLAG dok je on zauzet. Pri ulasku u rep CEKA transakcija unosi u rep P1 jedinica. Transakcija zauzima facility 15+5 vremenskih jedinica.

(Napomena: I bez specificiranja repa CEKA transakcija bi čekala dok je facility zauzet, ali se to čekanje ne bi mjerilo a možda ni zapazilo.)

Modifikacija toka transakcija

(1) Usmjeravanje transakcije



Usmjeravanje transakcije u sljedeći blok na nekoliko različitih načina. Operandi:
A: način usmjeravanja

1. Ako nije definiran, transakcija ide bezuvjetno u blok s labelom B.

Primjer:

TRANSFER , IZLAZ

Transakcija ide u blok s labelom IZLAZ.

2. Ako se napiše BOTH, tada transakcija pokušava najprije ući u blok specificiran u B, pa onda u blok specificiran u C (ako ulaz u B nije moguć). Ako ne može ući ni u jedan od ta dva bloka, onda se zahtjev za ulazom ponavlja pri sljedećoj promjeni stanja modela.

Primjer:

TRANSFER BOTH,,VAN

Transakcija redom pokušava ući u sljedeći blok pa u blok VAN. Ako ne može ući ni u jedan od ta dva bloka, transakcija čeka ispred ulaza u blok sve dok promjena stanja ne omogući da uđe u jedan od tih dvaju blokova.

3. Ako ovdje piše realni broj veličine od 0. do 1, tada se transakcija usmjeruje statistički. Pri tome broj zadan u A predstavlja vjerojatnost da transakcija uđe u blok specificiran u C (a vjerojatnost ulaska u blok specificiran u B jednaka je 1-A).

Primjer:

TRANSFER 0.65, ULAZ, IZLAZ

Transakcije u 65 % slučajeva idu u blok specificiran u C, a u ostalih 35 % slučajeva u blok specificiran u B.

(2) Testiranje statusa stalnih entiteta



Ispituje se status X stalnog entiteta A. Ako operand B nije zadan, tada je transakcija prihvaćena u blok ili odbijena (ovisno o ispunjavanju uvjeta X). Ako je B operand zadan, tada transakcija ide u sljedeći sekvenčnalni blok ako je uvjet X ispunjen, a u blok specificiran u B ako uvjet X nije ispunjen. Operandi:

A: ime ili broj stalnog entiteta.

B: alternativni izlaz ako uvjet nije ispunjen. Ako B nije definiran, tada transakcija ne može ući u blok ako uvjet nije ispunjen, pa mora čekati u prethodnom bloku dok se uvjet ne ispuni.

X: uvjet postavljen na stalni entitet A.

Za facility:

U	facility se koristi	('used')
NU	facility se ne koristi	('not used')

Za storage:

SF	storage je pun	('full')
SNF	storage nije pun	('not full')
SE	storage je prazan	('empty')
SNE	storage nije prazan	('not empty')

Primjeri:

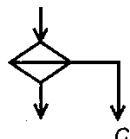
GATE NU AUTO

Transakcija može ući u blok samo ako facility AUTO nije zauzet. Dok se taj uvjet ne ispunii, transakcija mora čekati u prethodnom bloku.

GATE SE HOTEL, DALJE

Transakcija ulazi u blok ako je storage HOTEL prazan, a ako nije, ulazi u blok DALJE.

(3) Testiranje uvjeta



TEST X A, B, C

Uvjet X ispituje se operandima A i B. C je uvjetni izlaz (kao i operand B bloka GATE). Operandi:

A i B: numerički atributi koji se uspoređuju korištenjem logičkog operatorka X.

C: alternativni izlaz ako uvjet nije ispunjen. Ako C nije zadani, tada transakcija ne može ući u blok ako uvjet nije ispunjen, pa mora čekati u prethodnom bloku dok se uvjet ne ispunii.

X: logički operator

E A = B (odnosno: relacija je istinita ako je A = B)

NE A nije jednak B

L A < B

LE A <= B

G A > B

GE A >= B

Primjeri:

TEST LEP3, 20

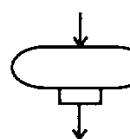
Transakcija može ući u blok samo ako je P3 <= 20, a inače čeka pred blokom da se taj uvjet ispunii.

TEST E Q1, FN\$DUZIN, VAN

Transakcija može ući u blok ako je Q1 = FN\$DUZIN, tj. dužina repa 1 jednaka je u tom času izračunanoj vrijednosti funkcije DUZIN. Ako to nije ispunjeno, transakcija ide u blok s labellom VAN.

Spremanje i pretraživanje informacija

(1) Spremanje informacija u save lokacije



SAVEVALUE A, B

Spremanje, dodavanje ili oduzimanje informacija iz operanda B u save lokaciju A.

Operandi:

A: ime ili broj save lokacije. Ako iza njega slijedi + ili - tada se vrijednost iz operanda B dodaje, odnosno oduzima od sadašnje vrijednosti save lokacije A.

B: vrijednost koja se sprema, dodaje ili oduzima.

Primjeri:

SAVEVALUE 4, S1

U save lokaciju 4 sprema se vrijednost sadašnjeg popunjena storagea 1 (broj jedinica u njemu).

SAVEVALUE P1+, Q2

Sadašnjoj vrijednosti one save lokacije čiji se broj nalazi u parametru 1 transakcije dodaje se tekući broj jedinica u repu 2.

7.3.3. Definicije entiteta

Definicije entiteta pišu se u sljedećem formatu:

2	8	19
labela	operacije	operandi

(1) Definicija storagea

Kapaciteti storagea (broj jedinica storagea koje storage može primiti) mogu se definirati na dva načina:

	STORAGE	S1, 25/S\$PARK, 40
ili		
	I STORAGE	25
	PARK STORAGE	40

(2) Definicija varijabli

Varijabla je entitet čija se vrijednost izračunava onda kada je referenciran. Varijabla se referencira (poziva) sa Vj (j je broj varijable) ili V\$ime.

Aritmetička varijabla se definira ovako:

labela	VARIABLE	aritmetički izraz
--------	----------	-------------------

Labela: broj ili ime varijable.

Aritmetički izraz: u njemu se mogu koristiti bilo koji numerički atributi te aritmetički operatori (+, -, *, /). U njemu ne smije biti praznina. Tijekom izračunavanja svi elementi izraza se pretvaraju u cijele brojeve, isto kao i izračunana vrijednost varijable.

Primjeri:

DUZIN	VARIABLE	Q1+Q3/2
5	VARIABLE	X\$GOD/12+V3

Pri referenciranju varijable 5 uzima se sadašnja vrijednost save vrijednosti GOD i izračunava vrijednost varijable 3, te se pomoću toga izračuna aritmetički izraz koji daje vrijednost varijable 5.

(3) Definiranje funkcija

Kao i kod varijabli, vrijednost funkcije se izračunava u času njezina referenciranja. To su funkcije jedne varijable. Argument funkcije je numerički atribut (npr. vrijednost parametra, varijable ili neke druge funkcije). Vrijednost funkcije u pravilu pretvara se u cijeli broj, osim kada se ona koristi kao funkcionalni modifikator u blokovima GENERATE, ADVANCE i ASSIGN ili kao argument druge funkcije.

Format naredbe za definiciju funkcije je:

ime FUNCTION A, B koordinate

Operandi su:

Ime: ime funkcije.

A: argument funkcije (numerički atribut).

B: tip funkcije, uz koji je dodan broj točaka koje je specificiraju.

Koordinate: parovi koordinata x, y koji definiraju funkciju, međusobno odijeljeni kosom crtom /.

Opisat ćemo tri tipa funkcija: kontinuirane (C), diskretne (D) i liste (L), i to primjerima sa slike 7.7a, b i c:

FUNK1 FUNCTION RN5, C4
0.8/0.40, 11/0.90, 12/1., 14

Kontinuirana funkcija definirana s četiri točke, čiji je argument slučajni broj generiran od generatora br. 5.

FUNK2 FUNCTION X1, D3
10,60/15,80/20,20

Diskretna funkcija definirana s tri točke, čiji je argument vrijednost save lokacije 1.

FUNK3 FUNCTION P3, L7
.20/.30/.35/.25/.15/.5/.5

Lista definirana sa sedam točaka, čiji je argument vrijednost parametra 3 transakcije. Pretpostavlja se da koordinate x točaka poprimaju redom vrijednosti 1,2,3,4 ..., tako da se njihova vrijednost ne mora pisati.

Različite teorijske funkcije razdoblje vjerojatnosti potrebne u simulaciji definiraju se aproksimativno, i to obično s dvadeset do trideset točaka.

(4) Inicijalizacija vrijednosti save lokacija

Vrijednosti save lokacija mogu se inicijalizirati prije početka izvođenja simulacije s naredbom:

INITIAL A, B

gdje su operandi:

A: ime ili broj save lokacije (ispred kojih je oznaka X),

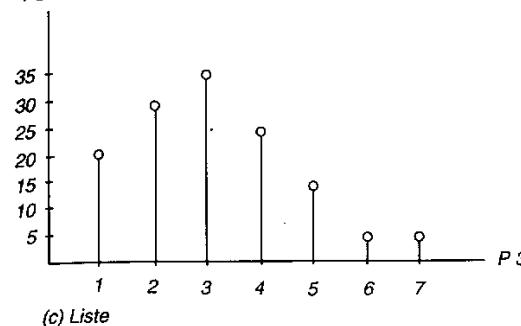
B: vrijednost na koju se save lokacija inicijalizira.

Primjeri:

INITIAL X1, 20

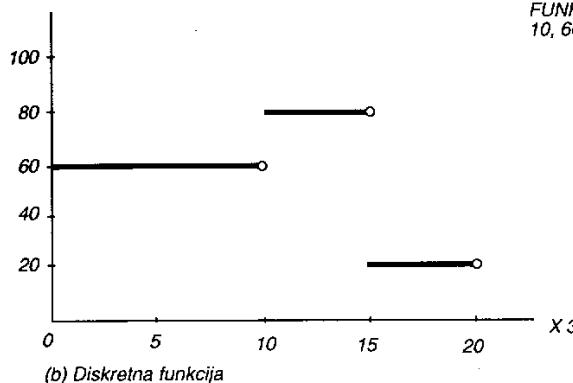
INITIAL X\$SUMA, 1000

FUNK3



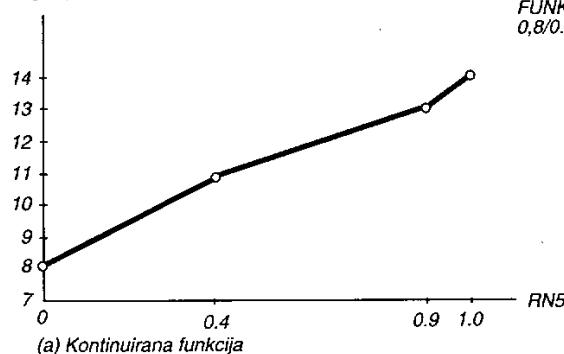
FUNK3 FUNCTION P3, L7
.20/.30/.35/.25/.15/.5/.5

FUNK2



FUNK2 FUNCTION X1, D3
10,60/15,80/20,20

FUNK1



FUNK1 FUNCTION RN5,C4
0.8/0.40, 11/0.90, 12/1., 14

Slika 7.7. Primjeri definiranja GPSS funkcija

7.3.4. Atributi

Numerički atributi su vrijednosti povezane za model i njegove entitete. Te se vrijednosti izračunavaju pri referenciranju atributa.

Sistemski atributi

C1: sadašnje vrijeme simulacijskog sata.

Atributi blokova

Nj: ukupan broj transakcija ušlih u blok j.

Wj: trenutni broj transakcija u bloku j.

Atributi transakcija

Pj: parametar j transakcije.

M1: vrijeme 'života' transakcije od generiranja do sada.

PR: prioritet transakcije.

Atributi stalnih entiteta

Fj: status facilityja j

(ili F\$ime, a analogni način referenciranja imenom entiteta vrijedi za gotovo sve atribute koji slijede).

Sj: trenutni broj jedinica (ne transakcija) u storageu j.

Rj: preostali slobodni kapacitet storagea j, odnosno broj jedinica koji u njega još stane.

Qj: trenutni broj jedinica (ne transakcija) u repu j.

Save lokacije

Xj: trenutni sadržaj save lokacije j.

Računski entiteti

Vj: trenutna vrijednost varijable j.

FNj: trenutna vrijednost funkcije j.

RNj: vrijednost slučajnih brojeva j

(vrijednost slučajnog broja generira se tek pri njegovu referenciranju!).

Logički atributi

za facility:

Uj: korištenje facilityja (1-korišten, 0-nije korišten)

NUj: korištenje facilityja (1-nije korišten, 0-korišten)

za storage:

SFj: storage je pun (1-pun, 0-nije pun)

SNFj: storage nije pun (1-nije pun, 0-pun)

SEj: storage je prazan (1-prazan, 0-nije prazan)

SNEj: storage nije prazan (1-nije prazan, 0-prazan)

7.3.5. Indirektno adresiranje

Direktno *specificiranje entiteta* je ili prema broju entiteta ili prema imenu (pri tome, GPSS interno vodi uvijek entitete prema njihovu broju, čak i onda kada se entitet u

programu naziva imenom). Parametri transakcija i slučajni brojevi mogu se specificirati samo brojevima. Entiteti se specificiraju u operandima blokova namijenjenih specifikaciji određenih tipova entiteta, pa nije potrebno navesti tip entiteta koji se specificiraju. Za tu svrhu služe npr. prvi operandi naredbi SEIZE, RELEASE, ENTER, LEAVE, QUEUE, DEPART, SAVEVALUE i ASSIGN.

Primjeri direktnog specificiranja entiteta prema broju:

ENTER	4,
RELEASE	12,
QUEUE	5,
ASSIGN	3,500

i prema imenu:

LEAVE	PARK,
QUEUE	REP,
RELEASE	ALAT,
SAVEVALUE	AKUMUL, 20

Referenciranje atributa izvodi se referenciranjem numeričkih atributa čija se vrijednost uzima ili proračunava (u slučaju funkcija i varijabli) u času referenciranja. Referenciranje atributa može se koristiti u svim operandima u kojima se traži numerička vrijednost, bilo da se traži broj entiteta, bilo da se numerička vrijednost koristi za druge svrhe. Budući da se na tim mjestima mogu referencirati numerički atributi različitih entiteta, mora se navesti tip entiteta ispred broja ili imena entiteta na koji se atribut odnosi.

Primjeri za *indirektno referenciranje broja entiteta* preko vrijednosti atributa su:

QUEUE	P3	(broj traženog repa je vrijednost parametra 3 transakcije koja je ušla u blok),
ENTER	FN1	(broj traženog storagea je vrijednost funkcije 1),
SEIZE	V\$OPREMA	(broj traženog facilityja je izračunana vrijednost varijable OPREMA).

Indirektno referenciranje entiteta omogućuje da se jednim blokom opiše referenciranje većeg broja različitih entiteta. Time se omogućuje smanjenje broja blokova u GPSS programu, tj. više procesa istog tipa mogu se opisati istom grupom GPSS blokova. Kada transakcija prolazi tim blokovima, trenutne vrijednosti numeričkih atributa će davati brojeve entiteta koji se u blokovima referenciraju. Tako će u seriji blokova:

SEIZE P1
ADVANCE V1
RELEASE P1

traženi facility biti facility broj 1,2,3..., ovisno o vrijednosti parametra 1 transakcije, a vrijeme zadržavanja u bloku ADVANCE bit će izračunana vrijednost varijable V1.

Primjeri za referenciranje drugih numeričkih podataka u operandima pomoću numeričkih atributa su:

GENERATE	10,FN5,
ADVANCE	P2,

```

ENTER      3,V1,
ADVANCE   V$TIME,
TEST       G Q$REP2,15,VAN,
QUEUE     3,X$TRECI.

```

Ovdje se ne referenciraju brojevi entiteta već druge numeričke vrijednosti potrebne u radu modela.

Još je jedna snažnija mogućnost indirektnog adresiranja u kojoj se koristi ekvivalentnost referenciranja parametara transakcija na ova dva načina:

Pj,
*j.

Drugi od ova dva načina služi za referenciranje entiteta čiji je broj sadržan u parametru j. Primjeri korištenja:

- | | |
|-------------|--|
| ENTER P*2 | (Zahtjev za zauzimanje onog storagea čiji broj se dobije iz vrijednosti parametra broj 2; dakle ako je $P2=5$ a $P5=3$, broj storagea je u parametru $P*2=P5$, dakle to je storage 3.) |
| ADVANCE Q*4 | (Zadržavanje transakcije toliko vremenskih jedinica koliki je sadržaj repa čiji je broj vrijednost parametra 4; ako je $P4=1$, to je sadržaj repa broj 1.) |

7.3.6. Tip izlaznih rezultata

Izvođenje GPSS programa daje niz izlaznih rezultata korisnih kako za testiranje ispravnosti programa tako i za analizu rada sistema koji se simulira. Neki od osnovnih tipova podataka su:

1. Vrijeme izvođenja simulacije

Apsolutni sat mjeri vrijeme od početka simulacije ili poslije izvođenja posljednje CLEAR naredbe za pražnjenje modela.

Relativni sat modela mjeri vrijeme poslije izvođenja posljednje RESET naredbe za brisanje statistike modela.

2. Podaci o blokovima

Za svaki blok dobiva se ukupan broj transakcija koji je prošao kroz blok i sadašnji broj transakcija u tom bloku u času završetka simulacije.

3. Facility

Za svaki facility dobiva se prosječno korištenje, broj ušlih transakcija, prosječno vrijeme po transakciji u bloku i broj transakcije koja sada zauzima facility.

4. Storage

Za svaki storage dobiva se kapacitet, prosječni sadržaj, prosječno korištenje, broj ušlih jedinica, prosječno vrijeme po transakciji, sadašnji broj jedinica i maksimalan broj jedinica u storageu tijekom cijelog izvođenja simulacije.

5. Queue

Za svaki rep dobivaju se njegov maksimalni i prosječni sadržaj tijekom simulacije, ukupan broj jedinica koje su u njega ušle, broj i postotak jedinica koje nisu čekale u repu, prosječno vrijeme u repu za sve transakcije, kao i za transakcije koje nisu čekale.

7.4. MODELIRANJE I SIMULACIJA NEKOLIKO JEDNOSTAVNIH PROBLEMA JEZIKOM GPSS

Da bismo demonstrirali način konstrukcije GPSS blok-dijagrama i programa te način njihova rada, prikazali smo nekoliko jednostavnih problema čija rastuća složenost zahtijeva dodavanje sve moćnijih GPSS blokova. Svi ti problemi izviru iz jednog osnovnog problema kojem su postepeno dodavane sve realističnije karakteristike. Prilikom rješavanja problema ujedno je predložen i sistematski pristup modeliranju problema (u potpunosti proveden kod problema 1).

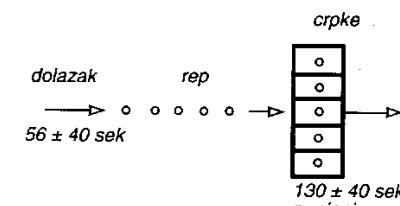
(1) BENZINSKA STANICA 1

Problem

Na benzinsku stanicu dolaze vozila svakih 60 ± 40 sekundi i čekaju u zajedničkom repu na slobodnu crpu. Na stanici je 5 crpki i uz svaku jedan radnik. Vozilo se puni 100 ± 30 sekundi i nakon toga odlazi. Treba simulirati radni dan od 8 sati rada stanice.

Prikaz sistema

Skica strukture sistema dana je na slici 7.8a.



(a) Prikaz sistema

Entiteti

Vozila – transakcije

Crpke – storage PUMPA (kapacitet 5)

Rep – rep čekanja REP pred crpkama

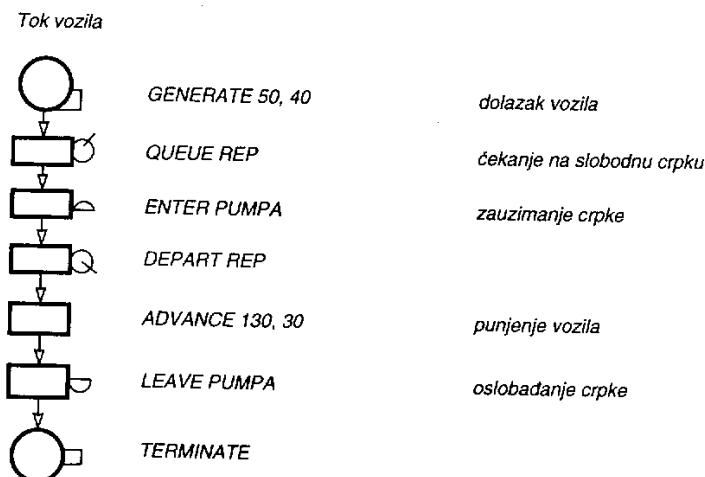
Vremenska jedinica

Sekunda

GPSS blok dijagram

GPSS blok dijagram problema prikazan je na slici 7.8b. Sastoji se od dva nezavisna dijela, jedan koji opisuje tok vozila i drugi koji dovodi do završetka simulacije nakon željenog vremena.

Ovdje naredbe za ulazak i izlazak transakcije iz repa dolaze ispred i iza naredbe ENTER koja može zaustaviti gibanje transakcije ako su sve crpke u času dolaska transakcije zauzete.



Slika 7.8. Benzinska stanica I

Rad programa počinje generiranjem transakcija koje predstavljaju vozila u prvoj naredbi GENERATE. One zauzimaju jednu crpku ako ima slobodnih crpki ili čekaju u repu pred crpkama ako nijedna od njih nije slobodna. Kada je završeno posluživanje na crpki prikazano blokom ADVANCE, transakcija izlazi iz modela i u taj isti čas simulacijskog vremena sljedeća transakcija može iz bloka QUEUE ući u blok ENTER. Tako u modelu može u istom času biti veći broj transakcija u raznim ili istim blokovima. U času 28800, GENERATE blok kontrole završetka simulacije generira jednu transakciju koja odmah u bloku TERMINATE smanji brojač završetka u naredbi START na nulu, što dovodi do završetka rada simulacije.

GPSS program

GPSS program koji odgovara ovom blok dijagramu prikazan je na slici 7.9.

Prevodenje i izvođenje GPSS programa

Na slici 7.10. vidi se listing prevodenja i izvođenja ovog GPSS programa. Ni u prevodenju ni u izvođenju nije bilo grešaka.

```

1: * ----- benzinska stanica 1
2: * --- svaka pumpa ima svog radnika
3: * --- vremenska jedinica je sekunda
4: * ----- SIMULATE
5: * PUMPA   STORAGE 5      5 benzinskih crpki
6: * ----- tok vozila
7: * GENERATE 50, 40      dolazak vozila
8: * QUEUE REP           cekanje na slobodnu pumpu
9: * ENTER PUMPA         zauzimanje pumpu
10: * DEPART REP          punjenje vozila gorivom
11: * ADVANCE 130, 30     oslobodenje pumpu
12: * LEAVE PUMPA         TERMINATE
13: * ----- kontrola zavrsetka simulacije
14: * GENERATE 28800      8 sati simulacije
15: * TERMINATE 1
16: * ----- START 1
17: * ----- END
18: * ----- Slika 7.9. GPSS program za benzinsku stanicu I
19: * ----- 1: * ----- benzinska stanica 1
20: * 2: * --- svaka pumpa ima svog radnika
21: * 3: * --- vremenska jedinica je sekunda
22: * 4: * ----- SIMULATE
23: * 5: * PUMPA   STORAGE 5      5 benzinskih pumpi
24: * 6: * ----- tok vozila
25: * 7: * 1: GENERATE 50, 40      dolazak vozila
26: * 8: * 2: QUEUE REP           cekanje na slobodnu pumpu
27: * 9: * 3: ENTER PUMPA         zauzimanje pumpe
28: * 10: * 4: DEPART REP          punjenje vozila gorivom
29: * 11: * 5: ADVANCE 130, 30     oslobadanje pumpe
30: * 12: * 6: LEAVE PUMPA         TERMINATE
31: * 13: * ----- kontrola vozila
32: * 14: * GENERATE 28800      8 sati simulacije
33: * 15: * TERMINATE 1
34: * 16: * ----- START 1
35: * 17: * ----- END

```

Storage symbols and corresponding numbers
1: PUMPA

Queue symbols and corresponding numbers
1: REP

No errors detected

Relative clock 28800 Absolute clock 28800

Block counts

Block	Current	Total
1	0	580
2	0	580
3	0	580
4	0	580
5	1	580
6	0	579
7	0	579
8	0	1
9	0	1

Storage	Capacity	Average Contents	Average Utilisation	Entries	Average Time/tran	Current Contents	Maximum Contents
1	5	2.635	0.527	580	130.669	1	5
Queue	Maximum contents	Average contents	Total entries	Zero entries	Percent zeros	Average time/trans	Current contents
1	3	0.006	580	573	98.793	0.300	0

Slika 7.10. Benzinska stanica 1 – prevođenje i izvođenje GPSS programa

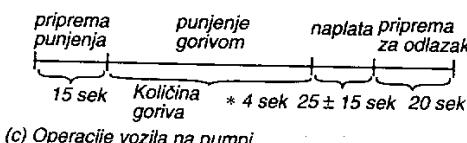
(2) BENZINSKA STANICA 2

Problem

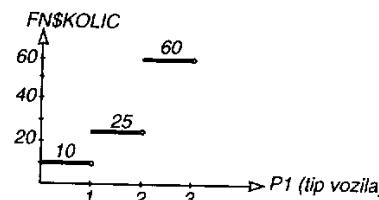
Na istu benzinsku stanicu dolaze 3 različita tipa vozila, i to svakih 200 ± 150 , 120 ± 60 odnosno 180 ± 120 sekundi (prema tipu vozila). U vozila treba natočiti prosječno 10, 25, odnosno 60 litara goriva, ovisno o tipu vozila. Vrijeme punjenja jedne litre goriva je 4 sekunde. Prije punjenja vozilo treba oko 15 sekundi za pripremu za punjenje, naplata traje 25 ± 15 sekundi, a oko 20 sekundi je potrebno da se vozilo pripremi za napuštanje crpke. I ova benzinska stanica ima 5 crpki, a svaka crpka svojega radnika. Treba simulirati 8 sati rada benzinske stanice.

Prikaz sistema

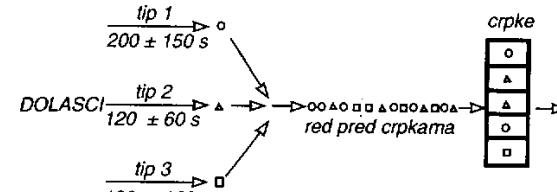
Na slici 7.11a prikaz je sistema, a na slici 7.11c slijed operacija vozila na crpkama.



(c) Operacije vozila na pumpi



(b) Količina goriva za punjenje – prema tipu vozila



(a) Prikaz sistema

Slika 7.11. Benzinska stanica 2 – Prikaz sistema i njegovih karakteristika

Entiteti

Tip vozila – parametar 1 transakcije vozila

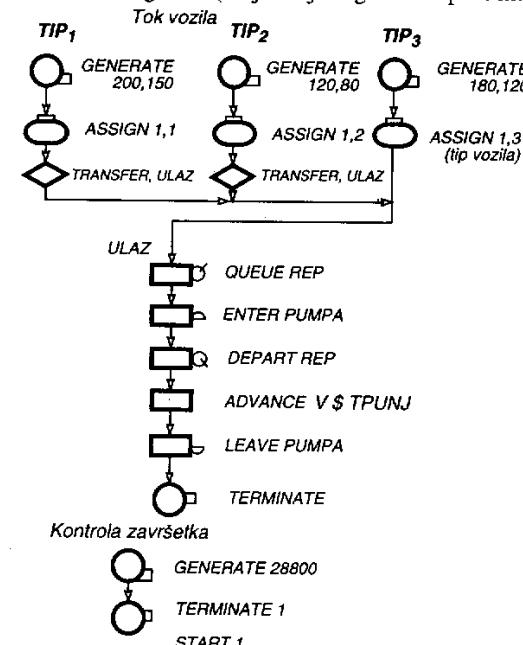
Količina goriva – funkcija KOLIC, čiji je argument tip vozila, prikazana je na slici 7.11b

Vrijeme naplate – funkcija TNAPL. To je funkcija slučajne varijable, prikazana na slici 7.13.

Vrijeme punjenja – varijabla TPUNJ=15+4*FN\$KOLIC+FN\$TNAPL+20

GPSS blok dijagram

GPSS blok-dijagram prikazan je na slici 7.12. Transakcije vozila generiraju se sa tri odvojena GENERATE bloka s različitim ritmom generiranja. Zatim se iz svakog od njih postavlja tip vozila u parametar 1 transakcije i nakon toga se transakcije slijevaju u zajednički niz blokova. U tom nizu blokova različita trajanja vremena posluživanja različitih tipova vozila rješavaju se izračunavanjem varijable TPUNJ, čiji je jedan pribrojnik vrijeme punjenja koje se dobiva korištenjem funkcije KOLIC što daje prosječan zahtjev za količinom goriva (a njezin je argument tip vozila P1).



Slika 7.12. Benzinska stanica 2 — GPSS blok dijagram

Prevođenje GPSS programa

Na slici 7.13. dan je listing prevođenja GPSS programa.

```

* ----- *
* benzinska stanica 2
* -----
*   - 3 tipa vozila (prema potražnji za gorivom)
*   - svaka pumpa ima svog radnika
*   - vremenska jedinica je sekunda
* ----- *

          SIMULATE

* PUMPA    STORAGE 5
KOLIC    FUNCTION P1, D3      5 benzinskih pumpi
         1,10/22,25/3,60          potražnja vozila za gorivom
TNAPL    FUNCTION RN2, C2      vrijeme naplate
         0,0,10/1,0,41
TPUNJ    VARIABLE 15+4*FN$KOLIC+FN$TNAPL+20  vrijeme vozila na pumpi
*
*   tok vozila
*
*   GENERATE 200,150      dolazak vozila tipa 1
ASSIGN 1,1           tip vozila 1
TRANSFER ,ULAZ
*
*   GENERATE 120, 80       dolazak vozila tipa 2
ASSIGN 1,2           tip vozila 2
TRANSFER ,ULAZ
*
*   GENERATE 180, 120      dolazak vozila tipa 3
ASSIGN 1,3           tip vozila 3
*
ULAZ     QUEUE REP          čekanje na slobodnu pumpu
ENTER PUMPA        zauzimanje pumpe
DEPART REP
ADVANCE V$TPUNJ      punjenje vozila gorivom
LEAVE PUMPA        oslobođanje pumpe
TERMINATE
*
*   kontrola završetka simulacije
*
*   GENERATE 28800        8 sati simulacije
TERMINATE 1
*
START 1
*
END

```

Slika 7.13. Benzinska stanica 2 — GPSS program

(3) BENZINSKA STANICA 3

Problem

Problem je isti kao prethodni, osim što svaka crpka više nema svojega radnika već su prisutna 2 radnika koji poslužuju svih 5 crpki. Radnik mora natočiti gorivo i naplatiti ga, nakon čega može posluživati vozilo na nekoj drugoj crpki. Vozila koja dolaze odustaju od čekanja i odlaze ako je rep pred benzinskom stanicom duži od 10 vozila.

Rješenje ovog problema nećemo ovdje prikazivati zbog ograničenja prostora, ali je ono u potpunosti sadržano u proširenju problema što slijedi.

(4) BENZINSKA STANICA 4

Problem

Problem je isti kao i prethodni, osim što treba ispitati da li se više isplati imati 2, 3, 4 ili 5 radnika na benzinskoj stanici koja ima 5 benzinskih crpki. Pri tome se zna da je dobit po prodaji 1 litre goriva 0.65 dinara, a da je dnevni trošak po jednom radniku 500 dinara. Ocjena o najpovoljnijem broju radnika donosi se na temelju postizanja najviše prosječne dnevne dobiti benzinske stanice.

Entiteti

- Save vrijednost XRADN — ukupan broj radnika na crpkama (inicijalizira se na početku simulacije).
- Save vrijednost UKKOL — povećava se prilikom svakog punjenja za vrijednost količine natočenoga goriva FN\$KOLIC.
- Save vrijednost DOBIT — izračunava se na kraju simulacije u kontrolnom procesu kao V\$VDOB.
- Varijabla VDOB — dnevna dobit $0.65 \times \$UKKOL - 500 \times \$XRADN$.
- Radnici — storage čiji je kapacitet 2, 3, 4 ili 5. (u uzastopnim izvođenjima simulacije)
- Rep RADN — čekanje na dostupnost storagea RADN.

Prevođenje GPSS programa

Na slici 7.14. prikazan je listing prevođenja GPSS programa ovog problema. Vidimo da je iz posluživanja vozila na crpki izdvojen dio u kojem sudjeluje radnik, te u tom dijelu treba biti dustupan storage RADN.

```

* ----- *
* benzinska stanica 4
* -----
*   - 3 tipa vozila (po potražnji za gorivom)
*   - ispitivanje varijanti: 2, 3, 4, ili 5 radnika (za svih 5 pumpi)
*   - traženje varijante s maksimalnom dobiti
*   - vozila na čekanju ako je rep duži od 10 vozila
*   - vremenska jedinica je sekunda
* ----- *

```

```

SIMULATE
*
INITIAL X$XRADN, 2
INITIAL X$UKKOL, 0
INITIAL X$DOBIT, 0
2 radnika
uk. količina natočenoga goriva
uk. dnevna dobit

* CRPKA      STORAGE 5           5 benzinskih crpki
RADN      STORAGE 2           2 radnika
* KOLIC      FUNCTION P1, D3    potražnja vozila za gorivom
1,10/2,25/3,60
TNAPL      FUNCTION RN2, C2    vrijeme naplate
0,0,10/1,0,41
* TPUNJ      VARIABLE 4*FN$KOLIC+FN$TNAPL  vrijeme punjenja i naplate
VDOB      VARIABLE (65/100)*X$UKKOL—500*X$XRADN  dnevna dobit
*          tok vozila
*          GENERATE 200, 150      dolazak vozila tipa 1
ASSIGN 1,1      tip vozila 1
TRANSFER ,ULAZ
*          GENERATE 120,80       dolazak vozila tipa 2
ASSIGN 1,2      tip vozila 2
TRANSFER ,ULAZ
*          GENERATE 180,120      dolazak vozila tipa 3
ASSIGN 1,3      tip vozila 3
*          ULAZ      TEST LE Q$REP1,10, VAN
QUEUE REPI
ENTER PUMPA
DEPART REPI
ADVANCE 15
QUEUE REP2
ENTER RADN
DEPART REP2
ADVANCE V$TPUNJ
SAVEVALUE UKKOL+,FN$KOLIC
LEAVE RADN
ADVANCE 20
LEAVE PUMPA
TERMINATE
*          VAN      TERMINATE
*          kontrola završetka simulacije
*          GENERATE 28800      8 sati simulacije
SAVEVALUE DOBIT, V$VDOB
TERMINATE !
*          START !
*          END

```

Slika 7.14. Benzinska stanica 4 — GPSS program

Izvođenje simulacije

Izvedena je simulacija za sve zadane vrijednosti broja radnika na crpkama. Na slici 7.15. je pregled osnovnih rezultata izvođenja simulacije.

Vidimo da benzinska crpka ostvaruje najveću dobit kada na njoj rade 3 radnika. Ako na crpkama rade 2 radnika tada je ukupna količina natočenog goriva mnogo manja, jer gotovo jedna trećina vozila odustaje od čekanja pred velikim repom ispred benzinske

crpke. Ukupna je dobit tada manja iako se plaćaju samo 2 radnika. Sa 3 radnika posluže se već sva vozila koja su došla na benzinsku stanicu, tako da je daljnje dodavanje radnika samo veći trošak a ne i povećanje dobiti. Iskorištenje crpki i radnika je već vrlo visoko, a čekanje na crpke i radnike je prihvatljivo (oko 1 minute). Dalnjim povećanjem broja radnika smanjuje se iskorištenost radnika te dužine repova i vremena čekanja na crpke i radnike, ali se ne povećava ukupna količina natočenoga goriva.

Ovaj je primjer demonstrirao, osim načina konstrukcije programa, i način te mogućnosti analize problema simulacijskim modeliranjem uz korištenje simulacijskog jezika GPSS.

	BROJ RADNIKA NA PUMPAMA			
	2	3	4	5
Ukupna količina natočenog goriva X\$UKKOL	11 780	16 985	16 985	16 985
Ukupna dnevna dobit X\$ DOBIT	6 656	9 540	9 040	8 540
Broj posluženih vozila	375	539	539	539
Broj vozila koja su odustala od čekanja	153	0	0	0
Prosj. iskorištenje				
— crpki	0.99	0.87	0.71	0.70
— radnika	0.99	0.95	0.71	0.57
Prosječno čekanje (sek)				
— na crpke	710	72	4	4
— na radnike	188	45	3	0

Slika 7.15. Benzinska stanica 4 — analiza alternativnih konfiguracija

8. STVARANJE POVJERENJA U SIMULACIJSKE MODELE

Da bi se stvoren simulijski model mogao s pouzdanjem koristiti, potrebno je uložiti velik napor da se ispita njegova ispravnost. Budući da je vrlo često riječ o modelima složenih dinamičkih i stohastičkih sistema, postupak stvaranja povjerenja nije nimalo jednostavan. Iako nema recepata ni gotovih procedura za proces stvaranja povjerenja, mogu se koristiti različite metode i postupci koji su za to razvijeni. Onaj tko razvija model treba utvrditi koje su od tih metoda primjenljive i korisne za provjeru određenog simulijskog modela. U ovom poglavljiju opisali smo osnovne pretpostavke procesa stvaranja povjerenja u simulijske modele, a same postupke podijelili smo na dvije osnovne grupe: one za verifikaciju računarskog modela i one za vrednovanje konceptualnih modala.

8.1. PROVJERA ISPRAVNOSTI SIMULACIJSKIH MODELA

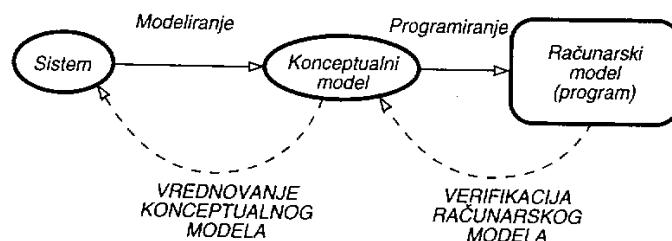
Osnovni razlog izgradnje simulijskih modala jest omogućavanje ispitivanja rada nekog sistema i donošenje odgovarajućih odluka u vezi s tim sistemom. Pri tome se, naravno, pretpostavlja da *model adekvatno zamjenjuje sistem*, tj. da eksperimenti provedeni modelom sistema mogu zamjeniti eksperimentiranje sa samim sistemom.

Ta se pretpostavka mora *opravdati*, kako bi oni koji donose odluke mogli imati povjerenje u korištenje modela kao zamjene za sam sistem. Time se ujedno smanjuje vjerojatnost donošenja pogrešne odluke, koja može biti povezana za značajne gubitke. Traži se, dakle, da ponašanje simulijskog modela bude veoma slično ponašanju realnog ili budućeg predviđenog sistema.

Više je različitih pristupa provjeri ispravnosti simulijskih modala. Dva su osnovna koraka u stvaranju povjerenja u simulijski model (slika 8.1):

- *vrednovanje konceptualnog modela*, koje je ispitivanje slaganja ponašanja konceptualnog modela i realnog sistema
- *verifikacija računarskog modela*, koja je ispitivanje slaganja ponašanja računarskog modela i konceptualnog modela.

Vrednovanje i verifikacija simulijskih modala imaju cilj da na najmanju mjeru svedu pojavu raznih vrsta mogućih grešaka modela (u logici konceptualnog i računarskog modela, matematičkim relacijama, programiranju, mjerenu i obradbi ulaznih podataka, načinu korištenja modela, obradbi i interpretaciji rezultata i sl.). Za to se koriste različite statističke i informatičke tehnike i sposobnost kritičke subjektivne procjene eksperata.



Slika 8.1. Vrednovanje i verifikacija simulacijskog modela

Stvaranje povjerenja u simulacijski model znači razvijati *prihvatljivi nivo pouzdanja* da su zaključci koji se donose na temelju ponašanja modela ispravni i primjenljivi na stvarni sistem. Vrednovanje treba gledati kao *stupanj zadovoljavanja*, a ne kao određivanje da li model potpuno zadovoljava ili uopće ne zadovoljava (Shannon, 1975). Ovaj zaključak temelji se na argumentima:

- model je nužno aproksimacija sistema
- elementi modela dobiveni mjerjenjem na stvarnom sistemu sadrže u sebi greške mjerjenja koje uzrokuju inherentnu netočnost u rezultatima modela
- točnost rezultata statističke analize izlaza modela ovisi o veličini promatranoj uzorku, što znači da i povećanje te točnosti zahtjeva povišenje cijene dobivanja rezultata.

Porastom stupnja valjanosti modela raste i cijena i vrijeme razvoja modela, pa to svakako utječe na mogućnost stvaranja modela s veoma visokim stupnjem valjanosti.

8.2. OSNOVNE PREPOSTAVKE PROCESA STVARANJA POVJERENJA U SIMULACIJSKI MODEL

Proces stvaranja povjerenja u simulacijski model ne može se opisati kao jednoznačna i nepromjenljiva procedura s točno definiranim koracima. Ipak, moguće je dati neke opće odrednice kojih se potrebno podržavati kako bi se sa što većom izvjesnošću došlo do modela koji je adekvatna slika stvarnog sistema. Neke od osnovnih karakteristika procesa stvaranja povjerenja u simulacijske modele te zahtjeva koji se postavljuju na ovaj proces jesu:

(1) Stvaranje povjerenja u simulacijski model ne smije se shvatiti kao korak koji se izvodi tek kada je proces modeliranja završen: stvaranje povjerenja je proces koji je bitno vezan za izgradnju simulacijskog modela, pa se stoga i mora ostvarivati *tijekom cijelog procesa izgradnje modela*. Njegovo odgadjanje za sam kraj procesa modeliranja može ugroziti kvalitetu simulacijskog modela i onemogućiti izvođenje eventualnih nužnih izmjena u modelu zato što je raspoloživo vrijeme za izgradnju modela proteklo.

(2) Koraci u procesu stvaranja povjerenja u simulacijski model nisu uvijek jednaki, već su vezani za one aspekte ponašanja sistema koji su *relevantni za određenu odluku* zbog koje se simulacijsko modeliranje i provodi. Pri tome većina metoda koje se u procesu stvaranja povjerenja u simulacijski model koriste nisu u obliku precizno

definiranog postupka, već su povezani za *način gledanja i razumijevanja procesa modeliranja* analitičara i donosioca odluka.

(3) U procesu stvaranja povjerenja u simulacijski model moraju, kao i u ostalim dijelovima procesa modeliranja, *kontinuirano suradivati i analitičar* (ekspert za simulacijsko modeliranje) i *donosilac odluke* (ekspert za sam sistem). U onim fazama procesa u kojima je potrebna primjena statističkih ili drugih metoda potrebno je, naravno, sudjelovanje i odgovarajućih eksperata za te discipline. Samo tako može se osigurati da proces modeliranja bude voden korektno i efikasno, te da se izbjegne da razvoj modela ide u smjeru eventualno interesantnom za analitičara ali ne nužno i razumnom sa stajališta donosioca odluke. Praksa je dovoljno jasno pokazala da preuzimanje zadatka na početku simulacijskog procesa i predavanje gotovih rezultata na kraju procesa bez dovoljno suradnje tokom procesa gotovo bez iznimke dovodi do promašenih projekata, a često i do gubljenja povjerenja u simulacijsko modeliranje.

(4) Budući da je cilj simulacijskog modeliranja da pomogne donošenju odluka, potrebno je osigurati da i *simulacijski model i njegove rezultate* (dobivene eksperimentiranjem) *prihvate donosioci odluke*. To znači da ne samo da procesi vrednovanja i verifikacije modela trebaju biti provedeni, već to moraju znati i donosioci odluke. Proses prihvaćanja modela značajno se olakšava ako su donosioci odluke bili angažirani u procesu razvoja i vrednovanja modela. Tome može pomoći i adekvatna prezentacija modela kako tijekom njegova razvoja, tako i u završnom izvještaju projekta; korištenje grafike u samom modelu (animacija simulacije) i njegovoj prezentaciji može pri tome biti veoma korisna.

8.3. VERIFIKACIJA RAČUNARSKOG MODELA

Verifikacija računarskog modela odnosi se na slaganje računarskog i konceptualnog modela. Postupak verifikacije nije izravno povezan za karakteristike stvarnog sistema, već za ispitivanje da li simulacijski program zadovoljavajuće prikazuje odgovarajući konceptualni model.

Verifikacija računarskog modela obuhvaća ove aspekte (Law i Kelton, 1982):

(1) Već pri razvoju *simulacijskog programa* preporučuje se korištenje metoda softverskog inženjerstva u onoj mjeri u kojoj su one primjenljive za upotrebljeni programski jezik simulacije. Time se značajno smanjuje vjerojatnost pojava grešaka u stvorenom simulacijskom programu.

Preporučuje se npr. primjena ovih postupaka:

- Prilaz sa-vrha-naniže ('top-down'), tj. postupni razvoj glavnog programa i osnovnih procedura u obliku kostura, te njihovo testiranje. Tada se kosturu dodaju detaljnije procedure nižeg nivoa koje se zatim testiraju. Ovaj se postupak nastavlja sve dok se kompletan programski zadatak ne završi.

- Strukturno programiranje, koje nastoji strukturu programa što više približiti strukturi problema. Može se koristiti zajedno s prilazom 'sa-vrha-naniže', a pri tome je posebno olakšanje ako se koristi programski jezik koji odgovara problemu što se rješava. Tako je za razvoj pouzdanih simulacijskih programa veoma povoljno ako se koriste specijalni simulacijski jezici, gotove biblioteke simulacijskih rutina i posebno automatski generatori simulacijskih programa.

- Modularno programiranje, koje potiče razvoj i testiranje programskih cjelina (modula) razumne veličine u konstrukciji složenih programskih proizvoda. Svaki modul

prikazuje jednu zatvorenu funkciju sistema, pa se stoga moduli mogu lakše i pouzdanije pojedinačno verificirati prije verifikacije cijelog simulacijskog programa.

Preporučuje se ujedno korištenje adekvatnog jezika za razvoj simulacijskog programa. Specijalni i opći simulacijski jezici uglavnom rezultiraju u kompaktnijem i čitljivijem programskom kodu. Oni također, u pravilu, rezultiraju (u usporedbi s modelima razvijenim u općem programskom jeziku) s manje grešaka u prevodenju i izvođenju koje se lakše otkrivaju i ispravljaju, zbog postojanja adekvatnijih alata za otkrivanje grešaka specifičnih za simulacijsko modeliranje. Postojanje adekvatne programske dokumentacije, odnosno pisanje programa koji su sami sebi dokumentacija, također znatno olakšava otkrivanje i ispravljanje grešaka u programu.

(2) Testiranje *ispravnosti i točnosti* simulacijskog programa. Testiraju se statička svojstva, tj. struktura programa, i dinamička svojstva programa dobivena izvođenjem programa u različitim uvjetima. Za to se preporučuje:

- Usporedba rezultata računarskog modela (programa) s rezultatima konceptualnog modela izračunanim provođenjem ručne simulacije uz neka pojednostavljenja. Mogu se npr. zamijeniti neke slučajne varijable modela konstantnim (npr. srednjim) vrijednostima kako bi se lakše mogla provesti manuelna simulacija rada konceptualnog modela.

- Testiranje detalja izvođenja programa ('variable trace'): pri svakoj promjeni stanja sistema mogu se štampati vrijednosti varijabli stanja sistema (lista događaja, dužina repova itd.). Rezultati izvođenja usporeduju se s rezultatima ručne simulacije modela. Nekada je za otkrivanje greške u programu dovoljan samo ispis slijeda izvođenja naredbi programa ('flow trace') ili brojač aktivacije pojedinih blokova programa (ukupan broj prolaza privremenih entiteta kroz blok i preostali broj tih entiteta u bloku na kraju izvođenja).

- Grafičko predstavljanje izlaza programa (animacija simulacije i prikaz statističkih rezultata) može znatno olakšati testiranje izvođenja programa, budući da se dinamika i logika odvijanja simulacije mogu vizualno veoma efikasno testirati.

- Strukturni prolaz ('structured walkthrough') uključuje zajednički prolaz svih članova simulacijske projektne ekipe kroz program. Svaka se pojedina naredba prihvata tek kada su svi članovi ekipe zadovoljni njezinom ispravnosću.

- Ako se pojavi indicija da postoji neka greška u programu, korisno je imati prihvaćenu proceduru za traženje uzroka greške i njezinu korekciju. Tu trebaju biti uključeni npr. opis točnih okolnosti u kojima je greška nastala, što se točno desilo i gdje se u programu dogodila greška. Nakon identifikacije pretpostavljenog uzroka greške i njegova uklanjanja potrebno je testiranjem potvrditi ispravnost korekcije te izvođenjem prethodnih testova provjeriti da uklanjanjem jedne greške nije uvedena druga.

8.4. VREDNOVANJE KONCEPTUALNOG MODELA

Vrednovanje konceptualnog modela odnosi se na ispitivanje slaganja ponašanja konceptualnog modela i realnog sistema. Kao i kod verifikacije računarskog modela neki pristupi su orijentirani na ispitivanje vanjskog ponašanja modela uz tretiranje i modela i sistema kao 'crne kutije', dok se drugi više bave detaljnijim opisom rada sistema i modela. Neki pristupi uključuju uglavnom formalne statističke procedure usporedbе, a drugi se temelje na ocjenama eksperata za sistem.

Vrednovanje simulacijskog modela sadrži tri osnovna tipa ispitivanja modela. *Replikativna valjanost* modela označava sposobnost modela da generira podatke koji

su u skladu s već prije prikupljenim podacima sistema koji radi u istim uvjetima. *Strukturalna valjanost* označava da model ne samo što ispravno reproducira ponašanje sistema već i točno reflektira način na koji stvarni sistem radi. *Prediktivna valjanost* označava mogućnost modela da ispravno predviđa buduće ponašanje sistema.

8.4.1. Replikativno vrednovanje modela

Replikativno vrednovanje modela obuhvaća ispitivanje slaganja ponašanja modela s ponašanjem stvarnog sistema u istim uvjetima; ispitivanje je moguće ako sistem koji modeliramo postoji i na njemu su moguća mjerena.

Za to se koriste različite tehnikе:

a) Korištenje statističkih testova

Usporedba ponašanja sistema i modela nije tako jednostavna kao što se to može činiti. Razlog je što su izlazi većine realnih sistema *nestacionarni* (distribucije vjerojatnosti mijenjaju se u vremenu) i *autokolerirani* (uzastopne vrijednosti varijabli u vremenu međusobno su korelirane). To znači da klasični statistički testovi, temeljeni na opažanjima s nezavisnom identičnom razdiobom, nisu direktno primjenljivi.

Također se smatra da, budući da je model samo aproksimacija sistema, testiranje hipoteza o jednakosti modela i sistema nema mnogo smisla već da je korisnije testirati da li su *razlike* između sistema i modela značajne.

Mogu se koristiti različiti pristupi:

- usporedba nekih statističkih pokazatelja modela i sistema (sredine uzorka, varijance i sl.) bez korištenja formalne statističke procedure

- korištenje intervala pouzdanosti (za što treba prilično velik broj podataka)

- analiza vremenskih serija izlaznih podataka modela i sistema

- ako su ulazni i izlazni podaci za sistem potpuni, moguće je napraviti točniju usporedbu ponašanja sistema i modela time što bi se u modelu koristile originalne empiričke razdiobe ulaznih varijabli (umjesto izvedenih teorijskih razdioba)

- prilikom korištenja historijskih ulaznih podataka proces vrednovanja modela može se pretvoriti u *kalibraciju modela*, i to tako da se nakon usporedbе dijela podataka modela i sistema promijene parametri modela tako da slaganje podataka poraste. S preostalim dijelom podataka model se zatim može sve bolje kalibrirati u više koraka.

b) Izvođenje 'Turing testa' s ekspertima

Ekspertima se pokažu izlazni rezultati sistema i modela uz pitanje mogu li ih razlikovati. Ako ih ne mogu razlikovati, to je indicija da model dobro opisuje sistem. Ako ih mogu razlikovati, tada njihovo objašnjenje može indicirati slabe točke modela.

c) Analiza osjetljivosti

Ispituje se jesu li izlazne varijable u sistemu i modelu podjednako osjetljive na promjenu parametara, i imaju li isti smjer promjene (npr. da li prosječna dužina repa pred mjestom posluživanja raste ako vrijeme posluživanja raste).

d) Ako stvarni sistem ne postoji

Ako stvarni sistem ne postoji, može se napraviti usporedba s već postojećim modelima (struktura, parametri) ili paralelno razvijati i drukčiju vrstu modela s kojom će se osnovni model usporediti (npr. pojednostavljeni matematički modeli analitičkog tipa).

8.4.2. Struktorno vrednovanje modela

Struktorno vrednovanje modela obuhvaća ispitivanje točnosti prepostavki na kojima se temelji model, apstrakcija učinjenih u modelu, točnosti logike rada modela i ulaznih podataka kojima se model koristi. Pri tome može biti obuhvaćeno:

- ispitivanje statičkih prepostavki modela (logika, ovisnosti, međudjelovanja) od eksperata
- ispitivanje točnosti apstrakcija učinjenih u modelu procjenama eksperata koji dobro poznaju sistem što se modelira, temeljenih obično na analizi konceptualnog modela
- ispitivanje dinamike rada (ponašanja) sistema numeričkim pokazateljima, tokovima privremenih entiteta kroz model te korištenjem animacije rada modela
- ispitivanje kvalitete ulaznih podataka, koja ovisi o načinu i kvaliteti mjerjenja i prikupljanja ulaznih podataka, veličini odabranog uzorka te o kvaliteti statističke analize ulaznih podataka (testiranje konzistentnosti podataka, ispitivanju podataka koje treba odbaciti, određivanju razdiobe vjerojatnosti i njezinih parametara, određivanju veza među podacima, ispitivanju homogenosti populacija ako je napravljena mješavina više različitih tipova podataka itd.)
- analiza osjetljivosti vrijednosti izlaznih varijabli na male promjene parametara simulacijskog modela, njihove razdiobe vjerojatnosti, način rada (npr. izbor disciplina repa: FIFO, LIFO itd.). Time se mogu dobiti granice do kojih vrijedi model kod varijacije faktora modela; ako je utjecaj nekog faktora na izlazne varijable veoma mali, on se može izbaciti iz modela ili se može zamijeniti prosječnom vrijednostju slučajne varijable; ako su izlazne varijable jako osjetljive na neki procijenjeni parametar, tada to može biti indicija da je potrebna bolja procjena vrijednosti tog parametra.

8.4.3. Prediktivno vrednovanje modela

Modél se mora adekvatno modificirati da bi mogao prikazati pretpostavljeni izgled sistema u budućnosti, s očekivanjem promjena vanjskih uvjeta, strukture i parametara rada sistema.

Prije izgradnje sistema moguće je testirati model nezavisnim skupom podataka o radu sistema, npr. nekim starijim podacima o radu sistema ili podacima koji namjerno nisu korišteni pri razvoju modela već su čuvani upravo za povećanje pouzdanja u prediktivna svojstva modela.

Pošto se sistem izgradi (ili promijeni) te se prikupe podaci o njegovu radu u tom novom obliku, moguće je usporediti rad modela i sistema. Time se stvara dodatno povjerenje u model i ujedno omogućuje njegovo dotjerivanje i prilagođavanje za donošenje što kvalitetnijih odluka u budućnosti.

9. OSNOVNI ELEMENTI VJEROJATNOSTI I STATISTIKE

Simulacijski modeli u pravilu sadrže slučajne varijable što opisuju elemente sistema koji mogu poprimiti više mogućih vrijednosti (npr. vrijeme posluživanja, tip vozila). Postojanje slučajnih varijabli zahtijeva odgovarajući tretman simulacijskih modela korištenjem teorije vjerojatnosti i statistike, i to posebno u ovim fazama simulacijskog procesa: (1) odabiranju razdoba vjerojatnosti za ulazne varijable, (2) generiranju slučajnih varijabli koje odgovaraju tim razdiobama, (3) vrednovanju simulacijskih modela, (4) planiranju simulacijskih eksperimenta i (5) analizi izlaznih rezultata simulacije.

Zbog važnosti slučajnih varijabli i njihova pravilnog tretmana u simulacijskom procesu, ukratko ćemo opisati osnovne ideje i način zaključivanja teorije vjerojatnosti i statistike potrebnih za razumijevanje i korištenje simulacijskog modeliranja. Pri opisu statističkih metoda u simulacijskom modeliranju u ovom poglavljiju i nekoliko sljedećih velikim dijelom smo slijedili pristup iz izvanredne knjige (Law i Kelton, 1982).

9.1. SLUČAJNE VARIJABLE I NJIHOVA SVOJSTVA

Više je razloga zbog kojih varijablu možemo smatrati slučajnom. Prije svega, varijabla se može ponašati tako da točan ishod njezina ponašanja nije poznat. Osim toga, ponekad varijabla zapravo nije slučajna ali potpuna informacija o njezinom ponašanju nije poznata, tako da se ona mora opisati kao slučajna varijabla. Treća je mogućnost da je potpuna informacija o varijabli dostupna, ali je toliko složena da je varijablu pogodnije opisati kao slučajnu.

Efekti fluktuacije slučajnih varijabli se, zbog postojanja međudjelovanja među varijablama tijekom razvoja sistema u vremenu, reflektiraju kroz cijeli sistem. Posljedica toga je da većina veličina od interesa u sistemu pokazuje slučajne fluktuacije u ponasanju.

Pojednostavljeni rečeno, *slučajna varijabla* je veličina čije su vrijednosti određene *ishodom slučajnog eksperimenta* (bacanje kocke, mjerjenje vremena posluživanja i sl.). Iako točan niz vrijednosti koje slučajna varijabla može poprimiti nije moguće znati unaprijed, područje vrijednosti u kojem ona može varirati i vjerojatnost s kojom ona može poprimiti vrijednosti mogu biti poznati. Slučajne varijable ćemo stoga opisivati *funkcijama* koje opisuju *vjerojatnost da varijable poprime različite vrijednosti*.

Dalje u tekstu *slučajne varijable* označavat ćeemo velikim slovima (npr. X, Y, Z) a *vrijednosti* koje one poprimaju malim slovima (npr. x, y, z).

9.1.1. Diskrete slučajne varijable

Reći ćemo da je slučajna varijabla X *diskretna* ako može poprimiti diskretan broj (konačan ili prebrojivo beskonačan) vrijednosti. Primjeri diskretnih slučajnih varijabli su vrijednost bacanja kocke, broj proizvoda koje će kupiti kupac u samoposluživanju, broj paketa na traci za prtljagu na aerodromu i sl.

Vjerojatnost $p(x_i)$ da diskretna slučajna varijabla X poprimi vrijednost x_i zove se *funkcija vjerojatnosti*. Ona određuje razdiobu ishoda slučajnog eksperimenta, tj. razdiobu vrijednosti varijable dobivenih slučajnim eksperimentom. Za nju vrijedi :

$$p(x_i) \geq 0, \quad \text{za svaki } x_i, i=1, 2, \dots$$

Ako slučajna varijabla može poprimiti n različitih vrijednosti x_i ($i=1, 2, \dots, n$), tada je

$$\sum_{i=1}^n p(x_i) = 1,$$

tj. suma vjerojatnosti svih mogućih ishoda slučajnih eksperimenata jednaka je 1.

Vjerojatnost $F(x)$ da slučajna varijabla X poprimi vrijednosti manje od x ili jednake x zove se *funkcija razdiobe*:

$$F(x) = P(X \leq x).$$

Funkcija razdiobe ima ova svojstva:

1. $0 < F(x) < 1, \quad -\infty < x < \infty$
2. Ako $x_1 < x_2$, tada je $F(x_1) < F(x_2)$
3. $\lim_{x \rightarrow \infty} F(x) = F(\infty) = 1, \quad \lim_{x \rightarrow -\infty} F(x) = F(-\infty) = 0$.

Na slici 9.1. prikazan je primjer diskrette slučajne varijable s njezinom funkcijom vjerojatnosti $f(x)$ i funkcijom razdiobe $F(x)$. U tom su primjeru vrijednosti funkcija ovakve:

x	1	2	3	4	5	6
$f(x)$	2/25	8/25	7/25	5/25	2/25	1/25
$F(x)$	2/25	10/25	17/25	22/25	24/25	25/25

Očekivana vrijednost, ili očekivanje, diskrette slučajne varijable X koja se označava sa μ , ili $E(X)$, je

$$\mu = \sum_{i=1}^n x_i p(x_i).$$

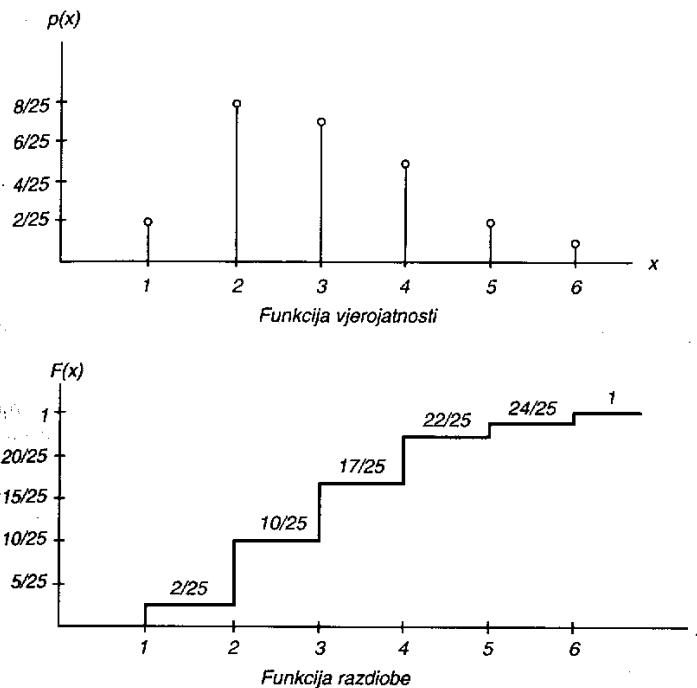
Za varijablu sa slike 9.1. je

$$\mu = 1(2/25) + 2(8/25) + 3(7/25) + 4(5/25) + 5(2/25) + 6(1/25) = 75/25 = 3.0$$

Ako se slučajni eksperiment provodi velik broj puta i na kraju se izračuna prosječna vrijednost ishoda, ovaj izračunani prosjek trebao bi se približavati očekivanju slučajne varijable.

Varijanca diskrette slučajne varijable X , koja se označava sa σ^2 ili $Var(X)$, je

$$Var(X) = E((X - \mu)^2) = \sum_{i=1}^n (x_i - \mu)^2 p(x_i)$$



Slika 9.1. Primjer funkcije vjerojatnosti i odgovarajuće funkcije razdiobe za diskretnu slučajnu varijablu

Varijanca se može prikazati i u sljedećem obliku:

$$\begin{aligned} Var(X) &= E(X^2 - 2X\mu + \mu^2) = E(X^2) - 2E(X)\mu + \mu^2 = \\ &= E(X^2) - 2\mu^2 + \mu^2 = E(X^2) - \mu^2. \end{aligned}$$

Za varijablu sa slike 9.1. je

$$\begin{aligned} Var(X) &= (1-3.0)^2(2/25) + (2-3.0)^2(8/25) + (3-3.0)^2(7/25) + \\ &+ (4-3.0)^2(5/25) + (5-3.0)^2(2/25) + (6-3.0)^2(1/25) = \\ &= 4(2/25) + 1(8/25) + 0(7/25) + 1(5/25) + 4(2/25) + 9(1/25) = \\ &= 38/25 = 1.52. \end{aligned}$$

Dakle, varijanca slučajne varijable, koja je uvijek nenegativna, mjeri prosječno kvadratno odstupanje od sredine (tj. očekivane vrijednosti). Očekivana vrijednost i varijanca su pogodan način za opis značajnih aspekata razdiobe vjerojatnosti slučajne varijable.

Standardna devijacija diskrette slučajne varijable koja se označava sa σ , ili $S(X)$, je

$$\sigma = \sqrt{Var(X)} = \sqrt{E(X^2) - \mu^2}$$

Za varijablu sa slike 9.1. je

$$\sigma = \sqrt{1.52^{1/2}} = 1.2329$$

Ako imamo dvije diskretne slučajne varijable X i Y , tada *kovarijanca* $\text{Cov}(X,Y)$ mjeri njihovu linearnu ovisnost :

$$\text{Cov}(X,Y) = E((X - \mu_x)(Y - \mu_y))$$

Može se pokazati da je

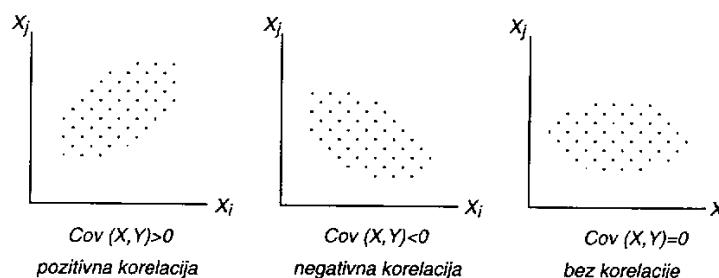
$$\text{Cov}(X,Y) = E(XY) - \mu_x \mu_y.$$

Ako je $\text{Cov}(X,Y) = 0$, tada za slučajne varijable X i Y kažemo da su *nekorelirane*, tj. njihove vrijednosti nisu povezane ni statistički. Može se pokazati da je ako su X_i i X_j *nezavisne* slučajne varijable, tada $\text{Cov}(X,Y) = 0$. Obrat ne vrijedi u općem slučaju. Međutim, ako X i Y imaju normalnu razdiobu, tada iz $\text{Cov}(X,Y) = 0$ slijedi da su varijable X i Y nezavisne.

Ako je $\text{Cov}(X,Y) > 0$, za X i Y kažemo da su *pozitivno korelirane*. Tada vrijedi da malim vrijednostima jedne slučajne varijable odgovaraju pretežno male vrijednosti druge varijable, a velikim vrijednostima jedne varijable velike vrijednosti druge.

Ako je $\text{Cov}(X,Y) < 0$, za X i Y kažemo da su *negativno korelirane*. Tada vrijedi da malim vrijednostima jedne slučajne varijable odgovaraju pretežno velike vrijednosti druge varijable, a velikim vrijednostima jedne varijable male vrijednosti druge.

Na slici 9.2. prikazani su primjeri pozitivno i negativno koreliranih te nekoreliranih slučajnih varijabli.



Slika 9.2. Korelacija slučajnih varijabli

Budući da kovarijanca $\text{Cov}(X,Y)$ nije bezdimenzionalna veličina, definiramo *koeficijent korelacije*, koji označavamo sa r ili $\text{Cor}(X,Y)$, kao

$$r = \frac{\text{Cov}(X,Y)}{\sqrt{\sigma_x^2 \sigma_y^2}}$$

i uzimamo je kao primarnu mjeru zavisnosti među X i Y . Korelacija ima isti predznak kao kovarijanca, a osim toga je i normirana :

$$-1 \leq r \leq 1.$$

9.1.2. Kontinuirane slučajne varijable

Slučajna varijabla X je *kontinuirana* ako može poprimiti kontinuum vrijednosti. Budući da kontinuirana slučajna varijabla može poprimiti neprebrojivo beskonačno mnogo vri-

jednosti, računanje vjerojatnosti prebrojavanjem (kao što se to čini kod diskretnog slučaja) nije moguće. Štoviše, vjerojatnost da kontinuirana slučajna varijabla poprimi bilo koju pojedinačnu vrijednost u svojoj domeni jednaka je nuli.

Primjeri kontinuirane slučajne varijable su vrijeme posluživanja na blagajni, vrijeme trajanja telefonskog razgovora itd.

Opis kontinuirane slučajne varijable dobiva se preko kontinuirane *funkcije razdiobe* dane sa:

$$F(x) = P(X \leq x).$$

Područje slučajne varijable X može se sastojati od jednog ili više intervala, a funkcija razdiobe $F(x)$ mora biti glatka krivulja u svakom od tih intervala i to takva da se njezina derivacija može integrirati preko svih intervala.

Sada se *funkcija vjerojatnosti*, koja se označava sa $f(x)$, dobiva kao derivacija funkcije razdiobe

$$f(x) = \frac{dF(x)}{dx}.$$

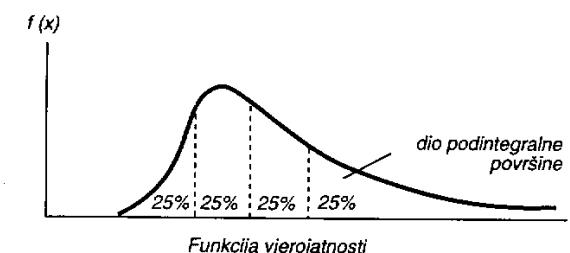
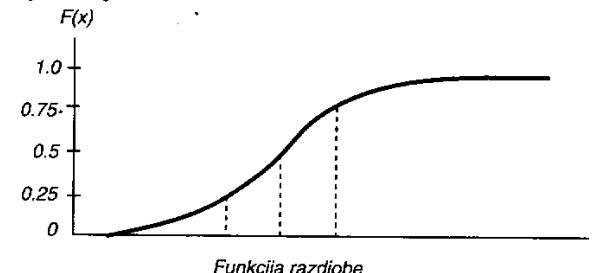
Funkcija vjerojatnosti daje razdiobu vjerojatnosti slučajne varijable X u njezinoj domeni. Za nju vrijedi:

$$\begin{aligned} f(x) &= 0 && \text{ako } x \text{ nije u domeni varijable } X \\ f(x) &\geq 0 && \text{inače} \end{aligned}$$

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

$$F(x_1) = \int_{-\infty}^{x_1} f(x) dx.$$

Na slici 9.3. prikazan je primjer funkcije razdiobe i funkcije vjerojatnosti kontinuirane slučajne varijable.



Slika 9.3. Primjer funkcije razdiobe i funkcija vjerojatnosti kontinuirane slučajne varijable

Očekivanje kontinuirane slučajne varijable X , koje označavamo sa μ , odnosno $E(X)$, je :

$$\mu = \int_{-\infty}^{\infty} xf(x)dx.$$

Varianca od X je :

$$\text{Var}(X) = \int_{-\infty}^{\infty} (x - \mu)^2 f(x)dx.$$

Standardna devijacija od X je :

$$\sigma = \sqrt{\text{Var}(X)}$$

Kovarijanca i korelacija dviju kontinuiranih slučajnih varijabli definiraju se jednako kao i za diskretne slučajne varijable.

9.2. UZORCI I STATISTIČKO ZAKLJUČIVANJE O SVOJSTVIMA POPULACIJE

U simulacijskim se modelima radi s *uzorcima* vrijednosti slučajnih varijabli, uzetih iz neke razdiobe vjerojatnosti slučajne varijable (tj. *populacije*). Tako pri mjerjenju ulaznih podataka, generiranju vrijednosti nezavisnih varijabli u simulacijskim eksperimentima ili analizi izlaznih podataka simulacijskih eksperimenata uvijek radimo s nekim podskupom vrijednosti varijable (opažanja) izvućenim iz razdiobe vjerojatnosti varijable. Na temelju dobivenih uzoraka stvaramo zaključke o karakteristikama populacije (tipu razdiobe, parametrima razdiobe, očekivanim vrijednostima, varijanci i sl.). Osnovna su pitanja stoga: kako ispravno izabrati ili stvoriti uzorak i kako na temelju uzorka ispravno zaključiti o karakteristikama populacije koja nas zanima. O kvaliteti tih postupaka ovisi točnost procjena o populaciji koje će se napraviti na temelju uzorka.

9.2.1. Stvaranje slučajnog uzorka

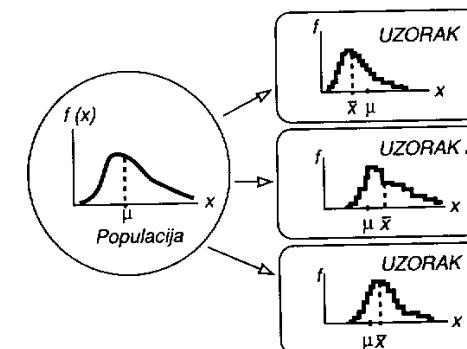
Slučajni uzorci moraju osigurati da se odabiranje iz populacije izvodi na slučajan način. U poglavljaju o generiranju uzorka bit će opisane metode generiranja slučajnih varijabli koje stvaraju uzorak vrijednosti slučajne varijable iz dane razdiobe vjerojatnosti. Na temelju tako generiranih vrijednosti nezavisnih slučajnih varijabli simulacijski eksperimenti daju uzorce izračunanih zavisnih varijabli simulacije na temelju dinamike rada modela i međudjelovanja njegovih varijabli.

Pri mjerjenju ulaznih varijabli simulacijskog modela također je potrebno stvoriti slučajni uzorak vrijednosti ulaznih varijabli. Metode osiguravanja slučajnosti uzorka prelaze namjenu ovog teksta i mogu se naći u statističkoj literaturi.

9.2.2. Statističko zaključivanje o populaciji na temelju uzorka

Parametri populacije (npr. očekivanje i varijanca) u pravilu nisu poznati, a njihove vrijednosti nastojimo procijeniti na temelju dobivenog uzorka iz populacije. Slika 9.4.

pokazuje razdiobu kontinuirane slučajne varijable s njezinim parametrima, te karakteristike nekoliko uzoraka uzetih iz te razdiobe. Kao što vidimo, procijenjeni parametri razlikuju se od uzorka do uzorka.

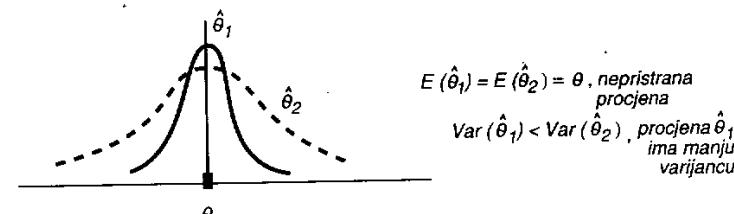


Slika 9.4. Razdioba vrijednosti kontinuirane slučajne varijable i nekoliko uzoraka uzetih iz razdiobe

Dva osnovna tipa zaključivanja o populaciji na temelju uzorka jesu *procjena* vrijednosti parametara populacije i *testiranje hipoteza* o parametrima populacije. Elementi statističkog zaključivanja su sam zaključak te mjera njegove kvalitete (koja pokazuje kolika je vjerojatnost da je donesen ispravan zaključak).

Procjena vrijednosti parametara populacije može biti točkasta i intervalna.

Točkasta procjena daje jedan broj $\hat{\theta}$ koji je procjena za traženi parametar θ populacije. Teži se tome da se dobije nepristrana procjena, tj. da je $E(\hat{\theta}) = \theta$, koja uz to ima minimalnu varijancu oko θ (slika 9.5a).



(a) Točkasta procjena

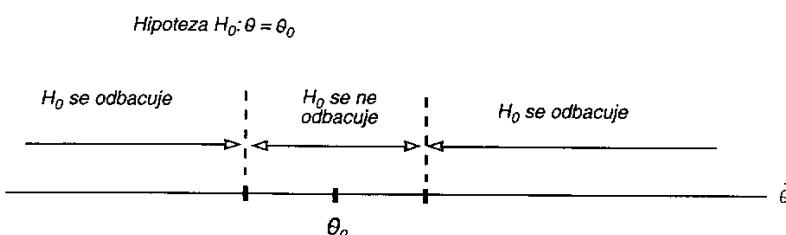


(b) Intervalna procjena

Slika 9.5. Točkasta i intervalna procjena parametra θ razdiobe

Intervalna procjena (interval pouzdanosti) interval je koji će sadržati vrijednost parametra θ sa specificiranim pouzdanostima (slika 9.5b). Dobra intervalna procjena daje najmanji interval oko θ , uz određenu pouzdanost i veličinu uzorka.

Testiranje hipoteza blisko je povezano za interval pouzdanosti. Postavlja se nul hipoteza H_0 o odabranom parametru θ populacije, i njoj alternativna hipoteza H_1 . Hipoteza H_0 se ili odbacuje ili ne odbacuje, na temelju obrade informacija iz uzorka. Ako je hipoteza H_0 odbačena, tada prihvaćamo alternativnu hipotezu H_1 . Pravilo odluke o ne-odbacivanju nul hipoteze H_0 postavlja se prema odabranoj statističkoj mjeri uzorka $\hat{\theta}$ (slika 9.6).



Slika 9.6. Područja prihvatanja i odbacivanja hipoteze H_0

9.3. STATISTIČKO ZAKLJUČIVANJE O SREDINI I VARIJANCI

9.3.1. Točkaste procjene sredine i varijance

Neka su X_1, X_2, \dots, X_n varijable (ishodi) slučajnog eksperimenta s nezavisnom identičnom razdiobom (NIR), s očekivanjem μ i varijancom σ^2 .

Tada je *sredina uzorka*

$$\bar{X}(n) = \frac{1}{n} \sum_{i=1}^n X_i$$

nepristrana točkasta procjena za μ , tj. $E(\bar{X}(n)) = \mu$.

Slično tome, *varijanca uzorka*

$$s^2(n) = \left(\frac{1}{(n-1)} \right) \sum_{i=1}^n (X_i - \bar{X}(n))^2$$

je nepristrana procjena za σ^2 , tj. $E(s^2(n)) = \sigma^2$.

Da bi se $\bar{X}(n)$ koristio kao procjena od μ , potrebna je dodatna informacija o tome koliko blizu je $\bar{X}(n)$ do μ . Procjena odstupanja provodi se konstrukcijom *intervala pouzdanosti* za μ . Prvi korak u konstrukciji intervala pouzdanosti je procjena varijance sredine uzorka $\sigma^2(\bar{X}(n))$. Budući da je varijanca sredine uzorka:

$$\sigma^2(\bar{X}(n)) = \sigma^2\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \left(\frac{1}{n^2}\right) \sigma^2\left(\sum_{i=1}^n X_i\right) =$$

9.3. STATISTIČKO ZAKLJUČIVANJE O SREDINI I VARIJANCI

$$= \left(\frac{1}{n^2}\right) \sum_{i=1}^n \sigma^2(X_i) \quad (\text{zbog nezavisnosti } X_i) \\ = \left(\frac{1}{n^2}\right) n \sigma^2 = \left(\frac{1}{n}\right) \sigma^2$$

Odatle je jasno da je sredina uzorka $\bar{X}(n)$ to bliže očekivanju populacije μ što je veći uzorak (tj. što je veći n).

Nepristranu procjenu od $\sigma^2(\bar{X}(n))$ dobivamo dalje zamjenom σ^2 sa $s^2(n)$, pa tako imamo procjenu odstupanja sredine uzorka $\bar{X}(n)$ od sredine populacije μ :

$$\sigma^2(\bar{X}(n)) = \left(\frac{1}{n}\right) s^2(n) = \left(\frac{1}{n(n-1)}\right) \sum_{i=1}^n (X_i - \bar{X}(n))^2$$

Svi se ovi rezultati temelje na pretpostavci da su X_i nezavisni, dakle i nekorelirani.

Osnovni je problem pri analizi *izlaznih varijabli simulacije* to da su oni praktično *uvijek korelirani*. To znači da gornji rezultati nisu *direktno* primjenljivi pri analizi izlaza simulacije. Međutim, kao što ćemo kasnije vidjeti (u poglavljiju koje opisuje analizu izlaznih rezultata simulacije), često je moguće izlazne podatke simulacije *grupirati* u nova "opažanja" na koja se mogu primjeniti formule temeljene na opažanjima o nezavisnim identičnim razdiobama.

9.3.2. Intervali pouzdanosti za sredinu

Neka su X_1, X_2, \dots, X_n slučajne varijable s nezavisnom identičnom razdiobom, sa sredinom μ i varijancom σ^2 .

Tada, kao posljedicu centralnog graničnog teorema, dobivamo da slučajna varijabla $t^{(n)}$ definirana na sljedeći način:

$$t^{(n)} = \frac{1}{\sqrt{\frac{s^2(n)}{n}}} (\bar{X}(n) - \mu)$$

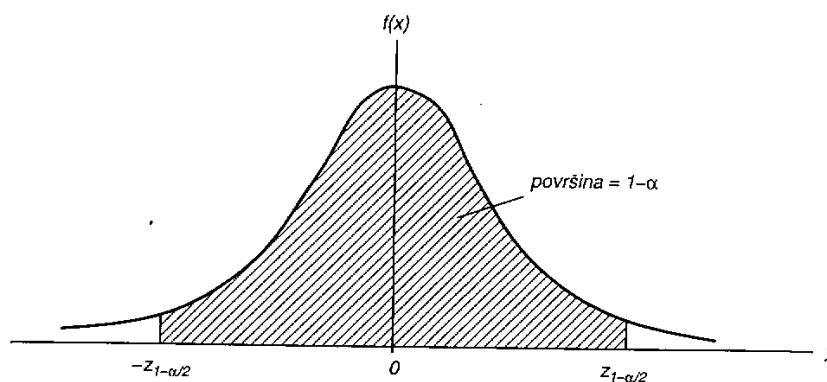
ima za "dovoljno velik" n funkciju vjerojatnosti koja se može dobro aproksimirati s funkcijom vjerojatnosti standardne normalne varijable (za koju je $\mu = 0$ i $\sigma^2 = 1$).

Zbog toga slijedi da je za velike n

$$P\left\{-z_{1-\alpha/2} \leq \frac{\bar{X}(n) - \mu}{\sqrt{s^2(n)/n}} \leq z_{1-\alpha/2}\right\} = \\ = P\left\{\bar{X}(n) - z_{1-\alpha/2} \sqrt{\frac{s^2(n)}{n}} \leq \mu \leq \bar{X}(n) + z_{1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}\right\} \\ \approx 1 - \alpha$$

Ovdje $z_{1-\alpha/2}$ (za $0 < \alpha < 1$) predstavlja $(1-\alpha/2)$ kritičnu točku za standardnu normalnu razdiobu, kao što je prikazano na slici 9.7. Dakle, ako je n dovoljno velik, približna *intervalna procjena* za μ s pouzdanostu $(1-\alpha)$, tj. interval pouzdanosti (ili preciznost procjene) za μ , dana je sa:

$$\bar{X}(n) \pm z_{1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}$$



Slika 9.7. Funkcija vjerojatnosti za standardnu normalnu razdiobu, i njezine kritične točke

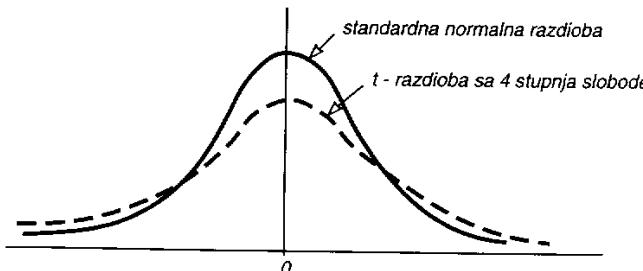
To znači da će, ako se konstruira velik broj intervalnih procjena za μ s pouzdanošću $(1-\alpha)$ svaki temeljen na n opažanja, dio tih intervala koji sadrže (*pokrivaju*) μ biti $(1-\alpha)$. Ako je broj opažanja n premalen, tada će stvarno pokrivanje intervala pouzdanosti biti manje od $(1-\alpha)$.

Ako su X_i slučajne varijable s *normalnom razdiobom*, može se pokazati da slučajna varijabla $t^{(n)}$ ima studentovu t-razdiobu sa $n-1$ stupnjeva slobode. Tada je točna intervalna procjena za μ s pouzdanošću $(1-\alpha)$, za veličinu uzorka $n \geq 2$, dana sa

$$\bar{X}(n) \pm t_{n-1,1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}$$

gdje je $t_{n-1,1-\alpha/2}$ gornja $1-\alpha/2$ kritična točka za t-razdiobu sa $(n-1)$ stupnjeva slobode.

Kao što se vidi na slici 9.8, t-razdioba ima niži vrh i šire repove od normalne razdiobe. U praksi je razdioba X_i -ova rijetko normalna; budući da je $t_{n-1,1-\alpha/2} > z_{1-\alpha/2}$, interval pouzdanosti temeljen na studentovoj t-razdiobi širi je od onog temeljenog na z-razdiobi, pa je njegovo pokrivanje (pouzdanost) bliže željenom nivou $(1-\alpha)$. Stoga se preporučuje da se interval pouzdanosti konstruira na osnovi studentove t-razdiobe.



Slika 9.8. Funkcija vjerojatnosti za t-razdiobu sa 4 stupnja slobode i za standardnu normalnu razdiobu

9.3.3. Testovi hipoteza za sredinu

Neka su X_1, X_2, \dots, X_n slučajne varijable s *normalnom* ili *približno normalnom* razdiobom, a mi želimo testirati nul-hipotezu H_0 da je $\mu = \mu_0$ (μ_0 je fiksna vrijednost). Intuitivno domišljamo: ako je $|\bar{X}(n) - \mu_0|$ veliko, H_0 vjerojatno nije istinit.

Ako je hipoteza H_0 *istinita*, tada statistika $t^{(n)} = \left(1/\sqrt{s^2(n)/n}\right)(\bar{X}(n) - \mu_0)$ ima t-razdiobu sa $(n-1)$ stupnjeva slobode.

Zato je oblik testa hipoteze za $\mu = \mu_0$ ovakav:

Ako je $|t^{(n)}| > t_{n-1,1-\alpha/2}$ H_0 se odbacuje
 $\leq t_{n-1,1-\alpha/2}$ H_0 se ne odbacuje.

Dio područja vrijednosti varijable X koji odgovara odbacivanju H_0 ($|X| > t_{n-1,1-\alpha/2}$) zove se *kritično područje* testa. Vjerojatnost da statistika $t^{(n)}$ pada u kritično područje ako je H_0 istinit (koja je jednaka α), naziva se nivo testa. Obično se izabire nivo α jednak 0.05 ili 0.1.

Pri izvođenju testa hipoteze moguća su dva tipa greške. *Greška tipa I* – odbacivanje je hipoteze H_0 kada je ona istinita – njezina vjerojatnost je α . Ako je npr. $\alpha = 0.05$ a mi odbacimo H_0 , možemo biti 95% sigurni da smo ispravno odlučili, a vjerojatnost greške u odlučivanju je 5%.

Greška tipa II – prihvatanje je hipoteze H_0 kada je ona pogrešna, tj. kada je alternativna hipoteza H_1 istinita. Vjerojatnost δ greške tipa II ovisi o obliku H_1 . Veličinu $1-\delta$, koja je jednaka vjerojatnosti odbacivanja pogrešne hipoteze H_0 , nazivamo *snagom* testa. Za fiksne α i H_1 , snaga testa raste samo s brojem stupnjeva slobode n . Budući da snaga testa može biti niska i nepoznata, tada ćemo, kada $t^{(n)}$ ne leži u kritičnom području, reći da "nismo uspjeli odbaciti H_0 " (umjesto "prihvatali smo H_0 "). Tada ne znamo s bilo kojom sigurnošću da li je H_0 istinit ili pogrešan, jer naš test nije možda dovoljno jak da ustani bilo kakvu razliku između H_0 i H_1 .

10. GENERIRANJE UZORAKA

Postojanje slučajnih veličina u simulacijskom modelu zahtjeva postojanje mehanizama koji u toku izvođenja simulacijskih eksperimentata mogu generirati vrijednosti varijabli iz različitih razdoba vjerojatnosti. Niz generiranih vrijednosti slučajne varijable je uzorak iz razdobe vjerojatnosti kojom je ta varijabla opisana. U ovom su poglavljju najprije opisani slučajni brojevi, koji čine osnovu za generiranje slučajnih varijabli. Prikazani su različiti načini generiranja slučajnih brojeva, te su detaljnije opisani linearni kongruentni generatori slučajnih brojeva i navedene osnovne metode testiranja generatora slučajnih brojeva. Zatim je opisana metoda inverzne transformacije za generiranje vrijednosti kontinuiranih i diskretnih slučajnih varijabli zadanih teorijskim ili empiričkim razdiobama vjerojatnosti. Na kraju je opisana metoda prihvaćanja i odbijanja kojom se mogu generirati vrijednosti slučajnih varijabli ako diskrete metode ne daju rezultate ili nisu dovoljno efikasne.

10.1. KORIŠTENJE SLUČAJNIH BROJEVA U SIMULACIJSKIM EKSPERIMENTIMA

Simulacijski modeli sistema ili procesa koji sadrže komponente koje se slučajno ponašaju zahtjevaju odgovarajuće metode generiranja slučajnih brojeva (Law i Kelton, 1982; Banks i Carson, 1984). Tijekom izvođenja simulacijskog eksperimenta može se npr. tražiti generiranje vrlo velikog broja vrijednosti vremena posluživanja, veličine zahtjeva ili međuvremena dolazaka koji pripadaju nekim razdiobama vjerojatnosti. Zato je potrebno imati kvalitetan i efikasan način generiranja vrijednosti slučajnih brojeva i varijabli. Pri tome naziv "generiranje slučajnih varijabli" nije najprecizniji, tj. pod tim pojmom zapravo mislimo na generiranje numeričkih vrijednosti slučajnih varijabli iz odgovarajućih razdoba vjerojatnosti. Niz generiranih vrijednosti slučajnih varijabli čini uzorak iz razdobe vjerojatnosti te varijable.

Korištenje slučajnih brojeva i varijabli u simulacijskom modelu omogućuje dakle reproduciranje nepravilnosti ponašanja elemenata sistema bez potrebe da model sadrži izvanredno detaljan opis tog ponašanja. Slučajni brojevi i varijable opisuju, dakle, nepravilno ponašanje u komprimiranom obliku.

10.2. ŠTO SU SLUČAJNI BROJEVI

S pojmovima slučajni broj i slučajna varijabla treba biti oprezan, jer nije jednostavno reći da li je određeni niz brojeva slučajan, iako nam se za neki niz čini da nije slučajan

(npr. 1, 2, 3, 4, 1, 2, 3, 4, ...), dok bismo za drugi rekli da jest slučajan (npr. 2, 7, 5, 9, 3, 6, ...). Da li je npr. slučajan niz brojeva dobiven uzimanjem uzastopnih znamenki telefonskih brojeva s proizvoljne stranice telefonskog imenika?

Na ovakvo pitanje ne može se ispravno odgovoriti na temelju poznavanja načina na koji je niz brojeva stvoren. Jedini pristup koji može dati zadovoljavajući odgovor jest da se taj niz brojeva podvrgne odgovarajućim *statističkim testovima* koji će pokazati da li je on uzorak slučajnog broja i koliko. Upravo se tako, što ćemo detaljnije opisati, ispituje i kvaliteta generatora slučajnih brojeva.

Precizno govoreći, pod pojmom "slučajni broj" podrazumijevamo kontinuiranu slučajnu varijablu s uniformnom razdiobom na intervalu [0,1]. Tu razdiobu označavat ćemo $U(0,1)$. Premda je to najjednostavnija kontinuirana razdioba, ona je *izvanredno važna* jer se slučajne varijable svih drugih razdioba vjerojatnosti (normalna, gama, binomna itd.) mogu dobiti *transformacijom* slučajne varijable $U(0,1)$ s nezavisnom identičnom razdiobom.

10.3. KAKO SE MOGU GENERIRATI SLUČAJNI BROJEVI

U dugoj i zanimljivoj povijesti metoda generiranja slučajnih brojeva (Hull i Dobell, 1962) korišteni su različiti načini generiranja.

10.3.1. Fizički pribori

Fizički pribori izgledaju najprirodniji za generiranje slučajnih brojeva, jer samim načinom generiranja sugeriraju slučajan karakter dobivenih brojeva. To vrijedi kako za bacanje kocke i okretanje ruleta, kao dobro poznatih sredstava za izbjegavanje sistematičnosti odabiranja brojeva (koji su zapravo najčešće i mentalna podloga za predodžbu pojma slučajnosti), tako i za složenije fizičke pribore i modele. Tako su 1930. godine Kendall i Babington-Smith upotrijebili brzo rotirajući disk za pripremanje tablica od 100 000 slučajnih znamenki. U istu svrhu korišten je i električni sklop temeljen na slučajnom pulsiranju vakuumske cijevi. Zanimljivo je da se pokazalo da telefonski brojevi uzeti iz telefonskog imenika ne zadovoljavaju testove slučajnosti.

No navedene metode pokazale su više nedostataka, od kojih su osnovni ti da su pribori dosta komplikirani, teško ih je održavati u stanju u kojem bi generirali niz iz iste razdiobe, a njima nije moguće izvesti ni simulacije u kojima bi se ponavljalo generiranje istog niza slučajnih brojeva. Ideja da se već generirani niz brojeva spremi u obliku tablice rješava neke od tih problema, ali zato zahtijeva dodatnu memoriju, a ujedno trpi i brzina rada jer je doseg podataka s periferne memorije prilično spor.

10.3.2. Korištenje iracionalnih brojeva

Čini se da znamenke u iracionalnim matematičkim konstantama (kao što su e, π itd.) ne bi trebale imati nikakve pravilnosti u sebi, pa su one isprobane i kao generatori slučajnih brojeva. Pokazalo se međutim da one imaju relativno slabe karakteristike slučajnosti, tako da se više uopće za to ne koriste.

10.3.3. Aritmetički procesi

Kako su računala (a s njima i simulacijsko modeliranje) postajala sve popularnija, istraživanje se počelo orijentirati prema aritmetičkim procesima za generiranje slučajnih brojeva. To kao da proturječi intuitivnom razumijevanju pojma slučajnosti, jer se čini nevjerojatno da bi se strogo determinističkim aritmetičkim operacijama moglo doći do slučajnih brojeva. Pokazalo se međutim da način generiranja brojeva ne mora biti ograničenje na kvalitetu generiranih brojeva. Tako su upravo neke od metoda generiranja slučajnih brojeva pomoću aritmetičkih procesa najbolje i najviše korištene.

Generatori slučajnih brojeva temeljeni na aritmetičkim procesima obično rade tako da se slučajni brojevi generiraju u nizu, i to tako da se *svaki novi broj* izračunava na temelju jednoga svojeg prethodnika ili više njih korištenjem fiksne matematičke formule

$$x_{i+1} = f(x_i)$$

odnosno

$$x_{i+1} = f(x_i, x_{i-1}, \dots)$$

Zbog ove determiniranosti načina generiranja, aritmetički generatori se nazivaju i *pseudoslučajnim* generatorima.

Jedan od prvih primjera takvih metoda generiranja je Von Neumannova i Metropolisova metoda *sredine kvadrata*, razvijena 1940-ih godina. Prema toj metodi krene se od početnog p-znamenkastog broja, a sljedeći broj dobiva se kvadranjem prethodnog broja i uzimanjem srednjih p-znamenki tog kvadrata za sljedeći broj. Slučajni broj dobije se stavljanjem decimalne točke ispred prve znamenke dobivenog p-znamenkastog broja. Primjer dobivenog niza četveroznamenkastih brojeva prikazan je u tablici 10.1.

Tablica 10.1. Primjer aritmetičkoga generiranja slučajnih brojeva metodom sredine kvadrata

i	n_i	n_i^2	U_i
1	5617	31550689	–
2	5506	30316036	0.5506
3	3160	09985600	0.3160
4	9856	97140736	0.9856
5	1407	01979649	0.1407
6	9796	95961616	0.9796
7	9616	92467456	0.9616
8	4674	21846276	0.4674
9	8462	71605444	0.8462
10	6054	36650916	0.6054

Iako se čini da ovaj način generiranja slučajnih brojeva ne unosi nikakvu sistematičnost, pokazalo se da se generirani nizovi ponavljaju u prilično kratkim ciklusima čija dužina ovisi o odabiru početne vrijednosti niza, te da niz nije dovoljno uniforman.

Dalja istraživanja pokazala su da dobar aritmetički generator slučajnih brojeva (kao i svaki drugi generator slučajnih brojeva) mora zadovoljavati neka svojstva. Prije svega, generirani brojevi moraju:

- (i) imati uniformnu razdiobu u intervalu $[0,1]$,
- (ii) biti nezavisni (tj. nekorelirani), odnosno da se vrijednosti iz razdiobe generiraju u slučajnom slijedu.

Osim toga, povoljno je da niz :

- (iii) bude reproducibilan (zbog mogućnosti usporedbe alternativnih konfiguracija sistema u što sličnijim uvjetima),
- (iv) da se ne ponavlja (odnosno da je period ponavljanja što duži),
- (v) da se generira brzo,
- (vi) da traži malo centralne memorije.

Dalje u tekstu opisat ćemo najviše korišteni generator slučajnih brojeva, te osnovne testove statističkih svojstava generatora slučajnih brojeva.

10.4. LINEARNI KONGRUENTNI GENERATORI

Većina generatora slučajnih brojeva koji se danas upotrebljavaju aritmetički su generatori, i to linearni kongruentni generatori (Lehmer, 1951). Niz generiranih cijelih brojeva x_1, x_2, \dots definira se rekurzivnom formulom:

$$x_{i+1} = (a x_i + c) \pmod{m}, \quad (10.1)$$

gdje su

m , modul,

a , množilnik,

c , inkrement,

x_0 , sjeme (početna vrijednost niza)

nenegativni cijeli brojevi.

Dakle, x_{i+1} je ostatak pri cijelobrojnem dijeljenju $a \cdot x_i + c$ sa m (npr. $15 \pmod{4} = 3$, $17 \pmod{5} = 2$). Zbog toga vrijedi:

$$0 \leq x_i \leq m-1,$$

pa željeni slučajni broj U_i iz intervala $[0,1]$ dobivamo iz:

$$U_i = \frac{x_i}{m} \quad (10.2)$$

Osim nenegativnosti, cijeli brojevi m, a, c i x_0 trebaju zadovoljavati relacije : $0 < m$, $a < m$, $c < m$ i $x_0 < m$.

Ako je u generatoru $c = 0$, on se zove 'multiplikativni kongruentni generator'

$(x_{i+1} = a x_i) \pmod{m}$; ako je $c \neq 0$, zove se 'miješani kongruentni generator'.

Najvažnije su karakteristike linearних kongruentnih generatora ove:

- (1) Jasno je da x_i nisu zaista slučajni brojevi, jer su oni potpuno određeni parametrima m, a, c i x_0 . Međutim, pažljivim odabirom tih četiriju parametara može se postići da x_i (odnosno U_i) izgleda kao slučajni broj kada se na njega primijene odgovarajući statistički testovi.

- (2) Iz relacije 10.2 vidi se da U_i može poprimiti samo racionalne vrijednosti

$$0, \frac{1}{m}, \frac{2}{m}, \dots, \frac{m-1}{m}$$

Zato je, na primjer, "vjerojatnost" da U_i bude između $0.1/m$ i $0.9/m$ jednaka 0, iako bi trebala biti jednak $0.8/m$ (zbog uniformnosti slučajnog broja). Ali, kako ćemo vidjeti, modul m je obično veoma velik (recimo 10^9 ili veći), tako da su točke u intervalu $[0,1]$ u koje U_i može pasti veoma guste. Za $m \geq 10^9$ generator može dati bar milijardu jednakih udaljenih mogućih vrijednosti za U_i u intervalu $[0,1]$, što je za većinu potreba dobra aproksimacija za pravu $U(0,1)$ razdiobu.

- (3) Linearni kongruentni generatori imaju cikluse u kojima se brojevi x_i ponavljaju.

Neka je npr. $m = 34$, $a = 14$, $c = 9$, $x_0 = 8$, tj. niz se generira rekurzivnom relacijom $x_{i+1} = (14 x_i + 9) \pmod{34}$. Tada dobivamo generirani niz slučajnih brojeva kao u tablici 10.2.

Tablica 10.2. Primjer niza slučajnih brojeva generiranih linearnim kongruentnim generatorom ($m = 34$, $a = 14$, $c = 9$, $x_0 = 8$)

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
x_i	8	19	3	17	9	33	29	7	5	11	27	13	21	31	1	23	25	19	3	17

Vidimo da je $x_{17}=x_1=19$, tj. nakon toga se isti niz brojeva ponavlja. Dužinu ciklusa nazivamo periodom p generatora ($p=16$ za prikazani niz). Budući da je $0 \leq x_i \leq m-1$, slijedi da je $p \leq m$, tj. najduži period generatora je $p = m$.

Kako se tijekom simulacijskih eksperimenata može zahtijevati generiranje jako dugih serija slučajnih brojeva, nužno je da generatori slučajnih brojeva imaju što duži period, tj. da je modul m velik i da je, po mogućnosti, $p = m$.

(4) Operacija dijeljenja sa m da bi se dobio ostatak relativno je spora aritmetička operacija. Ona se može izbjegći ako je $m = 2^b$ (gdje je b broj bitova u jednoj rječi računala minus 1). Tada se kod prikaza broja $(a \cdot x_i + c)$ u računalu izgube svi viši (lijevi) bitovi broja i ostane upravo ostatak koji bismo dobili dijeljenjem ovog broja sa m . Ako je npr. $b \geq 31$, tada je $m \geq 2^{31} (\approx 2.1 \cdot 10^9)$.

10.5. TESTIRANJE GENERATORA SLUČAJNIH BROJEVA

Postoji niz empirijskih testova za generatore slučajnih brojeva koji testiraju generirani niz brojeva. Većina testova ispituje svojstva uniformiranosti dobivene razdiobe i odsutnosti serijske korelacije generiranog niza brojeva. Izbor testova koji će se u određenom slučaju primijeniti ovisi o načinu na koji se slučajni brojevi koriste u modelu te o željenoj preciznosti rezultata simulacije.

Neki od najčešće korištenih empirijskih testova (Banks i Carson, 1984; Law i Kelton, 1982) jesu:

10.5.1. Test frekvencija

Testom frekvencija ispituje se *uniformnost* razdiobe slučajnih brojeva. Interval $[0,1]$ dijeli se u k jednakih podintervala i prebrojava se koliko slučajnih brojeva padne u pojedini interval (frekvencija). Ako je uzorak veličine n brojeva, očekivana frekvencija slučajnih brojeva po svakom intervalu je n/k .

Testiranje slaganja generiranih i očekivanih frekvencija po intervalima izvodi se pomoću hi-kvadrat testa slaganja sa $(k-1)$ stupnjeva slobode, ili pomoću Kolmogorov-Smirnova testa koji omogućuje provjeru hipoteze da je uzorak podataka izvučen iz određene kontinuirane razdiobe. Kolmogorov-Smirnov test je neparametrijski i egzaktan je za sve veličine uzorka.

10.5.2. Test serija

Test serija proširenje je testa frekvencija. Uzimaju se uzastopni parovi generiranih brojeva $(x_1, x_2), (x_3, x_4) \dots$ i broji se njihova frekvencija u odabranim elementima jediničnog kvadrata $[0,1] \times [0,1]$ podijeljenog na jednake kvadratične elemente stranice $1/k$. Ta se frekvencija uspoređuje s očekivanom frekvencijom parova po elementima jediničnog kvadrata. Test se analogno proširuje na trojke, četvorke itd. uzastopnih slučajnih brojeva.

10.5.3. Test porasta

Test porasta ispituje *nezavisnost* generiranih slučajnih brojeva. U generiranom nizu U_i traže se neprekidni podnizovi monotono rastućih brojeva U_i ('runs-up' nizovi). Iz niza U_i dužine n nađemo frekvenciju takvih podnizova dužine $1, 2, 3, 4, 5$ i $g \geq 6$.

Odgovarajućim statističkim postupkom prihvaćamo ili odbacujemo nul-hipotezu da su brojevi U_i *nezavisi* s identičnom razdiobom (NIR). Analogno se radi i pri analizi podnizova neprekidno padajućih brojeva, odnosno s testovima podnizova koji su iznad ili ispod srednje vrijednosti cijelog niza.

10.5.4. Testovi autokorelacije

Testovi autokorelacije ispituju *korelaciju* podskupa slučajnih brojeva x_i, x_{i+k}, x_{i+2k} , međusobno odmaknutih za k brojeva. Uspoređuje se korelacija uzorka s očekivanom korelacijom nula (tj. s nepostojanjem korelacijske).

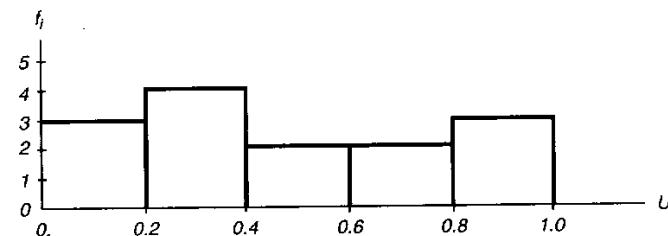
Na slici 10.1. prikazani su slučajni brojevi kompletog perioda linearog kongruentnog generatora iz tablice 10.2. Demonstriran je i njihov prikaz u obliku pogodnom za analizu pomoću nekoliko metoda testiranja generatora.

Složenost problematike generiranja i testiranja slučajnih brojeva ilustrira činjenica da se čak i kod nizova brojeva koji su dobro prošli testove slučajnih brojeva mogu naći podnizovi slabih karakteristika slučajnosti.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
U_i	.559	.088	.500	.265	.971	.853	.206	.147	.324	.794	.382	.618	.912	.029

(a) Prvih 14 generiranih slučajnih brojeva ($u_i = x_i / 34$)

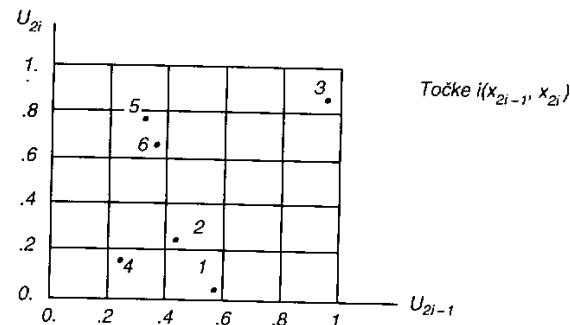
$$\bar{x} = \frac{\text{velič. uzorka}}{\text{broj klasa}} = \frac{14}{5} = 2.80$$



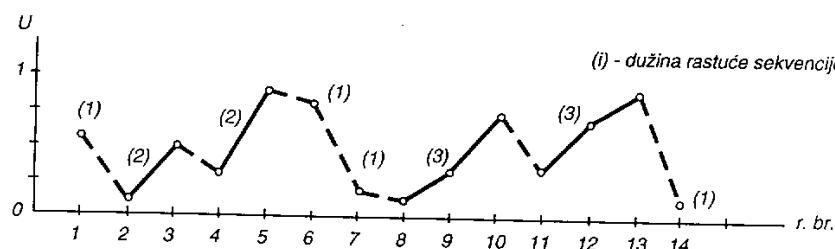
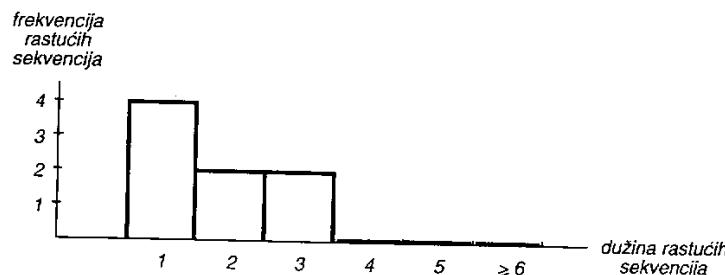
Interval	$\geq 0 - 0.2$	$> 0.2 - 0.4$	$> 0.4 - 0.6$	$> 0.6 - 0.8$	$> 0.8 - 1$
Frekvencija	3	4	2	2	3

(b) Frekvencija slučajnih brojeva po intervalima šrine 0.2 i histogram frekvencija (za test frekvencija)

Slika 10.1. Niz slučajnih brojeva generiranih linearnim kongruentnim generatorom $x_{i+1} = (14x_i + 9) \pmod{34}$, uz $x_0 = 8$



(c) Frekvencija uzastopnih parova generiranih slučajnih brojeva u elementima širine 0.2×0.2 . Točke $1(x_1, x_2), 2(x_3, x_4), \dots, 6(x_9, x_{10})$ — označene su sve točke (za test serija)



(d) Niz generiranih slučajnih brojeva i histogram sekvenčije rastućih brojeva dužine $1, 2, 3, 4, 5$ i ≥ 6 (za test porasta)

Slika 10.1. Niz slučajnih brojeva generiranih linearnim kongruentnim generatorom $x_{i+1} = (14x_i + 9) \pmod{34}$, uz $x_0 = 8$

10.6. SLUČAJNE VARIJABLE I NJIHOVO GENERIRANJE

Slučajne varijable u simulacijskim modelima, koje se pojavljuju kod opisa međuvremena dolaska, vremena posluživanja, grananja, generiranja atributa entiteta i sl., mogu imati različite razdiobe vjerojatnosti (eksponencijalnu, normalnu, gama itd.).

Kao što smo već naglasili, osnovni element potreban za generiranje slučajnih varijabli je generator slučajnih brojeva, tj. generator nezavisnih brojeva s identičnom uniformnom razdiobom $U(0,1)$. Bez njega nije moguće korektno generirati slučajne varijable bilo koje razdiobe.

Više je različitih metoda generiranja slučajnih varijabli (Law i Kelton, 1982; Banks i Carson, 1984), od kojih ćemo neke ovdje opisati. Izbor odgovarajuće metode ovisi o više faktora:

- točnosti generiranja
- efikasnosti generiranja (sa stajališta potrebne memorije i vremena izvođenja)
- složenosti algoritma generiranja
- robustnosti algoritma generiranja (tj. efikasnosti za široki opseg vrijednosti parametara).

Procjenom značenja tih faktora za određeni model ili slučajnu varijablu izabiremo odgovarajuće metode generiranja slučajne varijable.

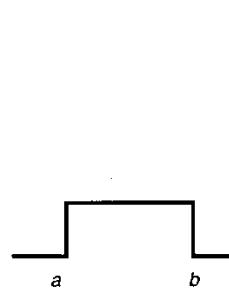
10.7. JEDNOSTAVNE TRANSFORMACIJE SLUČAJNE VARIJABLE

Najprije ćemo navesti kako se slučajni brojevi mogu upotrijebiti za generiranje nekoliko jednostavnih razdioba korištenjem jednostavnih transformacija slučajne varijable (Paul i Balmer, 1991).

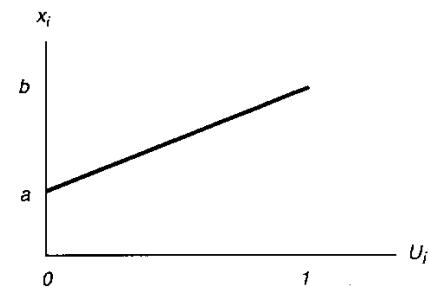
(1) Neka je x_i kontinuirana slučajna varijabla s uniformnom razdiobom iz intervala (a, b) . Ona se može dobiti iz slučajnih brojeva U_i , s uniformnom razdiobom u intervalu $[0, 1]$, transformacijom:

$$x_i = a + (b - a) \cdot U_i$$

prikazanom na slici 10.2a.



Razdioba $U(a,b)$



Transformacija $x_i = a + (b - a)U_i$

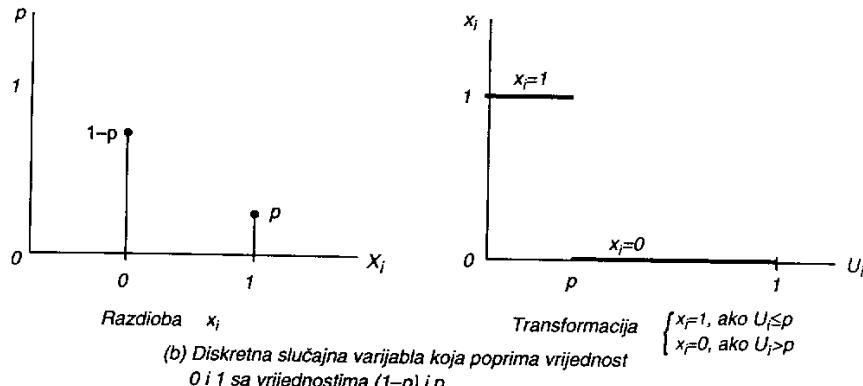
(a) Kontinuirana slučajna varijabla s razdiobom $U(a,b)$

(2) Neka je x_i diskretna slučajna varijabla koja može poprimiti vrijednosti 0 ili 1 s odgovarajućim vjerojatnostima $(1-p)$ i p .

Odgovarajuća transformacija:

$$\begin{array}{ll} x_i = 1 & , \text{ako } U_i \leq p \\ x_i = 0 & , \text{ako } U_i > p \end{array}$$

prikazana je na slici 10.2b.



Slika 10.2. Neke jednostavne transformacije slučajnih varijabli

Te primjere možemo promatrati kao specijalne slučajeve opće metode transformacije zvane inverzna transformacija.

Sada ćemo prikazati osnovne karakteristike najviše korištenih općih metoda transformacije slučajnih brojeva.

10.8. METODA INVERZNE TRANSFORMACIJE

10.8.1. Kontinuirane slučajne varijable

(1) Teorijske razdiobe

Želimo generirati slučajnu varijablu X koja je kontinuirana i ima kontinuiranu i striktno rastuću funkciju razdiobe F . Neka je F^{-1} inverz funkcije F (F^{-1} za danu vrijednost funkcije F daje vrijednost argumenta funkcije).

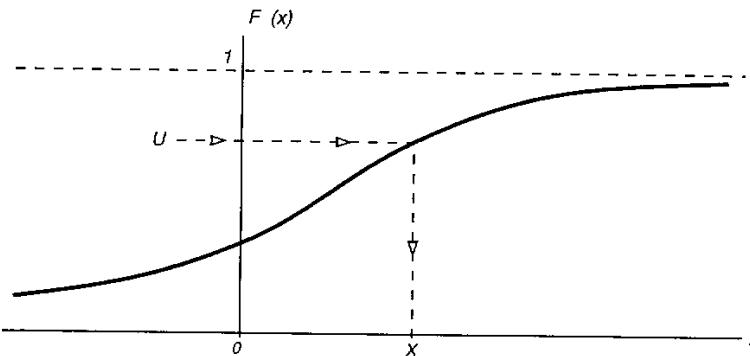
Algoritam za generiranje slučajne varijable X metodom inverzne transformacije je:

1. generiraj slučajni broj $U \sim U(0,1)$

(~ znači 's razdiobom kao')

2. nađi vrijednost x za koju je $F(x) = U$, tj. $x = F^{-1}(U)$.

Algoritam je prikazan grafički na slici 10.3.



Slika 10.3. Inverzna transformacija za kontinuirane slučajne varijable

Primjenu metode inverzne transformacije demonstrirat ćemo za slučaj negativne eksponencijalne razdiobe. Iz :

$$f(x) = m e^{-mx}, \quad x \geq 0$$

slijedi

$$\begin{aligned} F(x) &= \int_0^x m e^{-mx} dx = m \left(\frac{1}{m} e^{-mx} \right) \Big|_0^x = -e^{-mx} + 1 = \\ &= 1 - e^{-mx} \end{aligned}$$

Da bismo našli F^{-1} , stavimo

$$u = F(x) = 1 - e^{-mx},$$

$$\text{tj.} \quad 1 - u = e^{-mx} \quad / \ln$$

$$-mx = \ln(1-u)$$

$$x = -\left(\frac{1}{m}\right) \ln(1-u).$$

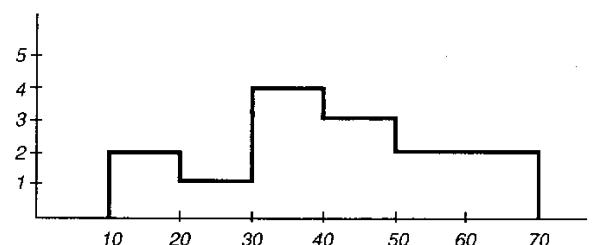
Budući da $(1-u)$ ima istu razdiobu kao u , tj. kao $U(0,1)$, dobivamo:

$$X = -\frac{1}{m} \ln U.$$

Dakle, X je slučajna varijabla s negativnom eksponencijalnom razdiobom čije vrijednosti dobivamo tako da najprije generiramo vrijednosti varijable u s uniformnom razdiobom iz $U(0,1)$, i tada izračunamo $x = -\frac{1}{m} \ln u$.

Ako se inverz funkcije ne može naći analitičkim putem (npr. kod normalne razdiobe), provode se *numeričke procedure* za rješenje jednadžbe $u = F(x)$ po x za svaku generiranu vrijednost slučajnog broja u .

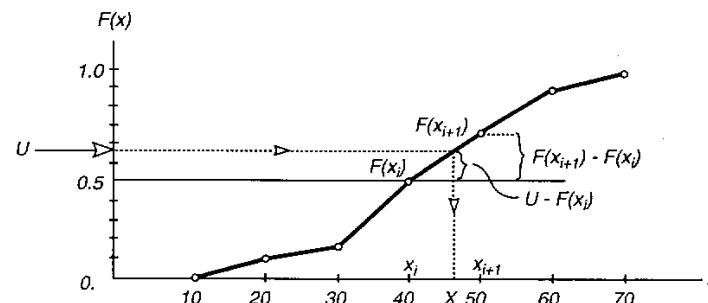
(2) Empirijske razdiobe



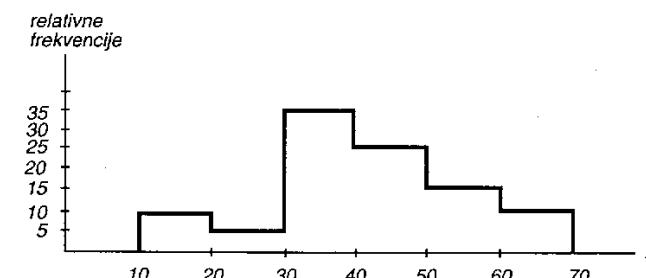
(d) Dobiveni histogram vjerojatnosti generiranih slučajnih varijabli

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_i	42.36	18.80	40.00	33.29	67.10	56.87	31.60	29.40	34.97	52.93	36.63	44.72	61.20	12.90

(c) Generiranje vrijednosti slučajne varijable pomoću niza slučajnih brojeva iz sl. 10.1a korištenjem metode inverzne transformacije za empirijske kontinuirane varijable



(b) Funkcija razdiobe



(a) Histogram relativnih frekvencija

Slika 10.4. Histogram relativnih frekvencija i funkcija razdiobe kontinuirane empirijske razdiobe

10.8. METODA INVERZNE TRANSFORMACIJE

Ako se kontinuirana slučajna varijabla ne može dobro opisati teorijskim razdiobama vjerojatnosti, nego se koriste empirijske razdiobe, tada se metoda inverzne transformacije može primijeniti ovako (Law i Kelton, 1982): neka je empirijska razdioba zadana histogramom relativnih frekvencija kao na slici 10.4a. Na temelju histograma možemo konstruirati funkciju razdiobe koja je linearna po odsjećima (za svaki stupac histograma), kao što je prikazano na slici 10.4b.

Na slici 10.4b ujedno je prikazan i način korištenja metode inverzne transformacije za generiranje empirijske razdiobe vjerojatnosti. Algoritam generiranja je:

1. Generiraj $u \sim U(0,1)$.
2. Generirani slučajni broj u je između $F(x_i)$ i $F(x_{i+1})$, tj.
 $F(x_i) \leq u \leq F(x_{i+1})$,

a vrijednost generirane slučajne varijable x dobije se linearnom interpolacijom vrijednosti funkcije razdiobe $F(x)$ između x_i i x_{i+1} :

$$x = x_i + \frac{u - F(x_i)}{F(x_{i+1}) - F(x_i)}(x_{i+1} - x_i)$$

Na slici 10.4c prikazan je niz dobivenih vrijednosti slučajne varijable x , čiji je histogram zadan na slici 10.4a. Niz je dobiven korištenjem prvih 14 vrijednosti generiranih od linearog kongruentnog generatora slučajnih brojeva $x_{i+1} = (14 \cdot x_i + 9) \pmod{34}$ sa $x_0 = 8$, prikazanih na slici 10.1a. Tako se npr. slučajni broj $u_6 = 0.853$ nalazi između $F(50) = 0.750$ i $F(60) = 0.900$, pa je:

$$x_6 = 50 + \frac{0.853 - 0.750}{0.900 - 0.750}(60 - 50) = 50 + 0.687 \cdot 10 = 50 + 6.87 = 56.87$$

Odgovarajući histogram generiranih vrijednosti varijable x prikazan je na slici 10.4d. Iako je uzorak veoma mali ($n = 14$), vidimo da razdioba vjerojatnosti generiranog uzorka pokazuje sličnost s originalnom empirijskom razdiobom vjerojatnosti varijable.

10.8.2. Diskretnе slučajne varijable

Kod diskretnih slučajnih varijabli ista metoda generiranja vrijedi za varijable zadane teorijskom ili empirijskom razdiobom.

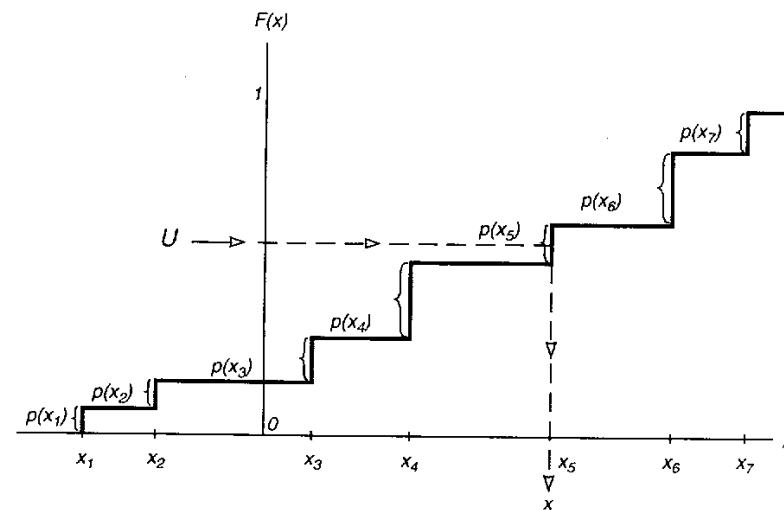
Pretpostavimo da X može poprimiti samo diskretne vrijednosti $x_1, x_2, \dots, x_p, \dots$, te daje $x_1 < x_2 < x_3 \dots$. Pri tome je :

$$F(x) = P(X \leq x) = \sum_{\{i : x_i \leq x\}} p(x_i), \quad \text{funkcija razdiobe ,}$$

$$\text{a } p(x_i) = P(X = x_i), \quad \text{funkcija vjerojatnosti .}$$

Algoritam generiranja, prikazan na slici 10.5, je:

1. Generiraj $u \sim U(0,1)$.
 2. Odredi najmanji pozitivni cijeli broj I takav da je $u \leq F(x_i)$, i stavi $X = x_I$.
- (Dakle, kada je $0 \leq u \leq p_1$, tada je $X = x_1$
 $p_1 < u \leq p_1 + p_2$, tada je $X = x_2$ itd.)



Slika 10.5. Inverzna transformacija za diskretne slučajne varijable

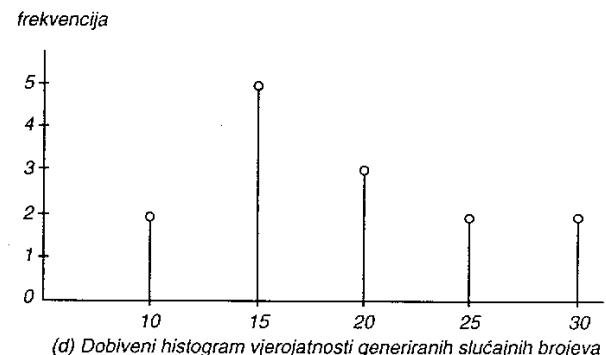
Ova metoda funkcioniра и када дискретна варijабла може попримити бесконачно mnogo vrijednosti (npr. kod Poissonove i geometrijske razdiobe).

Veća efikasnost u primjeni ove metode može se postići korištenjem odgovarajućih teknika sortiranja i pretraživanja.

Na slici 10.6. prikazan je niz vrijednosti slučajne varijable dobiven iz zadane razdiobe vjerojatnosti korištenjem prvih 14 vrijednosti (prikazanih na slici 10.1a) koje je generirao linearni kongruentni generator slučajnih brojeva $x_{i+1} = (14x_i + 9) \pmod{34}$ sa $x_0 = 8$.

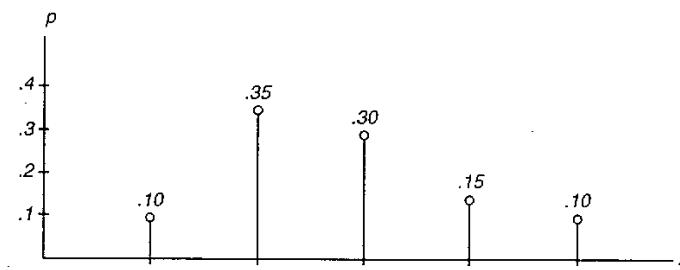
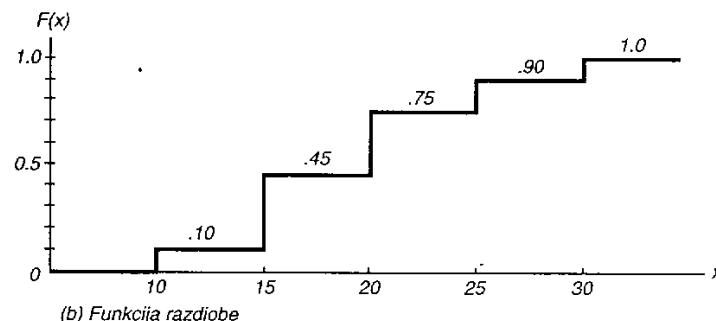
Dvije druge poznate *direktne* metode generiranja slučajnih varijabli, tj. metode koje direktno rade sa željenom razdiobom slučajne varijable (Law i Kelton, 1982) jesu:

- *metoda kompozicije* koja se primjenjuje kada funkcija razdiobe F može biti prikazana kao konveksna kombinacija više drugih funkcija razdiobe F_1, F_2, \dots iz kojih se uzorci mogu dobiti jednostavnije nego za originalni F
- *metoda konvolucije* koja se primjenjuje za nekoliko važnih razdioba (npr. gama razdioba) za koje željena slučajna varijabla X može biti prikazana kao suma drugih slučajnih varijabli s nezavisnom i identičnom razdiobom (NIR) koje se mogu generirati jednostavnije nego X .



i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_i	20	10	20	15	30	25	15	15	15	25	15	20	30	10

(c) Generirane vrijednosti slučajne varijable s danim generatorom

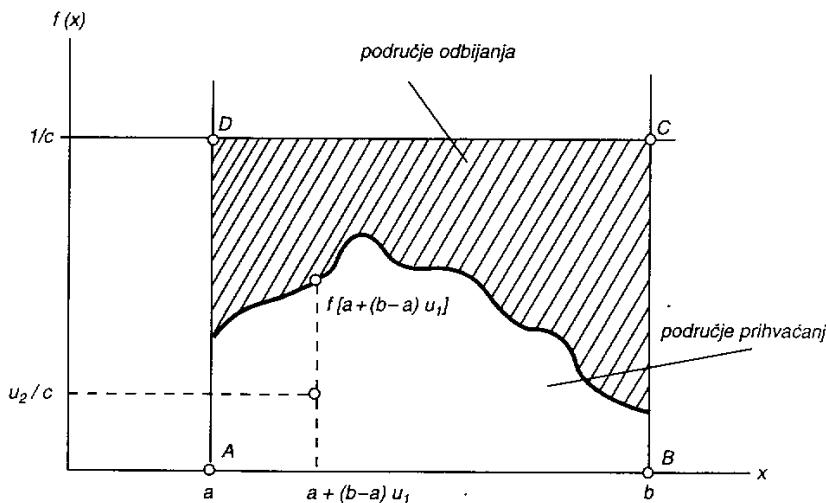


Slika 10.6. Diskretna slučajna varijabla i generiranje njezinih vrijednosti pomoću niza slučajnih brojeva iz slike 10.1a korištenjem metode inverzne transformacije

10.9. METODA PRIHVAĆANJA I ODBIJANJA

Metoda prihvaćanja i odbijanja manje je direktna metoda od prethodno navedenih, i može biti korisna kada direktnе metode ili ne daju rezultat ili su neefikasne. Metoda je veoma slična i za kontinuirane i za diskretne slučajne varijable. Prikazat ćemo generiranje *kontinuirane* slučajne varijable.

Neka je funkcija vjerojatnosti $f(x)$ ograničena, i neka je uvijek manja od vrijednosti $1/c$. Neka varijabla X ima konačno područje vrijednosti $a \leq x \leq b$. Metoda prihvaćanja i odbijanja prikazana je na slici 10.7.



Slika 10.7. Metoda prihvaćanja i odbijanja

Ideja metode je da se generira slučajna točka u pravokutniku ABCD. Ako ta točka leži u osjenčanom području pravokutnika, ona se *odbija* i generira se druga točka, i to sve dok se ne generira točka koja leži u neosjenčanom području pravokutnika. Ta se točka *prihvata* i njezina x koordinata služi kao slučajna vrijednost (opažanje) željene razdiobe.

Slučajna točka u pravokutniku može se generirati pomoću generiranja para slučajnih brojeva (u_1, u_2) sa $U(0,1)$ razdiobom. Tada su koordinate slučajne točke $(a + (b-a)u_1, u_2/c)$.

Algoritam generiranja je :

1. Izabere se konstanta c za koju vrijedi

$$f(x) \leq \frac{1}{c} \quad \text{za domenu } a \leq x \leq b.$$

2. Generira se par slučajnih brojeva (u_1, u_2) iz $U(0,1)$.

Neka je $x = a + (b-a) u_1$ i

$$y' = \frac{u_2}{c},$$

tj. točke (x', y') imaju jednaku vjerojatnost da padnu u bilo koji dio pravokutnika ABCD.

3. Ako par (x', y') ne zadovoljava relaciju

$$y' \leq f(x')$$

tada se točka *odbija* i ponovo se ide na korak 2, a ako je zadovoljava, tada je *prihvaćena* generirana vrijednost slučajne varijable:

$$x = a + (b-a) u_1.$$

Može se dokazati da ova metoda generira uzorak iz željene funkcije vjerojatnosti f, te da je očekivani broj pokušaja prije nego što se nađe uspešan par jednak $(b-a)/c$.

10.10. GENERIRANJE NEKIH ZNAČAJNIH RAZDIOBA

Za generiranje vrijednosti slučajnih varijabli iz nekih često korištenih razdioba razvijene su specifične metode koje su efikasnije od općih metoda generiranja (Law i Kelton, 1982). Ovdje ćemo navesti neke od njih.

(a) Kontinuirane slučajne razdiobe

- (1) *Eksponencijalna razdioba* $f(x) = m e^{-mx}$

Algoritam smo pokazali prije :

1. Generiraj $U \sim U(0,1)$

2. Stavi $X = - (1/m) \ln U$.

- (2) *Normalna razdioba* $N(m, \sigma^2)$

Jedna od često korištenih metoda generiranja je :

1. Generiraj n slučajnih brojeva u_i iz $U(0,1)$

2. Stavi $x = \sqrt{\sigma \frac{12}{n}} \left(\sum_{i=1}^n u_i - \frac{n}{2} \right) + m$.

Veličina uzorka n mora pri tome biti bar reda veličine 10.

(b) Diskretne slučajne varijable

- (1) *Poissonova razdioba* $p(x) = (1/x!) e^{-\lambda} \lambda^x$

Algoritam generiranja :

1. Neka je $a = e^{-\lambda}$, $b = 1$, $i = 0$

2. Generiraj $U_{i+1} \sim U(0,1)$, i zamjeni b sa $b \cdot U_{i+1}$.

Ako je $b < a$, stavi $x = i$,

ako nije, idи na korak 3.

3. Zamjeni i sa $i+1$, i vrati se na korak 2.

11. ANALIZA ULAZNIH PODATAKA

Ukupna kvaliteta ostvarenja simulacijskih procesa ne ovisi samo o kvaliteti simulacijskog modela već i o kvaliteti ulaznih podataka modela o kojima ovisi dinamika odvijanja simulacijskog eksperimenta i dobiveni rezultati ponašanja sistema. Ulazni podaci koji su slučajnog karaktera moraju se stoga prikupiti i analizirati na odgovarajući način, kako bi u modelu bili što vjernije prikazani. Poseban je problem analiza mogućih veza među ulaznim podacima, koje također moraju biti modelirane odgovarajućom točnošću. Osim rješavanja navedenih problema u ovom ćemo poglavlju opisati i najvažnije teorijske razdiobe vjerojatnosti što se susreću u simulacijskom modeliranju, i služe kako za opis ulaznih podataka, tako i izlaznih rezultata simulacije.

11.1. TRETMAN ULAZNIH PODATAKA SIMULACIJSKOG MODELA

Da bi se mogla provesti simulacija sistema koji sadrži slučajne varijable, nužno je specificirati *razdiobu vjerojatnosti* tih varijabli. To omogućuje da se tijekom simulacijskog eksperimenta generiraju vrijednosti tih ulaznih slučajnih varijabli u skladu s njihovim razdiobama. Opisat ćemo kako se provodi analiza i modeliranje slučajnih ulaznih varijabli.

Prikupljeni ulazni podaci mogu u simulacijskom modelu biti prikazani ili kao izvorne *empirijske razdiobe* ili se metodama statističkog zaključivanja načini njihova odgovarajuća *teorijska razdioba*. Osim u izuzetnim slučajevima, koristit ćemo se pristupom preko teorijske razdiobe, jer se time izbjegava uvođenje slučajnih fluktuacija specifičnih za određena očekivanja.

Razvoj adekvatnog modela ulaznih podataka sadrži četiri osnovna koraka:

- (1) Skupljanje ulaznih podataka.
- (2) Postavljanje hipoteze o porodici razdiobe vjerojatnosti ulaznih podataka.
- (3) Procjena vrijednosti parametara odabrane razdiobe.
- (4) Testiranje slaganja odabrane razdiobe (s njezinim procijenjenim parametrima) i ulaznih podataka.

Osim toga, ulazni podaci mogu biti međusobno povezani, pa ćemo opisati i način određivanja veze među njima, korištenjem regresijske analize.

Prije prikaza modeliranja ulaznih podataka opisat ćemo najčešće korištene razdiobe vjerojatnosti u simulacijskom modeliranju.

11.2. KORISNE RAZDIOBE VJEROJATNOSTI

Neke od razdioba vjerojatnosti slučajnih varijabli susreću se u velikom broju različitih sistema koji se opisuju simulacijskim modelima, bilo kao ulazne, bilo kao izlazne varijable (Law i Kelton, 1982). Opisat ćemo njihova osnovna obilježja.

11.2.1. Teorijske razdiobe

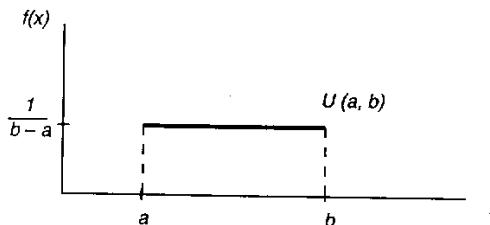
Kontinuirane razdiobe

(1) Uniformna razdoba $U(a,b)$

upotrebljava se za varijable koje slučajno fluktuiraju između vrijednosti a i b .

Funkcija vjerojatnosti (slika 11.1):

$$f(x) = \begin{cases} 1/(b-a) & , \text{ za } a \leq x \leq b \\ 0 & , \text{ inače} \end{cases}$$



Slika 11.1. Uniformna razdoba

Funkcija razdiobe:

$$F(x) = \begin{cases} 0 & , \text{ za } x < a \\ (x-a)/(b-a) & , \text{ za } a \leq x \leq b \\ 1 & , \text{ za } b < x \end{cases}$$

Sredina : $(a+b)/2$

Varijanca : $(b-a)^2/12$

Procjena parametara na osnovi ulaznih podataka metodom procjene maksimalne vjerojatnosti (PMV) :

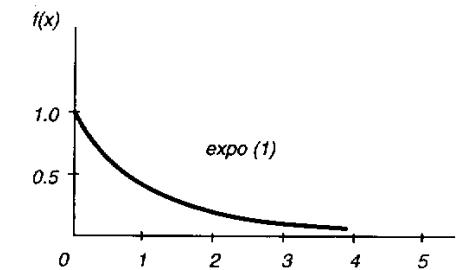
$$a = \min_{1 \leq i \leq n} X_i, \quad b = \max_{1 \leq i \leq n} X_i$$

(2) Eksponencijalna razdoba $\exp(\beta)$ (negativna eksponencijalna razdoba)

upotrebljava se najčešće za opis međuvremena dolaska te vremena između kvarova opreme (tj. "vrijeme života" opreme). Opisuje vrijeme između nezavisnih slučajnih događaja koji se dešavaju konstantnom brzinom.

Funkcija vjerojatnosti (slika 11.2):

$$f(x) = \begin{cases} (1/\beta) e^{-x/\beta} & , \text{ ako } x \geq 0 \\ 0 & , \text{ inače} \end{cases}$$



Slika 11.2. Eksponencijalna razdoba

Funkcija razdiobe:

$$F(x) = \begin{cases} 1 - e^{-x/\beta} & , \text{ ako } x \geq 0 \\ 0 & , \text{ inače} \end{cases}$$

Sredina: β

Varijanca: β^2

Procjena parametra PMV metodom : $\hat{\beta} = \bar{x}(n)$

(3) Gama-razdoba gama (α, β)

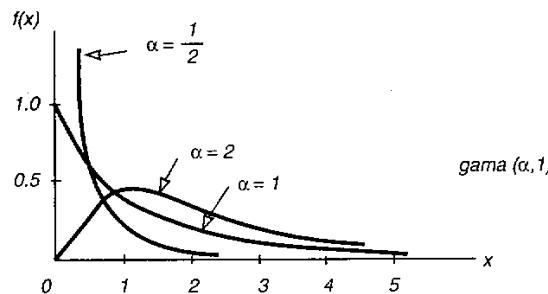
upotrebljava se za opis vremena izvršenja zadataka (posluživanje, popravak opreme i sl.).

Funkcija vjerojatnosti (slika 11.3):

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)} \beta^{-\alpha} x^{\alpha-1} e^{-x/\beta} & , \text{ ako } x > 0 \\ 0 & , \text{ inače} \end{cases}$$

gdje je $\Gamma(\alpha)$ gama funkcija :

$$\Gamma(z) = \int_0^\infty t^{(z-1)} e^{-t} dt, \quad z > 0$$



Slika 11.3. Gama razdioba

Funkcija razdiobe:

$$F(x) = 1 - e^{-\frac{x}{\beta}} \sum_{j=0}^{\alpha-1} \left(\frac{x}{\beta}\right)^j \frac{1}{j!}, \quad \text{ako } x > 0$$

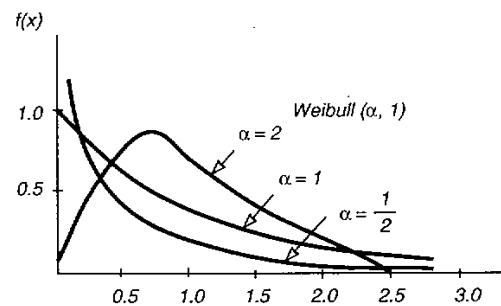
$$= 0, \quad \text{inače}$$

Sredina : $\alpha\beta$ Varijanca: $\alpha\beta^2$

Napomene :

- a) gama (α, β) je expo (β) razdioba
b) za pozitivan cijeli broj m , gama (m, β) razdioba zove se i Erlang (β) razdioba.

(4) Weibulova razdioba Weibull(α, β)
upotrebljava se za modeliranje "vremena života" opreme.
Funkcija vjerojatnosti (slika 11.4):



Slika 11.4. Weibulova razdioba

$$f(x) = \alpha \beta^{-\alpha} x^{\alpha-1} e^{-\left(\frac{x}{\beta}\right)^\alpha}, \quad \text{ako } x > 0$$

$$= 0, \quad \text{inače}$$

Funkcija razdiobe: $F(x) = 1 - e^{-\left(\frac{x}{\beta}\right)^\alpha}$

$$F(x) = 1 - e^{-\left(\frac{x}{\beta}\right)^\alpha}, \quad \text{ako } x > 0$$

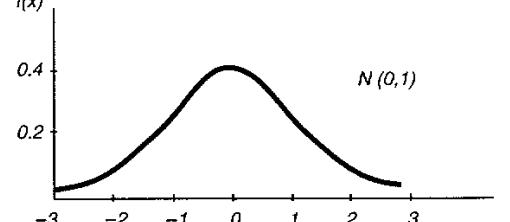
$$= 0, \quad \text{inače}$$

Sredina : $\frac{\beta}{\alpha} \Gamma\left(\frac{1}{\alpha}\right)$ Varijanca : $\frac{\beta^2}{\alpha} \left[2\Gamma\left(\frac{2}{\alpha}\right) - \frac{1}{\alpha} \left(\Gamma\left(\frac{1}{\alpha}\right) \right)^2 \right]$ Napomena. Weibull ($1, \beta$) je expo (β) razdioba.(5) Normalna razdioba $N(\mu, \sigma^2)$

upotrebljava se za opis vremena posluživanja, te niza drugih mjerjenih veličina (težina, dimenzija i sl.) i grešaka mjerjenja.

Funkcija vjerojatnosti (slika 11.5):

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad \text{za sve realne } x$$



Slika 11.5. Normalna razdioba

Funkcija razdiobe ne može se izraziti eksplizitno (postoje tabelirane vrijednosti).

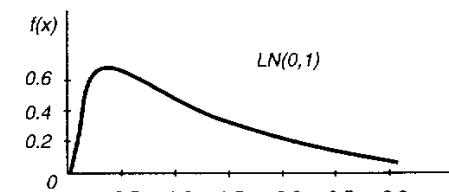
Sredina : μ Varijanca : σ^2 Procjena parametara PMV metodom : $\hat{\mu} = \bar{X}(n)$, $\hat{\sigma} = \sqrt{\frac{n-1}{n}} s^2(n)$ Napomena. Razdioba sume n slučajnih varijabli s nezavisnom identičnom razdiobom teži normalnoj razdiobi kada $n \rightarrow \infty$ (centralni granični teorem)(6) Lognormalna razdioba LN (μ, σ^2)

opisuje vrijeme izvršenja zadatka.

Funkcija vjerojatnosti (slika 11.6):

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}, \quad \text{ako } x > 0$$

$$= 0, \quad \text{inače}$$



Slika 11.6. Lognormalna razdioba

Funkcija razdiobe ne može se izraziti eksplicitno.

$$\text{Sredina} : e^{\frac{\mu + \sigma^2}{2}}$$

$$\text{Varijanca: } e^{2\mu + \sigma^2} \left(e^{\sigma^2} - 1 \right)$$

Procjena parametara PMV metodom:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \ln X_i, \quad \hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\ln X_i - \hat{\mu})^2}$$

Napomena. Ako $\ln X_1, \ln X_2, \dots, \ln X_n$ imaju normalnu razdiobu $N(\mu, \sigma^2)$ tada X_1, X_2, \dots, X_n imaju lognormalnu razdiobu $LN(\mu, \sigma^2)$.

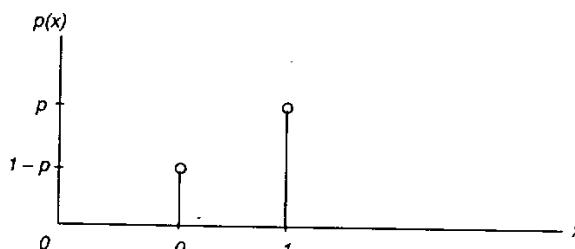
Diskrete razdiobe

(1) Bernoullijeva razdioba Bernoulli(p)

upotrebljava se za opis događaja koji imaju dva moguća ishoda, te za generiranje binomne, geometrijske i drugih razdioba.

Funkcija vjerojatnosti (slika 11.7):

$$\begin{aligned} p(x) &= 1 - p && \text{, ako } x = 0 \\ &= p && \text{, ako } x = 1 \\ &= 0 && \text{, inače} \end{aligned}$$



Slika 11.7. Bernoullijeva razdioba

Funkcija razdiobe :

$$\begin{aligned} F(x) &= 0 && \text{, ako } x < 0 \\ &= 1 - p && \text{, ako } 0 < x < 1 \\ &= 1 && \text{, ako } 1 < x \end{aligned}$$

Sredina : p

Varijanca : $p(1 - p)$

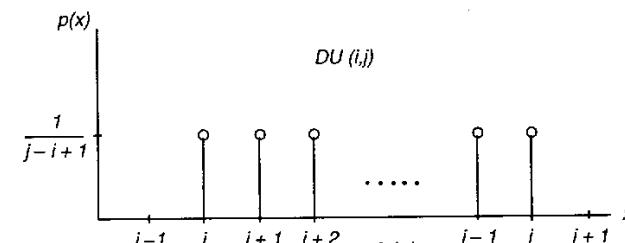
Procjena parametara PMV metodom : $\hat{p} = \bar{X}(n)$

(2) Diskretna uniformna razdioba DU(i, j)

opis je događaja s više mogućih ishoda, od kojih je svaki jednako vjerojatan.

Funkcija vjerojatnosti (slika 11.8):

$$\begin{aligned} p(x) &= \frac{1}{j-i+1} && \text{, ako } x \in (i, i+1, \dots, j) \\ &= 0 && \text{, inače} \end{aligned}$$



Slika 11.8. Diskretna uniformna razdioba

Funkcija razdiobe :

$$\begin{aligned} F(x) &= 0 && \text{, ako } x < i \\ &= (\lfloor x \rfloor - i+1)/(j-i+1) && \text{, ako } i \leq x \leq j \\ &= 1 && \text{, ako } j < x \end{aligned}$$

gdje $\lfloor x \rfloor$ označava najveći cijeli broj $\leq x$

Sredina : $(i+j)/2$

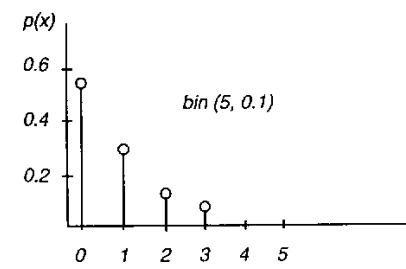
Varijanca : $((j-i+1)^2 - 1)/12$

Procjena parametara PMV metodom :

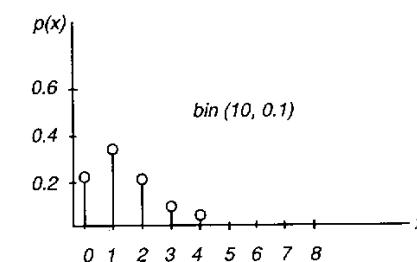
$$\hat{i} = \min_{1 \leq k \leq n} X_k, \quad \hat{j} = \max_{1 \leq k \leq n} X_k$$

(3) Binomna razdioba bin(n, p)

opis je n uzastopnih Bernoullijevih pokušaja s vjerojatnošću p za uspjeh svakog do-gađaja. Upotrebljava se za opis broja defektivnih komada u grupi veličine n, broja komada koji će se tražiti sa skladišta i sl.



Slika 11.9. Binomna razdioba



Funkcija vjerojatnosti (slika 11.9):

$$p(x) = \begin{cases} \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}, & \text{ako } x \in \{0, 1, \dots, n\} \\ 0, & \text{inače} \end{cases}$$

gdje je $\binom{n}{x}$ binomni koeficijent, $\binom{n}{x} = \frac{n!}{x!(n-x)!}$

Funkcija razdiobe:

$$\begin{aligned} F(x) &= 0 && \text{ako } x < 0 \\ &= \sum_{i=0}^{\lfloor x \rfloor} \binom{n}{i} p^i (1-p)^{n-i} && \text{ako } 0 \leq x \leq n \\ &= 1 && \text{ako } n < x \end{aligned}$$

Sredina : $n p$

Varijanca : $n p (1-p)$

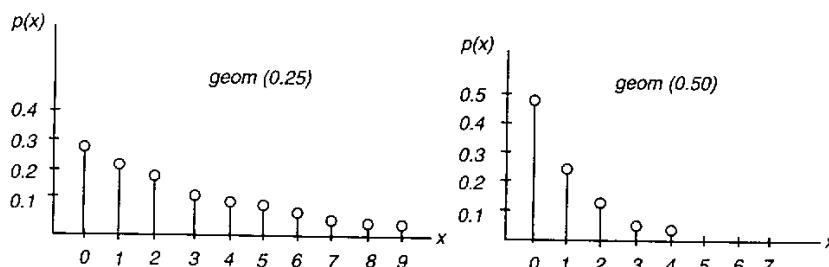
(4) Geometrijska razdioba geom(p)

opis je broja pokušaja između dvaju uspjeha (ili neuspjeha) u nizu nezavisnih Bernoulli-jevih pokušaja s vjerojatnošću p za uspjeh svakog pokušaja. Upotrebljava se za opis broja pripadnika grupe slučajne veličine, broja komada koji se potražuju sa skladišta i sl.

Funkcija vjerojatnosti (slika 11.10):

$$p(x) = p(1-p)^x \quad \text{ako } x \in \{0, 1, \dots\}$$

$$= 0 \quad \text{inače}$$



Slika 11.10. Geometrijska razdioba

Funkcija vjerojatnosti:

$$F(x) = \begin{cases} 1 - (1-p)^{\lfloor x \rfloor + 1}, & \text{ako } x \geq 0 \\ 0, & \text{inače} \end{cases}$$

Sredina : $(1-p)/p$

Varijanca : $(1-p)/p^2$

Procjena parametara PMV metodom : $\hat{p} = 1/\bar{X}(n)+1)$

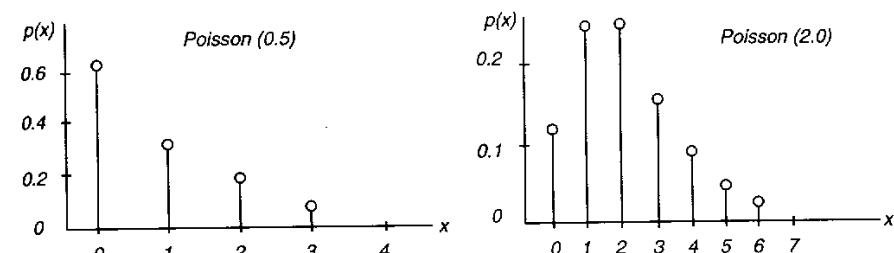
(5) Poissonova razdioba Poisson (λ)

opisuje broj rijetkih nezavisnih događaja koji se dešavaju u nekom vremenskom intervalu, odnosno broj komada u grupi i sl.

Funkcija vjerojatnosti (slika 11.11):

$$p(x) = e^{-\lambda} \frac{\lambda^x}{x!}, \quad \text{ako } x \in \{0, 1, \dots\}$$

$$= 0, \quad \text{inače}$$



Slika 11.11. Poissonova razdioba

Funkcija razdiobe:

$$F(x) = 0 \quad \text{ako } x < 0$$

$$= e^{-\lambda} \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!} \quad \text{ako } 0 \leq x$$

Sredina : λ

Varijanca : λ

Procjena parametara PMV metodom : $\hat{\lambda} = \bar{X}(n)$.

11.2.2. Empirijske razdiobe

Ako ne možemo pronaći adekvatnu teorijsku razdiobu za skupljene podatke, koristićemo se empirijskom razdiobom što su zapravo sami skupljeni podaci (Law i Kelton, 1982).

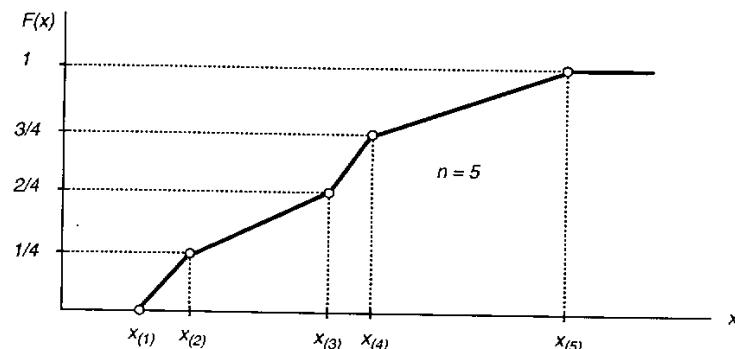
Ako je riječ o *kontinuiranoj slučajnoj varijabli*, način konstruiranja empirijske razdiobe ovisi o tome imamo li originalne podatke opažanja ili samo frekvencije tih podataka po klasama.

- a) Ako imamo originalne podatke opažanja X_1, X_2, \dots, X_n , sortirat ćemo ih prema rastućim vrijednostima $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n)}$. Tada se funkcija razdiobe konstruira kao linearna funkcija po intervalima između točaka

$$(X_{(1)}, 0), (X_{(2)}, 1/(n-1)), \dots, (X_{(i)}, (i-1)/(n-1)), \dots, (X_{(n)}, 1),$$

dakle $F(X_{(i)}) = (i-1)/(n-1)$, što je za velike n upravo proporcija opažanja X koji su manji od $X_{(i)}$.

Primjer tako konstruirane funkcije razdiobe prikazan je na slici 11.12. Jedna od loših strana tako konstruirane razdiobe jest to da generirane vrijednosti slučajne varijable ne mogu biti izvan intervala $X_{(1)}$ do $X_{(n)}$.



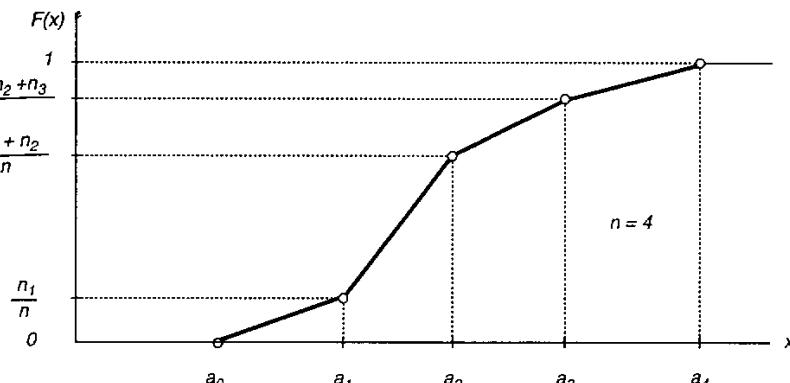
Slika 11.12. Kontinuirana linearna empirijska razdioba po intervalima iz originalnih podataka

- b) Ako su podaci grupirani u klase (tj. poznata su samo grupiranja, a ne i originalni podaci), potreban je drukčiji pristup konstruiranju empirijske razdiobe. Neka je n podataka grupirano u k susjednih intervala $[a_0, a_1], [a_1, a_2], \dots, [a_{k-1}, a_k]$, tako da j -ti interval sadrži n_j opažanja. Tada se može konstruirati funkcija razdiobe linearna po intervalima između točaka

$$(a_0, 0), (a_1, n_1/n), (a_2, (n_1+n_2)/n), \dots, (a_p, (n_1+n_2+\dots+n_p)/n), \dots, (a_n, 1).$$

Primjer tako konstruirane razdiobe je na slici 11.13. I kod ovog načina konstrukcije razdiobe vjerojatnosti generirane vrijednosti slučajne varijable ograničene su na interval a_0 do a_k .

Moguće je konstruirati i drukčije definirane razdiobe, npr. takve koje dodaju "repove" razdiobe s obje strane.



Slika 11.13. Kontinuirana linearna empirijska razdioba po intervalima za podatke grupirane prema klasama

Ako je pak riječ o *diskretnoj slučajnoj varijabli*, empirijska se distribucija jednostavno konstruira pomoću:

- (a) originalnih podataka X_1, X_2, \dots, X_n : funkcija vjerojatnosti $p(x)$ definira se kao proporcija X_i -ova koji su jednakvi vrijednosti x , ili se
- (b) podacima grupiranim u klase dobije tako da je suma $p(x)$ -ova za sve moguće vrijednosti od x u danom intervalu jednaka proporciji X_i -ova u tom intervalu.

11.3. SKUPLJANJE ULAZNIH PODATAKA

Kao što smo rekli, prvi korak u razvoju modela ulaznih podataka je skupljanje ulaznih podataka. To je u pravilu dosta dugotrajan proces koji nije jednostavan i nema potpuno predvidiv sadržaj ni tok.

Ulagni podaci koriste se kako za određivanje razdiobe slučajnih ulaznih (nezavisnih) varijabli, tako i za vrednovanje simulacijskog modela. Najčešće je potrebno skupiti podatke za više različitih vrsta procesa (dinamika dolazaka, karakteristike dolaznih entiteta, vremena posluživanja raznih resursa sistema itd.).

Zbog dugog trajanja, te cijene i raznovrsnosti ulaznih podataka, preporučljivo je da se proces skupljanja ulaznih podataka obvezno planira. Potrebno je planirati što će se mjeriti, kako, kada, gdje i tko će mjeriti.

- (1) Što će se mjeriti: izbor varijabli od interesa (nezavisne i zavisne varijable; nezavisne varijable koje mogu biti u međusobnoj povezanoći —npr. vrijeme posluživanja i karakteristike objekta koji se poslužuju).

- (2) Kako će se mjeriti: način mjerjenja, mjerne naprave, formulari, kontrola mjerjenja i sl. (u vezi s načinom mjerjenja može se npr. kod posluživanja mjeriti vrijeme posluživanja svakog pojedinog objekta mjerjenja, a kod brzih posluživanja često je bolje mjeriti broj posluženih entiteta u jedinici vremena).
- (3) Točnost mjerjenja
· (da li se mjeri u sekundama, minutama i sl., ovisno o dinamici fenomena koji promatramo).
- (4) Veličina uzorka
(broj mjerena za svaki tip fenomena koji se mjeri).
- (5) Kada će se mjeriti
(da li se mjeri u bilo kojoj fazi rada sistema ili možda samo u vršnjim opterećenjima, jer se način i brzina posluživanja mogu mijenjati u različitim okolnostima – u pravilu se npr. vrijeme posluživanja mjeri samo kada je pred mjestom posluživanja rep od bar nekoliko objekata).
- (6) Gdje će se mjeriti
(u kojem se dijelu sistema mjeri, gdje su promatrači koji mjeru kako bi imali dobar pregled ali da ne utječu na proces koji mjeri).
- (7) Tko će mjeriti
(broj mjeraca u ekipi koji osigurava pouzdano mjerjenje i eventualnu mogućnost kontrole mjerjenja; sinkroniziran raspored ekipa za mjerjenja na različitim mjestima i njihovo premještanje s mjesta na mjesto, kako bi se izmjerili svi fenomeni od interesa i dobro iskoristile eklipe koje mjeri).

Zbog boljeg planiranja i izvođenja mjerena preporučuju se i prethodna kraća *pi-lot mjerena* kako bi se steklo iskustvo, eliminirale eventualne greške i pronašao zadovoljavajući način mjerena.

Nakon mjerena podaci se unose na prikidan medij, a zatim se radi eventualna transformacija podataka, kontrola podataka (npr. da li su u dopuštenim granicama) i izbacivanje neadekvatnih podataka.

11.4. POSTAVLJANJE HIPOTEZE O PORODICI RAZDIOBA

Sljedeći korak u modeliranju ulaznih podataka je izbor porodice razdioba ulaznih podataka. Ponekad nam prethodno znanje o karakteristikama slučajne varijable ili području dopuštenih vrijednosti omogućuje da izaberemo razdiobu ulaznih podataka ili bar da isključimo neke od razdioba. Najčešće to ipak nije moguće, pa se jedino možemo koristiti nekim postupcima koji mogu pomoći pri izboru odgovarajuće razdiobe. Obično je moguće razlikovati da li je riječ o kontinuiranoj ili diskretnoj razdiobi, pa ćemo te slučajeve posebno razmotriti.

11.4.1. Kontinuirane razdiobe

Opisat ćemo tri heurističke (neegzaktne) procedure koje se koriste za izbor kontinuirane razdiobe.

(1) Točkasta statistika

Neke od funkcija pravih parametara razdioba mogu poslužiti kao karakterizacija razdioba. Na primjer, koeficijent varijacije razdiobe, $\delta = \sqrt{\text{Var}(X)/E(X)}$, razlikuje se za pojedine razdiobe.

Budući da δ ovisi o *stvarnim* parametrima razdiobe, njezina vrijednost se može procijeniti iz podataka:

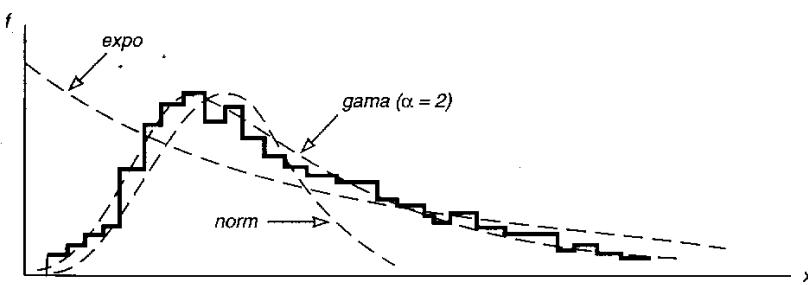
$$\hat{\delta}(n) = \frac{\sqrt{s^2(n)}}{\bar{X}(n)}$$

Ova je procjena brza i jednostavna, ali ne određuje oblik razdiobe na jedinstven način (Law i Kelton, 1982). Zato se koriste i druge funkcije parametara razdiobe, npr. koeficijenti asimetričnosti i spljoštenosti. I vrijednosti tih drugih funkcija treba međutim procijeniti iz podataka, što je zbog fluktuacija podataka vrlo gruba mjeru za procjenu razdiobe.

(2) Histogrami

Histogrami su grafičke procjene oblike funkcije vjerojatnosti koja odgovara razdiobi podataka X_1, X_2, \dots, X_n . Zbog prepoznatljivog oblika različitih funkcija vjerojatnosti ovaj pristup omogućava dosta dobru procjenu razdiobe.

Histogrami se konstruiraju tako da se područje vrijednosti varijable podijeli u intervale jednake širine i odrede se relativne frekvencije (broj opažanja varijable koja padnu u taj interval podijeljen ukupnim brojem opažanja varijable) što se nacrtaju kao funkcija od vrijednosti varijable. Taj se histogram *usporeduje* s grafičkim prikazom funkcija vjerojatnosti različitih razdioba vjerojatnosti, i to isključivo na osnovu *oblika*, kao što pokazuje primjer na slici 11.14.



Slika 11.14. Izbor kontinuirane teorijske razdiobe pomoću histograma

Ova je metoda jednostavna, opće primjenljiva i omogućava lagunu vizualnu interpretaciju. Poteškoća je izbor intervala i njihova širina; stoga se preporučuje isprobavanje nekoliko različitih širina intervala, dok se ne nađe širina koja daje histogram koji je najgladi i najbolje izgleda. Izvjestan gubitak točnosti kod ove je metode neizbjegjan zbog grupiranja podataka.

(3) Crteži vjerojatnosti

Dok su histogrami procjene oblika funkcije vjerojatnosti, crteži vjerojatnosti su grafičke usporedbe funkcije razdiobe podataka s teorijskim razdiobama. Međutim, kako je dosta teško zapaziti razlike u oblicima funkcija razdioba (S-krivulja), ideja je crteža vjerojatnosti da se ispitivanje svede na to da li točke crteža leže približno na pravcu. Ovdje ćemo dati osnovnu ideju konstrukcije jednog od mogućih crteža vjerojatnosti.

Prije svega, ulazni podaci X_1, X_2, \dots, X_n sortiraju se prema rastućem slijedu u niz $X_{(1)}, X_{(2)}, \dots, X_{(n)}$. Razumna aproksimacija za funkciju razdiobe $F_{(x)}$ je proporcija X_i -ova manjih ili jednakih od x , tj. $\tilde{F}_n(X_{(i)}) = i/n$ ili $\tilde{F}_n(X_{(i)}) = (i-0.5)/n$ (da ne bismo dobili $\tilde{F}_n = 1$ za konačnu vrijednost od x , $x = X_{(n)}$).

Uspoređivat ćemo tzv. *kvantile* (percentile) razdioba; za $0 < q < 1$, q -ti kvantil razdiobe F je broj x_q koji zadovoljava relaciju $F(x_q) = q$ (dakle, do $x = x_q$ je kumuliran q -ti dio cijele razdiobe). Može se pokazati: ako se dvije funkcije razdiobe F i G ne razlikuju oblikom (nego eventualno samo položajem i skalom), tada za q -te kvantile od F i G vrijedi da je $y_q = a + b x_q$, za svaki q . Tada crtež kvantila (x_q, y_q) , zvan i Q-Q crtež, ima oblik pravca.

Kada izaberemo neku teorijsku razdiobu F (npr. eksponencijalnu, normalnu ili sl.), usporedit ćemo procijenjeni F_n^{-1} i teorijski F^{-1} pomoću parova kvantila za vrijednosti $q = (i-0.5)/n$ ($i = 1, 2, \dots, n$) crtanjem točaka:

$$(X_i, F^{-1}((i-0.5)/n)),$$

budući da je po definiciji $(i-0.5)/n$ kvantil od \tilde{F}_n jednak $X_{(i)}$, dok je isti kvantil od F jednak $F^{-1}((i-0.5)/n)$.

Pri tome se inverz $F^{-1}((i-0.5)/n)$ za neke razdiobe (uniformna, eksponencijalna) može dobiti u obliku formula, a za neke razdiobe ga treba izračunati numeričkim metodama (gama, normalna).

Na slici 11.15. demonstriran je način izbora kontinuirane teorijske razdiobe pomoću crteža vjerojatnosti.

Prednost ove metode je u općenitosti, lakoći interpretacije i nepostojanju zahtjeva za grupiranjem podataka. Nedostaci su, naprimjer, teškoće pri invertiranju funkcija razdiobe. Više o ovoj metodi može se naći u (Lewis i Orav, 1989).

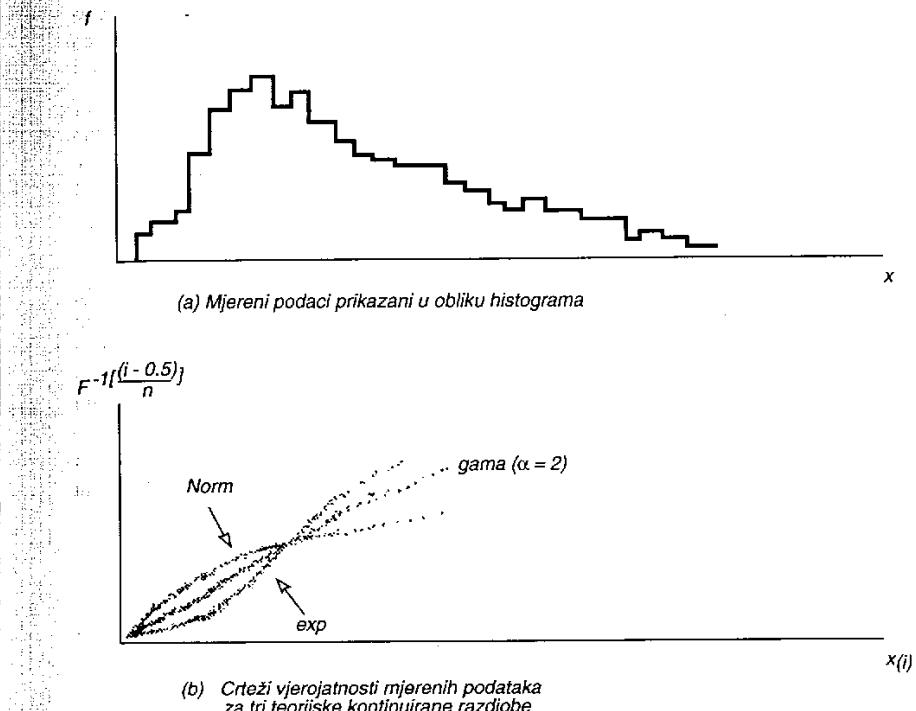
11.4.2. Diskretnе razdiobe

Kod diskretnih razdioba koriste se slične heuristike kao kod kontinuiranih razdioba. Q-Q crtež vjerojatnosti nije općenito moguć, jer $F^{-1}(y)$ nije uvijek dobro definiran kod diskretnih razdioba.

(1) Točkaste statistike

Omjer $\tau = \text{Var}(X)/E(X)$ može se iskoristiti za razlikovanje razdioba, a za procjenu od τ može služiti (Law i Kelton, 1982)

$$\hat{\tau}(n) = \frac{s^2(n)}{\bar{X}(n)}$$



Slika 11.15. Izbor kontinuirane teorijske razdiobe pomoću crteža vjerojatnosti

Kao i kod kontinuiranih razdioba, ova procjena ne određuje oblik razdiobe na jedinstven način.

(2) Linijski grafovi

Funkcije vjerojatnosti diskretnih podataka mogu se vrlo jednostavno procijeniti pomoću proporcija vrijednosti slučajne varijable, koje se uspoređuju s teorijskim razdiobama vjerojatnosti prema sličnosti oblika.

Linijski grafovi su slični histogramima, ali kod njih nije potrebno grupirati podatke.

11.5. PROCJENA PARAMETARA RAZDIOBA

Nakon izbora porodice razdiobe potrebno je napraviti procjenu parametara odabранe razdiobe. Procjena se radi istim podacima s nezavisnom identičnom razdiobom (NIR) X_1, X_2, \dots, X_n koji su služili za traženje porodice razdiobe. Dakle, traži se procjena vrijednosti nepoznatih parametara na temelju ulaznih podataka.

Procjena je numerička funkcija podataka. Najčešće korištena metoda procjene je tzv. *procjena maksimalne vjerodostojnosti* (PMV). Ova metoda ima jasno intuitivno tumačenje, ima dobra svojstva i važna je u potvrđivanju hi-kvadrat testa (Law i Kelton, 1982).

Ideju PMV metode objasniti ćemo na slučaju *diskretnih slučajnih varijabli*. Neka smo izabrali porodicu diskretnih razdioba koja ima jedan parametar θ čija vrijednost se traži, i neka je $p_\theta(x)$ odgovarajuća funkcija vjerojatnosti. Uz dane izmjerene podatke X_1, X_2, \dots, X_n definiramo funkciju vjerodostojnosti:

$$L(\theta) = p_\theta(X_1) p_\theta(X_2) \dots p_\theta(X_n).$$

Dakle, $L(\theta)$ je složena funkcija vjerojatnosti (budući da su podaci nezavisni), tj. vjerojatnost dobivanja svih opaženih podataka ($i X_1$ i $X_2 \dots$ i X_n) ako nepoznati parametar ima vrijednost θ . PMV procjena nepoznate vrijednosti od θ , koju označavamo sa $\hat{\theta}$, ona je vrijednost od θ koja *maksimalizira* $L(\theta)$. Dakle, $L(\hat{\theta}) \geq L(\theta)$ za sve moguće vrijednosti od θ , tj. $\hat{\theta}$ daje najveću složenu vjerojatnost za dobivanje svih opaženih podataka.

Za *kontinuiranu slučajnu varijablu* PMV se definira analogno. Ako je $f_\theta(x)$ pretpostavljena funkcija vjerojatnosti za odabranu porodicu razdioba, funkcija vjerodostojnosti je :

$$L(\theta) = f_\theta(X_1) f_\theta(X_2) \dots f_\theta(X_n).$$

PMV procjena $\hat{\theta}$ od θ ona je vrijednost θ koja *maksimalizira* $L(\theta)$.

Ponekad se do procjene $\hat{\theta}$ može doći traženjem uvjeta za maksimum u matematičkoj analizi.

Tako je npr. za *eksponencijalnu* razdiobu:

$$\theta = \beta (\beta > 0), f_\beta(x) = \frac{1}{\beta} e^{-\frac{x}{\beta}} \quad \text{za } x \geq 0$$

funkcija vjerodostojnosti:

$$L(\beta) = \left(\frac{1}{\beta} \right)^n e^{-\frac{x_1}{\beta}} \cdot \left(\frac{1}{\beta} \right)^n e^{-\frac{x_2}{\beta}} \dots \left(\frac{1}{\beta} \right)^n e^{-\frac{x_n}{\beta}} = \beta^{-n} \exp \left(- \left(\frac{1}{\beta} \right) \sum_{i=1}^n X_i \right)$$

Dakle, traži se vrijednost od β koja maksimalizira $L(\beta)$ za sve $\beta > 0$. Uvođenjem logaritamske funkcije vjerodostojnosti $l(\beta) = \ln L(\beta)$ možemo iz relacije $dl(\beta)/d\beta = 0$ dobiti procjenu za :

$$\hat{\beta} = \frac{1}{n} \sum_{i=1}^n X_i = \bar{X}(n)$$

Može se pokazati da je to upravo maksimum, tj. da je $(d^2 l / d\beta^2)_{\beta=\hat{\beta}} < 0$.

PMV-procjena parametara za neke razdiobe može se provesti analognom procedurom kao i za eksponencijalnu razdiobu, ali za neke razdiobe se ili ne može koristiti funkcija vjerodostojnosti (npr. za uniformnu razdiobu) ili rješavanje $dL/d\theta = 0$ ne ide algebarskim putem, već se moraju primijeniti numeričke metode (npr. kod gamma-razdiobe i Weibull razdiobe). Takve se procjene često prikazuju u tabelarnom obliku. U paragrafu 11.2. dane su procjene PMV-metodom za neke od parametara različitih kontinuiranih i diskretnih razdioba. Proširenje funkcije vjerodostojnosti na slučaj

razdiobe s većim brojem parametara je obično jednostavno, ali je traženje procjene parametara kod maksimaliziranja takve funkcije vjerodostojnosti najčešće komplikirano.

11.6. TESTOVI SLAGANJA

Nakon izbora porodice razdiobe i procjene parametara razdiobe, potrebno je ispitati da li je *dobivena prilagođena* razdioba u skladu s opaženim podacima X_1, X_2, \dots, X_n , odnosno jesu li ti podaci uzorak iz prilagođene razdiobe s funkcijom razdiobe F .

Može se postaviti nul-hipoteza

$$H_0 : X_i \text{-ovi su NIR slučajne varijable s funkcijom razdiobe } \hat{F}$$

Ovakav test hipoteze zove se *test slaganja*, jer on testira kako će dobro odabrana prilagođena razdioba slaže s opaženim podacima. Ovdje nemogućnost odbacivanja H_0 ne znači prihvatanje, već test hipoteze H_0 treba shvatiti kao pokušaj sistematskog otkrivanja većih neslaganja između podataka i prilagođene razdiobe. Treba imati na umu i to da veći broj testova slaganja ima malu snagu (npr. hi-kvadrat i Kolmogorov-Smirnov testovi), tj. nisu jako osjetljivi na finija neslaganja između podataka i prilagođene razdiobe.

Opisat ćemo nekoliko osnovnih testova slaganja.

11.6.1. Neformalna vizualna procjena

Kao približna neformalna ocjena slaganja može poslužiti vizualna usporedba funkcije vjerojatnosti odabrane razdiobe s histogramom podataka u kontinuiranom slučaju, odnosno funkcije razdiobe s linijskim grafom (vidi prethodni paragraf) u diskretnom slučaju.

Tako se u *kontinuiranom slučaju* uspoređuje histogram frekvencije podataka s *funkcijom* $g(x) = \Delta b \hat{f}(x)$, gdje je Δb širina intervala histograma a $\hat{f}(x)$ funkcija vjerojatnosti prilagođene razdiobe dobivene u prethodnom postupku modeliranja podataka. Uspoređuje se tako da se na istom crtežu prikažu obje funkcije. Druga mogućnost usporedbe je usporedba frekvencija histograma po intervalima s izračunatom *očekivanom frekvencijom* opažanja koja bi pala u taj interval ako je prilagođena razdioba prava razdioba.

U *diskretnom slučaju* vizualno se uspoređuju usporedno nacrtane funkcije vjerojatnosti prilagođene razdiobe s linijskim grafom podataka.

11.6.2. Hi-kvadrat test

Hi-kvadrat test zahtijeva podjelu mjerenja X_1, X_2, \dots, X_n u k uzastopnih intervala $[a_0, a_1], [a_1, a_2], \dots, [a_{k-1}, a_k]$. Pri tome je moguće da je $a_0 = -\infty$ i $a_k = +\infty$. Neka je :

N_j frekvencija opažanja X_i u j -tom intervalu $[a_{j-1}, a_j]$,

E_j očekivana frekvencija za j -ti interval

gdje je, $E_j = n p_j$, a

p_j je očekivana relativna frekvencija za j -ti interval za prilagođenu razdiobu (ako bi ona bila istinita razdioba):

$$p_j = \int_{a_{j-1}}^{a_j} \hat{F}(x) dx, \quad \text{za kontinuirani slučaj}$$

$$p_j = \sum_{\{i: a_{j-1} \leq x_i \leq a_j\}} \hat{p}(x_i), \quad \text{za diskretni slučaj}$$

$$\text{Statistika testa je: } \chi^2 = \sum_{j=1}^k \frac{(N_j - E_j)^2}{E_j}$$

Može se očekivati da je χ^2 mali ako je prilagodavanje dobro. Hipoteza H_0 se odbacuje ako je

$$\chi^2 \geq \chi^2_{k-1, 1-\alpha}$$

gdje je $\chi^2_{k-1, 1-\alpha}$ gornja $(1-\alpha)$ kritična točka za hi-kvadrat razdiobu sa $k-1$ stupnjem slobode. Pri tome se preporučuje da izbor intervala bude takav da je približno: $p_1=p_2=\dots=p_k$, te da vrijedi $n p_j > 5$ za gotovo sve j -ove. Također se preporučuje da je broj intervala $k < 40$.

Iako je hi-kvadrat test osjetljiv na izbor intervala i općenito je točan samo za velike uzorke, ipak se dosta koristi jer se može primijeniti na bilo koju razdiobu vjerojatnosti.

11.6.3. Kolmogorov-Smirnov test

Kolmogorov-Smirnov test uspoređuje funkcije razdiobe. Pri tome on ne traži grupiranje podataka i točan je za proizvoljnu veličinu uzorka n ; međutim, razdioba mora biti kontinuirana (ili grupirana diskretna razdioba) i parametri razdiobe ne bi smjeli biti procijenjeni iz podataka (iako se to često radi) već nezavisno prepostavljeni.

Neka je:

$F_n(x)$ empirijska funkcija razdiobe dobivena iz podataka X_1, X_2, \dots, X_n ,

$F(x)$ prilagođena funkcija razdiobe.

Pri tome ćemo empirijsku funkciju razdiobe izračunati iz:

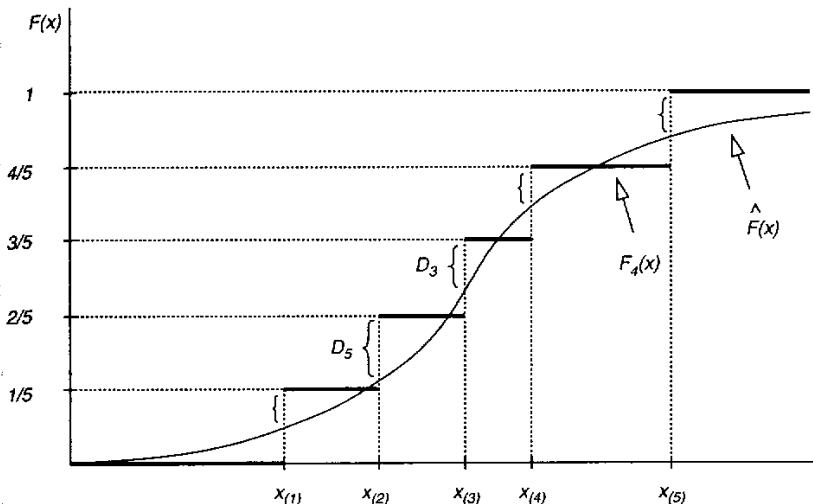
$$F_n(x) = \frac{1}{n} (\text{broj } X_i \text{-ova } < x), \quad \text{za sve realne brojeve } x.$$

$F_n(x)$ je funkcija sa skokom vrijednosti takva da je $F_n(X_{(i)}) = i/n$ za svaki $i=1, 2, \dots, n$.

Mjeru bliskosti razdioba F_n i \hat{F} opisat ćemo *najvećom udaljenošću* D_n između $F_n(x)$ i $\hat{F}(x)$ za sve vrijednosti x :

$$D_n = \max (|F_n(x) - \hat{F}(x)|).$$

Primjer za nalaženje D_n prikazan je na slici 11.16.



Slika 11.16. Geometrijska interpretacija Kolmogorov - Smirnov testa

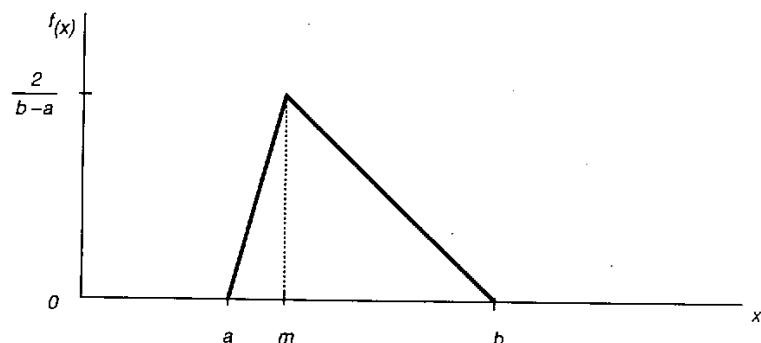
Velike vrijednosti D_n su indikacija za slabo slaganje. Hipoteza H_0 se odbacuje ako je D_n veći od neke konstante $d_{n, 1-\alpha}$, gdje je α specificirani nivo testa. Vrijednosti za $d_{n, 1-\alpha}$ su tabelirane, te ovise o načinu na koji je prilagođena razdioba specificirana i o tome koja je to porodica razdioba.

11.7. IZBOR RAZDIOBE U NEDOSTATKU PODATAKA

Ponekad nije moguće prikupiti podatke o slučajnoj varijabli sistemu, npr. kada sistem koji modeliramo još ne postoji. Opisat ćemo jednu jednostavnu heurističku metodu izbora razdiobe vjerojatnosti koja se u takvim slučajevima koristi (Law i Kelton, 1982).

Pretpostavimo da je riječ o kontinuiranoj slučajnoj varijabli, i recimo da ona opisuje trajanje posluživanja ili neke druge aktivnosti. U prvom koraku pokušat ćemo identificirati interval $[a, b]$ (gdje su a i b realni brojevi, te $a < b$) za koji se vjeruje da predstavlja područje u kojem se mogu nalaziti sve moguće vrijednosti slučajne varijable. *Subjektivna* procjena za a i b može se dobiti od eksperata koji će ocijeniti najoptimističnije (a) i najpesimističnije (b) vrijeme trajanja aktivnosti koje opisuje slučajna varijabla.

U drugom se koraku pokušava u intervalu $[a, b]$ definirati odgovarajuća funkcija vjerojatnosti. U *metodi trokuta* koju opisuјemo eksperți će dati i svoju procjenu najvjerojatnijeg vremena m potrebnog da se izvrši aktivnost. Pretpostavlja se da slučajna varijabla X ima trokutnu razdiobu, prikazanu na slici 11.17.



Slika 11.17. Trokutna razdioba (za izbor razdiobe u nedostatku podataka)

Osnovna svojstva trokutne razdiobe su:

Funkcija vjerojatnosti:

$$f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(m-a)}, & \text{ako } a \leq x \leq m \\ \frac{2(b-x)}{(b-a)(b-m)}, & \text{ako } m < x \leq b \end{cases}$$

Funkcija razdiobe:

$$\begin{aligned} F(x) &= 0 && \text{ako } x < a \\ &= \frac{(x-a)^2}{(b-a)(m-a)} && \text{ako } a \leq x \leq m \\ &= 1 - \frac{(b-x)^2}{(b-a)(b-m)} && \text{ako } m < x \leq b \\ &= 1 && \text{ako } b < x \end{aligned}$$

Sredina : $\frac{1}{3}(a+b+m)$

Varijanca : $\frac{1}{18}(a^2 + b^2 + m^2 - ab - am - bm)$

11.8. ANALIZA VEZA MEĐU PODACIMA

Ulažni podaci za simulacijske modele mogu biti povezani, i tada se vrsta veze među njima treba odrediti i uzeti u obzir u modelu. Tako npr. vrijeme posluživanja može ovisiti o karakteristikama entiteta koji se poslužuje ili o dužini repa pred mjestom posluživanja. Važno je da se takve ovisnosti opišu u simulacijskom modelu jer one

mogu utjecati na rezultate izvođenja simulacijskih eksperimenata. Najčešće korištena tehnika za prikazivanje veze među varijablama je regresijsko modeliranje (Banks i Carson, 1984; Kleijnen, 1987).

11.8.1. Jednostavna linearna regresija

Pokazat ćemo osnovne korake regresijskog modeliranja za jednostavnu linearnu regresiju.

(1) Postavljanje regresijskog modela

Pretpostavimo da tražimo vezu između slučajnih varijabli x i y , i da je ta veza linearna. Tu vezu možemo prikazati jednostavnim linearnim regresijskim modelom ovako:

$$y = \beta_0 + \beta_1 x + e,$$

gdje je e slučajna greška sa sredinom nula i s konstantnom variancom.

(2) Procjena parametara modela

Parametre modela β_1 i β_2 možemo procijeniti na temelju dobivenih očekivanja ulaznih podataka. Ako imamo n parova očekivanja $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, tada za procjenu možemo upotrijebiti *metodu najmanjih kvadrata*. Individualna očekivanja možemo opisati relacijama:

$$y_i = \beta_0 + \beta_1 x_i + e_i, \quad i=1,2,\dots,n,$$

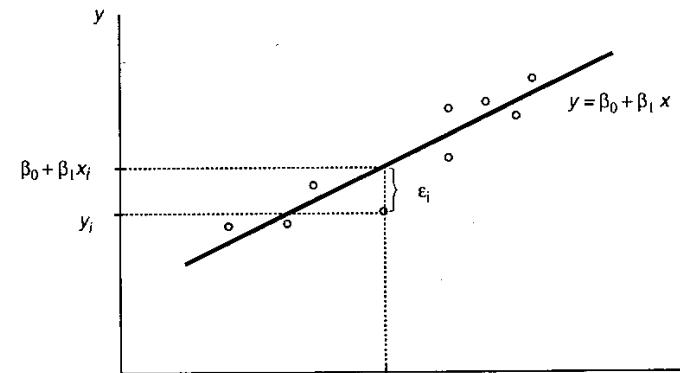
gdje se za e_1, e_2, \dots, e_n pretpostavlja da su nekorelirane slučajne varijable. Slučajne greške e_i su dakle:

$$e_i = y_i - (\beta_0 + \beta_1 x_i),$$

gdje su

$$y_i, \quad \text{mjerena vrijednost varijable } y$$

$\beta_0 + \beta_1 x_i$, očekivane vrijednosti varijable y prema regresijskome modelu.



Slika 11.18. Veza među elementima jednostavnoga linearnog regresijskog modela

Na slici 11.18. vidi se veza među elementima jednostavnog linearog regresijskog modela.

Procjena vrijednosti parametara regresijskog modela pomoću *metode najmanjih kvadrata* traži minimalizaciju sume kvadrata slučajnih grešaka e_i :

$$L = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n [y_i - (\beta_0 + \beta_1 x_i)]^2$$

Jednostavni linearni regresijski model obično se pretvara u oblik:

$$y_i = \beta'_0 + \beta'_1 (x_i - \bar{x}) + e_i$$

gdje su

$$\beta'_0 = \beta_0 + \beta_1 \bar{x}, \quad \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

Tada se traženjem minimuma izraza L dobiva:

$$\hat{\beta}'_0 = \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i.$$

$$\hat{\beta}'_1 = \frac{\sum_{i=1}^n y_i (x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \hat{\beta}'_0 + \hat{\beta}'_1 \bar{x}.$$

(3) Testiranje značajnosti regresijskog modela

Prije korištenja dobivene veze među varijablama potrebno je testirati značajnost postavljenog regresijskog modela. Prikazat ćemo jedan od načina testiranja, koji pretpostavlja da greške e_i imaju normalnu razdiobu.

Postavimo alternativne hipoteze:

$$H_0 : \beta_1 = 0 \quad (\text{nema veze između } y \text{ i } x)$$

$$H_1 : \beta_1 \neq 0 \quad (\text{ima veze između } y \text{ i } x)$$

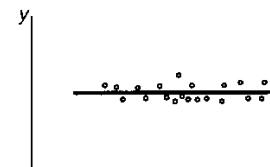
Test značajnosti može se provesti korištenjem statistike:

$$t_0 = \frac{\hat{\beta}_1}{\sqrt{\frac{MSE}{S_{xx}}}}$$

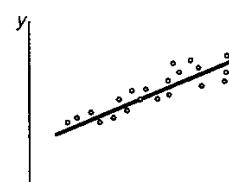
gdje je
 $MSE = \frac{1}{n-2} \sum_{i=1}^n e_i^2$ nepristrasna procjena od $\text{Var}(e_i)$
 $e_i = y_i - \bar{y}$

$$S_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2$$

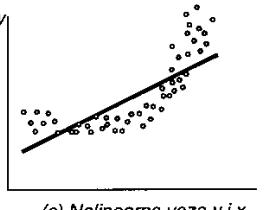
Statistika t_0 ima studentovu t-razdiobu s $(n-2)$ stupnja slobode. Nul hipoteza H_0 se odbacuje ako je $|t_0| > t_{\alpha/2, n-2}$



(a) Hipoteza $H_0 : \beta_1 = 0$ ne može se odbaciti



(b) Linearna veza y i x



(c) Nelinearna veza y i x

Hipoteza $H_0 : \beta_1 = 0$ je odbacena

Slika 11.19. Primjeri regresijske analize

Na slici 11.19a prikazan je slučaj kada se ne može odbaciti hipoteza H_0 , tj. da varijabla x ima malo značenje u objašnjavanju varijabiliteta varijable y. Na slici 11.19b prikazan je slučaj odbacivanja hipoteze H_0 , kada je linearna veza između y i x zadovoljavajuća. Na slici 11.19c prikazan je također slučaj odbacivanja H_0 , ali se vidi da je nužno, iako je linearna regresija značajna, isprobati nelinearni model veze među y i x :

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots + e$$

Posljednji primjer jasno govori o tome kako pri modeliranju nije razumno prihvatiću dobivene vrijednosti koje daje model bez pokušaja da se one vizualno prikažu. Vizualna i intuitivna kontrola rezultata (a nekada i prepostavki) može otkriti pogreške u pristupu, ili dati ideje za drukčiji pristup modeliranju, koji same vrijednosti ne pokazuju.

Osim ovim testom, često se F-testom ispituje slaganje dobivenog regresijskog modela (s njegovim procijenjenim parametrima) s izmjerenim podacima. Također se preporučuje stvaranje određenog broja novih opažanja koji se uspoređuju s predviđanjima modela korištenjem t-testa. Ako se nova opažanja slažu s regresijskim modelom, ona se mogu iskoristiti i za proširenje baze opažanja na temelju koje se izračunavaju nove vrijednosti parametara regresijskog modela.

Napokon, treba reći da se regresijskom analizom treba *oprezno koristiti*, jer je na temelju nje lako napraviti pogrešne zaključke. Na primjer, ne preporučuje se da se linearna veza regresijskog modela extrapolira izvan opsega originalnog područja podataka gdje možda ne vrijedi dobivena ovisnost. Također treba pažljivo analizirati da li dobivena ovisnost zaista pokazuje na uzročnu povezanost varijabli ili je ta veza među varijablama prividna.

11.8.2. Uključivanje međudjelovanja nezavisnih varijabli u model

Pretpostavimo da model uključuje jednu zavisnu varijablu y i tri nezavisne varijable x_1 , x_2 i x_3 . Tada jednostavni linearni regresijski model ima oblik:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + e_i \quad , \quad i=1,2,\dots,n$$

gdje je

i , r. br. opažanja,

e_i , slučajna fluktuacija.

Općenitiji model uključuje i međudjelovanja ulaznih varijabli:

$$\begin{aligned} y_i = & \beta_0 + (\beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}) + \\ & + (\beta_{12} x_{i1} x_{i2} + \beta_{13} x_{i1} x_{i3} + \beta_{23} x_{i2} x_{i3}) + e_i \quad , \quad i=1,2,\dots,n \end{aligned}$$

gdje parametri modela β_{12} , β_{13} , β_{23} označavaju međudjelovanje nezavisnih varijabli regresijskog modela.

Ako ovakav regresijski model nije odbačen u testiranju, to znači sljedeće: ako su x_1 , x_2 i x_3 i y *ulazne* varijable, tada se y ne mjeri *nezavisno*, nego se na temelju x_1 , x_2 i x_3 izračunava iz dobivenog regresijskog modela.

Ako u dalnjem testiranju regresijski model nije prihvaćen, moguće mu je dodavati i trofaktorska međudjelovanja (iako je takav model teško interpretirati), pokušati transformacije varijabli (npr. $1/x_i$ umjesto x_i koja daje nelinearnost u y), ili smanjiti područje vrijednosti varijabli u kojem model vrijedi.

11.8.3. Koristenje regresijskih modela ulaznih podataka u simulaciji

Ako je regresijski model prihvaćen kao značajan, tada se on može koristiti u simulaciji. Ako se modeliraju veze među ulaznim podacima, ti se modeli mogu koristiti kod generiranja vrijednosti ulaznih varijabli u simulacijskim eksperimentima.

Generiranje neke vrijednosti y' varijable y izvodi se iz relacije

$$y' = \hat{y} + e,$$

gdje je

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x, \text{ procjena srednje vrijednosti od } y \text{ za dani } x$$

e imaju normalnu razdiobu sa sredinom nula i procijenjenom varijancom

$$MS_E = \frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y})^2$$

Regresijski model ulaznih podataka može se koristiti na dva načina za generiranje podataka u simulacijskim eksperimentima:

(1) Izmjerena je neka vrijednost ulazne varijable x (npr. broj stvari koje je kupac što čeka pred blagajnom u samoposluživanju kupio). Na temelju te vrijednosti izračuna se y (npr. prosječno vrijeme posluživanja na blagajni za taj broj kupljenih stvari x), i generira se vrijednost slučajne greške e s normalnom razdiobom. Odatle se dobije vrijednost zavisne varijable $y' = \hat{y} + e$ (vrijednost iz razdiobe vremena posluživanja).

(2) Vrijednost ulazne varijable x (npr. broj kupljenih stvari u samoposluživanju) generira se na slučajni način iz odgovarajuće razdiobe. Zatim se ponovi prethodni postupak za izračunavanje vrijednosti zavisne varijable y (to može biti npr. vrijeme posluživanja kupca na blagajni).

12. PLANIRANJE SIMULACIJSKIH EKSPERIMENTATA

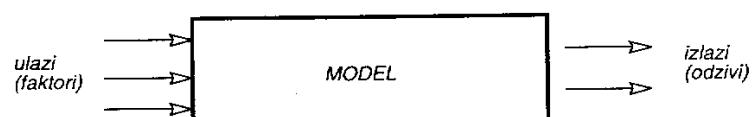
Izvođenje simulacijskih eksperimenata mora se planirati, kako bi bilo što racionalnije te kako bi se dobile što korektnije i preciznije informacije koje omogućuju adekvatnu analizu sistema. Planiranje simulacijskih eksperimenata uključuje dizajn eksperimenata koji treba omogućiti da se unaprijed izaberu konfiguracije sistema što će se simulirati, te tehnike za redukciju varijanci kojima je cilj da uz jednako ukupno vrijeme izvođenja simulacijskih eksperimenata što više smanje varijancu izlaznih varijabli modela. U ovom poglavlju opisali najvažnije i najčešće korištene metode dizajna simulacijskih eksperimenata i tehnike redukcije varijance.

12.1. OSNOVNE IDEJE PLANIRANJA SIMULACIJSKIH EKSPERIMENTATA

Simulacijski eksperimenti u pravilu trebaju odrediti kako različiti faktori utječu na rad modela (odnosno sistema koji model predstavlja). Faktori mogu biti različiti: broj blagajni u samoposluživanju, vrsta izabranih strojeva u proizvodnom pogonu, disciplina posluživanja, očekivana dinamika dolazaka zahtjeva za posluživanje itd. Karakteristike sistema koje se ispituju mogu biti: vrijeme čekanja na posluživanje, dužine repova čekanja, propusnost sistema, iskorištenost opreme itd.

Osnovni ciljevi planiranja simulacijskog eksperimenta jesu: (1) određivanje načina kojim će se u eksperimentu izabirati različite varijante sistema (tj. kombinacije faktora) te (2) određivanje načina kako će se generirati slučajni uzorci da bi se željena informacija dobila s minimalnim vremenom izvođenja simulacije.

Elementi eksperimentiranja sa simulacijskim modelom prikazani su simbolički na slici 12.1, na kojoj vidimo da simulacijski eksperiment uključuje podvrgavanje modela ulazima (faktorima) na različitim nivoima (npr. različite vrijednosti srednjeg vremena obrade na stroju) i dobivanje različitih efekata na izlazima (odzivima) koje simulacijski eksperiment daje (npr. različita vremena čekanja u repu). Tako simulacijski eksperimenti mogu dati odgovor na pitanje kako npr. korištenje različitih tipova strojeva za pranje utječe na ukupan dnevni kapacitet pralnice rublja i na prosječno vrijeme čekanja da se rublje opere.



Slika 12.1. Prikaz elemenata simulacijskog eksperimentiranja

Faktore koji utječu na rad modela možemo podijeliti na dvije osnovne grupe:

- parametri koji definiraju *konfiguraciju sistema*, koja uključuje i izbor tehnologije te organizaciju rada sistema (npr. broj šaltera u banci, vrste uređaja korištenih na šalteru, pravila davanja prioriteta za različite tipove klijenata)
- *slučajni uzoreci* iz različitih razdioba vjerojatnosti (npr. dinamika dolazaka klijenata, vremena posluživanja).

Planiranje simulacijskih eksperimenata povezano za izbor konfiguracija sistema zove se *dizajn simulacijskih eksperimenata*, a ono povezano za izbor slučajnih uzoraka naziva se *tehnikama za redukciju varijance*. Dok je dizajn eksperimenata poznat i razvijen u sklopu statistike neovisno o simulaciji, tehnike za redukciju varijance specifične su upravo za simulacijske eksperimente.

12.2. DIZAJN SIMULACIJSKIH EKSPERIMENTA

Cilj dizajna simulacijskih eksperimenata jest da unaprijed izabere koje će se konfiguracije sistema simulirati, i to tako da se na što efikasniji način dobiju tražene informacije o relevantnim karakteristikama sistema i o značenju utjecaja pojedinačnih faktora i njihovih kombinacija na performanse sistema.

Ulagani parametri i pretpostavke o načinu rada sistema nazivaju se *faktori*. Faktori mogu biti kvantitativni i kvalitativni:

- *kvantitativni faktori* su oni koji prirodno poprimaju numeričke vrijednosti (npr. broj blagajni u samoposluživanju, maksimalno tolerirano vrijeme čekanja na benzinskoj crpkici)
- *kvalitativni faktori* najčešće opisuju strukturalne pretpostavke modela koje prirodno nemaju numeričke vrijednosti (npr. disciplina čekanja u repu: FIFO, LIFO, po prioritetu; način ukrcavanja u avion s podjelom mjesta prije ukrcavanja ili bez nje).

Gledano sa stajališta simulacijskih eksperimenata, faktori se mogu podijeliti na kontrolabilne i nekontrolabilne:

- *kontrolabilni faktori* (variable odlučivanja) oni su elementi na koje se u modelu sistema može utjecati (npr. broj šaltera u banci, disciplina čekanja u repu)
- *nekontrolabilni faktori* su oni na koje se u stvarnom sistemu ne može utjecati (npr. dinamika dolazaka automobila na benzinsku crpkicu, prosječan broj transakcija koje traži mušterija u banci), a koji također utječu na rezultate rada sistema.

Vrijednosti faktora zovu se *nivoi faktora* (npr. nivoi broja šaltera u banci mogu biti 4, 5 i 6). Kombinacija faktora na određenim nivoima zove se *tretman*. Zavisne varijable modela čije vrijednosti dobivamo kao rezultat simulacijskog eksperimenta nazivamo *odzivima* (to su npr. vremena čekanja, propusnost sistema, iskorištenje opreme).

Izbor koji će parametri i strukturalne pretpostavke biti konstante u modelu a koji će biti promjenljivi eksperimentalni faktori postavlja se u samom cilju simulacijske studije i ovisi o prirodi problema koji se simulacijom rješava.

Posebna važnost dizajna simulacijskih eksperimenata je to da se umjesto nesistematičnog *ad hoc* izvođenja simulacijskih eksperimenata s različitim alternativama si-

stema priđe planiranju eksperimenata *unaprijed*, i to na *sistematičan* način, koristeći se poznatim karakteristikama faktora određenog simulacijskog modela. Pri tome se dizajnu eksperimenata treba prići tako da se u ranim fazama eksperimentiranja prije svega nastoji ustanoviti *koji su faktori značajni* za ponašanje sistema i *kako oni utječu na performanse sistema*. Kada je to poznato, mogu se, u principu, ispitivati i *optimalne kombinacije faktora* sa stajališta dane performanse sistema (iako se to u simulacijskim studijama prilično rijetko čini).

Prikazat ćemo osnove nekoliko najčešće korištenih metoda dizajna simulacijskih eksperimenata. Ostale su metode detaljnije opisane u (Kleijnen, 1974; Kleijnen, 1987; Law i Kelton, 1982).

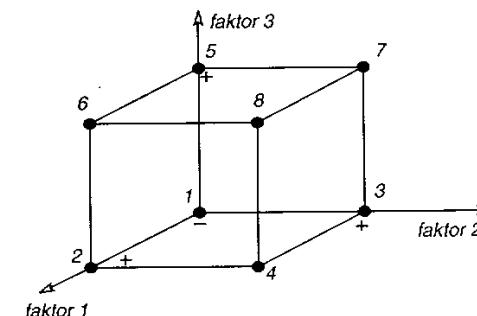
12.2.1. Potpuni 2^k faktorski dizajn

Ako model ima samo jedan faktor, simulacijski se eksperiment izvodi za svaki od nivoa tog faktora. Zbog postojanja slučajnih varijabli za svaki se nivo faktora eksperiment treba izvesti veći broj puta (dobivanje statističkog uzorka), što omogućuje dobivanje intervala pouzdanosti za odziv modela.

Ako postoji k faktora, što je i najčešće, cilj je simulacijskih eksperimenata da omoguće određivanje utjecaja kako *svakog pojedinačnog faktora*, tako i *međudjelovanja faktora* (da li efekt pojedinog faktora ovisi o nivou drugih faktora) na odzive sistema.

Najjednostavniji način na koji se može odrediti efekt pojedinog faktora na odzive sistema je da se eksperiment dizajnira tako da se nivoi svih ostalih $k-1$ faktora fiksiraju na nekom skupu vrijednosti te da se simulacijski eksperimenti izvode za svaki od nivoa odabranog faktora. Tako se dobiva utjecaj samo tog faktora na odziv (izlaz) sistema. Isti tip eksperimenata zatim se ponavlja i za sve ostale faktore. Slabost ovog pristupa je velik broj potrebnih eksperimenata i nemogućnost određivanja međudjelovanja faktora, pa se zato on i ne upotrebljava.

Ekonomičniji dizajn eksperimenta je tzv. *potpuni* ili 2^k *dizajn*, kod kojeg se svaki faktor prikazuje sa samo *dva nivoa*, a simulacijski eksperimenti se izvode za svaku od 2^k mogućih kombinacija nivoa faktora. Izbor nivoa za faktore je proizvoljan i temelji se na intuiciji. Nivoi faktora obično se označavaju —i +. U tablici 12.1. prikazan je mogući potpuni 2^3 dizajn eksperimenta sa tri faktora ($k=3$). Vrijednost O_i u tablici odgovara odzivu sistema sa i-tom kombinacijom nivoa faktora. Tablica nivoa faktora zove se i *matrica dizajna*. Grafički je prikaz tog dizajna na slici 12.2.



Slika 12.2. Grafički prikaz potpunog 2^3 faktorskog dizajna iz tablice 12.1.

Tablica 12.1. Matrica dizajna za potpuni 2^3 faktorski dizajn

R.br. kombinacije faktora	Faktor 1	Faktor 2	Faktor 3	Veličina odziva
1	-	-	-	O_1
2	+	-	-	O_2
3	-	+	-	O_3
4	+	+	-	O_4
5	-	-	+	O_5
6	+	-	+	O_6
7	-	+	+	O_7
8	+	+	+	O_8

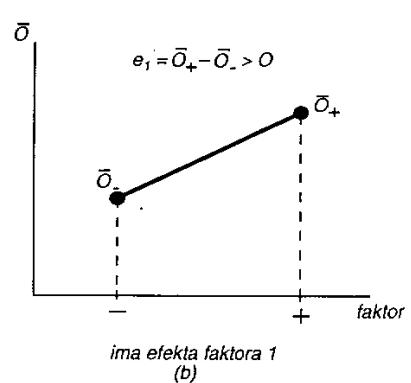
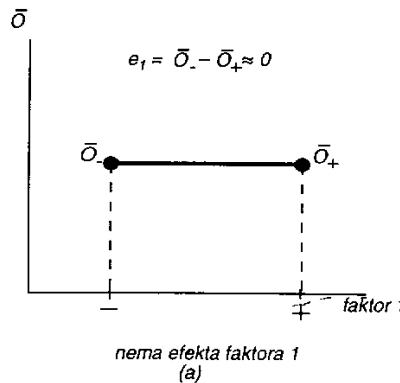
Glavni efekt faktora j je prosječna promjena u odzivu sistema zbog promjene faktora j s njegova - na njegov + nivo, uz sve ostale faktore fiksirane. Tako je za 2^3 faktorski dizajn iz tablice 12.1. glavni efekt faktora 1

$$e_1 = \frac{1}{4}[(O_2 - O_1) + (O_4 - O_3) + (O_6 - O_5) + (O_8 - O_7)],$$

jer su u kombinacijama (1,2), (3,4), (5,6) i (7,8) nivoi faktora 2 i 3 fiksni, i mijenja se samo nivo faktora 1. Slično dobivamo npr. za glavni efekt faktora 3:

$$e_3 = \frac{1}{4}[(O_5 - O_1) + (O_6 - O_2) + (O_7 - O_3) + (O_8 - O_4)].$$

Na slici 12.3a vidimo da faktor 1 ne utječe značajno na odziv O , za razliku od situacije na slici 12.3b, gdje je njegov utjecaj značajan. Sa \bar{O} su na slici prikazane prosječne vrijednosti odziva O za svaki od nivoa faktora 1. Tako su za 2^3 faktorski dizajn iz tablice 12.1:



Slika 12.3. Prosječne vrijednosti odziva za različite nivoje jednog faktora

12.2. DIZAJN SIMULACIJSKIH EKSPERIMENTA

$$\bar{O}_+ = \frac{1}{4}(O_2 + O_4 + O_6 + O_8), a$$

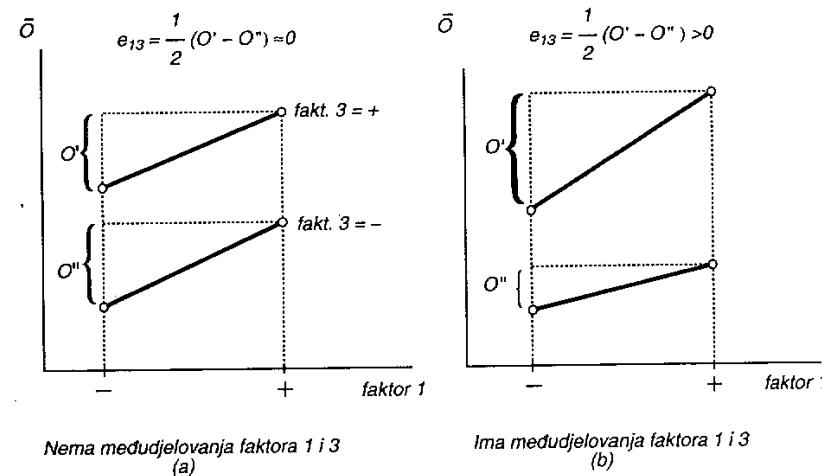
$$\bar{O}_- = \frac{1}{4}(O_1 + O_3 + O_5 + O_7).$$

Moguće je i da efekt faktora j_1 ovisi o nivou nekog drugog faktora j_2 , tj. da ti faktori međudjeluju. Stupanj međudjelovanja mjeri se dvofaktorskim efektom međudjelovanja e_{jj_2} . Taj se efekt definira kao polovica razlike između prosječnog efekta faktora j_1 kada je faktor j_2 na svojem + nivou (a svi ostali faktori su konstantni) i prosječnog efekta faktora j_1 kada je faktor j_2 na svojem - nivou. Tako je, na primjer, za 2^3 faktorski dizajn iz tablice 12.1. efekt međudjelovanja e_{13} koji mjeri stupanj ovisnosti efekta faktora 1 o nivou faktora 3:

$$e_{13} = \frac{1}{2} \left\{ \frac{1}{2} [(O_6 - O_5) + (O_8 - O_7)] - \frac{1}{2} [(O_2 - O_1) + (O_4 - O_3)] \right\}$$

Ovdje je $((O_6 - O_5) + (O_8 - O_7))/2$ prosječni efekt faktora 1 kada je faktor 3 na + nivou, a $((O_2 - O_1) + (O_4 - O_3))/2$ prosječni je efekt faktora 1 kada je faktor 3 na - nivou. Može se pokazati da su dvofaktorski efekti međudjelovanja simetrični, tj. da je $e_{jj_2} = e_{j_2 j_1}$.

Na slici 12.4a vidimo da ne postoji dvofaktorski efekt međudjelovanja faktora 1 i 3, jer faktor 3 ne utječe na promjenu prosječnog odziva zbog faktora 1, a na slici 12.4b vidimo da je dvofaktorsko međudjelovanje tih faktora dosta veliko.



Slika 12.4. Prosječna vrijednost odziva za različite nivoje dvaju faktora

Analogno je moguće definirati i višefaktorske efekte međudjelovanja.

Usporedbom dobivenih vrijednosti efekata faktora dobiva se odnos utjecaja pojedinih faktora i njihovih međudjelovanja na odziv sistema.

Primjer rezultata primjene jednog potpunog 2^2 dizajna dan je u tablici 12.2.

Tablica 12.2. Rezultati primjene jednog potpunog 2^2 faktorskog dizajna

R.br. faktora	Faktor kombinacije	Faktor	Prosječni odziv
	1	2	
1	-	-	30
2	+	-	40
3	-	+	60
4	+	+	52

Ovdje su glavni efekti faktora:

$$e_1 = \frac{1}{2}[(O_2 - O_1) + (O_4 - O_3)] = \frac{1}{2}[(40 - 30) + (52 - 60)] = \frac{10 - 8}{2} = \frac{2}{2} = 1$$

$$e_2 = \frac{1}{2}[(O_4 - O_2) + (O_3 - O_1)] = \frac{1}{2}[(52 - 40) + (60 - 30)] = \frac{12 + 30}{2} = \frac{42}{2} = 21$$

Vidimo da je glavni efekt faktora 1 zanemarive veličine, dok je glavni efekt faktora 2 dosta velik (i to pozitivnog predznaka, tj. prosječni odziv poraste kada faktor 2 prelazi sa svojeg – nivoa na + nivo).

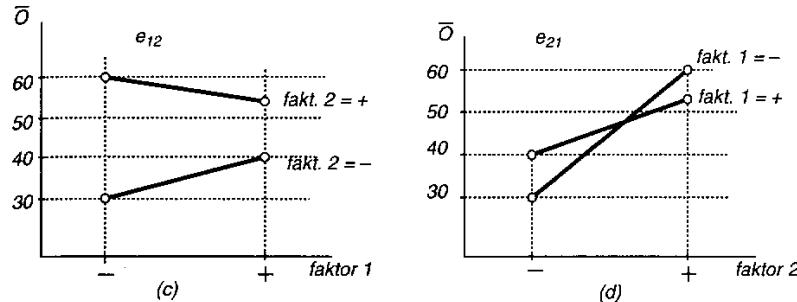
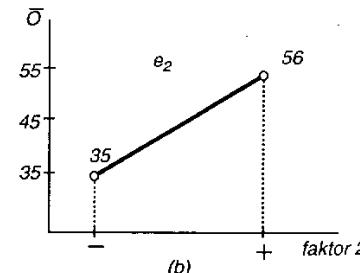
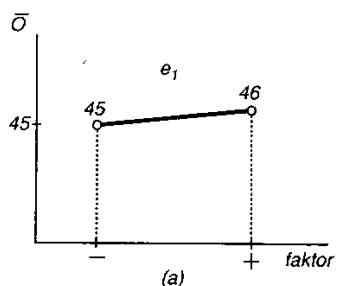
Dvofaktorski efekti međudjelovanja su:

$$e_{12} = \frac{1}{2}[(O_4 - O_3) - (O_2 - O_1)] = \frac{1}{2}[(52 - 60) + (40 - 30)] = \frac{-8 - 10}{2} = \frac{-18}{2} = -9$$

$$e_{21} = \frac{1}{2}[(O_4 - O_2) - (O_3 - O_1)] = e_{12} = -9$$

Vidimo da postoji efekt međudjelovanja faktora 1 i 2 (i to negativnog predznaka, tj. efekt svakog od faktora pada kada drugi faktor prelazi sa svojeg – nivoa na + nivo).

Odgovarajući grafovi prikaza glavnih i dvofaktorskih efekata dani su na slikama 12.5a, b, c i d.



Slika 12.5. Grafovi glavnih i dvofaktorskih efekata potpunog 2^2 dizajna iz tablice 12.2.

12.2.2. Djelomični 2^{k-p} faktorski dizajn

Za model sa k faktora potpuni 2^k faktorski dizajn ima 2^k kombinacija faktora, pa je za dobivanje slučajnih uzoraka veličine n za svaku kombinaciju potrebno izvesti ukupno $n \cdot 2^k$ simulacijskih eksperimenata. Ako npr. imamo 6 faktora tada je $2^k = 2^6 = 64$, pa je za uzorce veličine $n = 5$ potrebno izvesti ukupno $n \cdot 2^k = 5 \cdot 64 = 320$ simulacijskih eksperimenata. Budući da ti eksperimenti mogu trajati relativno dugo (vrijeme izvođenja na računalu), korisno je upotrijebiti neke od oblika dizajna eksperimenata koji zahtijevaju manji broj eksperimenata.

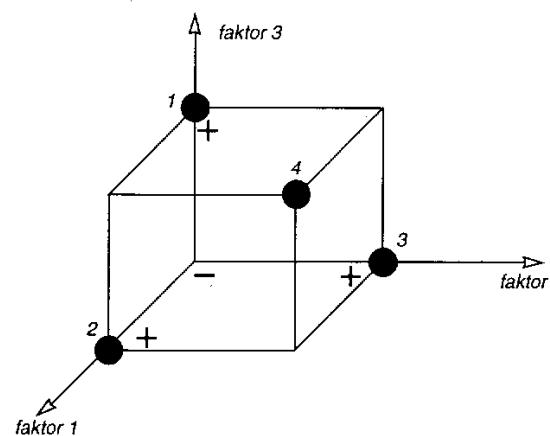
Djelomični faktorski dizajn omogućuje da se dobiju procjene glavnih efekata i efekata međudjelovanja nižeg reda *izvođenjem samo jednog dijela potpunog 2^k faktorskog dizajna*. Taj tip dizajna posebno je koristan kao prvi korak u eksperimentiranju sa velikim brojem faktora, u kojem se žele eliminirati oni faktori koji imaju mali utjecaj na odziv sistema s izvođenjem relativno malog broja simulacijskih eksperimenata. Nakon toga se, korištenjem potpunog faktorskog dizajna, može iscrpne analizirati utjecaj preostalih značajnih faktora sistema.

2^{k-p} djelomični faktorski dizajn izvodi se tako da se odabere podskup od 2^{k-p} kombinacija faktora za koje se izvode simulacijski eksperimenti. Način izbora $k-p$ kombinacija složen je problem koji ovdje nećemo opisivati. Zbog smanjenja broja kombinacija faktora uključenih u dizajn smanjuje se i broj glavnih efekata i efekata međudjelovanja koji se takvim dizajnom mogu procijeniti (o načinu izbora kombinacija koje ostaju u matrici dizajna ovisi koji su efekti međusobno povezani i ne mogu se eksperimentima odvojiti).

Primjer djelomičnog 2^{3-1} dizajna prikazan je u tablici 12.3. Odgovarajući grafički prikaz je na slici 12.6.

Tablica 12.3. Matrica dizajna za nepotpuni 2^{3-1} faktorski dizajn

R.br. kombinacije faktora	Faktor 1	Faktor 2	Faktor 3	Prosječni odziv
1	-	-	+	O_1
2	+	-	-	O_2
3	-	+	-	O_3
4	+	+	+	O_4



Slika 12.6. Grafički prikaz nepotpunog 2^{3-1} faktorskog dizajna iz tablice 2.3.

12.2.3. Traženje optimalnih kombinacija faktora metodologijom površine odziva

Nakon pronalaženja faktora koji značajno utječe na odziv sistema (uz pomoć potpunog ili djelomičnog faktorskog dizajna) moguće je naći i optimalnu kombinaciju faktora, tj. kombinaciju faktora koja maksimalizira ili minimalizira odziv sistema. Problem se rješava grupom metoda koje se nazivaju metodologijom površine odziva (Law i Kelton, 1982).

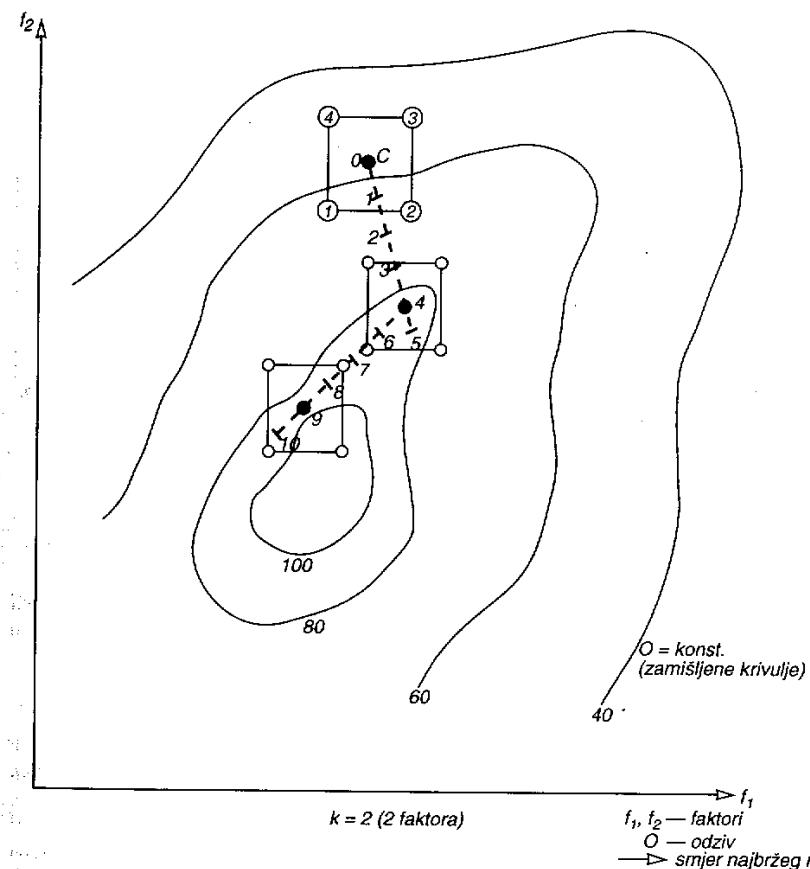
Površina odziva očekivani su odzivi sistema prikazani kao funkcija faktora. Ako postoje dva faktora ($k=2$), tada je površina odziva ploha u trodimenzionalnom prostoru. Primjer prikazan na slici 12.7. pokazuje zamišljene krivulje konstantne vrijednosti odziva (vrijednosti odziva procjene su odziva za određenu veličinu uzorka koje bi se dobile nezavisnim izvođenjem simulacijskog eksperimenta za zadane vrijednosti faktora). Konstrukcija površine odziva zahtijevala bi očito izvanredno velik broj eksperimenata, tako da procedure metodologije površine odziva nastoje naći optimum mnogo kraćim traženjem po površini odziva.

Problem traženja optimalnog rješenja je već i u determinističkom slučaju veoma složen, a prisutnost slučajnih varijabli ga još više komplicira. Stoga je kod primjene metodologije površine odziva nužan veliki oprez. Jedan od mogućih pristupa traženju optimalnih vrijednosti faktora (Myers, 1971) ilustriran je na slici 12.7. Taj pristup ima ove korake:

- (1) Izaberu se prve četiri točke potpunog 2^k faktorskog dizajna.
- (2) Izračunaju se prosječni odzivi u te četiri točke dizajna izvođenjem odgovarajućih simulacijskih eksperimenata za dane vrijednosti faktora.
- (3) Površina odziva u području tih četiri točaka aproksimira se linearnim regresijskim modelom (tj. ravninom za slučaj dva faktora):

$$E(O(f_1, f_2)) = \beta_0 + \beta_1 f_1 + \beta_2 f_2,$$

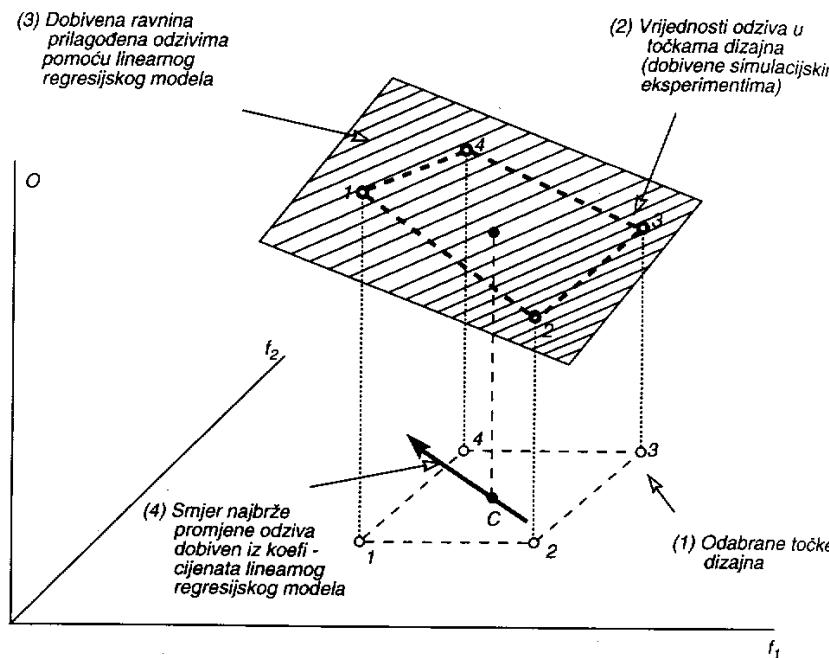
prilagođenim na četiri točke dizajna (metodom minimalnih kvadrata odstupanja od točaka dizajna).



Slika 12.7. Način rada metodologije površine odziva

(4) Pomoću tako izračunane procjene koeficijenata $\hat{\beta}_0$, $\hat{\beta}_1$ i $\hat{\beta}_2$ nalazi se smjer najbrže promjene odziva (slika 12.8).

(5) Iz centra početnog dizajna C, tj. iz odziva centra dizajna izračunanog iz dobivenog regresijskog modela, kreće se u tom smjeru najbrže promjene odziva. U uzastopnim točkama konstantnih odsječaka pravca u tom smjeru izvode se simulacijski eksperimenti koji daju vrijednosti odziva u tim točkama.



Slika 12.8. Traženje smjera najbrže promjene odziva u metodologiji površine odziva

(6) Taj se postupak nastavlja sve dok ne prestane porast odziva (ako se traži maksimum odziva), odnosno pad odziva (ako se traži minimum odziva), kao što se vidi na slici 12.9.

(7) Kada prestane porast (odnosno pad) odziva, vraćamo se na prethodnu točku i pomoću 2^k faktorskog dizajna tražimo novi smjer najbrže promjene.

(8) Proces završava izborom para vrijednosti faktora (f_1, f_2) u centru posljednjeg dizajna za koji je prilagođeni regresijski model dao plohu veoma malog nagiba, tj. veoma male promjene veličine odziva (manju od neke unaprijed odabrane vrijednosti).

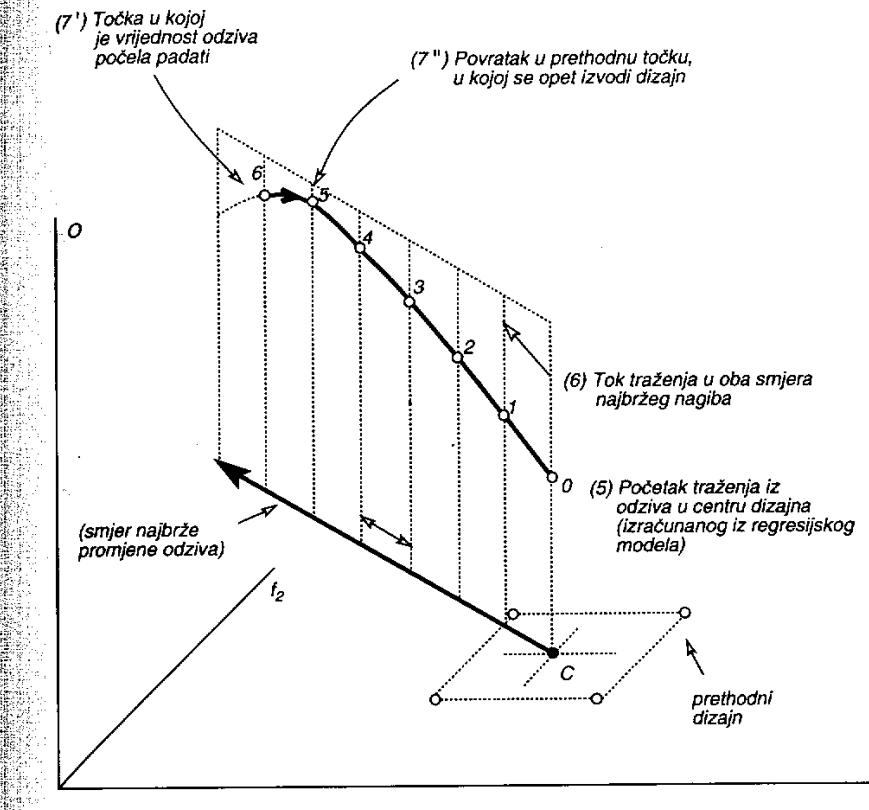
(9) U ovoj završnoj točki moguće je napraviti finiji dizajn (s većim brojem nivoa faktora) koji npr. omogućuje prilagođavanje kvadratičnog regresijskog modela na odzive točaka dizajna dobivene simulacijskim eksperimentima.

(10) Tada se optimalne vrijednosti faktora ne određuju kao koordinate točke centra dizajna, već kao koordinate optimalne veličine odziva (slika 12.10) dobivene iz kvadratičnog regresijskog modela

$$E(O(f_1, f_2)) = \beta_0 + \beta_1 f_1 + \beta_2 f_2 + \beta_{12} f_1 f_2 + \beta_{11} f_1^2 + \beta_{22} f_2^2$$

pomoću standardnih matematičkih tehniki.

U traženju optimalne kombinacije faktora značajni su ovi elementi: način izbora početne točke traženja, testiranje slaganja regresijskog modela u svakom koraku (i eventualno korištenje regresijskog modela višeg reda ako linearni ne može zadovoljiti), te mogućnost dolaska do lokalnog umjesto globalnog optimuma.

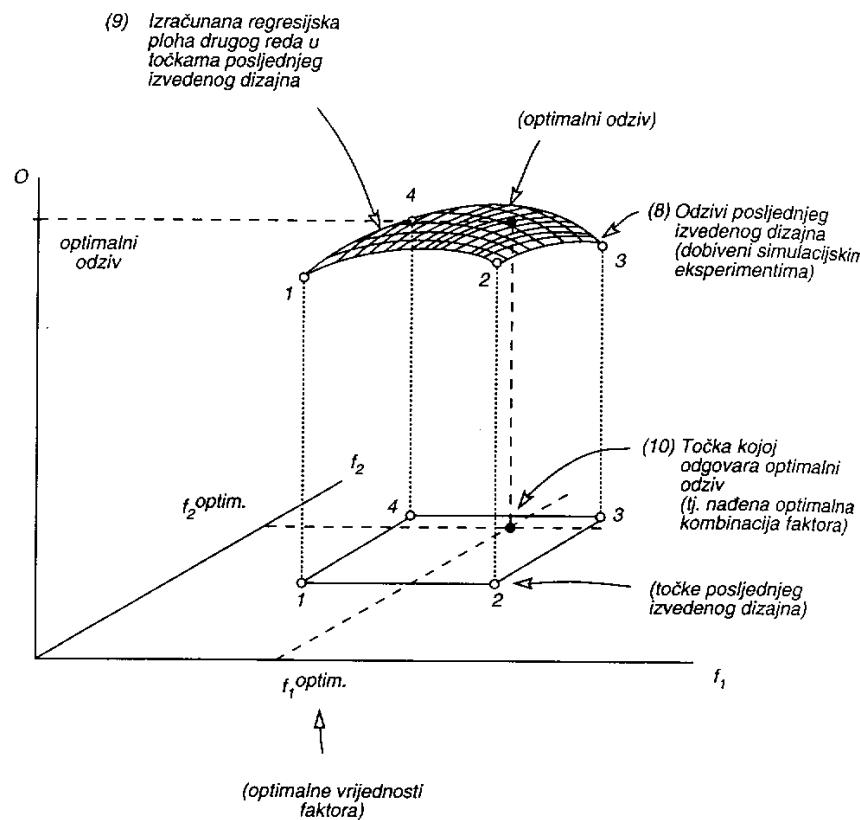


Slika 12.9. Napredovanje u smjeru najbrže promjene odziva u metodologiji površine odziva

12.2.4. Faktorski dizajn sa slučajnim varijablama

Većina sistema koje simuliramo ima slučajni karakter, pa su tako i odzivi O_i slučajne veličine. Stoga se i eksperimenti za isti tretman u dizajnu moraju izvesti više puta s različitim nizovima slučajnih brojeva, te se mora napraviti i odgovarajuća analiza rezultata simulacijskih eksperimenta. U knjizi (Law i Kelton, 1982) predloženo je da se svi tretmani dizajna ponove nezavisno n puta, i tako dobije n nezavisnih vrijednosti za svaki od faktora dizajna e_i, e_{ij} itd. Nakon toga se primjenom tehnika statističke procjene mogu dobiti procjene njihovih srednjih vrijednosti, varijanci i intervala pouzdanosti.

Najčešće primjenjivan pristup kod slučajnih eksperimentalnih dizajna je pristup pomoći analize variance, koji je opisan u knjigama (Kleijnen, 1987; Banks i Carson, 1984). Ukratko ćemo opisati osnovnu ideju takva pristupa prema tekstu (Banks i Carson, 1984), i to za primjer potpunog slučajnog eksperimentalnog dizajna s jednim faktorom.



Slika 12.10. Traženje točke optimalnog odziva u poslijednjem dizajnu (s malim nagibom), u metodologiji površine odziva

Pretpostavimo da imamo jedan faktor V (tzv. varijablu odlučivanja) koji može utjecati na varijablu odziva Y. Neka faktor V ima k različitih nivoa. Dizajn će biti potpuni slučajni dizajn ako se izvode nezavisni eksperimenti za sve nivoe faktora V, a pri tome broj ponavljanja eksperimenta za svaki nivo faktora može biti različit.

Statistički model za analizu ovog eksperimentalnog dizajna je:

$$Y_{rj} = \mu + \tau_j + \epsilon_{rj}, \quad r = 1, 2, \dots, R_j, \quad j = 1, 2, \dots, k$$

Y_{rj} je r-to opažanje varijable Y za j-ti nivo faktora,

μ je sveukupni prosjek varijable Y,

τ_j je efekt j-tog nivoa faktora,

ϵ_{rj} je slučajna varijacija odziva Y_{rj} oko sredine $\mu + \tau_j$,

R_j je broj opažanja za j-ti nivo faktora
(s različitim slučajnim brojevima).

Analiza eksperimenta može se koristiti statističkim testom hipoteze da nivoi faktora nemaju nikakav utjecaj na odziv:

$$H_0 : \tau_j = 0, \quad j = 1, 2, \dots, k$$

Pri tome se za statistički test može koristiti jednosmjerna analiza varijance, a test se sastoji od računanja F-statistike i uspoređbe dobivene vrijednosti s odgovarajućom kritičnom vrijednošću (ako je $k=2$, to je t-test). Ako hipoteza H_0 nije odbačena, može se zaključiti da faktor V nema utjecaja na odziv. Ako je hipoteza H_0 odbačena, nivoi faktora imaju utjecaja na odziv, i tada se može ispitivati relativna važnost pojedinih nivoa faktora i dobivene srednje vrijednosti. Odgovori na ta pitanja mogu se dobiti različitim testovima (testom područja, Scheffeeovim testom i sl.).

Test analize varijance temelji se na podjeli varijabilnosti odziva Y_{rj} na dva dijela, jedan uzrokovani nivoom faktora (tretmanom) i drugi uzrokovani slučajnim varijacijama samog simulacijskog eksperimenta. Ako su

\bar{Y}_j prosjek odziva po opažanjima j-tog nivoa faktora
(opažanja su rezultati simulacijskih eksperimenta),

$\bar{Y}_{..}$ ukupan prosjek odziva za sva opažanja faktora,

varijacija odziva oko ukupnog prosjeka uzorka može se prikazati kao

$$Y_{rj} - \bar{Y}_{..} = (\bar{Y}_j - \bar{Y}_{..}) + (Y_{rj} - \bar{Y}_j)$$

a odatle dobijemo da je suma kvadriranih varijacija

$$\sum_{rj} (\bar{Y}_j - \bar{Y}_{..})^2 = \sum_j R_j (\bar{Y}_j - \bar{Y}_{..})^2 + \sum_{rj} (Y_{rj} - \bar{Y}_j)^2$$

odnosno

$$SS_{UKUPNI} = SS_{TRETMAN} + SS_{DEVIJAC}$$

Tada se računa statistika testa

$$F = \frac{\frac{1}{k-1} SS_{TRETMAN}}{\frac{1}{R-k} SS_{DEVIJAC}}$$

gdje je $R = \sum_j R_j$.

Test hipoteze H_0 je da se:

odbaci H_0 ako $F > F_{1-\alpha}$,

ne odbaci H_0 ako $F < F_{1-\alpha}$,

gdje je $(1-\alpha)$ vjerojatnost da je F-statistika sa $k-1$ i $R-k$ stupnjeva slobode ispod kritične vrijednosti $F_{1-\alpha}$ razdiobe F.

Ponekad su zanimljiva i pitanja razlikovanja utjecaja različitih nivoa faktora na odziv, pa se tada mogu testirati npr. hipoteze:

$$H_0 : \tau_1 = \tau_2 , \quad \text{i slično.}$$

Za odgovor na ta pitanja mogu se koristiti različite metode i testovi, npr. metoda ortogonalnih kontrasta ili Duncanov test višestrukih područja (Montgomery, 1976).

12.3. TEHNIKE REDUKCIJE VARIJANCE

Tehnike redukcije varijance vezane su za postojanje slučajnih varijabli i *stvaranje slučajnih uzoraka* u simulacijskim eksperimentima. Ako npr. izvedemo dva simulacijska eksperimenta s nezavisnim slučajnim brojevima, dobiveni će rezultati biti različiti (slika 12.11). S porastom broja eksperimenata (tj. povećanjem veličine uzorka) povećava se preciznost zaključaka simulacije, tj. smanjuje se varijanca dobivene zavisne veličine (slika 12.12). Time se ujedno olakšava razlučivanje utjecaja slučajne varijacije uzorka i efekata konfiguracije sistema na varijaciju rezultata simulacije. Svrha tehniku redukcije varijance jest da *smanje varijancu izlaznih varijabli* (i to bez promjene njihovih srednjih vrijednosti) s *istom količinom simulacije* (tj. istim ukupnim vremenom izvođenja simulacijskih eksperimenata), odnosno da postignu specificiranu preciznost izlaznih varijabli s manje simulacijskih eksperimenata.

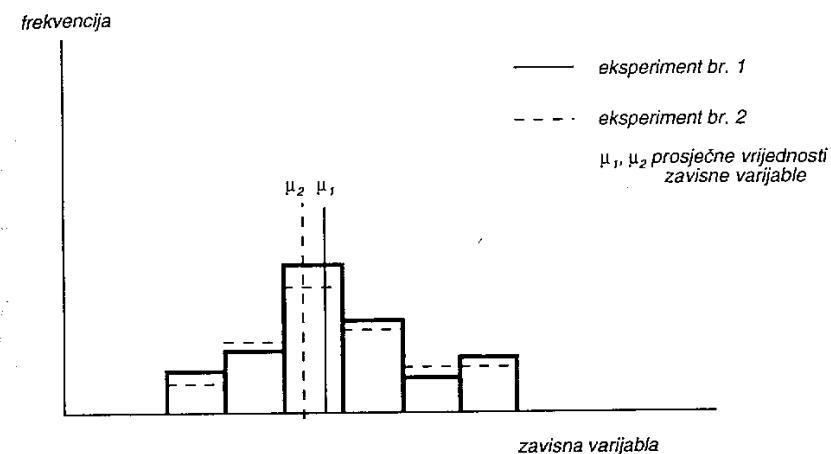
Za neke od tehnik redukcije varijance potrebno je detaljno poznavati rad modela. Također je, u principu, nemoguće unaprijed znati kako se velika varijanca može postići, i da li ćemo je moći reducirati u odnosu prema direktnoj upotrebi simulacije. Stoga je korisno izvesti *preliminarna* izvođenja simulacije što daju indikaciju efekta koji se može dobiti primjenom metoda redukcije varijance.

Prikazat ćemo nekoliko najčešće korištenih metoda redukcije varijance. Opis drugih metoda može se naći npr. u (Kleijnen, 1974; Law i Kelton, 1982).

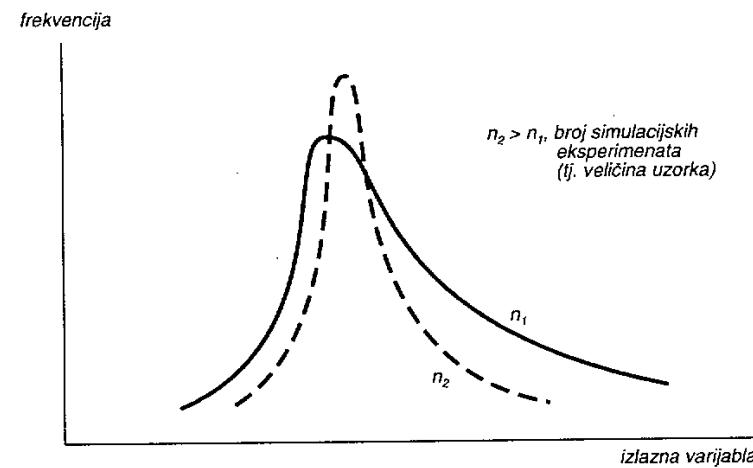
12.3.1. Zajednički slučajni brojevi

Metoda zajedničkih slučajnih brojeva primjenjuje se pri *uspoređenju rada dviju ili više alternativa sistema*. Pri tome želimo usporediti alternativne sisteme u *sto sličnijim eksperimentalnim uvjetima*, tako da razlike u performansama sistema dobivene simulacijom odražavaju razlike u karakteristikama sistema, sa što manje utjecaja uvjeta simuliranja (tj. korištenih nizova generiranih slučajnih varijabli). Stoga se koriste *isti nizovi slučajnih brojeva* za generiranje slučajnih varijabli različitih alternativa sistema. Tako se nastoje generirati iste vrijednosti slučajnih varijabli u istim uvjetima (npr. generiranje istih vremena međudolazaka i posluživanja u istim okolnostima). Ova se metoda zove i *stvaranje koreliranih uzoraka*, jer je njezin cilj da se uvede pozitivna korelacija između izvođenja simulacije za različite alternative sistema.

Pretpostavimo da imamo *dvije* alternativne sisteme, te da su odgovarajući izlazni rezultati alternativa sadržani u varijablama X i Y. Ako bi uzorci za alternativе bili *nezavisni slučajni uzorci* (tj. bez zajedničkih slučajnih brojeva), tada bi varijabla razlike rezultata alternativa



Slika 12.11. Razdioba zavisne varijable za dva izvođenja simulacijskog eksperimenta



Slika 12.12. Promjena varijance zavisne varijable za različit broj izvođenja simulacijskih eksperimenata

$$Z = X - Y$$

imala varijancu

$$\text{var}(Z) = \text{var}(X) + \text{var}(Y),$$

a za *korelirane uzorke* (tj. zajedničke slučajne brojeve) ta je varijanca

$$\text{var}(Z) = \text{var}(X) + \text{var}(Y) - 2 \text{cov}(X, Y).$$

Kod koreliranih uzoraka je vrlo često $\text{cov}(X, Y) > 0$, što znači da je dobivena varijanca razlike izlaza alternativa smanjena (uz jednak veličinu uzorka, odnosno jednak vrijeđenja simulacijskih eksperimenata, kao i za slučaj nezavisnih uzoraka).

Problem pri korištenju ove metode jest to što ona ne garantira smanjenje varijance. Stoga se ponekad osigurava *sinkronizacija* upotrebe istih vrijednosti slučajnih brojeva u istoj situaciji, ali to zahtijeva dodatne intervencije u simulacijskom programu. Za to se koriste različite tehnike: npr. za generiranje vrijednosti svake slučajne varijable preporučuje se korištenje drugoga generatora slučajnih brojeva (npr. generator br. 1 uvijek je za generiranje dolazaka, generator br. 2 za jednu vrstu posluživanja itd.). Nekoliko drugih tehnika opisano je u knjizi (Law i Kelton, 1982).

12.3.2. Antitetske varijable

Metoda antitetičkih varijabli koristi se za simulaciju *jedne konfiguracije sistema*. Pri tome se nastoji uvesti *negativnu korelaciju* između posebnih izvođenja simulacije, i to tako da se simulacijski eksperimenci izvode u dvije serije: u odgovarajućim *parovima* generiranih slučajnih brojeva i varijabli nastoji se osigurati da se male vrijednosti očekivanja iz jednog izvođenja dopunjaju velikima iz drugoga, i obrnuto. Tada će *projekti* odgovarajućih očekivanja parova biti *blize* očekivanoj vrijednosti nego kod nezavisnih izvođenja, tj. varijanca prosjeka će biti manja.

To se nastoji postići tako da se u prvoj seriji izvođenja simulacijskih eksperimenata koriste slučajni brojevi U_k , a u drugoj $(1-U_k)$. Ako prvo izvođenje daje odziv sistema X_1 , a drugo odziv X_2 , tada ćemo tražiti prosječan odziv

$$\bar{X} = \frac{1}{2}(X_1 + X_2)$$

koji je nepristrana procjena od $E(X)$, i čija je varijanca:

$$\text{var}(X) = \frac{1}{4} [\text{var}(X_1) + \text{var}(X_2) + 2 \text{cov}(X_1, X_2)]$$

Tehnika antitetičkih varijabli nastoji uvesti negativnu korelaciju $\text{cov}(X_1, X_2) < 0$ između X_1 i X_2 , kako bi smanjila varijancu od X , uz jednak veličinu uzorka kao kod nezavisnih uzoraka.

Ni kod ove metode nema garancije da će uvesti željenu negativnu korelaciju, zbog složenih transformacija ulaznih varijabli koje se zbivaju u svakom iole složenjem simulacijskog modela. Naime, generiranje slučajnih varijabli mora biti: ako mali U_k daje, recimo, male vrijednosti ulazne slučajne varijable, tada veliki $(1-U_k)$ daje njihove velike vrijednosti. Isti takav odnos mora biti i između ulaznih varijabli i varijabli odziva. Napokon, i ovdje je potrebna *sinkronizacija* generiranja parova varijabli X_1 i X_2 , tj.

par U_k i $(1-U_k)$ mora uvijek djelovati u istim okolnostima, što navodi na slične probleme kao kod tehnike zajedničkih slučajnih brojeva.

12.3.3. Selektivni uzorci

Metoda selektivnih uzoraka temelji se na identificiranju sljedećih dvaju osnovnih izvora grešaka uzorka u simulacijskim eksperimentima (Pidd, 1984):

Efekt skupa su greške skupa vrijednosti dobivenog procesom stvaranja uzorka. Naime, razdiobe slučajnih uzoraka razlikuju se od teorijskih razdioba iz kojih se ti uzorci generiraju. Tako će se i sredine i varijance uzorka razlikovati od očekivanja i varijance teorijske razdiobe.

Efekt nizova su greške uzrokovane slijedom u kojem su vrijednosti skupa generirane. Taj izvor grešaka se ne može u potpunosti eliminirati čak ni ako eliminiramo greške koje nastaju zbog efekta skupa, jer bi za njegovu eliminaciju bio potreban beskonačno velik uzorak.

Budući da u izvođenju simulacije imamo potpunu kontrolu nad generiranjem uzorka, greške uzrokovane efektom skupa možemo minimalizirati, što je i osnova većine metoda redukcije varijance.

Metoda selektivnih uzoraka (odnosno deskriptivnih uzoraka) na jednostavan način osigurava da generirane slučajne varijable budu najbliže moguće obliku odgovarajuće teorijske razdiobe vjerojatnosti. Da bi se dobio selektivni uzorak veličine n slučajne varijable X koja ima funkciju razdiobe $F(X)$, provest ćemo ovu proceduru:

1. Područje $(0,1)$ podijelit ćemo u n jednakih vjerojatnih podintervala sa središnjim točkama:

$$U_i = \frac{1}{2n} + \frac{i-1}{n}, \quad i = 1, 2, \dots, n$$

koje se koriste kao vrijednosti slučajnog broja potrebnog za generiranje slučajne varijable. Tako se minimaliziraju greške uzrokovane efektom skupa, jer smo izabrali slučajne brojeve koji su ravnomjerno raspodijeljeni.

2. Te vrijednosti U_i transformiramo metodom inverzne transformacije u uzorak tražene slučajne varijable

$$X_i = F^{-1}(U_i), \quad i = 1, 2, \dots, n$$

3. Dobiveni skup vrijednosti varijable X_i tada pomiješamo na slučajni način da bismo dobili uzorak slučajne varijable X sa slučajnim slijedom vrijednosti.

Kod ove metode potrebno je moći dobiti inverznu funkciju razdiobe. Također, potrebno je unaprijed znati traženu veličinu uzorka potrebnog za izvođenje simulacijskog eksperimenta.

13. ANALIZA IZLAZA SIMULACIJSKIH EKSPERIMENTATA

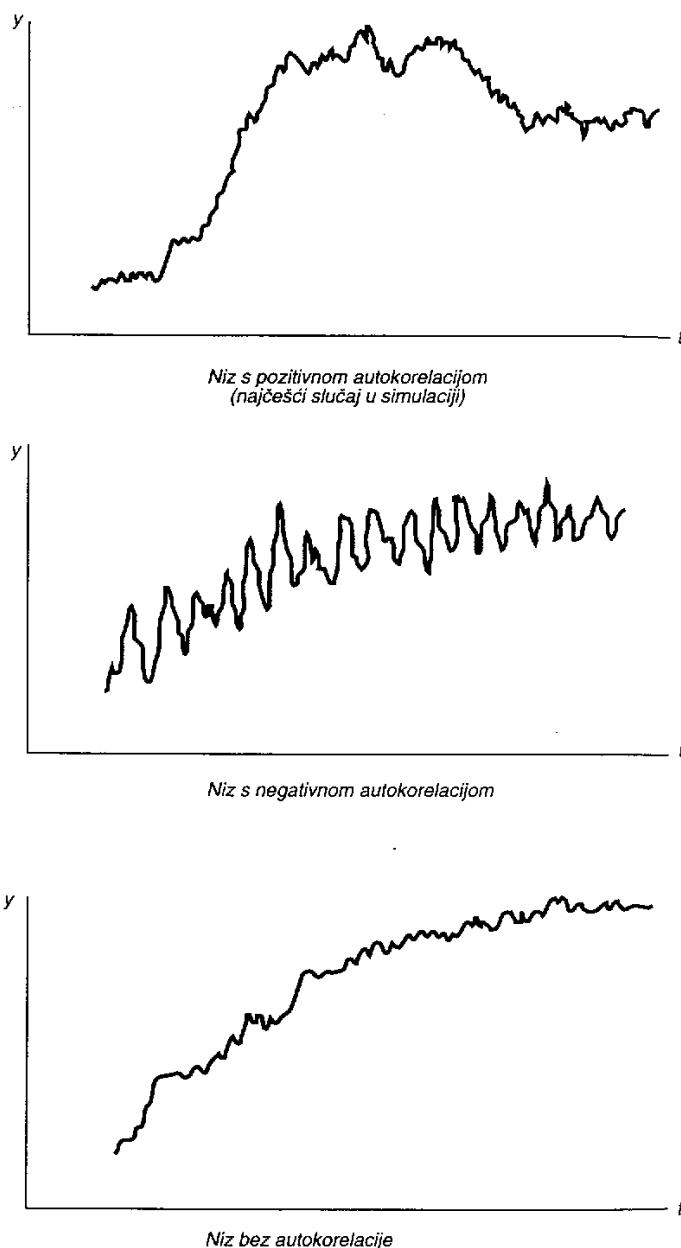
Cilj simulacijske studije je dobivanje performansi sistema korištenjem simulacijskih modela, tj. dobivanje odgovarajućih izlaznih varijabli modela koje odgovaraju relevantnim performansama sistema. S obzirom na slučajni karakter većine sistema od važnosti, dobivene izlazne rezultate simulacijskog eksperimenta potrebno je podvrgnuti i statističkoj analizi kako bi se izvukli ispravni zaključci. U ovom poglavlju opisana su najprije dva osnovna tipa simulacijskih eksperimentata, terminirajuće i stacionarne simulacije, te najznačajnije mjeru performanse sistema. Nakon toga opisano je nekoliko najčešće korištenih procedura za određivanje veličina uzoraka i dobivanje odgovarajućih procjena sredina i intervala pouzdanosti za slučaj analize izlaznih podataka jednog sistema te usporedbe alternativnih sistema.

13.1. OSNOVNE IDEJE ANALIZE IZLAZA SIMULACIJSKIH EKSPERIMENTATA

13.1.1. Uvod

Postojanje slučajnih ulaznih varijabli u simulacijskom modelu dovodi do slučajnog karaktera izlaznih varijabli dobivenih simulacijskim eksperimentima. Tako slučajne fluktuacije vremena posluživanja dovode do slučajnih fluktuacija vremena čekanja i dužine repa pred tim sredstvom posluživanja. To znači da će izvođenja simulacijskih eksperimentata s različitim nizovima slučajnih brojeva dati različite vrijednosti izlaznih varijabli modela. Zbog toga se mora provoditi odgovarajuća statistička analiza izlaza simulacijskih eksperimentata koja će dati potrebne statističke procjene (sredine, intervale pouzdanosti, percentile i sl.) performansi sistema od važnosti.

Osnovna teškoća pri analizi izlaznih podataka simulacije je to da oni u pravilu *nisu nezavisni*, pa se tako klasična statistička analiza opažanja s nezavisnim identičnim razdiobama (NIR) ne može direktno primijeniti. Primjer međusobne zavisnosti izlaznih varijabli su uzastopna vremena čekanja entiteta u repu. Vrijeme sljedećeg entiteta koji čeka u repu zavisno je od vremena jednog ili više prethodnih entiteta, jer se entitet ne može poslužiti sve dok pred njim ima drugih entiteta koji čekaju u istom repu. Za neke slučajevе zavisnosti izlaznih varijabli još nema adekvatnih procedura za analizu varijabli, a neke od postojećih procedura vrlo su složene ili zahtijevaju dosta vremena računala. Tipovi zavisnih vremenskih serija s autokorelacijom, tj. međusobnim utjecajem uzastopnih vrijednosti iste varijable, prikazani su na slici 13.1.



Slika 13.1. Vremenski niz autokoreliranih vrijednosti izlazne varijable simulacije (Banks i Carson, 1984)

Budući da se na temelju rezultata simulacijskih studija u pravilu donose značajne odluke (nabava opreme, izbor konfiguracije sistema), to je analizi izlaza simulacijskih eksperimenata potrebno dati odgovarajuće značenje. U simulacijskim studijama, međutim, dosta se često najveći dio raspoloživog vremena i drugih resursa potroši na izradu simulacijskog modela i programa, tako da se nerijetko događa da ostaje premašno vremena za mjerjenje i analizu ulaznih podataka, planiranje simulacijskih eksperimenata i analizu izlaza simulacijskih eksperimenata. Tako se ponekad izvodi samo jedan simulacijski eksperiment i na temelju njega zaključci, a to zbog moguće velike varijance izlaznih slučajnih varijabli može dovesti do potpuno pogrešnih procjena performansi sistema, što znači i do potpuno pogrešnih odluka koje se na temelju tih rezultata donose. Stoga je potrebno metodološki ispravno izvoditi i analizirati simulacijske eksperimente te stvoriti izlazni uzorak odgovarajuće veličine.

Pristup analizi izlaza simulacijskih eksperimenata u ovom poglavlju slijedi onaj iz knjige (Law i Kelton, 1982).

13.1.2. Tipovi simulacijskih eksperimenata

Dva osnovna tipa simulacijskih eksperimenata su terminirajuća simulacija i stacionarna simulacija.

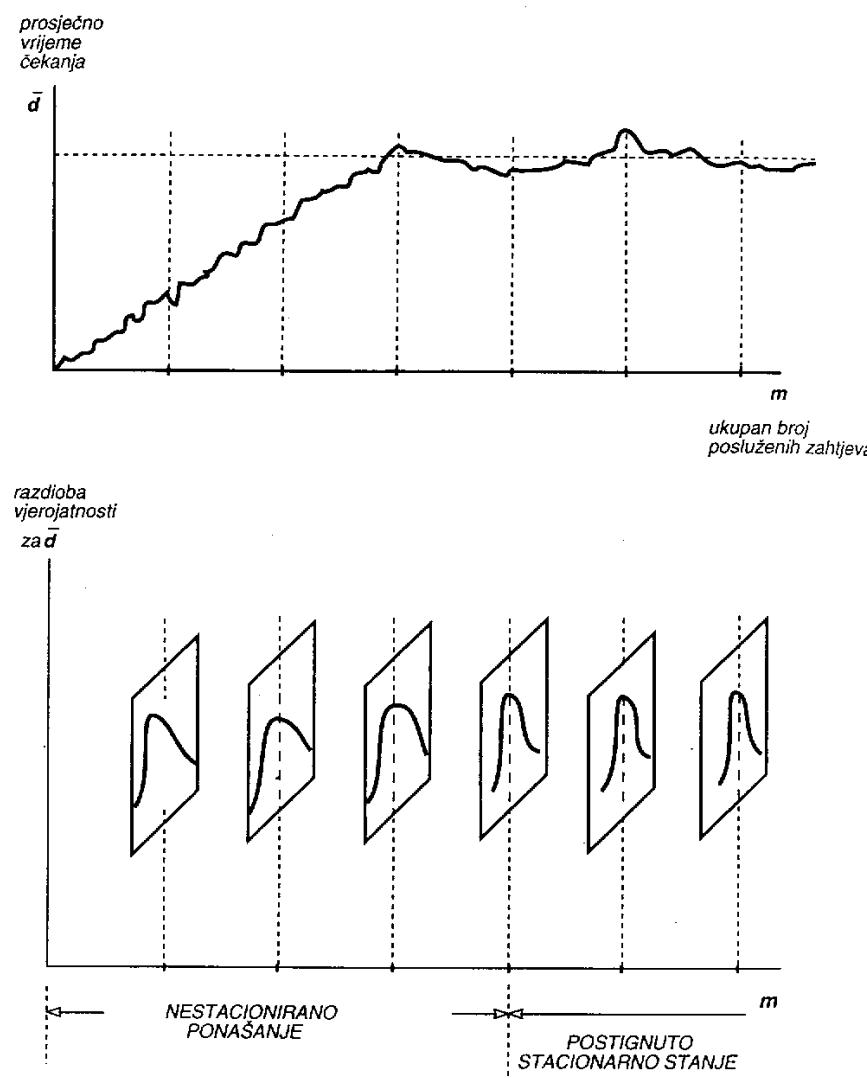
Terminirajuće simulacije su one koje završavaju kada se dogodi određeni događaj D specificiran prije početka simulacije. Mjere performanse sistema definirane su u odnosu prema intervalu $(0, T_D)$, gdje je T_D čas simulacije u kojem se desio događaj D. Mjere performanse kod terminirajućih simulacija ovise o stanju sistema u početnom času 0.

Primjer terminirajuće simulacije je npr. prodavaonica za samoposluživanje, koja radi od časa otvaranja do časa zatvaranja. Događaj D je vrijeme zatvaranja prodavaonice, početno stanje je prazna prodavaonica, a cilj simulacije može biti procjena maksimalnog čekanja mušterija na blagajni i prosječno iskorištenje blagajni.

Terminirajuća simulacija može također prikazivati jedan period najintenzivnijeg prometa sistema koji radi neprekidno. To može biti npr. simulacija prometa u gradu u razdoblju povratka s posla – iako gradski promet neprekidno funkcioniра, može se izvesti terminirajuća simulacija kojoj je početni čas 14 sati, a događaj završetka nastupa u 16 sati. Zajedničko svim terminirajućim simulacijama jest to da je rad sistema u periodu koji se simulira u pravilu nestacionaran.

Stacionarne simulacije su one za koje su mjere performanse definirane kao granične vrijednosti kada vrijeme simulacije traje vrlo dugo. Treba imati na umu da stacionarno stanje ne znači da vrijednosti izlaznih varijabli postaju konstantne, već da *razdiobe vjerojatnosti* tih varijabli postaju *nepromjenljive* u vremenu (kao što je prikazano na slici 13.2). Ovdje nema nekog prirodnog događaja koji uzrokuje završetak simulacije, pa vrijeme jednog izvođenja simulacije mora biti izabrano tako da se dobiju dovoljno dobre procjene veličina koje nas zanimaju.

Primjeri stacionarne simulacije su procesi koji rade kontinuirano (visoke peći, telefonske mreže) ili dovoljno dugo (velika računala s terminalima), i za koje je cilj simulacije nalaženje karakteristika sistema pošto je sistem veoma dugo u pogonu (npr. kapacitet proizvodnje, vrijeme čekanja na odziv sistema). Iako u stvarnom sistemu vanjsko opterećenje može varirati u vremenu, u simulaciji možemo ispitivati utjecaj maksimalnog opterećenja koje dugo traje da bismo dobili performanse sistema u periodu najvećeg opterećenja (npr. produženo ispitivanje rada telefonske mreže pod maksimalnim dnevnim opterećenjem).



Slika 13.2. Ponašanje prosječnog vremena čekanja u repu kod stacionarne simulacije (prosjek vremena čekanja za sve zahijeve poslužene do tog časa)

Zbog dugog trajanja stacionarne simulacije, utjecaj početnih uvjeta na rezultate simulacije je obično manje značajan nego kod terminirajuće simulacije.

Zaključno, možemo reći da jedna vrsta sistema ne mora uvjek značiti izbor jednog tipa simulacije, već odabrani tip simulacije ovisi i o cilju simulacije. Za isti sistem može se ispitivati kako nestacionarno ponašanje, tako i stacionarno ponašanje ili stacionarna aproksimacija vršnog perioda opterećenja sistema. Iskustvo pokazuje da su terminirajuće simulacije češće od stacionarnih simulacija.

13.1.3. Mjere performanse sistema

(1) Način dobivanja mjera performansi

Mjere performanse sistema različito se definiraju pri terminirajućoj i stacionarnoj simulaciji.

Pri *terminirajućoj* simulaciji te se mjere dobivaju kao posljedica izvođenja simulacije koja završava kod terminirajućeg dogadaja. Jedno izvođenje simulacije daje *jedno opažanje* (tj. jednu vrijednost) izlaznih varijabli koje opisuju performanse sistema (maksimalnu dužinu repa, iskorištenje opreme i sl.). Više nezavisnih izvođenja terminirajuće simulacije omogućuje dobivanje uzorka NIR performansi sistema na temelju kojeg se dobivaju potrebne procjene performansi.

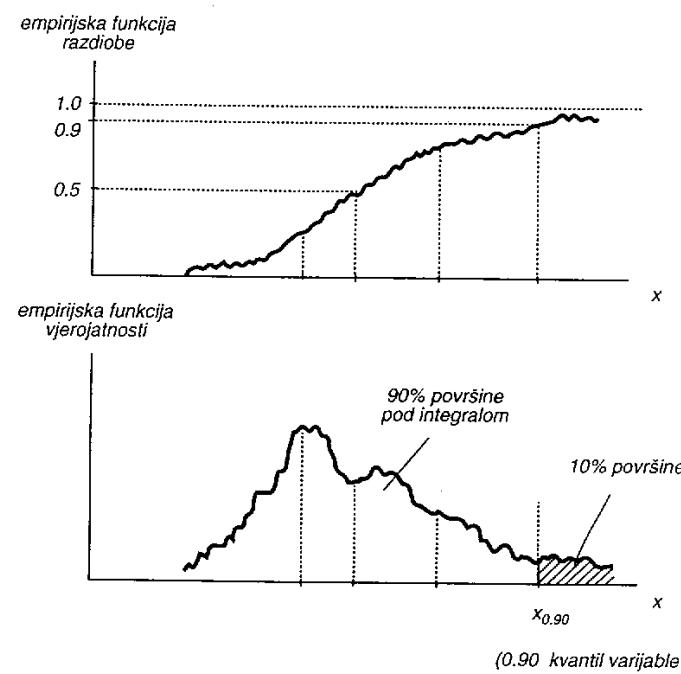
Pri *stacionarnoj* simulaciji povećanje točnosti procjena performansi sistema postiže se povećanjem dužine vremena izvođenja simulacije. Eliminaciju utjecaja početnih uvjeta simulacije olakšavamo i odbacivanjem statistike početnog perioda "zagrijavanja" sistema; ako npr. simulacija počinje praznim sistemom, odbacit ćemo statistiku sistema do vremena u kojem popunjenoj sistemu počinje fluktuirati oko svoje stacionarne vrijednosti.

(2) Korisne mjere performanse : kvantili i proporcije

S obzirom na fluktuaciju izlaznih varijabli simulacijskih modela, *projeci* nisu dovoljna mjera performansi sistema. Što više, same prosječne vrijednosti nisu dovoljne ni za precizan opis procjene očekivanih vrijednosti izlaznih varijabli, jer se projekti razlikuju od uzorka do uzorka.

Procjena stvarne vrijednosti performanse sistema mora stoga sadržati bar još i procjenu *intervala pouzdanosti* za očekivanu vrijednost performanse. To je interval u kojemu je sadržana očekivana vrijednost performanse. Za procjenu intervala pouzdanosti može se dati i pouzdanost kojom će ona biti ostvarena.

Mjera koja se često koristi za procjenu ekstremnih vrijednosti izlaznih varijabli je *kvantil* (percentil). Kvantil se definira tako da je p -ti kvantil izlazne varijable ona vrijednost varijable za koju je vjerojatnost da varijabla bude manja od te vrijednosti jednaka p (slika 13.3). Tako je 0.95 kvantil (odnosno 95% percentil) varijable čekanja u repu ona vrijednost vremena čekanja koja će biti premašena samo u 5% slučajeva (a 95% slučajeva će imati manju vrijednost vremena čekanja). Kvantili se koriste za (1) ocjenu kvalitete posluživanja (npr. određivanje dijela mušterija koje će na posluživanje u određenom sistemu čekati duže od najvećeg toleriranog vremena čekanja) i



Slika 13.3. Kvantil izlazne varijable

(2) ocjenu potrebnih veličina za oblikovanje sistema (npr. nalaženje potrebnog prostora za čekanje pred šalterom, tako da u 95% slučajeva taj prostor bude dovoljan za sve koji čekaju na posluživanje).

Osim toga, važno može biti određivanje *proporcija* (ili postotaka) očekivanja manjih od zadane vrijednosti izlazne varijable. To može npr. biti određivanje postotaka mušterija koje čekaju pred šalterom manje od 2 minute ili dijela korisnika koji pri ukucavanju šifre za korištenje računala dobiju odgovor da je računalo zauzeto (tj. da je na njega već priključen maksimalni dopušteni broj aktivnih terminala). Tako se mogu procijeniti i *proporcije klasa histograma* razdiobe, odnosno određivanja dijela vrijednosti izlazne varijable koje su unutar klasa određene širine (npr. broja mušterija koje pred blagajnom čekaju 0–2 min, 2–4 min, 4–6 min itd.).

Budući da je određivanje intervala pouzdanosti najčešće korištena i najjednostavnija mjerica performansi, u ovom ćemo tekstu njoj posvetiti najveću pažnju. Treba imati na umu da je često potrebno odrediti *istovremenu* procjenu za više mjeri performansi istog sistema, npr. dužine repova, iskorištenja mjesta posluživanja, ukupne propusne moći sistema i sl.

13.2. ANALIZA IZLAZNIH PODATAKA JEDNOG SISTEMA

Opisat ćemo način dobivanja intervala pouzdanosti za slučaj analize izlaznih podataka jednog sistema, i to za terminirajuće i stacionarne simulacijske eksperimente.

13.2.1. Intervali pouzdanosti za terminirajuće simulacije

Neka je napravljeno n nezavisnih izvođenja terminirajuće simulacije, od kojih je svaka počela iz jednakih početnih uvjeta i završila onda kada se desio specificirani terminirajući događaj D. Pri tome se *nezavisnost* eksperimenta postiže tako da se za svako izvođenje simulacije uzimaju različiti slučajni brojevi. Zbog jednostavnosti promatrati ćemo samo jednu mjeru performanse (tj. izlaznu varijablu).

Dobivene vrijednosti (opažanja) procjena mjeri performanse u pojedinim eksperimentima označimo X_1, X_2, \dots, X_n . Budući da su X_i -ovi NIR slučajne varijable, možemo upotrijebiti klasičnu statistiku za konstrukciju intervala pouzdanosti za očekivanu vrijednost $\mu = E(X)$.

(1) Procedure fiksne veličine uzorka

Uobičajan pristup konstrukciji intervala pouzdanosti je izvođenje fiksnog broja n ponavljanja simulacijskog eksperimenta. Ako su dobivene procjene X_1, X_2, \dots, X_n slučajne varijable s *normalnim razdiobama* (to je najčešće ispunjeno ako su oni prosjeci ili sume nekih veličina), tada je interval pouzdanosti za μ s pouzdanosti $(1-\alpha)$ jednak:

$$\bar{X}(n) \pm t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}.$$

Budući da X_i -ovi rijetko imaju normalne razdiobe, ovaj interval je zapravo *približan* interval pouzdanosti. Što je broj ponavljanja eksperimenta manji i što je više razdioba vjerojatnosti X_i -ova nesimetrična, to je pokrivanje očekivane vrijednosti s intervalom pouzdanosti manje od $(1-\alpha)$ (tj. izračunani interval pouzdanosti sadrži rjeđe nego $(1-\alpha)$ puta očekivanu vrijednost od X).

(2) Dobivanje intervala pouzdanosti zadane preciznosti

Uz fiksnu veličinu uzorka n, dobivena širina intervala pouzdanosti izlazne varijable ovisi o veličini varijance $s^2(n)$. Ako je dobiveni interval pouzdanosti preširok, potrebno je odrediti veličinu uzorka n koja će dati odgovarajuću širinu intervala.

Procedura za dobivanje intervala pouzdanosti željene preciznosti je: neka je γ željena vrijednost *relativne preciznosti* intervala pouzdanosti ($0 < \gamma < 1$):

$$\frac{t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}}{\bar{X}(n)}$$

Tada ćemo napraviti ove korake:

1. Izvest ćemo odabranih n_0 inicijalnih ponavljanja simulacije, i staviti ćemo $n = n_0$.
2. Na temelju dobivenih X_1, X_2, \dots, X_n izračunat ćemo $\bar{X}(n)$ i $t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}$.
3. Ako je dobivena relativna preciznost

$$\frac{t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}}{\bar{X}(n)} \leq \gamma,$$

tada ćemo prihvatiti

$$\bar{X}(n) \pm t_{n-1,1-\alpha/2} \sqrt{\frac{s^2(n)}{n}}$$

kao približni $(1 - \alpha)$ interval pouzdanosti za μ , i procedura je završena.

Ako je dobivena preciznost veća od željene preciznosti g , napraviti ćemo jedno dodatno ponavljanje simulacije ($n \rightarrow n+1$) i zatim ići na korak 2.

Iskustvo je pokazalo da se pokrivanje blisko $1 - \alpha$ dobiva za $n_0 \geq 10$ i $\gamma \leq 0.15$.

(3) Preporučeno korištenje procedura za izračunavanje intervala pouzdanosti

Procedura fiksne veličine uzorka može se koristiti u probnim (pilot) eksperimentima, u kojima širina intervala pouzdanosti nije suviše značajna.

Na osnovi probnih eksperimenta moguće je procijeniti troškove ponavljanja eksperimenta i varijancu uzorka, pa se na temelju toga može ocijeniti potreban broj ponavljanja eksperimenta da bi se postigla željena preciznost. Ako, recimo, tražimo da apsolutna preciznost (poluširina intervala pouzdanosti) bude

$$\beta = t_{n-1,1-\alpha/2} \sqrt{\frac{s^2(n)}{n}},$$

tada je približna procjena za veličinu uzorka potrebnog da se dobije apsolutna preciznost β jednaka

$$n^* \approx \frac{t^2}{\beta^2} \frac{n-1,1-\alpha/2}{s^2(n)}.$$

Bez obzira na cijenu izvođenja ponavljanja simulacije preporučuje se (Law i Kelton, 1982) izvođenje *bar triju ponavljanja* simulacijskog eksperimenta kako bi se ocijenila varijabilnost X_i -ova (izvođenje dva ponavljanja eksperimenta ima dosta veliku vjerojatnost da dovede do X_1 i X_2 koji se veoma malo razlikuju čak kada su X_i -ovi jako varijabilni). Ako dobiveni X_1 , X_2 i X_3 nisu međusobno veoma blizu, potrebno je provesti *daljnja ponavljanja* simulacijskih eksperimenta kako bi izvedeni zaključci bili ispravni.

(4) Izbor odgovarajućih početnih uvjeta simulacije

Vidjeli smo da mjere performansi terminirajuće simulacije ovise o stanju sistema u početnom času, pa stoga treba pažljivo izabrati početno stanje sistema. Na primjer, nije razumno da početno stanje u samoposluživanju u vršnom periodu opterećenja bude prazno samoposluživanje.

Jedan pristup izbora početnog stanja uzima na početku rada sistema (npr. samoposluživanja) da je sistem prazan i zatim se simulira odvijanje procesa do početka perioda vršnog opterećenja (period "ugrijavanja"). To je početno stanje sistema, koje odgovara času početka vršnog opterećenja. Slaba strana ovog pristupa je potrebitno dodatno vrijeme izvođenja simulacijskog eksperimenta.

Dруги приступ prepostavlja skupljanje podataka o stanju stvarnog sistema na početku vršnog opterećenja. Na temelju dobivene razdiobe vjerojatnosti broja kupaca u

samoposluživanju generira se broj kupaca koji je u času početka simulacije u samoposluživanju, što je početno stanje simulacije.

13.2.2. Intervali pouzdanosti za stacionarne simulacije

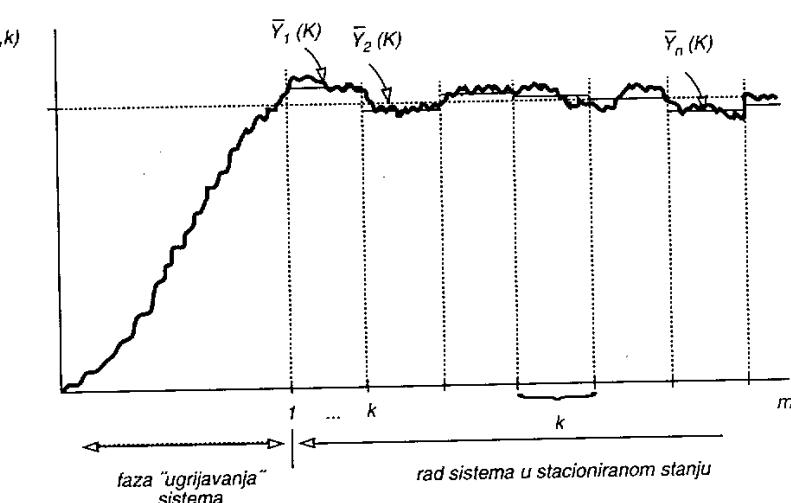
Kao što smo rekli, za rješavanje stvarnih problema stacionarne se simulacije koriste manje od terminirajućih simulacija. Procedure za konstrukciju intervala pouzdanosti za stacionarne simulacije su mnogo komplikiranije nego one za terminirajuće simulacije. Razlog za to je da su opažanja izlaznih varijabli u pravilu *autokorelirana*, tj. dobivene vrijednosti opažanja su međusobno zavisne. Opisat ćemo samo jednu takvu proceduru.

(1) Procedura sredine serija

Procedura *sredina serija* prepostavlja da opažanja izlazne varijable Y_i predstavljaju kovarijantno stacionarni proces (tj. proces u kojem su sredine i varijance stacionarne u vremenu, a kovarijanca između dva opažanja Y_i i Y_{i+j} ovisi samo o udaljenosti tih opažanja). Izvest će se *jedan* simulacijski eksperiment dužine m opažanja, i podijeliti dobivena opažanja Y_1, Y_2, \dots, Y_m u n jednakih uzastopnih serija dužine k ($m=nk$). Serija 1 sadržat će opažanja Y_1, Y_2, \dots, Y_k , serija 2 opažanja $Y_{k+1}, Y_{k+2}, \dots, Y_{2k}$ itd. Neka $\bar{Y}_j(k)$ ($j=1,2,\dots,n$) budu sredine opažanja serija (slika 13.4) i neka je ukupna sredina cijelog uzorka

$$\bar{Y}(n,k) = \frac{1}{n} \sum_{j=1}^n \bar{Y}_j(k) = \frac{1}{m} \sum_{i=1}^m Y_i$$

uzeta za procjenu očekivanja izlazne varijable.



Slika 13.4. Procedura sredina serija za nalaženje intervala pouzdanosti kod stacioniranih simulacija

Ako je *dužina serija k dovoljno velika*, tada će sredine serija biti približno nekorelirane (Law i Carson, 1979) i imat će približno normalnu razdiobu. Tada broj serija ne mora biti prevelik (prema nekim preporukama dovoljno je pet serija). Uvezši u obzir i svojstvo kovarijantne stacionarnosti, tada sredine serija $\bar{Y}_j(k)$, ako su dužine serija k dovoljno velike, možemo tretirati kao **NIR** normalne slučajne varijable sa sredinom v . Tada možemo konstruirati približni $(1-\alpha)$ interval pouzdanosti za v kao

$$\bar{Y}(n,k) \pm t_{n-1,1-\alpha/2} \sqrt{\frac{s_{\bar{Y}_j(k)}^2}{n}},$$

gdje je

$$s_{\bar{Y}_j(k)}^2 = \frac{1}{n-1} \sum_{j=1}^n (\bar{Y}_j(k) - \bar{Y}(n,k))^2.$$

Vidimo da je ovaj postupak analogan onome za računanje intervala pouzdanosti za terminirajuće simulacije.

Rezultati primjene ove procedure pokazuju da izbor odgovarajuće veličine m znatno ovisi o tipu simulacijskog modela. Također se pokazalo: ako je m relativno mali (zbog npr. skupih eksperimenata), metoda sredine serija je podjednako dobra kao i bilo koja druga procedura, a ujedno je vrlo jednostavna.

(2) Eliminacija utjecaja početnih uvjeta

Iako se zbog dugog vremena trajanja simulacije stacionarnog stanja smanjuje utjecaj početne nestacionarne faze razvoja simulacije, u toj se fazи ipak javljaju greške u ocjeni stacionarnih performansi rada sistema. Zbog toga se nastoji eliminirati utjecaj početnog perioda tzv. "ugrijavanja" sistema. Dva su osnovna načina da se to učini.

Prvi pristup eliminaciji utjecaja početnog perioda simulacije traži skupljanje podataka o radu sistema u stacionarnom stanju. Korištenjem prikupljenih podataka moguće je specificirati početno stanje modela koje odgovara stacionarnom stanju sistema, i tako izbjegći potreban period simulacije "zagrijavanja" sistema.

Drugi pristup uzima za početno stanje prazan sistem. Prva faza simulacije traje do prelaska sistema u stacionarno stanje. Tek nakon toga počinje skupljanje podataka o razvoju sistema tijekom simulacije, tako da statistički rezultati simulacije ne obuhvaćaju period "ugrijavanja" sistema (slika 13.4). Završetak perioda "ugrijavanja" sistema određuje se na osnovi toga što odgovarajuća izlazna varijabla treba poprimiti razdiobu vjerojatnosti blisku svojoj stacionarnoj razdiobi vjerojatnosti. Različite su procedure za određivanje prelaska sistema u stacionarno stanje, a najjednostavnija je (iako često i neadekvatna) analiza krivulje promjene vrijednosti izlazne varijable ili njezinih kumulativnih prosjeka u vremenu.

(3) Određivanje dužine izvođenja simulacije

Dužina izvođenja simulacije u stacionarnom stanju mjeri se bilo vremenom trajanja simulacije, bilo ukupnim brojem opažanja izlazne varijable. Odluka o dužini izvođenja simulacije obično se temelji na željenoj točnosti procjene izlazne varijable (koja se mjeri procjenom varijabilnosti izlazne varijable) te cijeni izvođenja simulacijskih eksperimenata. Ako simulacija počinje praznim sistemom, tada u ukupno vrijeme izvođenja

simulacije treba uračunati i vrijeme simulacije "ugrijavanja" sistema, koje se ne koristi za skupljanje opažanja izlazne varijable.

13.2.3. Određivanje ostalih mjera performansi

(1) Kvantili

Kao što smo rekli, p-ti kvantil x_p izlazne varijable X ona je vrijednost varijable za koju je vjerojatnost da varijabla bude manja od te vrijednosti jednaka p, tj.

$$P(X < x_p) = p, \quad 0 < p \leq 1.$$

Kvantil x_p možemo procijeniti (Kleijnen, 1987) tako da uredimo n opažanja izlazne varijable u rastućem poretku

$$x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n-1)} \leq x_{(n)}.$$

Tada se kao procjena kvantila populacije x_p uzima vrijednost uređenog opažanja varijable $x_{(pn+1)}$, od koje upravo p-ti dio opažanja varijable ima manju vrijednost.

Za granice intervalne procjene s pouzdanosću $(1-\alpha)$ mogu se, za uzorke veličine $n > 20$, uzeti uredena opažanja $x_{(L)}$ i $x_{(D)}$ (L – lijeva granica, D – desna granica):

$$P(x_{(L)} \leq x_p \leq x_{(D)}) = 1 - \alpha,$$

gdje je

$$L = np - z_{1-\alpha/2} \sqrt{np(1-p)}, \text{ zaokružen nagore,}$$

$$D = np + z_{1-\alpha/2} \sqrt{np(1-p)}, \text{ zaokružen nagore.}$$

Za stacionarnu simulaciju možemo napraviti isti pristup kao kod metode sredine serija, samo se iz svake serije (tj. grupe opažanja) ne uzima prosjek serije već p-ti kvantil serije.

(2) Proporcije

Kao što smo rekli, proporcija (postotak) dio je opažanja koji je manji od zadane vrijednosti izlazne varijable. Budući da su proporcije jedna vrsta prosjeka (prosjek udjela izlazne varijable u klasi manjoj od zadane kritične vrijednosti), oni se mogu procjenjivati statističkim procedurama koje se koriste za procjenu prosjeka (Law i Kelton, 1982).

13.2.4. Višestruke mjere performansi sistema

U većini simulacija potrebno je pronaći više mjera performansi sistema. Pretpostavimo da se traži m mjera performansi sistema, tj. da treba naći odgovarajuće $(1-\alpha_k)$ intervale pouzdanosti I_k za mjere performanse μ_k ($k=1,2,\dots,m$). Simulacija pri tome može biti bilo terminirajućeg, bilo stacionarnog tipa.

Tada se može dokazati da, bez obzira na to jesu li I_k nezavisni ili zavisni, vjerojatnost da svih m intervala pouzdanosti *istovremeno* sadrže odgovarajuće prave mjere performansi zadovoljava relaciju:

$$P(\mu_k \in I_k \text{ za svaki } k) \geq 1 - \sum_{k=1}^m \alpha_k \quad (\text{Bonferroni nejednadžba}).$$

Tako je npr. za 5 mjera performansi za koje se traži konstrukcija 90 postotnog intervala pouzdanosti ($\alpha_k = 0.1$) vjerojatnost da svaka od njih sadrži pravu mjeru pouzdanosti samo $P \geq 0.5$. Dakle, u zaključke ovakve studije ne može se imati previše *sveukupnog* pouzdanja, odnosno postoji mogućnost da bar neki od intervala pouzdanosti neće sadržati prave mjere performansi.

Mogući pristup rješenju ovog problema za slučaj malog broja m ($m \leq 10$) mjera performansi je: ako se želi dobiti sveukupni interval pouzdanosti od *bar* $(1-\alpha)$, izabrat ćemo α_k -ove takve da je $\sum \alpha_k = \alpha$. Pri tome će α_k koji odgovaraju značajnijim mjerama performansi biti manji po veličini. Osnovni je problem što će α_k -ovi biti relativno mali, tj. što će za njihovu konstrukciju trebati dosta veliki uzorci.

13.3. USPOREDBA ALTERNATIVNIH SISTEMA

Jedan od značajnijih ciljeva upotrebe simulacijskih modela je analiza izlaza nekoliko *različitih* simulacijskih modela, koji mogu prikazivati različite strukture sistema, tehnologije, načine upravljanja sistemom ili vanjske uvjete u kojima sistem radi. Na osnovi te analize izlaza mogu se donositi odluke o prihvatanju i implementaciji određene alternative.

Česta greška pri usporedbi alternativnih sistema jest *izvođenje samo jednog eksperimenta za svaku od alternativa*, na osnovi toga računanje nezavisnih procjena iste mjere performanse za svaku alternativu i zatim usporedba tako izračunanih procjena. Takav pristup može dovesti do veoma velike vjerojatnosti (od nekoliko desetaka posto) prihvatanja pogrešnog zaključka.

Osnovni zahtjev za korištenje bilo koje statističke metode za usporedbu alternativnih sistema je mogućnost prikupljanja NIR opažanja izlazne varijable koja odgovara traženoj mjeri performanse rada sistema. U slučaju terminirajuće simulacije to je lako postići izvođenjem nezavisnih ponavljanja eksperimenta, dok je kod stacionarne simulacije to teže postići (jedan od načina je korištenje sredina serija kao osnovnih NIR opažanja).

Opisat ćemo jednu proceduru za usporedbu dvaju sistema te osnovne ideje pristupa usporedbi većeg broja sistema.

13.3.1. Usporedba dvaju sistema

Dva sistema (odnosno dvije alternative istog sistema) uspoređivat ćemo na osnovi srednje vrijednosti jedne izlazne varijable. Usporedbu ćemo provesti konstrukcijom intervala pouzdanosti za *razliku* dviju srednjih vrijednosti.

Neka su X_{1j} i X_{2j} ($j=1,2,\dots,n$) uzorci veličine n NIR opažanja za dva sistema dobivena simulacijskim eksperimentima za svaki od sistema, i neka su $\mu_1 = E(X_{1j})$ i $\mu_2 = E(X_{2j})$ sredine izlazne varijable koje služe za usporedbu sistema. Prikazat ćemo metodu koja se temelji na formiranju *parova* razlike vrijednosti opažanja X_{1j} i X_{2j} . Tada će $Z_j = X_{1j} - X_{2j}$ biti NIR slučajne varijable, i cilj procedure će biti konstrukcija intervala pouzdanosti od očekivane vrijednosti razlike $E(Z_j) = \zeta$.

13.3. USPOREDBA ALTERNATIVNIH SISTEMA

Uz

$$\bar{Z}(n) = \frac{1}{n} \sum_{j=1}^n Z_j$$

$$\hat{\sigma}^2[Z(n)] = \frac{1}{n(n-1)} \sum_{j=1}^n [Z_j - \bar{Z}(n)]^2$$

formirat ćemo $(1-\alpha)$ interval pouzdanosti za ζ kao:

$$\bar{Z}(n) \pm t_{n-1, 1-\alpha/2} \sqrt{\hat{\sigma}^2[Z(n)]}.$$

Ovaj interval je točan ako Z_j imaju normalnu razdiobu, a inače je približan za velike uzorce n . Pri tome X_{1j} i X_{2j} ne moraju biti nezavisni, i njihove varijance ne moraju biti jednakе. Uvođenjem pozitivne korelacije među X_{1j} i X_{2j} (npr. metodom koreliranih uzoraka) može se reducirati varijanca od Z_j i tako smanjiti interval pouzdanosti.

Ako su npr. obje granice dobivenog intervala pouzdanosti veće od nule, tada možemo reći da se μ_1 razlikuje od μ_2 s pouzdanošću $(1-\alpha)$, te da alternativa 1 ima veće očekivanje. Neka se npr. ispituje maksimalno vrijeme čekanja u repu, i neka se dobije da je interval pouzdanosti za $(\mu_1 - \mu_2)$ s pouzdanošću 95% jednak (0.72, 3.24). To znači da prva alternativa ima duže maksimalno vrijeme čekanja u sistemu, i to za 0.72 do 3.24 vremenskih jedinica s pouzdanošću 95%. Alternative ovdje mogu biti npr. različite discipline repa, recimo FIFO i prioritetska. Dakle, sistem 2 je prihvatljiviji.

Ne smije se zaboraviti da su ovdje X_{1j} i X_{2j} slučajne varijable definirane za *cijelo ponavljanje* simulacijskog eksperimenta. Tako X_{1j} može npr. biti prosjek vremena čekanja u j-tom ponavljanju eksperimenta prvog sistema (a ne vrijeme čekanja nekog individualnog posjetioca).

13.3.2. Usporedba većeg broja sistema

Ako imamo veći broj alternativnih sistema, npr. k sistema ($k \geq 2$), tada je moguće napraviti više različitih usporedbi sistema, npr.:

- izbor najboljeg između svih k sistema
- izbor podskupa veličine m koji sadrži najbolji od svih k sistema
- izbor m najboljih između k sistema
- izbor sistema koji su bolji od zadanih standarda.

Radi usporedbe većeg broja sistema razvijen je niz različitih statističkih postupaka. Neki od tih postupaka koriste se konstantnim veličinama uzorka, a drugi sekvencialnim formiranjem uzorka.

Postupci *konstantne veličine uzorka*, koja je unaprijed zadana, koriste se za donošenje zaključaka testiranjem hipoteza ili procjenom intervala pouzdanosti. Tim postupcima se ponekad mogu eliminirati sistemi s vrlo slabim performansama, ali se često dobivaju preširoki intervali pouzdanosti na temelju kojih je teško donijeti dovoljno pouzdan zaključak.

Postupci *sekvencialnog formiranja uzorka* funkcioniraju tako da se tijekom prikupljanja podataka analiziraju dobivene procjene, i uzorak se povećava sve dok se ne postigne željena preciznost rezultata.

Za svaku od tih procedura potrebno je provesti i vrednovanje, za što su razvijeni odgovarajući statistički postupci.

S obzirom na složenost statističkih postupaka za usporedbu većeg broja alternativnih sistema, ovdje ih nećemo opisivati. Neki od tih postupaka mogu se naći u knjigama (Law i Kelton, 1982; Kleijnen, 1974 Vol. II).

13.4. MODELIRANJE VEZE IZMEĐU ULAZA I IZLAZA SIMULACIJE

U dosadašnjoj analizi izlaznih varijabli simulacijskih eksperimenata promatrali smo vrijednosti izlaznih varijabli dobivene uz određene vrijednosti ulaznih varijabli i parametara simulacijskog modela. Da bi se interpretirali podaci dobiveni simulacijskim eksperimentima s modelima koji imaju različite kombinacije vrijednosti ulaznih varijabli, potrebno je napraviti *modele* veze između ulaznih i izlaznih varijabli simulacijskih modela. Za to se najčešće koriste *regresijski modeli* koji omogućuju da se interpretiraju velike količine podataka koje se dobivaju iz simulacijskih eksperimenata. Time se postiže *generalizacija* rezultata simulacije i omogućuje stvaranje zaključaka o vezama među varijablama te o reakcijama sistema na promjene nezavisnih varijabli sistema, uključujući i analizu osjetljivosti i optimalizaciju. Razvoj i primjenu takvih regresijskih modela (odnosno *metamodela*, jer su to modeli ponašanja sistema temeljeni na rezultatima dobivenim simulacijskim modelima) razvio je osobito (Kleijnen, 1987).

U paragrafu 11.8. opisali smo osnove korištenja regresijskog modeliranja veza među podacima, koji se mogu primijeniti i za veze između ulaza i izlaza simulacije.

Interpretacija pojma opažanja kod simulacijskog eksperimentiranja dana je u paragrafu 13.1. Tako kod terminirajuće simulacije, koja je najčešći oblik simulacijskog eksperimentiranja, jedno opažanje izlazne varijable odgovara vrijednosti varijable dobivene jednim izvođenjem simulacijskog eksperimenta.

III. DIO

SISTEMSKA DINAMIKA

14. OSNOVNE IDEJE SISTEMSKE DINAMIKE

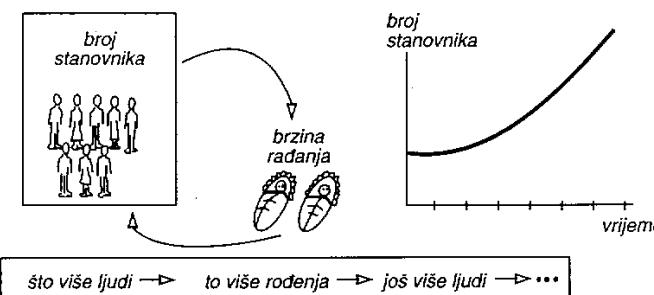
Sistemska dinamika je metoda za kontinuiranu simulaciju sistema s povratnom vezom. Njome se najčešće modeliraju i simuliraju ekonomski, društveni i biološki fenomeni. Pri tome se pojedinačni dogadaji tih složenih sistema agregiraju, tako da se mogu opisivati kontinuiranim tokovima. Sistemska dinamika će biti prikazana uglavnom u kvalitativnom obliku koji omogućuje razumijevanje i korištenje ove metode za široki krug korisnika koji nemaju dublju matematičku naobrazbu (u 17. poglavljtu navest ćemo i matematičke osnove ove metode, što omogućuje dublji uvid u metode simulacije i moguće probleme pri njezinoj realizaciji). U ovom su poglavljju opisani sistemi s povratnom vezom te osnovne ideje pristupa sistemske dinamike modeliranju i simulaciji sistema tog tipa.

14.1. SISTEMI S POVRATNOM VEZOM

Sistemi s povratnom vezom su osnovni tip sistema koji se modeliraju i simuliraju sistemskom dinamikom. Povratna veza zatvoreni je krug uzroka i posljedica, koji utječe na to da neki početni uzrok ima indirektan efekt na samoga sebe. Takav zatvoren krug uzroka i posljedica uzrokuje složeni razvoj ponašanja sistema u vremenu. Povratna veza može biti dosta jednostavna, tj. kružni lanac koji dovodi do utjecaja nekog uzroka na samoga sebe može biti kratak, ali može biti i dosta složena tako da je nije lako ustavoviti, a još je teže predvidjeti njezine posljedice.

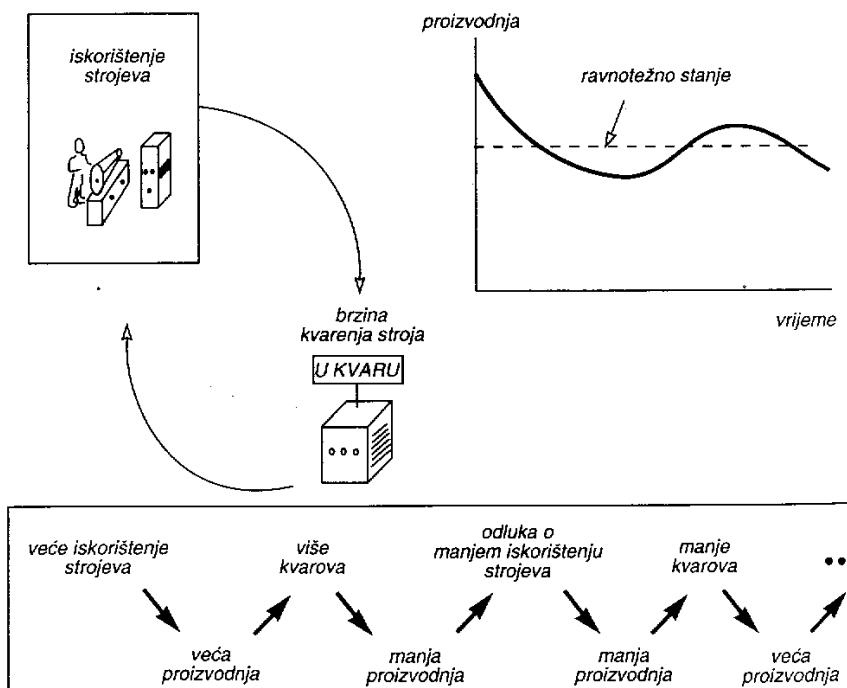
Povratna veza može biti pozitivna ili negativna. *Pozitivna povratna veza* znači da neki uzrok dovodi preko lanca posljedica do *promjena uvijek u istom smjeru*. Tako se stalnim povratnim djelovanjem postiže sve veći daljnji *stalni porast* ili *stalno smanjenje* tog uzroka. Jednostavni primjer pozitive povratne veze je utjecaj rađanja na ukupan broj stanovnika. Uz određenu stopu rađanja za neki početni broj stanovnika bit će određen broj novorođenčadi godišnje, čime će se povećati ukupan broj stanovnika. Zbog povećanog broja stanovnika na početku druge godine broj novorođenčadi u toj godini bit će još veći, što će dalje povećati broj stanovnika itd. Na slici 14.1. prikazana je ta povratna veza i vremenski razvoj broja stanovnika.

Negativna povratna veza znači da neki uzrok preko lanca posljedica dovodi do *promjene smjera vlastitog djelovanja*. Ako, dakle, uzrok poraste iznad nekog *ravnotežnog* stanja, povratna veza će smanjiti taj uzrok. No kada uzrok postane manji od ravnotežnog stanja, tada povratna veza uzrokuje povećavanje tog uzroka. Tako negativna povratna veza *stabilizira* uzrok oko nivoa njegova *ravnotežnog* stanja. Primjer negativne povratne veze je traženje takva iskorištenja strojeva u pogonu koje daje najveću moguću proizvodnju. Preveliko iskorištanje strojeva omogućuje veliku proizvodnju, ali uzrokuje i velik broj kvarova strojeva, što smanjuje proizvodnju. Zbog toga se odlučuje



Slika 14.1. Primjer pozitivne povratne veze

smanjiti iskorištenje strojeva. Manjim iskorištenjem strojeva smanjuje se proizvodnja ali i broj kvarova, posljedica čega je rast proizvodnje. Varijacijom stupnja iskorištenja strojeva traži se stanje s najvećom ukupnom proizvodnjom. Na slici 14.2. prikazana je povratna veza i vremenski razvoj proizvodnje.



Slika 14.2. Primjer negativne povratne veze

U pravilu, povratne se veze ne ostvaruju u ovako kratkim lancima uzroka i posljedica kakvi su prikazani u ovim primjerima, već oni nastaju u složenom nizu uzroka i posljedica. Isto tako, sistemi s povratnim vezama sadrže mješavini pozitivnih i negativnih povratnih veza koje određuju njihovo ponašanje u vremenu. Neke od primjera složenijih povratnih veza prikazat ćemo u sljedećem poglavlju.

Važnost analize sistema s povratnim vezama je pronalaženje *najznačajnijih faktora* koji utječu na upravljanje razvojem sistema. Sistemska dinamika omogućuje modeliranje i analizu vrlo složenih društvenih i ekonomskih fenomena modelima razumne veličine, te pronalaženje najvažnijih uzroka ponašanja sistema. Osim toga, sistemska dinamika omogućuje analizu *dinamike* promjena stanja sistema u vremenu. To je često vezano za ispitivanje *ravnotežnog* (stabilnog) stanja sistema, te dinamike dolaska u ravnotežno stanje koja ovisi kako o strukturi sistema tako i o njegovim parametrima. U pravilu, želi se postići *upravljanje* sistemom koje omogućuje dolazak sistema do željenog stanja.

14.2. OSNOVNE KARAKTERISTIKE SISTEMA S POVRATNOM VEZOM

Sistemska dinamika upotrebljava nekoliko osnovnih pojmova za opis sistema s povratnom vezom. To su nivoi, brzine i kašnjenja. Ti pojmovi povezani su za specifični prikaz procesa u sistemima koji se tipično modeliraju sistemskom dinamikom. Naime, procesi koji se opisuju u sistemskoj dinamici promatraju se sa stajališta različitih *resursa* (ljudi, narudžbe, novac i sl.) koji *prelaze iz stanja u stanje* (Wolstenholme, 1990).

14.2.1. Nivoi

Nivoi u sistemskoj dinamici su stanja resursa, odnosno različite *akumulacije resursa*. Nivoi su mjerljive veličine, a njihove dimenzije su u jedinicama resursa. U statičkom stanju nivoi se mogu primijetiti.

Nivoi mogu biti npr. broj stanovnika, broj kuća ili broj dijelova u skladištu. To mogu biti i veličine koje se fizički ne vide, npr. iznos novca na računima u banci. *Stanje sistema* stanje je svih nivoa u sistemu. Nivoima se može *upravljati*, i tako utjecati na brzinu njihove promjene i na njihovo ravnotežno stanje.

14.2.2. Brzine

Varijable brzine pokazuju brzinu pretvaranja resursa iz jednog u drugo stanje. One su dakle *akcije* ili *tokovi* koji dovode do povećanja ili smanjenja nivoa u sistemu u jedinici vremena, tj. koje *upravljaju stanjem nivoa*. One se izražavaju u jedinicama resursa u jedinici vremena. U statičkom stanju brzine su jednake nuli, tj. ne mogu se primijetiti. Ta činjenica olakšava identifikaciju tipa varijabli sistema.

Brzine se promatraju kao *prosječne* brzine u nekom *periodu vremena* (a ne u jednom času). Tako je ukupan broj stanovnika nekog područja nivo, a broj djece rođene u jednoj godini je brzina promjene tog nivoa. Isto tako, količina novca na nekom računu je nivo, a ulaganje novca na taj račun i vađenje s njega brzine su promjena tog nivoa.

Brzine se zovu i *funkcije odlučivanja*, budući da je često moguće odlučivati (upravljati) o tokovima koji povećavaju ili smanjuju vrijednosti pojedinih nivoa sistema. Na primjer, moguće je odlučivati o intenzitetu sadenja stabala što povećava broj stabala u

šumi (nivo), ili o intenzitetu ispuštanja vode iz umjetnog jezera što smanjuje razinu jezera. Na neke se brzine ne može utjecati, osobito na one na koje utječu samo vanjski faktori (npr. na potražnju tržišta).

U prirodnim sistemima brzine se mijenjaju bez odlučivanja, tj. na njih djeluju prirodni zakoni (npr. brzine povećavanja i smanjivanja mase oblaka ovise o fizičkim zakonima, uz određeno stanje rasporeda vlažnosti, pritiska, temperature i sl.).

14.2.3. Kašnjenja

Kako u tehničkim, tako se ni u društvenim ni u ekonomskim sistemima promjene ne dešavaju istovremeno kada su pokrenute. Na primjer, da bi se izvršila narudžba robe potrebno je određeno vrijeme. Naime, narudžba treba da stigne do proizvođača, da se kod njega izvedu potrebne knjigovodstvene i fizičke radnje pripreme za slanje robe te da poslana roba stigne do naručioca. Dio kašnjenja odnosi se na *materijalna kašnjenja*, a dio na *kašnjenja informacija*. Na neka se kašnjenja može utjecati (npr. na brzinu obradbe narudžbe kod proizvođača), a na neka ne može (npr. na vrijeme potrebno pošti da dostavi narudžbu).

Kašnjenja u sistemu mogu imati značajne posljedice na ponašanje sistema, i ona u pravilu uzrokuju *oscilacije* u ponašanju sistema. Ako je moguće smanjiti kašnjenja, tada se i oscilacije smanjuju i sistem radi ujednačenije. Tako se kod skladišta s dugim vremenom kašnjenja od časa slanja narudžbe do sticanja robe dešava da je nivo skladišta već vrlo nizak a narudžba još nije ispunjena. Skladište će u nervozni početi slati nove narudžbe, pa će se ono u jednom periodu početi naglo popunjavati. Zbog toga će se narudžbe bitno smanjiti. Ali, kada se skladište počne prazniti, neće biti dovoljno ispunjavanja narudžbi pa će se opet forsirati brojne narudžbe itd. Vidimo da kašnjenje u realizaciji narudžbi uzrokuje reakcije koje stvaraju veliku fluktuaciju nivoa skladišta, što je prikazano na slici 14.3a. Smanjenje kašnjenja realizacije narudžbi smanjuje i potrebu za prenaglim reakcijama (bilo povećanim bilo smanjenim naručivanjima) što omogućuje rad skladišta s mnogo manjim fluktuacijama nivoa skladišta, što se vidi na slici 14.3b.

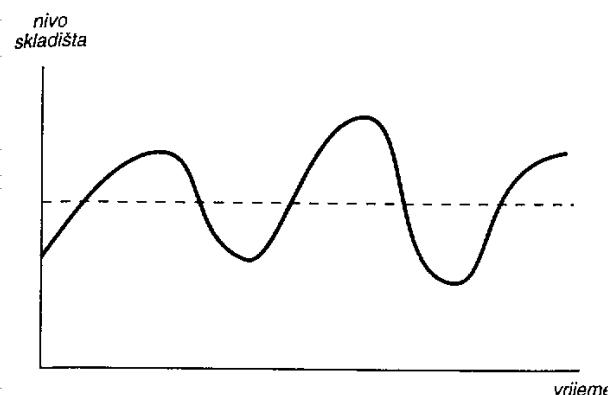
U sistemima s dužim lancima utjecaja, npr. kod hijerarhijskih sistema skladišta s narudžbama iz višeg nivoa skladišta, utjecaj kašnjenja na rad sistema je veoma izražen. Modeliranje i simulacija takvih sistema omogućuje da se razumije koji faktori utječu na nestabilnost rada sistema, te da se pronađu rješenja koja omogućuju stabilniji rad sistema.

14.2.4. Konstrukcija modela

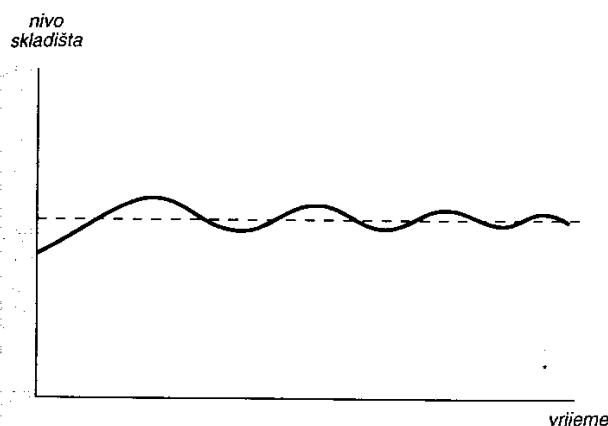
U pravilu su u modelima sistemske dinamike nivoi ovisni o brzinama, a brzine o nivoima (Wolstenholme, 1990). Izuzetak su samo kašnjenja koja su zapravo prikriveni nivoi u kojima su resursi zadržani neko vrijeme. Kašnjenja se mogu mjeriti. Zato i brzine mogu ovisiti o kašnjenjima, i obrnuto.

14.3. PRISTUP SISTEMSKE DINAMIKE

Sistemska dinamika je metoda koja podržava kontinuiranu simulaciju sistema. Sistemi što ih prikazuje sistemska dinamika složeni su sistemi s povratnom vezom. Opisuju se tako da se niz pojedinačnih događaja u sistemu *agregira* (povezuje) u jednostavnije



(a) Velika kašnjenja realizacije narudžbi



(b) Malá kašnjenja realizacije narudžbi

Slika 14.3. Fluktacija nivoa skladišta zbog kašnjenja u realizaciji narudžbi

koncepte *kontinuiranih tokova* (prosječni ravnomjerni tokovi entiteta). Na primjer, nabava skladišnih dijelova, koja se sastoji od niza diskretnih događaja sticanja pojedinačnih dijelova u skladište, prikazuje se kontinuiranim tokom dijelova, čiji se intenzitet može mijenjati tijekom vremena.

Treba naglasiti da procesi koji se odvijaju u sistemu u pravilu nisu lako uočljivi (Wolstenholme, 1990). Stoga je sistem potrebno promatrati iz "dovoljne udaljenosti", tj. u odgovarajućem vremenskom okviru i s odgovarajućim stupnjem agregacije.

Sistemska se dinamika koristi dvama tipovima grafičkih konceptualnih modela za prikaz sistema s povratnom vezom, dijagramima uzročnih petlji i dijagramima toka.

Dijagrami uzročnih petlji omogućuju jednostavan prikaz uzročno-posljedičnih veza između elemenata modela, uključujući i označavanje vrste veza (da li porast uzroka dovodi do porasta ili pada posljedica). Dijagrami toka omogućuju detaljniji opis tipova elemenata u modelu i karakteristika njihovih veza (uključujući i osnovne koeficijente veza).

Jednadžbe modela prikazuju se u obliku *diferencijskih jednadžbi*, tj. jednadžbi s konačnim razlikama vrijednosti varijabli modela. Tim se jednadžbama prikazuju promjene stanja sistema (tj. vrijednosti varijabli stanja) u vremenskim časovima udaljenim za tzv. *vremenski korak* modela (tj. odabranu najmanju vremensku jedinicu). Te se jednadžbe *integriraju* (tj. rješavaju) korištenjem različitih numeričkih integracijskih metoda.

Metode modeliranja i rješavanja jednadžbi modela realizirane su u raznim programskim jezicima sistemske dinamike, kao što su DYNAMO i STELLA. Moderni jezici sistemske dinamike omogućuju paralelan razvoj modela u obliku dijagrama toka i diferencijskih jednadžbi, te daju grafički izlaz ponašanja varijabli modela.

14.4. PODRUČJA PRIMJENE I CILJEVI UPOTREBE SISTEMSKE DINAMIKE

Sistemska dinamika ima vrlo široko područje primjene. Ipak, najviše je primjenjivana u području društveno-ekonomskih problema (rad poduzeća, odlučivanje u rukovođenju, simulacija tržišta, planiranje gradova, analiza budžeta, demografsko modeliranje, problemi u zdravstvu, energetici, okolišu i sl.). Osim toga, sistemska dinamika se koristi i u modeliranju procesa učenja, problema populacije, medicinskih problema te bioloških i kemijskih procesa.

Osnovni cilj upotrebe sistemske dinamike je omogućavanje razumijevanja utjecaja strukture sistema i strategija upravljanja sistemima na ponašanje sistema u vremenu. Ova metoda nije predviđena za dobivanje točnih rješenja problema, jer modelira nove strukture i strategije, već na poboljšavanje razumijevanja problema u toku procesa modeliranja. Iskustvo u modeliranju predstavlja ujedno i stjecanje iskustva koje omogućuje naalaženje boljih rješenja budućih problema u radu sa sistemom.

Neki od ostalih ciljeva upotrebe sistemske dinamike su (Wolstenholme,1990; Narchal,1988; Roberts i Dangerfield,1990):

- traženje struktura sistema i strategija upravljanja sistemom• koje poboljšavaju performanse rada sistema
- analiza efekata dugoročnih politika rada organizacija, industrija i vlada
- generiranje i analiza ponašanja sistema u vremenu
- analiza osjetljivosti sistema na promjene strukture sistema, upravljanja sistemom i vanjskih uvjeta (vladina politika, tržište i sl.)
- izbor strategije upravljanja sistema u realnom vremenu pomoću računala
- simulacijske igre koje omogućuju trening za rukovodioce koji nemaju dovoljno iskustva u radu sa sistemom (Meadows,1985; Sterman,1985).

Neke od suvremenih tendencija razvoja i primjene sistemske dinamike prikazane su u radu (Morecroft,1988).

15. KONCEPTUALNI MODELI SISTEMSKE DINAMIKE

Sistemska se dinamika koristi dvama tipovima konceptualnih modela: dijogramima uzročnih petlji i dijogramima toka, opisanim u ovom poglavlju. Dijagrami uzročnih petlji prikazuju uzročno-posljedične veze među elementima sistema, a dijagrami toka detaljnije opisuju veze između nivoa i brzina u sistemu. Konceptualni modeli omogućuju eksplicitni prikaz ideja koje ljudi imaju o radu sistema. Oni tako olakšavaju komunikaciju među ekspertima koji rade sa sistemom (inžinjerima, ekonomistima i sl.) i modelarima, a ujedno i olakšavaju razvoj simulacijskog programa i povećavaju njegovu pouzdanost.

15.1. DIJAGRAMI UZROČNIH PETLJI

15.1.1. Način opisa i osnovni pojmovi

Prvi je korak u analizi sistema s povratnom vezom pronaalaženje veza među elementima sistema te identifikaciju tipa tih veza i povratnih petlji u sistemu. Konceptualni modeli sistemske dinamike kojima se to prikazuje su dijagrami uzročnih petlji (Coyle,1977; Roberts i dr.,1983; Wolstenholme,1990).

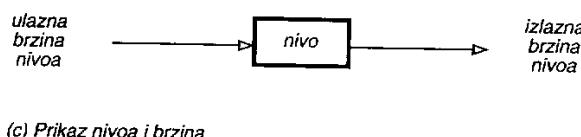
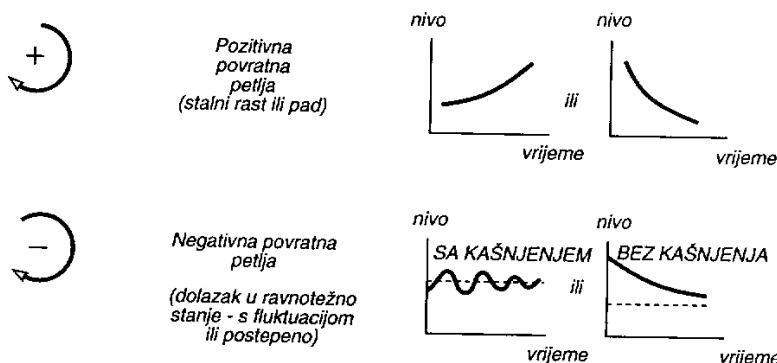
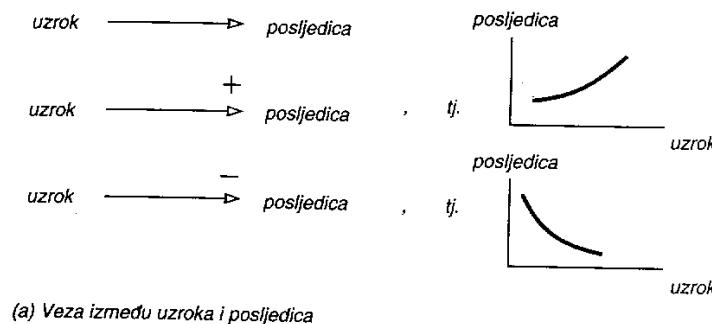
Dijagrami uzročnih petlji prikazuju uzročno-posljedičnu vezu među elementima sistema s povratnom vezom. Strelicom se označava smjer veze između uzroka i posljedice. Ako je na kraju strelice znak +, to znači da se uzrok i posljedica mijenjaju u istom smjeru: dakle, kada raste uzrok, raste i posljedica, a kada se smanjuje uzrok, smanjuje se i posljedica. Ako je na kraju strelice znak –, tada se uzrok i posljedica mijenjaju u suprotnom smjeru: dakle, kada uzrok raste, posljedica se smanjuje, i obratno. Ovi znakovi prikazani su na slici 15.1a.

Polukružna strelica u sredini petlje sa znakom + označava da je riječ o pozitivnoj povratnoj vezi u kojoj elementi petlje djeluju povratno na same sebe u istom smjeru. Zbog toga nastaje ili stalni rast ili stalno smanjenje veličine tih elemenata, i time nestabilno napuštanje ravnotežnog stanja sistema. Polukružna strelica u sredini petlje sa znakom – označava negativnu povratnu vezu u kojoj elementi petlje uzrokuju promjenu smjera vlastitog djelovanja. Zbog toga sistem ide prema ravnotežnom stanju, i to bilo postepeno bilo fluktuirajući. Ti znakovi prikazani su na slici 15.1b.

Način prikaza nivoa i brzina vidi se na slici 15.1c.

15.1.2. Određivanje tipa petlje

Za složenije uzročne petlje tip petlje se određuje prema tipu uzročno-posljedičnih veza unutar petlje. Ako su u petlji sve veze pozitivne, tada je to pozitivna povratna petlja.



Slika 15.1. Simboli korišteni u dijagramima uzročnih petlji

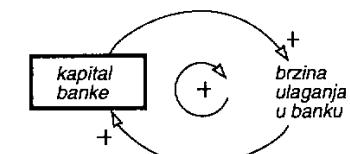
Ako u petlji ima negativnih veza, tip petlje ovisi o parnosti ukupnog broja negativnih veza: petlja s *parnim* brojem negativnih veza je pozitivna povratna petlja, a petlja s *neparnim* brojem negativnih veza je negativna povratna petlja. Dakle:

Samо pozitivne veze:	pozitivna povratna petlja
Paran broj negativnih veza:	pozitivna povratna petlja
Neparan broj negativnih veza:	negativna povratna petlja

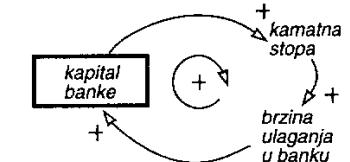
Ovo se pravilo može lako razumjeti, jer svako uvođenje negativne veze u povratnu petlju mijenja smjer promjena u petlji. Stoga svaki par negativnih veza u petlji vraća promjene u originalni smjer (tj. kao da negativnih veza i nema).

15.1.3. Pozitivne povratne petlje

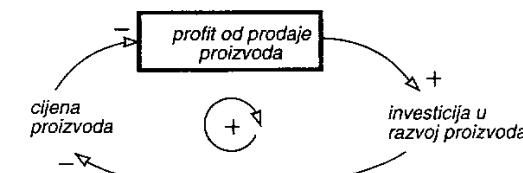
Najjednostavniji primjer *pozitivne povratne petlje* je onaj sa dva elementa petlje. Tako je na slici 15.2a prikazan kapital banke koji privlači ulagače u banku, a time se opet povećava kapital itd. U nešto složenijem primjeru veći kapital banke omogućuje povećanje kamata za ulagače, time se povećava broj ulagača, njihova ulaganja povećavaju kapital banke itd. Ova je petlja prikazana na slici 15.2b.



(a) Porast kapitala banke



(b) Porast kapitala banke povezan za porast kamatne stope

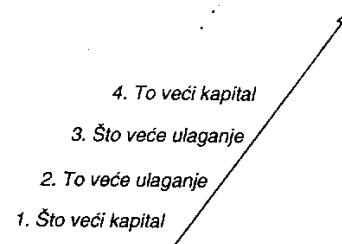


(c) Poduzeće koje investira u razvoj proizvoda (Wolstenholme, 1990)

Slika 15.2. Primjeri pozitivne povratne petlje

Na slici 15.2c prikazana je pozitivna povratna petlja poduzeća koje investira u razvoj proizvoda. Što je veći profit poduzeća, to više ono može investirati u razvoj proizvoda. Što je investicija u razvoj proizvoda veća, to proizvod može biti jeftiniji (npr. zbog razvoja nove efikasnije tehnologije ili proizvodnog procesa), a što je cijena proizvoda niža, profit je viši.

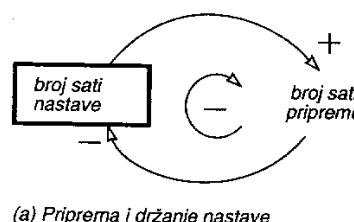
Na slici 15.3. prikazana je tendencija promjene veličina (Roberts i dr., 1983) u pozitivnoj povratnoj petlji sa slike 15.2a. Vidimo da kapital banke stalno raste.



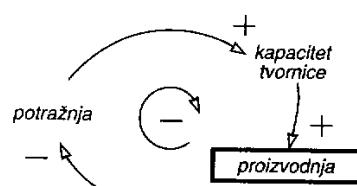
Slika 15.3. Tendencija promjene veličina iz pozitivne povratne petlje sa slike 15.2a (Roberts i dr., 1983)

15.1.4. Negativne povratne petlje

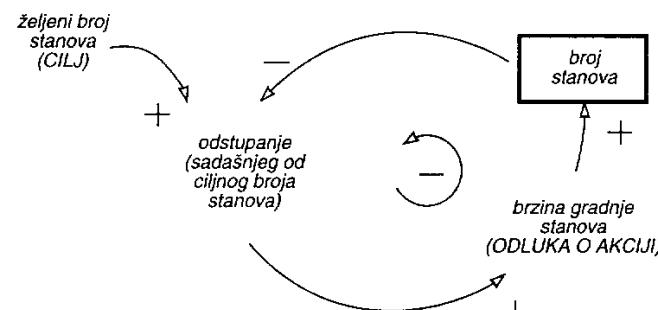
Jednostavan primjer negativne povratne petlje je model nastavnika koji priprema i drži nastavu. On za to troši određen broj sati tjedno (slika 15.4a). Što više sati nastave drži, to je duže vrijeme potrebno da bi tu nastavu dobro pripremio – ali što više sati priprema nastavu, to mu manje sati ostaje slobodno za držanje nastave itd. Na slici 15.4b prikazan je primjer proizvodnje i potražnje, u kojem povećana potražnja potiče ulaganja za proširenje kapaciteta tvornice, a veći kapacitet tvornice daje veću proizvodnju – što je veća proizvodnja, to je međutim potražnja više zadovoljena, pa se stoga potražnja smanjuje itd.



(a) Priprema i držanje nastave



(b) Potražnja i kapacitet proizvodnje



(c) Planiranje gradnje stanova

Slika 15.4. Primjeri negativne povratne petlje

Negativne povratne petlje vrlo često se susreću kod problema vezanih za ostvarenje određenih ciljeva. Na slici 15.4c prikazan je problem gradnje stanova u kojem je zadan cilj da se gradi određen broj stanova što nedostaje u nekom gradu. Što je cilj veći, tj. što je veći planirani broj stanova koji se trebaju sagraditi, to je veće *odstupanje* broja sadašnjih stanova od broja planiranih stanova. Što je to odstupanje veće, to će se stanovi brže sagraditi, a time više povećati ukupan broj stanova – ali što je veći ukupan broj stanova, to je odstupanje od planiranog broja stanova manje, tj. i brzina gradnje stanova će se smanjiti itd. Ovdje dakle veličina odstupanja od cilja utječe na odluku o intenzitetu akcije, a zadovoljavanje cilja smanjuje odstupanja pa time i intenzitet akcije (gradnja stanova).

Ovaj primjer demonstrira i uvođenje *egzogenih* (vanjskih) veličina koje utječu na rad sistema, ali na koje sistem nema utjecaja. U ovom primjeru egzogena veličina je planirani broj stanova, tj. zadan cilj. Sa proširenjem granica sistema, egzogene veličine se po potrebi također mogu uključiti u uzročno-posljedične petlje.

Na slici 15.5. prikazana je tendencija promjene veličina iz negativne veze prema slici 15.4a. Vidimo da se u ovom slučaju fluktuiraju vrijednosti elemenata petlje.



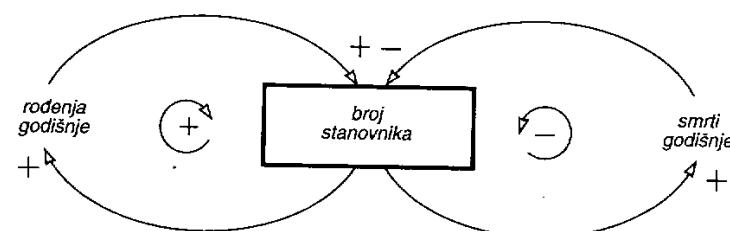
Slika 15.5. Tendencija promjena veličina iz negativne povratne petlje sa slike 15.4a (prema Roberts i dr., 1983)

15.1.5. Povezivanje pozitivnih i negativnih povratnih petlji

Gotovo da nema sistema od interesa koji se sastoji samo od jedne povratne petlje, već se sistemi opisuju većim brojem povratnih petlji povezanih preko elemenata koji sudjeluju u dvije petlje ili više njih. Analogno tome, i kod dijagrama ciklusa aktivnosti u simulaciji diskretnih događaja (poglavlje 4) međudjelovanje ciklusa života različitih entiteta odvija se u aktivnostima koje su zajedničke većem broju entiteta. I u jednom i u drugom slučaju je razvoj konceptualnog modela razumno napraviti pomoću razvoja pojedinačnih povratnih petlji (odnosno ciklusa života), i zatim njihovim povezivanjem u potpunim dijagramima uzročnih petlji (odnosno dijagram ciklusa aktivnosti). Druga mogućnost je postepeni razvoj međusobnih utjecaja elemenata sistema, i time izgradnja potpunog dijagrama.

Na nekoliko jednostavnih primjera demonstrirat ćemo povezivanje pozitivnih i negativnih povratnih petlji u dijagram uzročnih petlji ili jedan njegov dio (uvijek je naime pitanje odluke o granicama sistema koji promatramo da li ćemo dijagram uzročnih petlji smatrati potpunim).

Uzmimo najprije primjer sa slike 14.1. u kojem je prikazana pozitivna povratna petlja s godišnjim prirastom broja stanovnika, koja uzrokuje eksponencijalni porast ukupnog broja stanovnika. Na slici 15.6. prikazan je dodatni utjecaj negativne povratne petlje koja opisuje mortalitet na promjenu veličine populacije. Negativna povratna petlja dovodi do smanjenja efekta nataliteta, a može ga u potpunosti i anulirati ili čak uzrokovati stalni pad veličine populacije, ovisno o odnosu veličine stopa rada i umiranja.

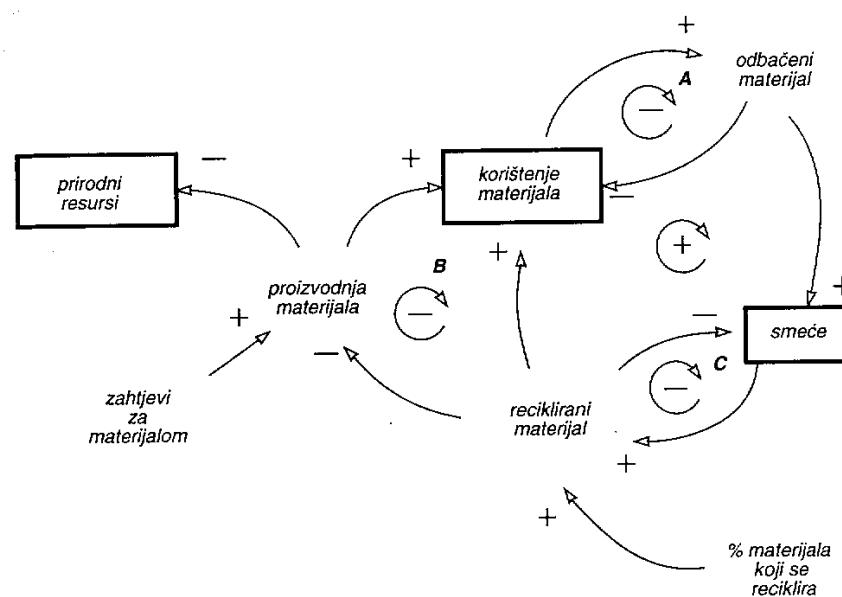


Slika 15.6. Model populacije s natalitetom i mortalitetom

Složeniji primjer sistema s povratnom vezom je model recikliranja prirodnih resursa (Roberts i dr., 1983). Zbog potrebe za materijalima raste njihova proizvodnja, a ona smanjuje ukupnu količinu prirodnih resursa potrebnih za proizvodnju. Što je više materijala proizvedeno, to je više materijala korišteno, pa je veća i količina iskorištenih odbačenih materijala. To utječe na porast količine smeća, čime se povećava količina recikliranih materijala koji sudjeluju u proizvodu (a ona raste i proporcionalno postotku materijala koji se reciklira). Porast količine recikliranog materijala povećava i korištenje materijala, a ujedno smanjuje potrebu za njegovom proizvodnjom. Dijagram uzročnih petlji modela prikazan je na slici 15.7.

U modelu možemo uočiti pozitivnu povratnu petlju: korištenje materijala – količina odbačenog materijala – količina smeća – količina recikliranog materijala, i tri negativne

povratne petlje. Petlja A: korištenje materijala – količina odbačenog materijala; petlja B: proizvodnja materijala – korištenje materijala – količina odbačenog materijala – količina smeća – količina recikliranog materijala; petlja C: količina smeća – količina recikliranog materijala. O odnosu veličine efekata u pozitivnim i negativnim povratnim petljama ovisi dominantno ponašanje sistema.



Slika 15.7. Model recikliranja prirodnih resursa (Roberts i dr., 1983)

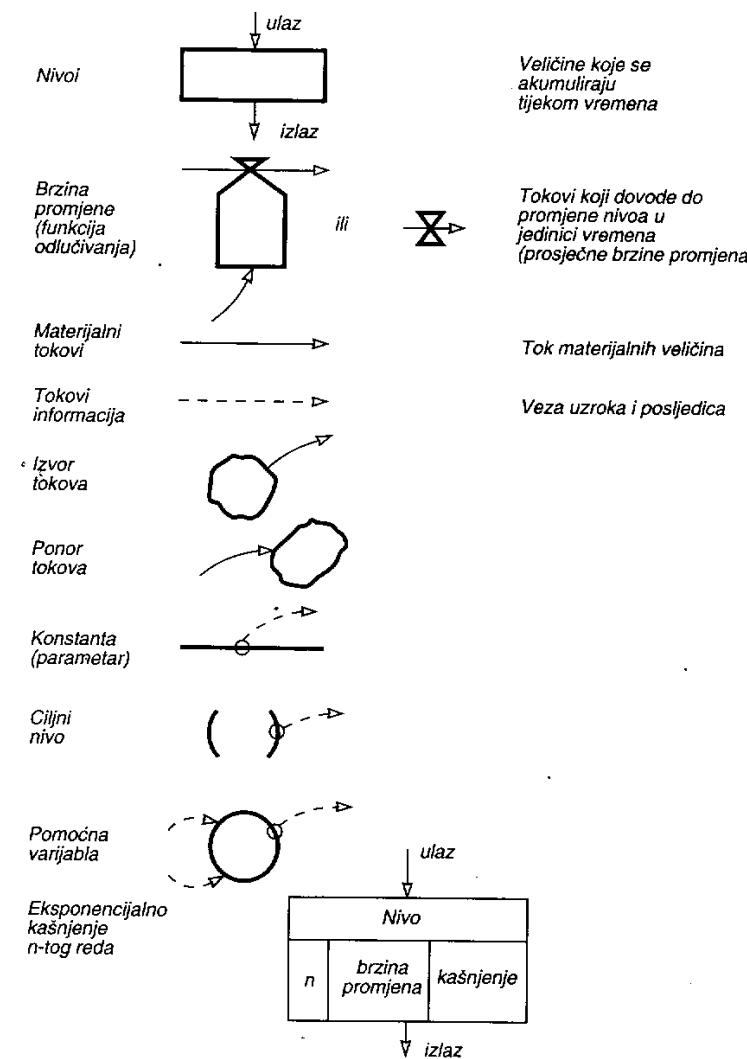
15.2. DIJAGRAMI TOKA

15.2.1. Način opisa i osnovni pojmovi

Pošto je konstruiran dijagram uzročnih petlji sistema s povratnom vezom koji promatramo, potrebno je identificirati nivoe, brzine i kašnjenja u sistemu, i tako preciznije prikazati model sistema. Konceptualni modeli kojima se to postiže su dijagrami toka (Forrester, 1961; Roberts i dr., 1983). Time se ujedno omogućuje jednostavniji i pouzdaniji prijelaz na formiranje računarskog modela (programa), odnosno opis jednadžbi modela u odabranom simulacijskom jeziku sistemske dinamike. Računarski model služi izvođenju simulacija, a time i oponašanju razvoja sistema u vremenu.

Osnovni simboli koji se koriste u dijagramima toka prikazani su na slici 15.8. Neki od korištenih pojmove detaljnije su opisani u paragrafu 14.3. Ukratko, nivoi su veličine koje se akumuliraju tijekom vremena (ulazni tokovi povećavaju a izlazni smanjuju

akumulaciju), a prosječne *brzine* (funkcije odlučivanja) tokovi su koji čine ulaze i izlaze nivoa i kojima se može upravljati. *Materijalni tokovi* su tokovi mjerljivih veličina, a *tokovi informacija* povezuju uzroke i posljedice u modelu. *Izvori* su počeci tokova koji su izvan sistema, a *ponori* su odredišta tokova koja su također izvan sistema. *Konstante* (parametri) nepromjenljivi su faktori modela, a *pomoćne varijable* omogućuju opis algebarskih operacija. *Eksponencijalna kašnjenja trećeg reda* bit će objašnjena nešto kasnije.



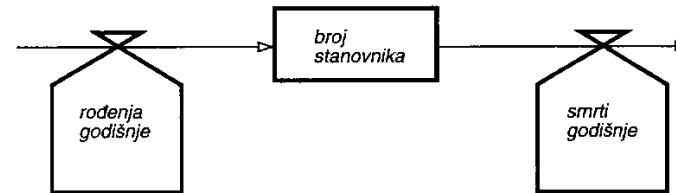
Slika 15.8. Simboli korišteni u dijagramima toka

Unaprijed se ne može reći što će u kojem modelu biti nivoi, tokovi itd., već to (kao i pri određivanju tipa entiteta u dijagramima ciklusa aktivnosti ili u simulacijskom jeziku GPSS) ovisi o sposobnosti apstrakcije i maštii onoga tko razvija model. Ta proizvoljnost svakako nosi određeni rizik stvaranja modela koji ne opisuju ispravno stvarni sistem, ili su nepotrebno komplikirani i dovode do neefikasnih simulacijskih programa. S druge strane, ta proizvoljnost pruža i izvjesne slobode i mogućnosti nalaženja neočekivanih prilagodavanja apstraktnih elemenata modeliranja objektima koje modeliramo. Time ni područje modeliranja nije unaprijed ograničeno, već se može proširiti i preko granica predviđenih kod stvaranja ove tehnike modeliranja. Pri određivanju tipa varijable sistema može pomoći promatranje statičke situacije u kojoj su nivoi prepoznatljivi, dok su brzine (tokova) jednake nuli.

15.2.2. Primjeri dijagrama toka

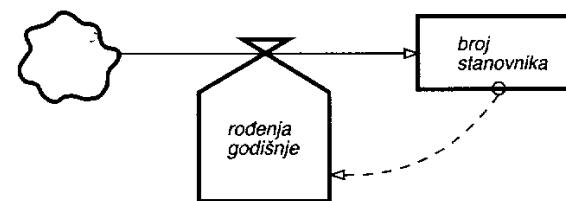
Prikazat ćemo nekoliko primjera konstruiranja dijagrama toka korištenjem opisanih simbola, kako bi se stekao osjećaj za vrstu objekata kojima se ti simboli tipično pridružuju.

U primjeru promjene veličine populacije broj stanovnika je nivo, a godišnje stopu rađanja i umiranja su brzine ulaza i izlaza tog nivoa, kao što se vidi na slici 15.9. Vidi se da broj stanovnika raste zbog rađanja, a pada zbog umiranja jedinki populacije. Strelice su pri tome tokovi promjene broja stanovnika, kako za njegovo povećavanje tako i za smanjivanje.



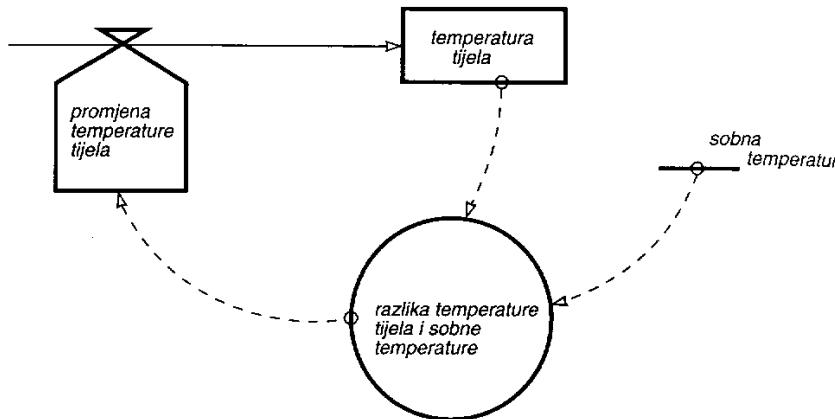
Slika 15.9. Primjer nivoa, brzine tokova i materijalnih tokova

Na slici 15.10. vidimo utjecaj koji broj stanovnika ima na rađanje: što je više stanovnika, to ih se više i rađa (uz određenu stopu nataliteta). Izvor toka broja stanovnika pokazuje da tokovi nastaju izvan granice sistema.



Slika 15.10. Primjer toka informacija i izvora tokova

Na slici 15.11. prikazana je pomoćna varijabla koja mjeri odstupanje stanja nivoa od planirane ciljne veličine nivoa. Ta varijabla djeluje dalje na upravljanje brzinom promjene nivoa. To je demonstrirano na primjeru promjene temperature tijela koje se hlađi (ili zagrijava) na sobnu temperaturu (ciljna temperatura). Sobna temperatura prikazana je kao konstanta.



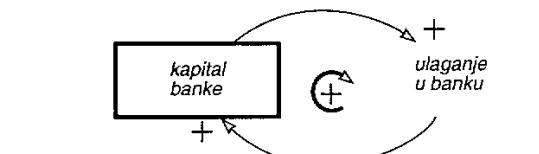
Slika 15.11. Primjer pomoćne varijable i konstante

15.2.3. Razvoj dijagrama toka na osnovi dijagrama uzročnih petlji

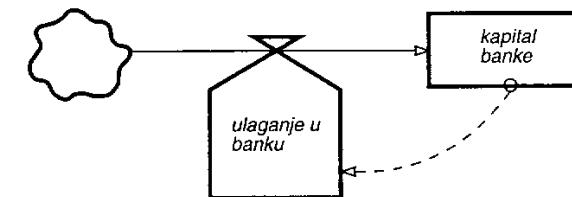
Dijagrami uzročnih petlji prikazuju model međusobnog utjecaja elemenata sistema. Da bismo ih pretvorili u dijagrame toka, potrebno je definirati tip elemenata (nivoi, brzine, pomoćne varijable itd.) te relevantne konstante sistema (karakteristike stabilnog stanja, konstante kašnjenja i sl.). Sa nekoliko karakterističnih primjera pokazat ćemo kako je moguće napraviti tu transformaciju između ova dva tipa konceptualnih modela sistemske dinamike.

Za početak opišimo transformaciju *jedne pozitivne povratne petlje* u dijagram toka. Na slici 15.12. prikazana je transformacija pozitivne povratne petlje porasta kapitala banke u dijagram toka. Kapital banke je nivo, a ulaganje u banku brzina. Nivo kapitala povećava se ulaganjem u banku, a informacija o kapitalu banke uzrokuje daljnje povećanje ulaganja u banku. Dakle, nivo direktno utječe na brzinu ulaznog toka u samoga sebe.

Transformacija jedne *negativne povratne petlje* s postavljenim *ciljem* prikazana je na slici 15.13. Tu je prikazan model gradnje stanova što se odvija u skladu s ciljnim (planiranim) brojem stanova. Broj stanova je nivo, a odstupanje od želenog broja stanova je pomoćna varijabla koja upravlja brzinom gradnje. Porast nivoa broja stanova S dovodi do smanjenja odstupanja ciljnog broja stanova S_c od postojećeg broja stanova ($S_c - S$), a to odstupanje direktno utječe na brzinu ulaznog toka gradnje stanova. Brzina gradnje upravljana je još i tzv. konstantom *vremena kašnjenja*. To je vrijeme povezano za kašnjenje gradnje stanova, tj. potrebno vrijeme da se realizira odluka o gradnji stanova. O različitim tipovima kašnjenja i njihovu prikazivanju govorit ćemo u sljedećem poglavljju, gdje ćemo opisati kašnjenja u jednadžbama modela.

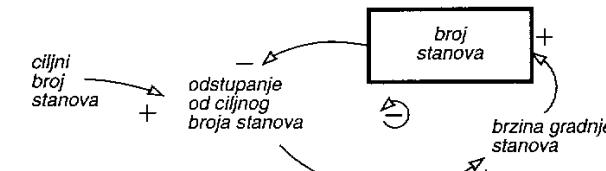


(a) Dijagram uzročnih petlji

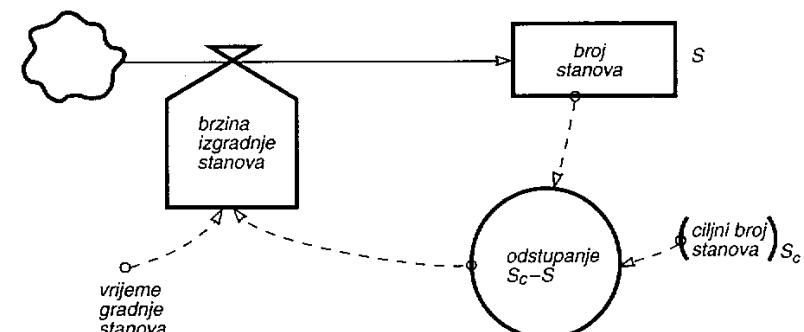


(b) Dijagram toka

Slika 15.12. Transformacija jedne pozitivne povratne petlje u dijagram toka — model porasta kapitala banke



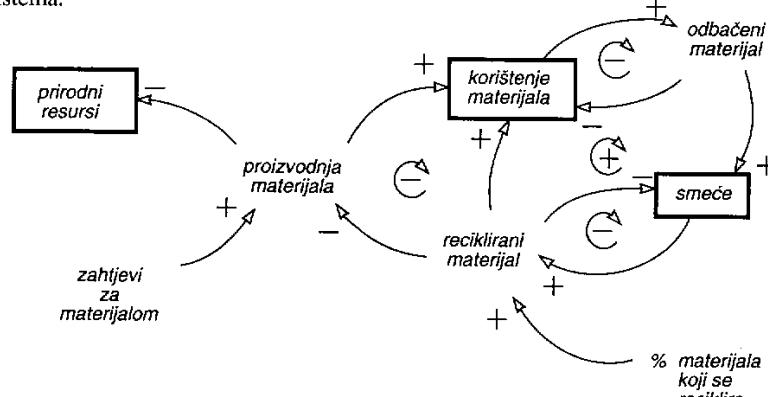
(a) Dijagram uzročnih petlji



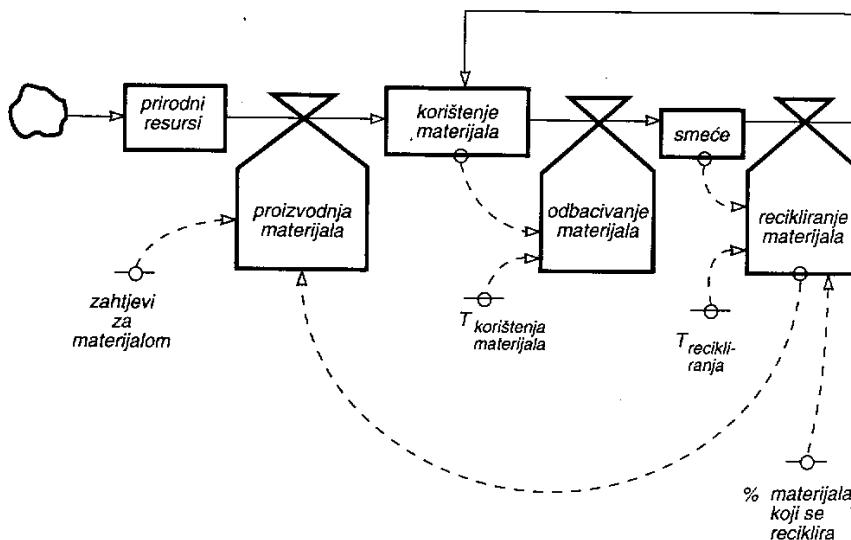
(b) Dijagram toka sistemske dinamike

Slika 15.13. Transformacija jedne negativne povratne petlje s postavljenim ciljem u dijagram toka — model gradnje stanova

Na slici 15.14. prikazan je *dijagram toka* razvijen iz *dijagrama uzročnih petlji* modela recikliranja prirodnih resursa. U njemu su *nivoi* prirodni resursi, korištenje materijala i stvoreno smeće. Proizvodnja, odbacivanje materijala i recikliranje su *brzine* koje utječu na promjenu nivoa modela. Tako proizvodnja materijala smanjuje nivo korištenih prirodnih resursa, te povećava korištenje materijala. Sama proizvodnja pri tome ovisi o dinamici pojavljivanja zahtjeva za materijalom te o iznosu recikliranog materijala. Uvedena su i dva kašnjenja, i to prosječno vrijeme korištenja materijala te prosječno vrijeme potrebno za recikliranje materijala, koja utječu na dinamiku rada sistema.



(a) Dijagram uzročnih petlji



(b) Dijagram toka

Slika 15.14. Dijagram uzročnih petlji i odgovarajući dijagram toka modela recikliranja prirodnih resursa

16. SIMULACIJSKE JEDNADŽBE, PROGRAMI I JEZIK DYNAMO

Na temelju dijagrama toka, kao detaljnog konceptualnog modela, stvara se računarski model (tj. simulacijski program). Simulacija se izvodi korištenjem odgovarajućih metoda za proračun razvoja sistema u vremenu. U ovom poglavlju opisan je način prikazivanja jednadžbi računarskog modela u simulacijskom jeziku DYNAMO, najpoznatijem programskom jeziku sistemske dinamike. To su diferencijske jednadžbe, tj. jednadžbe konačnih razlika, u kojima se vrijeme opisuje kao diskretna varijabla s konstantnim korakom promjene. Opisan je matematički pogled na modele sistemske dinamike, i to: diferencijalne jednadžbe modela sistemske dinamike, metode njihova numeričkog rješavanja te stabilnost i greške numeričkog rješenja. Posebno su opisani modeliranje kašnjenja i zaglađivanja informacija te izbor vremenskog koraka simulacije. Na kraju je kratak pregled simulacijskih jezika sistemske dinamike, stohastičke sistemske dinamike, optimizacije u sistemskoj dinamici te simulacijskog procesa u sistemskoj dinamici.

16.1. RAČUNARSKI MODELI SISTEMSKE DINAMIKE

Računarski modeli sistemske dinamike prikazuju kvantitativne veze među elementima sistema i omogućuju izvođenje kontroliranih simulacijskih eksperimenta s modelom sistema. Oni se prikazuju u obliku *sistema diferencijskih jednadžbi*, tj. jednadžbi konačnih razlika. Jednadžbe modela prikazuju promjene vrijednosti varijabli modela između uzastopnih vremenskih časova (trenutaka) udaljenih za određeni vremenski korak.

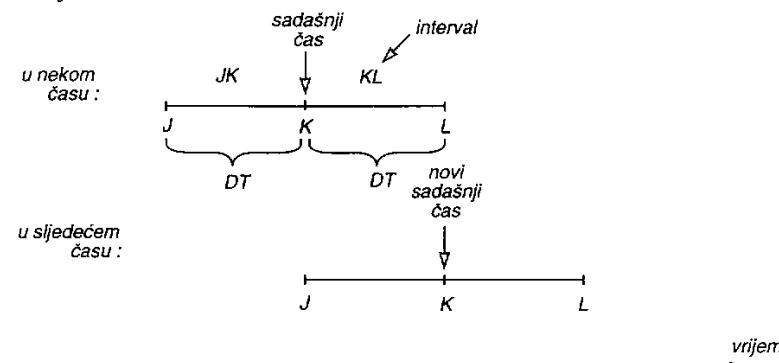
Da bi se dobilo ponašanje sistema u vremenu, odnosno izvela simulacija koja daje promjenu stanja modela u vremenu, potrebno je *integrirati* (tj. rješiti) sistem diferencijskih jednadžbi računarskog modela. Za to se koriste *numeričke metode integracije* koje daju rješenja sistema diferencijskih jednadžbi u uzastopnom slijedu vremenskih točaka međusobno udaljenih za zadani vremenski korak.

Različiti simulacijski jezici sistemske dinamike imaju izvjesne razlike u načinu opisivanja diferencijskih jednadžbi modela i koriste se različitim metodama numeričke integracije modela. Mi ćemo opisati način prikazivanja jednadžbi modela u najpoznatijem jeziku sistemske dinamike DYNAMO (DYNAMIC MOdels). Ovaj prikaz je vrlo specifičan i razlikuje se od standardnog matematičkog prikaza diferencijskih jednadžbi – ipak ga upotrebljavamo jer se on koristi (s manjim varijacijama) u cijelokupnoj literaturi sistemske dinamike.

16.2. OPISIVANJE PROMJENE VREMENA U JEDNADŽBAMA MODELA

Kao što smo rekli, vrijeme se pri izvođenju simulacije s modelima sistemskog dinamika pomicu u vremenskim koracima, koji se označuju DT. Veličinu vremenskog koraka bira modelar i o njoj, kao što ćemo to kasnije vidjeti, ovisi točnost rezultata i vrijeme izvođenja simulacije.

Tri uzastopna karakteristična trenutka u kojima se u sistemskoj dinamici opisuju promjene veličina modela u vremenu prikazani su na slici 16.1. Ti se vremenski trenuci označavaju:



Slika 16.1. Oznake za promjenu vremena u jeziku DYNAMO

- | | |
|---|--------------------------------|
| J | prethodni čas vremena, |
| K | sadašnji čas vremena (K=J+DT), |
| L | sljedeći čas vremena (L=K+DT), |

i oni su, kao što vidimo, međusobno udaljeni za vremenski korak DT. Vrijednosti nivoa u tim časovima označavaju se:

- | | |
|--------|-------------------------|
| nivo.J | nivo u prethodnom času, |
| nivo.K | nivo u sadašnjem času, |
| nivo.L | nivo u sljedećem času. |

Vrijednost prosječnih brzina promjena nivoa u intervalu između dvaju uzastopnih časova uzimaju se konstantnim, i označavaju se:

- | | |
|-----------|---|
| brzina.JK | brzina promjene nivoa između prethodnog i sadašnjeg časa, |
| brzina.KL | brzina promjene nivoa između sadašnjeg i sljedećeg časa. |

Dva osnovna tipa jednadžbi u jeziku DYNAMO su jednadžbe nivoa i jednadžbe brzine. Jednadžbe nivoa opisuju promjenu vrijednosti nivoa tijekom vremena, a jednadžbe brzina prikazuju način reguliranja tokova koji ulaze i izlaze iz nivoa. Proračun razvoja vrijednosti nivoa i brzina u vremenu odvija se ovako:

- (1) vrijeme se pomici na čas K;
- (2) na temelju jednadžbi nivoa izračunava se vrijednost nivoa u času K (uz pomoć vrijednosti nivoa u prethodnom času J, te brzine ulaznih i izlaznih tokova za taj nivo u prethodnom intervalu vremena JK);
- (3) na temelju jednadžbi brzina izračunavaju se vrijednosti brzina za sljedeći interval vremena KL (uz pomoć dobivenih vrijednosti nivoa u času K);

(4) nivo i brzine se preimenuju, tako da je K sada J, a L je K (zato da bi se nakon pomaka vremena moglo koristiti iste jednadžbe);

(5) vrijeme se pomiče za period DT, i novi sadašnji čas vremena ponovo nazivamo K. Cijeli ciklus se ponavlja sve dok simulacija ne završi (do zadatog vremena trajanja simulacije).

Da bi proračun mogao početi, moraju biti zadane potrebne vrijednosti veličina za trenutak početka simulacije (nivoi, brzine) te vrijednosti svih konstanti i parametara koje model sadrži.

Tako se dobiva uzastopni niz vrijednosti varijabli modela u vremenskim točkama međusobno udaljenim za vremenski korak DT. Taj diskontinuirani niz može predstavljati aproksimaciju kontinuiranog vremenskog ponašanja varijabli modela. Kako međutim u sistemskoj dinamici varijable stvarnog sistema često uopće nisu kontinuirane veličine, tada je dobiveni vremenski tok promjene vrijednosti varijabli diskontinuirani prikaz diskontinuirane varijable. Tada simulacija prikazuje aproksimaciju promjena u konstantnim vremenskim intervalima (čiju vrijednost modelar čak može slobodno mijenjati) dok se u stvarnom sistemu varijable tipično mijenjaju u nepravilnim vremenskim intervalima.

16.3. TIPOVI JEDNADŽBI MODELAA

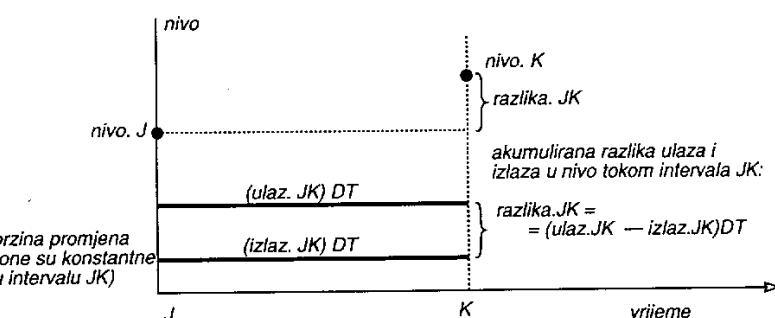
Osnovne jednadžbe modela su jednadžbe nivoa i jednadžbe brzine. Osim njih, koriste se i pomoćne jednadžbe, jednadžbe konstanti i jednadžbe početnih vrijednosti (termin jednadžbe koristi se u sistemskoj dinamici za sve vrste naredbi koje povezuju varijable i konstante modela).

(1) Jednadžbe nivoa

Jednadžbe nivoa prikazuju promjenu vrijednosti nivoa između dvaju uzastopnih vremenskih točaka. Jednadžba nivoa je vrijednost nivoa u novom sadašnjem času K, izražena vrijednošću istog nivoa u prethodnom času J i promjenom te vrijednosti koja je nastala zbog postojanja ulaznih i izlaznih tokova za taj nivo. Tipična jednadžba nivoa izgleda ovako:

$$\text{nivo.K} = \text{nivo.J} + (\text{DT}) (\text{ulaz.JK} - \text{izlaz.JK})$$

Ovdje je promjena vrijednosti nivoa izračunana kao prosječna promjena vrijednosti nivoa u jedinici vremena (ulaz.JK-izlaz.JK) pomnožena vremenom DT trajanja akumulacije tih konstantnih tokova između časova J i K. Promjena vrijednosti nivoa zbog razlike ulaznih i izlaznih tokova u nivo prikazana je na slici 16.2.



Slika 16.2. Promjena nivoa između časova J i K

(2) Jednadžbe brzina

Jednadžbe brzina prikazuju faktore koji utječu na tokove koji ulaze u nivo ili izlaze iz njega i kako utječu. Na brzine mogu utjecati vrijednosti pojedinih nivoa, a u jednadžbama se mogu pojaviti i različite konstante:

$$\text{brzina.KL} = f(\text{nivoi}, \text{konstante})$$

Tipičan oblik jednadžbe brzine je npr.

$$\text{brzina.KL} = \frac{1}{T} \text{nivo.K}$$

gdje je T konstanta s dimenzijom vremena. Ova jednadžba prikazuje brzinu promjene nivoa u periodu KL kao linearu funkciju veličine samog tog nivoa u prethodnom času K.

U sistemima kojima upravlja čovjek (tehnološki, ekonomski i sl.) jednadžbe brzina su pravila kojima se upravlja radom sistema, pa se one zovu i *funkcije odlučivanja*.

(3) Pomoćne jednadžbe

Pomoćne jednadžbe omogućuju pojednostavljeni pisanje jednadžbi brzina, i to tako da se one podijele u veći broj pomoćnih jednadžbi. Te se jednadžbe izvode poslije jednadžbi nivoa, a prije jednadžbi brzina.

(4) Jednadžbe konstanti

Jednadžbe konstanti omogućuju definiranje vrijednosti konstanti koje se u modelu koriste.

(5) Jednadžbe početnih uvjeta

Jednadžbe početnih uvjeta omogućuju postavljanje početnih vrijednosti nivoa i brzina za čas početka simulacije.

Različiti jezici sistemske dinamike prikazuju jednadžbe u oblicima koji se međusobno relativno malo razlikuju. Jezik DYNAMO zahtijeva označavanje tipa jednadžbe s karakterističnim slovom u prvom stupcu ispisa jednadžbe:

L ('Level')	jednadžbe nivoa,
R ('Rate')	jednadžbe brzina,
A ('Auxiliary')	pomoćne jednadžbe,
C ('Constant')	jednadžbe konstanti,
N ('Initial')	jednadžbe početnih vrijednosti.

16.4. MODELIRANJE KAŠNJENJA MATERIJALA

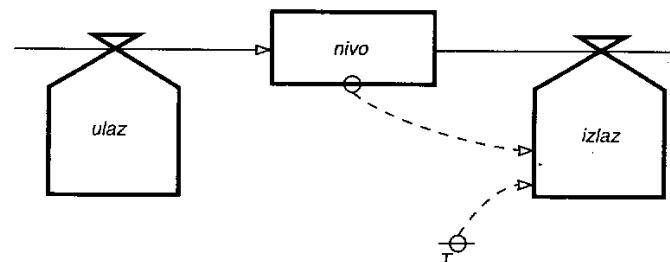
I materijali i informacije šire se konačnom brzinom, što znači da postoji određeno vrijeme *kašnjenja* od časa slanja materijala i informacija do časa njihova sticanja, odnosno vrijeme obradbe materijala i informacija (npr. vrijeme potrebno za sticanje naručenih dijelova, sticanje informacija o povećanoj potražnji za proizvodima, trajanje gradnje stanova ili vrijeme donošenja odluka). Što je više različitih tipova kašnjenja u sistemu, to više oni utječu na ponašanje sistema u vremenu, i to je teže procijeniti kakvo će to ponašanje biti. Stoga je potrebno adekvatno modelirati kašnjenja u sistemu kako bi model omogućio dobivanje dovoljno vjernih karakteristika sistema.

Neka kašnjenja u stvarnim sistemima mogu se prikazati jednostavnim pomakom za konstantni vremenski interval. Tako se na primjer pojedinačna narudžba realizira za određeno konstantno vrijeme poslije časa sticanja narudžbe. Međutim, ima kašnjenja koja su posljedica agregiranja velikog broja događaja (pojedinačnih kašnjenja) i koja imaju oblik postepenog reagiranja na inicijalnu promjenu. Primjer za to može biti sticanje velikog broja narudžbi koje se postepeno realiziraju nakon časa naručivanja. Kašnjenja toga tipa se vrlo često modeliraju sistemskom dinamikom, pa su ona stoga dosta detaljno razrađena i ovdje opisana.

16.4.1. Eksponencijalna kašnjenja prvog reda

Eksponencijalno kašnjenje je relativno jednostavan način prikaza kašnjenja koji se pokazao primjenljiv za većinu *oblika kašnjenja* u stvarnim sistemima. Oblik kašnjenja predstavlja oblik krivulje (odnosno tip odgovarajuće matematičke relacije) koja prikazuje dinamiku postepenog reagiranja varijabli modela na inicijalnu promjenu.

Eksponencijalno kašnjenje *prvog reda* u sistemskoj dinamici demonstrirat ćemo dijagramom toka prikazanim na slici 16.3. Tu se brzina izlaznog toka iz nivoa (tj. pražnjenje akumuliranog nivoa) kontrolira preko vrijednosti samog nivoa, uz određenu konstantu vremena kašnjenja T. Kašnjenje T prosječno je vrijeme potrebno izlaznom toku da se prilagodi trenutačnoj promjeni ulaznog toka. Odgovarajuće jednadžbe nivoa i brzina za to izgledaju ovako:



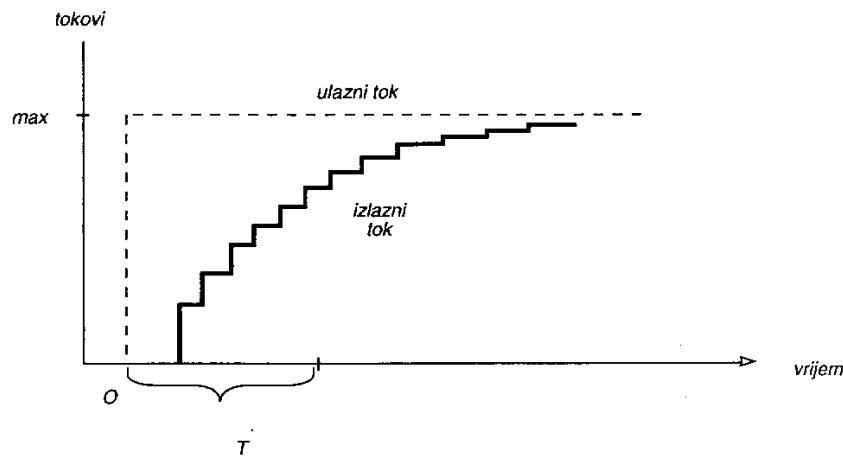
Slika 16.3. Dijagram toka za eksponencijalno kašnjenje prvog reda

$$\text{nivo.K} = \text{nivo.J} + (\text{DT}) (\text{ulaz.JK} - \text{izlaz.JK})$$

$$\text{izlaz.KL} = \frac{1}{T} \text{nivo.K}$$

tj. izračunana nova vrijednost nivoa u času K utječe na vrijednost brzine izlaznog toka iz tog nivoa u sljedećem vremenskom intervalu KL. Što je vrijednost kašnjenja T veća, to je manja dobivena vrijednost izlaznog toka iz nivoa, tj. to sporije će se izlazni tok prilagoditi nagloj promjeni ulaznog toka.

Prepostavimo da je sistem radio u stacionarnom stanju u kojem se vrijednost nivoa ne mijenja u vremenu, tj. izlazni tok jednak je ulaznom. Kada se brzina ulaznog toka naglo promjeni, npr. kada ona naglo poraste i ostane duže na toj vrijednosti, naglo poraste vrijednost nivoa. To uzrokuje porast brzine izlaznog toka, što ima kao posljedicu da se nešto uspori porast nivoa a time se nešto uspori i daljnji porast brzine izlaznog toka. Tako se brzina izlaznog toka postepeno približava promijenjenoj brzini ulaznog toka. Na taj način se kontrolom iznosa brzine izlaznog toka postiže postepeno prilagođavanje izlaznog toka ulaznom toku, i sistem se asymptotski (u beskonačnosti) približava novom stacionarnom stanju s povećanim izlaznim tokom i nivoom. Na slici 16.4. prikazana je promjena izlaznog toka koja odgovara naglom porastu ulaznog toka u nivo.



Slika 16.4. Prilagođavanje izlaznog toka naglom porastu ulaznog toka za eksponencijalno kašnjenje prvog reda

Porast izlaznog toka (pa prema tome i veličine nivoa, jer su te veličine proporcionalne) u vremenu ima eksponencijalni oblik. To se može vidjeti i iz prikaza jednadžbi kašnjenja prvog reda u obliku pripadnih diferencijalnih jednadžbi:

$$x_{t+dt} = x_t + dt (k - y),$$

$$y = \frac{1}{T} x,$$

gdje su

x nivo,

y brzina izlaznog toka,

k nova vrijednost ulaznog toka u času njegovog povećanja u $t = 0$,

odakle dobivamo

$$\frac{x_{t+dt} - x_t}{dt} = \left(k - \frac{x}{t} \right)$$

$$\frac{dx}{dt} = \frac{1}{T} (kT - x)$$

Uz pretpostavku da u času $t = 0$ nivo ima npr. vrijednost nula

$$x = 0 \quad \text{u času} \quad t = 0$$

dobivaju se rješenja diferencijalne jednadžbe u obliku:

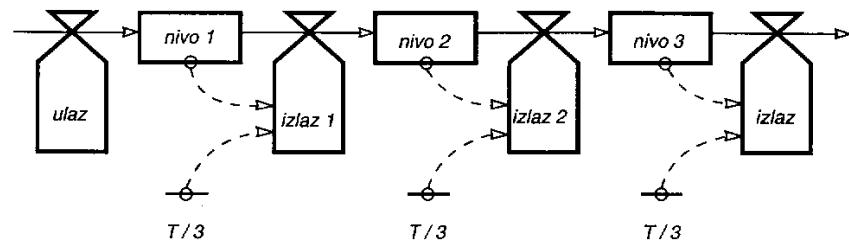
$$x = kT (1 - e^{-t/T}), \text{ tj.}$$

$$y = \frac{x}{T} = k \left(1 - e^{-t/T} \right),$$

koja imaju isti oblik (ali kontinuirani) kao i onaj na slici 16.4.

16.4.2. Eksponencijalna kašnjenja višeg reda

Eksponencijalna kašnjenja višeg reda imaju nešto promijenjeni oblik vremenske promjene vrijednosti varijabli modela u odnosu prema kašnjenju prvog reda. Takvi oblici kašnjenja dobivaju se serijom uzastopnih eksponencijalnih kašnjenja prvog reda. Na slici 16.5. prikazan je primjer dijagrama toka za eksponencijalno kašnjenje trećeg reda s konstantom kašnjenja $T/3$.



Slika 16.5. Dijagram toka za eksponencijalno kašnjenje trećeg reda

Jednadžbe nivoa i brzina za taj tip kašnjenja su:

$$\text{nivo } 1.K = \text{nivo1.J} + (DT) (\text{ulaz.JK} - \text{izlaz1.JK})$$

$$\text{nivo } 2.K = \text{nivo2.J} + (DT) (\text{izlaz1.JK} - \text{izlaz2.JK})$$

$$\text{nivo } 3.K = \text{nivo3.J} + (DT) (\text{izlaz2.JK} - \text{izlaz.JK})$$

$$\text{izlazl.KL} = \frac{1}{(T/3)} \text{nivo.K}$$

$$\text{izlaz2.KL} = \frac{1}{(T/3)} \text{nivo2.K}$$

$$\text{izlaz KL} = \frac{1}{(T/3)} \text{nivo3.K}$$

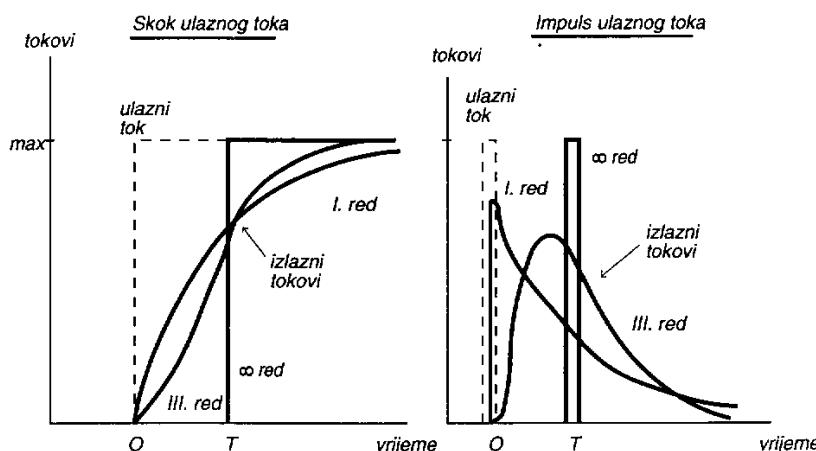
Pri tome je ukupno akumulirani iznos nivoa jednak

$$\text{nivo.K} = \text{nivo1.K} + \text{nivo2.K} + \text{nivo3.K}.$$

Nagla promjena brzine ulaznog toka se kod kašnjenja viših redova postepeno prenosi iz jednoga u drugi vremenski čas na porast nivoa 1, zatim porast brzine izlaznog toka 1, potom porast nivoa 2 itd. Time se postiže sporiji početak porasta ukupnog nivoa, ali se on postepeno ubrzava i u dalnjem toku vremena je brži nego što je to porast kod kašnjenja prvog reda.

Poseban slučaj je kašnjenje koje točno prenosi oblik promjene ulaznog toka ali s nekim vremenskim pomakom. Takav tip kašnjenja može se modelirati tzv. cjevovodom koji odgovara eksponencijalnom kašnjenju *beskonačnog reda*.

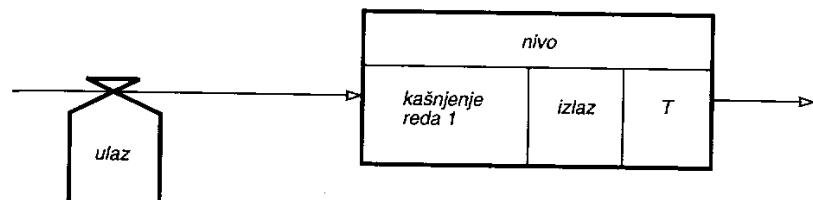
Praksa modeliranja sistemskom dinamikom pokazala je da se od svih tipova eksponencijalnog kašnjenja u stvarnim sistemima najviše koriste kašnjenja *prvoga, trećeg i beskonačnog reda*. Kašnjenje trećeg reda odgovara procesima s nekoliko tipova uključenih kašnjenja (npr. više faza dostave pošiljki), a kašnjenje beskonačnog reda odgovara odgođenim reagiranjima na promjene ulaznih tokova. Na slici 16.6. prikazani su oblici reakcije izlaznih tokova na dva tipa promjene ulaznih tokova: na nagli *skok* ulaznog nivoa uz trajno zadržavanje na njemu i na *impuls* promjene ulaznog toka koji naglo poraste i odmah se zatim vrati na početnu vrijednost.



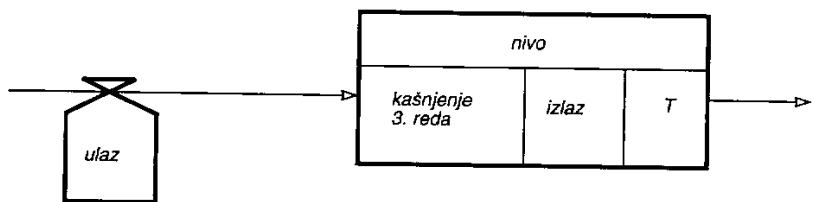
Slika 16.6. Reakcija izlaznih tokova na nagle promjene ulaznih tokova u sistemima s eksponencijalnim kašnjenjem prvog, trećeg i beskonačnog reda

16.4.3. Način označavanja eksponencijalnih kašnjenja

Eksponencijalno kašnjenje bilo kojeg reda možemo, umjesto crtanja složenih dijagrama toka s nizom serijski spojenih kašnjenja prvog reda, prikazati simbolom sa slike 15.8. Na slici 16.7. prikazani su skraćeni opisi kašnjenja prvoga i trećeg reda pomoću dijagrama toka koji odgovaraju potpunim opisima sa slika 16.3. i 16.5.



(a) Eksponencijalno kašnjenje 1. reda (potpun opis na slici 16.3)



(b) Eksponencijalno kašnjenje 3. reda (potpun opis na slici 16.5)

Slika 16.7. Skraćeni simboli za opis eksponencijalnog kašnjenja materijala

Isto tako moguće je napisati i skraćeni opis jednadžbi nivoa i brzina za eksponencijalna kašnjenja. Tako se, umjesto opisa kašnjenja trećeg reda pomoću sedam jednadžbi (vidi prethodni paragraf), može napraviti skraćeni opis korištenjem *standardne funkcije DELAY* iz biblioteke jezika DYNAMO:

$$\text{nivo.K} = \text{nivo.J} + (\text{DT}) (\text{ulaz.JK} - \text{izlaz.JK})$$

$$\text{izlaz.KL} = \text{DELAY3}(\text{ulaz.JK}, \text{T}).$$

16.4.4. Izbor reda eksponencijalnog kašnjenja

Kao što smo već naglasili, pravilan izbor oblika kašnjenja značajan je za dobar opis sistema koji modeliramo, iako se čini da točan izbor reda eksponencijalnog kašnjenja rijetko bitno utječe na ponašanje sistema. Metoda procjene reda kašnjenja temeljena na procjeni sredine i varijance razdiobe vremena kašnjenja opisao je (Holmes, 1970).

16.5. MODELIRANJE KAŠNJENJA INFORMACIJA

Kao i kod prijenosa materijala, tako je i za prijenos informacija i njihovo korištenje, odnosno donošenje odluka na temelju dostupnih informacija, potrebno vrijeme. Primjer kašnjenja u *prijenosu informacija* je vrijeme potrebno da bi informacije o prodaji proizvoda doprle do rukovodstva. Čak i kada su informacije dostupne, potrebno je vrijeme da se na određen način *iskoriste* (tj. obrade) kako bi mogle služiti za donošenje odgovarajućih odluka.

Eksponencijalno kašnjenje informacija ima svoje simbole dijagrama toka, koji su prikazani na slici 16.8a. Njima odgovara *standardna funkcija DLINF* ('delay of informations'), prikazana na primjeru:

$$\text{opažena vrijednost.K} = \text{DLINF3(vrijednost.K, T)}$$

koji prikazuje eksponencijalno kašnjenje trećeg reda opažene vrijednosti za stvarnom vrijednosti, uz konstantu kašnjenja T.

Odluke se često donose na osnovu *prognoze* o budućoj vrijednosti neke veličine napravljene korištenjem prethodnih informacija o vrijednosti te veličine. Prognoza na temelju jedne (posljednje) vrijednosti obično ne zadovoljava zbog fluktuacija veličina u vremenu, pa se u pravilu uzima veći broj prethodnih vrijednosti (vremenska serija) i na temelju njih se radi procjena prognozirane vrijednosti. Za to se najčešće koriste statističke metode temeljene na tzv. *zaglađivanju informacija*, koje na različite načine uzimaju prosjekе prethodnih promjena veličina od važnosti.

Jedna od tipičnih metoda zaglađivanja je tzv. metoda *pokretnih prosjeka*, gdje se procjena sljedeće vrijednosti uzima kao prosjek prethodnih opaženih vrijednosti:

$$\hat{y}_i = \frac{1}{n} \sum_{k=1}^n y_{i-k}$$

Različitim prethodnim vrijednostima moguće je dati *različitu težinu*. Tako se metoda *eksponencijalnog zaglađivanja* temelji na beskonačnom geometrijskom nizu članova različitih težina:

$$\hat{y}_i = \alpha \sum_{k=0}^{\infty} y_{i-1-k} (1-\alpha)^k$$

koja se može prikazati i u ovom obliku:

$$\hat{y}_{i+1} = \hat{y}_i + \alpha(y_i - \hat{y}_i)$$

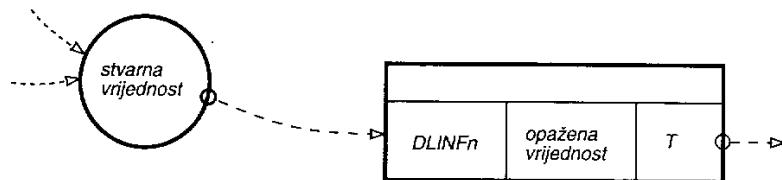
tj. sljedeća prognozirana vrijednost varijable \hat{y}_{i+1} dobiva se kao zbroj prethodne prognozirane vrijednosti \hat{y}_i i umnoška faktora α s greškom prognoze prethodne vrijednosti $(y_i - \hat{y}_i)$.

Zaglađivanje informacija dovodi do smanjenja utjecaja slučajne fluktuacije i također uvodi kašnjenje. Simbol dijagrama toka za zaglađivanje ('smoothing') informacija prikazan je na slici 16.8b. U jeziku DYNAMO koristimo se i standardnom funkcijom u ovom obliku:

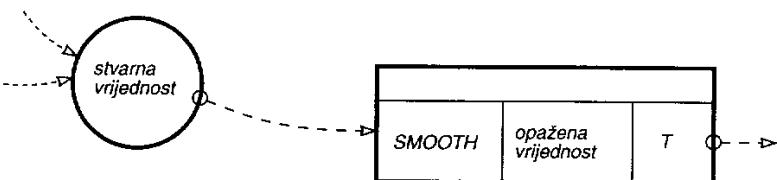
$$\text{projek.K} = \text{SMOOTH(ulaz.K, T)}$$

gdje je 'ulaz' varijabla koju želimo zagladiti (to može biti nivo, brzina, ili pomoćna varijabla), a 'projek' je izračunana zaglađena vrijednost te varijable.

16.6. PRIMJERI DYNAMO-PROGRAMA



(a) Eksponencijalno kašnjenje n-tog reda



(b) Zaglađivanje informacija (prognoza)

Slika 16.8. Simboli za opis kašnjenja informacija

16.6. PRIMJERI DYNAMO-PROGRAMA

Prikazat ćemo nekoliko primjera DYNAMO-programa za jednostavne sisteme s povratnom vezom. Opis naredbi potrebnih za specifikaciju ispisa rezultata dan je u sljedećem paragrafu, a ovdje su opisani samo elementi modela prikazani u dijagramu toka modela.

Prvi primjer je program koji odgovara *pozitivnoj povratnoj vezi* ulaganja u banku čiji je dijagram toka prikazan na slici 15.12. Odgovarajući program izgleda ovako:

L	KAPITAL.K=KAPITAL.J+(DT)(ULAGANJA.JK)
N	KAPITAL=1000
R	ULAGANJA.KL=KAPITAL.K/T
C	T=6

Pri tom je pretpostavljeno da je vremenska jedinica mjesec. Prva naredba (L) prikazuje jednadžbu porasta kapitala banke zbog ulaganja u banku tijekom jednog vremenskog koraka DT. Druga naredba (N) daje početnu vrijednost kapitala. Treća naredba (R) prikazuje brzinu ulaznog toka ulaganja proporcionalnu kapitalu, uz konstantu kašnjenja T. Posljednja naredba (C) postavlja vrijednost konstante kašnjenja na šest mjeseci.

Drugi primjer odgovara *negativnoj povratnoj vezi* pri gradnji stanova s upravljanjem brzine gradnje preko razlike ciljnog i sadašnjeg broja stanova, čiji je dijagram toka prikazan na slici 15.13. Odgovarajući program je:

```

L      STANOV.IK=STANOV.IJ+(DT)(IZGRAD.NJA.JK)
N      STANOV.I=40000
R      IZGRAD.NJA.KL=ODSTUPANJE.K/T
C      T=2
A      ODSTUPANJE.K=CILJ-STANOV.IK
C      CILJ=70000
  
```

Uzmimo da je vremenska jedinica godina. Konstanta kašnjenja gradnje stanova je dvije godine, sadašnji broj stanova je 40 000 a ciljni (planirani) je 70 000. Pomoćna varijabla (ODSTUPANJE) prikazuje odstupanje stvarnog broja stanova od planiranoga. Gradnja je to brža što više stanova nedostaje, a tada i broj stanova brže raste jer je odstupanje od cilja veće. Smanjenjem tog odstupanja smanjuje se i daljnja brzina gradnje.

16.7. SINTAKSA I DODATNE NAREDBE JEZIKA DYNAMO

Opisat ćemo osnove pravila pisanja kojima se koristi jezik DYNAMO, te dodatne naredbe za ispis i pokretanje izvođenja programa što omogućuju pisanje potpunih DYNAMO programa (Pugh,1976).

16.7.1. Sintaksa jezika DYNAMO

Naredbe jezika DYNAMO moraju biti ovakva oblike:

identifikator naredbe	praznina	tijelo naredbe
<i>Identifikator naredbe</i> je slovo ili riječ koja označava tip naredbe:		
*	glava (naslov) stranice	
L	nivo ('Level')	
R	brzina ('Rate')	
A	pomoćna varijabla ('Auxiliary variable')	
N	početna vrijednost nivoa ('Initial value')	
C	konstanta ('Constant')	
T	tablica ('Table')	
SPEC	specifikacija različitih veličina (DT, LENGTH, PLTPER, PRTPER)	
PLOT	naredba crtanja	
PRINT	naredba tiskanja	
RUN	kraj modela (izvođenje može početi)	
NOTE	komentar	
X	nastavak prethodne linije	

Praznina, ili više uzastopnih praznina, dijeli pojedine dijelove naredbe.

Tijelo naredbe je ili jednadžba ili funkcija. U pisanju jednadžbi ili funkcija ne smiju se koristiti praznine.

Linija može imati do 72 znaka, a od 73. znaka se daljnji znakovi ignoriraju. Linija se može nastaviti upisom znaka X u stupac 1 (iza njega mora biti praznina prije nastavka naredbe).

Slijed naredbi nije bitan, jer DYNAMO-prevodilac sam odreduje slijed izvođenja naredbi.

Jednadžbe se mogu koristiti ovim aritmetičkim operacijama:

+	zbrajanje
-	odbijanje
*	množenje → može biti i u obliku: (IZRAZ1)(IZRAZ2)
/	dijeljenje

16.7.2. Naredbe TABLE, STEP i PULSE

To su tri važne i često korištene naredbe.

(1) Naredba TABLE omogućuje specificiranje *nelinearnih funkcija* u modelu. Ona omogućuje da se definira neka pomoćna varijabla pomoću funkcije u obliku tablice:

A varijabla.K=TABLE(imetab,nezav.K,min,max,korak)

gdje je

varijabla.K,	pomoćna varijabla definirana pomoću funkcije u obliku tablice (vrijednost u času K),
imetab,	ime tablice,
nezav.K,	nezavisna varijabla za koju će se interpolirati vrijednost
tabelarno	zadane funkcije (vrijednost u času K),
min,	minimalni iznos ulazne varijable,
max,	maksimalni iznos ulazne varijable,
korak,	korak tabelarnih vrijednosti (razmak vrijednosti nezavisne varijable za koji su zadane vrijednosti funkcije).

Tabelarne vrijednosti funkcije zadaju se za uzastopne točke nezavisne varijable, i to od minimalne do maksimalne vrijednosti s razmakom za zadani korak:

T imetab=f₁/f₂/f₃/.../f_n

gdje je

f ₁ ,	vrijednost tabelarne funkc. u točki nezav=min,
f ₂ ,	vrijednost tabelarne funkc. u točki nezav=min+korak,
f _n ,	vrijednost tabelarne funkc. u točki nezav=max.

Primjer tablice je na slici 16.9a.

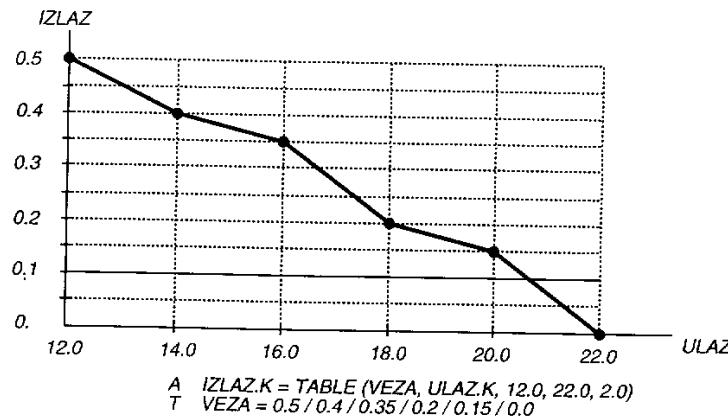
(2) Naredba STEP omogućuje simulaciju naglog skoka vrijednosti brzine toka na višu vrijednost, i ostajanje na toj vrijednosti. Ona se definira kao:

R varij.K=konst+STEP(visina,trenutak),

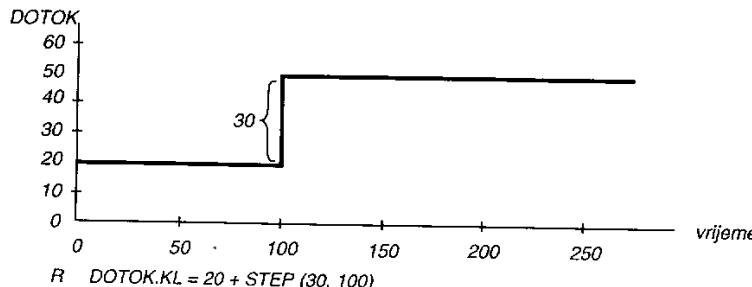
gdje je

visina, visina skoka (tj. povišenje vrijednosti brzine toka),
trenutak, vremenski trenutak skoka.

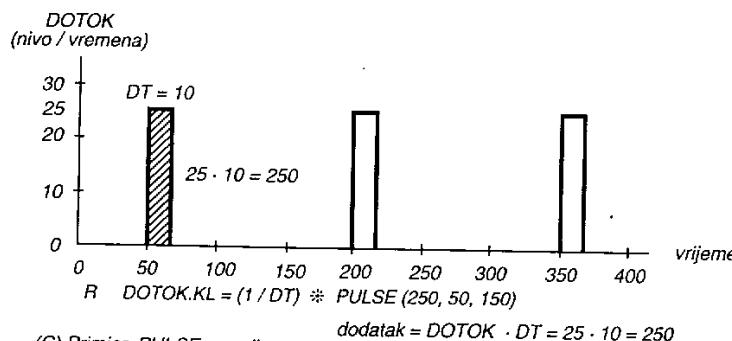
Primjer STEP funkcije na slici je 16.9b.



(a) Primjer TABLE naredbe



(b) Primjer STEP naredbe



(c) Primjer PULSE naredbe

Slika 16.9. Primjeri TABLE, STEP i PULSE naredbi

(3) Naredba PULSE omogućuje simulaciju pojave periodičkih naglih i kratkih impulsa vrijednosti brzine toka. Skok vrijednosti brzine toka na višu vrijednost traje jedan vremenski interval dužine DT, nakon čega vrijednost brzine toka pada natrag na prethodnu vrijednost. Naredba PULSE ima oblik:

R variј.K=PULSE(dodatak,trenutak,interval)

gdje je:

dodatak, dodatak nivou tijekom jednog pulsa (u dimenziji nivoa),
trenutak, vremenski trenutak prvog skoka,
interval, vremenski interval između uzastopnih pulseva brzine
ulaznog toka

(jer se dodatak nivou dobiva impulsom dotoka u trajanju DT, znači da se na temelju visine dotoka dodatak nivou može izračunati ovako:
dodatak = dotok * DT)

Primjer PULSE naredbe je na slici 16.9c.

16.7.3. Naredbe za izlaz i specifikaciju trajanja simulacije

Naredbe za izlaz i specifikaciju trajanja simulacije nisu dijelovi konceptualnog modela. One služe za definiciju izlaza modela te vremenskog koraka, trajanja simulacije i sl.

PLOT varijabla=znak(vrijedn1,vrijedn2)/...

PLOT naredba traži crtanje toka vrijednosti varijabli koje su u naredbi navedene, i to uz pomoć znakova koji su navedeni. Skala crtanja za svaku varijablu (ne mora se definirati u programu) ide od vrijednosti 'vrijedn1' do 'vrijedn2'.

PRINT varijabla1,varijabla2, ...

PRINT naredba traži tiskanje numeričkih vrijednosti navedenih varijabli u kolonama.

SPEC DT=broj/LENGTH=broj/PLTPER=broj/PRTPER=broj

SPEC naredba služi za specifikaciju vrijednosti:

DT vremenski interval simulacije,

LENGTH vrijeme trajanja simulacije (izraženo u vremenskim jedinicama simulacije),

PLTPER (PLot PERiod) vremenski interval nakon kojeg će se crtati vrijednosti označenih varijabli, izražen u vremenskim jedinicama simulacije,

PRTPER (PRinT PERiod) vremenski interval nakon kojeg se tiskaju vrijednosti označenih varijabli.

16.7.4. Naredbe za izvođenje simulacije

Dvije su naredbe za izvođenje simulacije. Prva od njih:

RUN

označava da je to kraj modela i da *izvođenje* simulacije može početi. Druga naredba je
ENTER RERUN CHANGES

Pošto je prethodno izvođenje završeno, može se ovako tražiti novo izvođenje programa. Iza ove naredbe mogu se također *promijeniti konstante* modela s odgovarajućim naredbama, i ponovo tražiti izvođenje naredbom **RUN**.

U novijim verzijama programa DYNAMO zahtjevi za izvođenjem daju se preko interaktivnog rada s menijima.

16.7.5. Primjer potpunog DYNAMO programa

Upotpunit ćemo program ulaganja u banku čiji je dijagram toka prikazan na slici 15.12.

```
*      ULAGANJE U BANKU
L      KAPITAL.K=KAPITAL.J+(DT)(ULAGANJA.JK)
N      KAPITAL=1000
NOTE   - BRZINA ULAGANJA KAPITALA -
R      ULAGANJA.KL=KAPITAL.K/T
C      T=6
NOTE   - CRTANJE I ISPIS VARIJABLI -
PLOT  KAPITAL=K/ULAGANJA=U
PRIN   KAPITAL,ULAGANJA
SPEC   DT=1/LENGTH=150/PLTPER=1/PRTPER=10
RUN
```

16.8. IZBOR VELIČINE VREMENSKOG KORAKA MODELA

Na početku ovog poglavlja uveli smo vremenski korak DT koji predstavlja minimalni iznos promjene vrijednosti vremena u modelu. Simulacija počinje od vremena 0 i pomiče se korak po korak (0, DT, 2DT, 3DT, ...), i u tim uzastopnim časovima simulacije mijenja se stanje sistema, odnosno vrijednosti varijabli sistema. Vremenski korak izražen je u odgovarajućim vremenskim jedinicama (sati, minute, sekunde, milisekunde i sl.) koje modelar odabire kao prikladne za sistem koji modelira.

Postavlja se pitanje kakvu veličinu vremenskog koraka DT treba izabrati za pojedini sistem. Intuitivno je jasno da za rastuće vrijednosti DT rezultati simulacije imaju u sebi sve veću grešku metode računanja koja potječe od sve grublje aproksimacije kontinuiranosti promjene vremena. Stoviše, može se pokazati da tada i rješenje postavljenih diferencijskih jednadžbi modela postaje *nestabilno*, te da rezultati simulacije ne odgovaraju razvoju stvarnog sistema u vremenu. Ovakvo ponašanje rezultata simulacije proistjeće iz karaktera numeričkog rješavanja jednadžbi modela, i poznato je u numeričkoj matematici. Pitanje izbora vrijednosti za DT je posebno značajno, jer je poznato da su metode integracije diferencijskih jednadžbi vrlo osjetljive na vrijednost vremenskog koraka DT i da teže širenju grešaka nastalih tijekom proračuna.

Kod padajućih vrijednosti DT točnost rješavanja jednadžbi simulacijskog modela je sve veća, ali uz cijenu sve dužeg vremena izvođenja simulacije jer se jednadžbe modela rješavaju to više puta što je manji vremenski korak DT.

Prema tome, potrebno je pronaći prikladnu vrijednost vremenskog koraka koja će osigurati zadovoljavajuću točnost rješavanja jednadžbi modela i stabilnost modela, ali pri tome neće zahtijevati predugo vrijeme računanja. Ne postoji egzaktni način kako da se odredi zadovoljavajuća vrijednost vremenskog koraka DT, ali su razvijena dva moguća pristupa ovom problemu.

1. Izvođenje probnih eksperimenata

Prije izvođenja eksperimenata za ispitivanje ponašanja sistema izvode se probni (pilot) eksperimenti relativno kratkog trajanja. U tim se eksperimentima kreće od neke relativno velike vrijednosti za DT, i ona se iz eksperimenta u eksperiment smanjuje. Prihvata se ona vrijednost DT za koju se rezultati simulacije prestaju značajno razlikovati od onih u kojima je DT imao prethodnu višu vrijednost (procjena značajnosti razlikovanja rezultata povezana je za prihvatljivu točnost rezultata simulacije).

2. Preporuke za izbor vremenskog koraka DT

Postoje neke preporuke (npr. Forrester, 1961) o najvećoj dopuštenoj vrijednosti DT, temeljene na karakterističnim kašnjenjima u sistemu. Za eksponencijalno kašnjenje prvog reda s konstantom kašnjenja T smatra se da DT treba obavezno biti manji od T/2 (neki autori smatraju da DT mora biti čak 3–10 puta manji od T). Ako ima više različitih kašnjenja u sistemu, uzima se kao mjerodavno najmanje kašnjenje (tj. ono s najmanjim T). Za kašnjenje n-tog reda traži se da je DT manji od T/2n, a ako ima više takvih kašnjenja, uzima se također najmanje kašnjenje kao mjerodavno za izbor DT.

Ova se dva pristupa izboru vrijednosti vremenskog koraka DT mogu kombinirati, i to tako da se najprije preporukom o izboru vremenskog koraka na temelju konstanti kašnjenja modela napravi procjena vrijednosti DT, a zatim da se izvedu probni eksperimenti s određenim rasponom vrijednosti DT oko procijenjene vrijednosti. Na temelju probnih eksperimenata definitivno se izabire prikladna vrijednost vremenskog koraka.

16.9. OSTALI PROGRAMSKI JEZICI SISTEMSKE DINAMIKE

Programski jezik DYNAMO razvijen je usporedno s razvojem ideje sistemske dinamike i postao je gotovo sinonimom za sistemsku dinamiku, osobito u pisaniju jednadžbi modela. Jezik DYNAMO u međuvremenu se dosta razvio, i njegova moderna varijanta je Professional DYNAMO (Pugh-Roberts Associates, 1986) koja npr. omogućuje automatsko prevođenje algebarskih imena u definiciju varijabli, indeksiranje varijabli, jednadžbi i dijagrama, grafički prikaz rezultata simulacije te interaktivni pristup izgradnji modela i izvođenju simulacije korištenjem menija. Osim toga razvijen je i prevodilac DYNASTAT koji omogućuje statističku analizu izlaznih rezultata DYNAMO programa, te jezik Gaming DYNAMO koji omogućuje izvođenje simulacijskih igara, i to tako što se simulacija prekida u pravilnim vremenskim intervalima te se sudionicima u igri omogućuje da analiziraju dotadašnji tok simulacije i mijenjaju vrijednosti varijabli odlučivanja.

Razvijeno je i nekoliko drugih jezika sistemske dinamike. Jedan od najpopularnijih jezika danas je STELLA (Richmond, 1985), jezik orijentiran na rad s računalima Macintosh, dakle i na grafički način rada. Grafičkim simbolima koji se izabiru i međusobno povezuju na ekranu gradi se dijagram toka, a pri tome sam softver kontrolira ispravnost uspostavljenih veza u skladu sa zadanim pravilima formiranja dijagrama. Jednadžbe modela moraju se koristiti upravo simbolima (imenima varijabli) definiranim na dijagramima, tako da se jednadžbe lako povezuju s dijagramima toka. Rezultati simulacije prikazuju se također u grafičkom obliku.

Jezik DYNSMAP (Cavana i Coyle, 1982) ima razvijenu interaktivnu okolinu u kojoj se jednostavno formiraju i prikazuju različiti tipovi grafičkih izlaza visoke rezolucije, te mijenjaju ili dodaju jednadžbe modela. Može se čak formirati graf jedne varijable prema vrijednosti te iste varijable iz prethodnih izvođenja simulacije.

Jezik DYNSMAP sadrži i modul koji izvodi potpunu dimenzionalnu analizu (jedinice mjerena za varijable u jednadžbama modela). Jezici Professional DYNAMO i STELLA omogućuju pridjeljivanje dimenzionalnosti koje se pojavljuje uz ispis jednadžbi modela.

16.10. MATEMATIČKE OSNOVE SISTEMSKE DINAMIKE

Prikazat ćemo osnovne elemente matematičkog prikaza modeliranja i rješavanja modela sistemske dinamike. Navest ćemo diferencijalne jednadžbe, njihovo pretvaranje u diferencijske jednadžbe, osnovne metode numeričkog rješavanja diferencijskih jednadžbi i svojstva tih numeričkih metoda. Prikazat ćemo i razliku pristupa matematičkog modeliranja i sistemske dinamike modeliranju i rješavanju modela.

Ovaj odjeljak teksta namijenjen je prije svega čitaocima koje zanima mehanizam rješavanja problema sistemske dinamike te njezina matematička podloga.

16.10.1. Diferencijalne i diferencijske jednadžbe

U fizici, biologiji, tehniči i drugim područjima fenomeni koji doživljavaju kontinuirane promjene u prostoru i vremenu u pravilu se predočavaju *diferencijalnim jednadžbama*, tj. jednadžbama koje opisuju beskonačno male promjene vrijednosti varijabli. Ako je npr. brzina promjene varijable y u ovisnosti o varijabli x zadana kao funkcija varijabli x i y , imat ćemo jednadžbu:

$$\frac{dy}{dx} = f(x, y) \quad (16.1)$$

uz početni uvjet u točki x_0 :

$$y(x_0) = y_0 \quad (16.2)$$

Ako ovakva diferencijalna jednadžba nema rješenje $y(x)$, tada se mora pribjegavati *numeričkom rješavanju* diferencijalne jednadžbe, i to tako što se diferencijali (beskonačno male razlike) zamjenjuju diferencijama (konačnim razlikama) vrijednosti varijabli. Tu se pojavljuje i problem pouzdanosti rješenja, jer sve veličine konačnih razlika vrijednosti varijabli ne daju točno i stabilno rješenje originalne diferencijalne jednadžbe.

Kao što smo vidjeli, u sistemskoj dinamici se problem originalno formulira u obliku *diferencijskih jednadžbi*, tj. jednadžbi konačnih razlika. Jedan od razloga za to je činjenica da sistemi koji se modeliraju i rješavaju sistemskom dinamikom (npr. ekonomski ili društveni) u pravilu nisu po svojoj prirodi kontinuiranog karaktera, već su to sistemi s mnogo diskretnih elemenata. Sistemska dinamika te elemente prikazuje u agregiranom obliku, a njihove promjene aproksimira diferencijskim jednadžbama. Drugi je razlog tradicija postavljanja jednadžbi u sistemskoj dinamici u obliku koji izavno vodi na numeričko rješavanje pomoću računala. U sistemskoj dinamici se dakle odmah formiraju jednadžbe konačnih razlika s određenim vrijednostima konačnih razlika nezavisne varijable (tj. vremenskog koraka), i te se jednadžbe mogu rješavati odabranim numeričkim metodama.

Budući da je numeričko rješavanje diferencijalnih jednadžbi veoma razvijeno područje numeričke matematike, metode koje su u tom području razvijene koriste se i u rješavanju jednadžbi modela sistemske dinamike. U ovom ćemo paragrafu navesti najviše korištene numeričke metode rješavanja diferencijalnih jednadžbi, jer su jednadžbe modela sistemske dinamike zapravo jednadžbe konačnih razlika za *prirodne diferencijalne jednadžbe*.

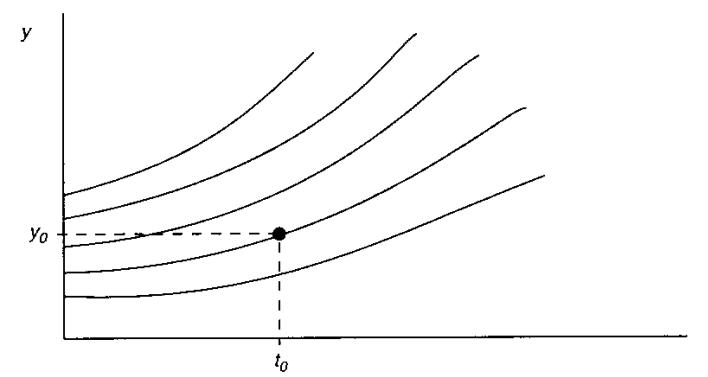
Promatrat ćemo diferencijalne jednadžbe s vremenom kao nezavisnom varijablim, jer sistemska dinamika opisuje *promjene sistema u vremenu*. Prototip diferencijalne jednadžbe prvog reda (koja uključuje prve derivacije varijable) u kojoj varijabla y ovisi o varijabli t je:

$$\frac{dy}{dt} = f(t, y) \quad (16.3)$$

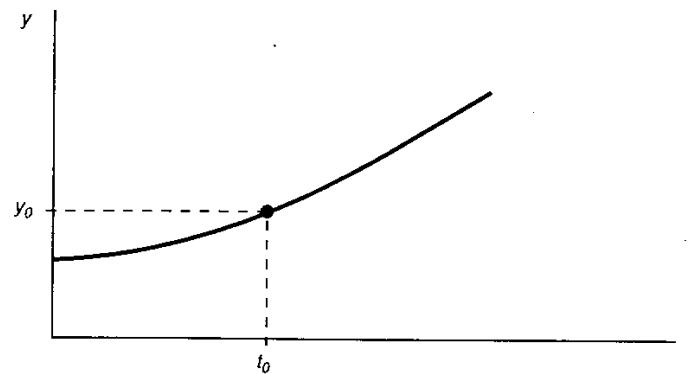
uz početni uvjet:

$$y(t_0) = y_0 \quad (16.4)$$

Rješenje diferencijalne jednadžbe bez početnih uvjeta možemo prikazati kao porodice krivulja (slika 16.10a). Definiranje početnih uvjeta ekvivalentno je izboru one krivulje te porodice koja prolazi kroz točku (t_0, y_0) , što se vidi na slici 16.10b.



(a) Porodica krivulja

(b) Krivulja koja prolazi kroz točku (t_0, y_0)

Slika 16.10. Rješenje diferencijalne jednadžbe

Specifičnost sistemske dinamike je to što ona opisuje promjenu varijabli (tj. nivoa) uzrokovana ulaznim i izlaznim tokovima za taj nivo. Stoga se modeli sistemske dinamike mogu opisati diferencijalnim jednadžbama tipa:

$$\frac{dy(t)}{dt} = g(y) - h(y) \quad (16.5)$$

gdje je

- y, varijabla (nivo),
- g(y), ulazni tok u nivo y,
- h(y), izlazni tok iz nivoa y.

Napominjemo i to da sistemski dinamika tretira *kašnjenje* na specifičan način (odjeljak 16.4.1), različit od matematičkog tretmana diferencijalnih jednadžbi s kašnjenjem. Pri porastu vrijednosti ulaznog toka na vrijednost k ($g(y)=k$) u sistemskoj dinamici se uzima da je izlazni tok $h(y)$ oblika:

$$h(y) = \frac{y}{T} \quad (16.6)$$

gdje je

- T, konstanta kašnjenja (procijenjeno kašnjenje).

Za razliku od toga, matematički tretman kašnjenja u diferencijalnim jednadžbama je takav da derivacija varijabli ovisi i o vrijednosti varijabli y u nekom prošlom vremenu $t-\tau$, tj.:

$$\frac{dy}{dt} = f(t, y(t), y(t-\tau)) \quad (16.7)$$

Takve jednadžbe dovode do fenomena nestabilnosti, periodičkog ponašanja i sl. (vidi npr. Harier i dr., 1987).

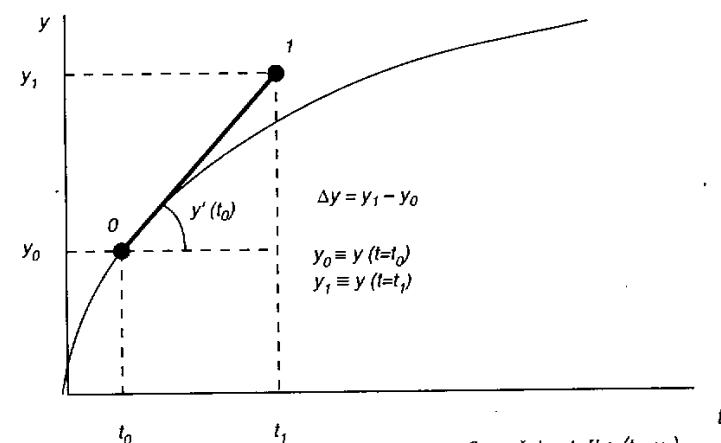
16.10.2. Numeričke metode rješavanja diferencijalnih jednadžbi

Jedna od prvih razvijenih i ujedno najjednostavnijih numeričkih metoda rješavanja diferencijalnih jednadžbi je *Eulerova metoda*. Ona ovako funkcioniра: poznata je početna točka (t_0, y_0) krivulje razvoja varijable y u vremenu. Metoda polazi od te točke i daljnje će se točke (t_i, y_i) izračunavati na svakih Δt korištenjem aproksimacije dane diferencijalne jednadžbe. Pomak od točke (t_0, y_0) do sljedeće točke (t_1, y_1) udaljene za Δt od t_0 (tj. $t_1=t_0+\Delta t$) dobit će se aproksimacijom luka krivulje od t_0 do t_1 s odreskom pravca koji ima nagib $y'(t_0)=dy/dt$ u $t=t_0$:

$$\begin{aligned} y_1 &= y_0 + y'(t_0) \Delta t \\ &= y_0 + f(t_0, y_0) \Delta t, \end{aligned} \quad (16.8)$$

kao što se vidi na slici 16.11. Na isti način se ide redom dalje, i svakih Δt se tom aproksimacijom računaju daljnje točke na krivulji:

$$(x_0, y_0) \rightarrow (x_1, y_1) \rightarrow (x_2, y_2) \rightarrow \dots \rightarrow (x_i, y_i) \rightarrow \dots$$

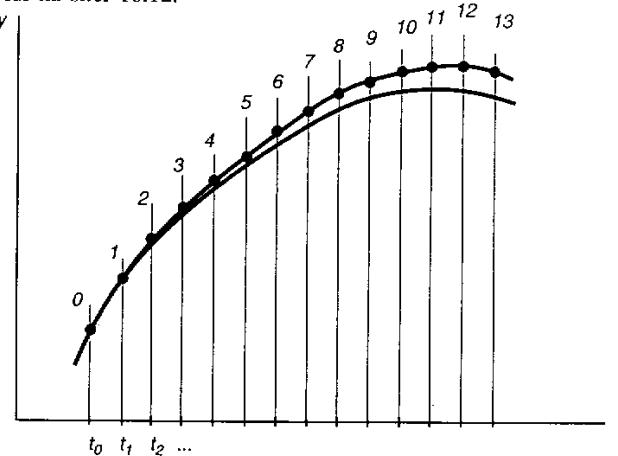


0 - početna točka (t_0, y_0)
1 - procijenjena sljedeća točka (t_1, y_1)

$$\frac{y_1 - y_0}{\Delta t} \approx y'(t_0) \quad \text{Eulerova aproksimacija}$$

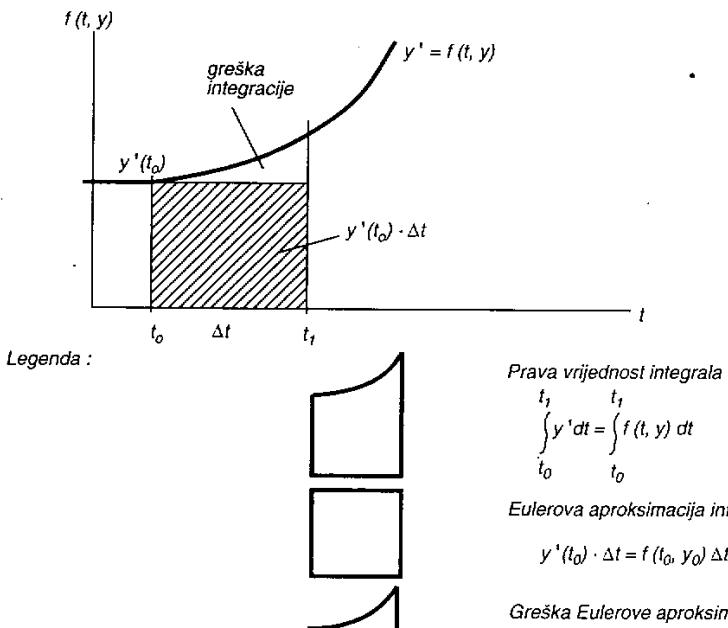
Slika 16.11. Eulerova metoda nalaženja točaka krivulje
(tj. numeričkog rješavanja diferencijalne jednadžbe)

kao što se vidi na slici 16.12.



Slika 16.12. Tok aproksimacije Eulerovom metodom

Greška koja se unosi svakim korakom računanja odgovara pogrešci integracije diferencijalne jednadžbe (16.3). Umjesto točne vrijednosti integrala pod krivuljom $f(t,y(t))$ između točaka t_0 i t_1 (slika 16.13)



Slika 16.13. Greška integracije u jednom koraku kod Eulerove metode

$$\int_{t_0}^{t_1} f(t, y(t)) dt$$

uzima se aproksimacija

$$y'(t_0) \Delta t.$$

Izbor veličine koraka Δt kod numeričke integracije očito utječe i na točnost i na brzinu približnog rješavanja diferencijalne jednadžbe. Kao što indicira i slika 16.12, može se pojaviti i problem stabilnosti: naime, procjena od y' u točkama unosi malu grešku u proračun dalnjih vrijednosti y , koja teži povećavanju u dalnjim koracima, i time sve većem odstupaju od prave vrijednosti.

Originalna Eulerova metoda se rijetko koristi, već je doživjela različite izmjene. Tako poboljšana Eulerova metoda računa novu vrijednost varijable kao:

$$y_1 = y_0 + \frac{1}{2}(f(t_0, y_0) + f(t_1, y_0 + f(t_0, y_0))\Delta t)\Delta t, \quad (16.9)$$

tj. brzina promjene varijable y aproksimira se s prosjekom nagiba u točkama (t_0, y_0) i $(t_1, y_0 + f(t_0, y_0) \Delta t)$.

Modificirana Eulerova metoda računa novu vrijednost varijable kao:

$$y_1 = y_0 + f\left(t_0 + \frac{\Delta t}{2}, y_0 + \frac{\Delta t}{2} f(t_0, y_0)\right) \Delta t, \quad (16.10)$$

tj. brzina promjene varijable y aproksimira se s nagibom u prosjeku (odnosno na pola razmaka između) točaka t_0 i t_1 , tj. u točki $(t_0 + \Delta t/2, y_0 + f(t_0, y_0) \Delta t/2)$.

I originalna Eulerova metoda i njezini derivati su poseban tip tzv. Runge-Kutta metoda koje su metode jednog koraka, tj. za prijelaz u sljedeći točku t_1 koriste se samo informacije iz prethodne točke t_0 . Kod nekih metoda se korak metode Δt može mijenjati tijekom integracije, i to tako da se prilagodi nepravilnostima funkcije $f(t, y)$. Runge-Kutta metode su dosta jednostavne za realizaciju, ali je kod njih teško procijeniti grešku integracije.

Koriste se i druge metode numeričke integracije diferencijalnih jednadžbi koje omogućuju procjenu grešaka integracije. Jedna klasa metoda toga tipa su tzv. prediktor-korektor metode. U njima se za računanje sljedeće vrijednosti varijable umjesto formule koriste različiti iterativni postupci, tj. ponavljanje proračuna u petlji dok se ne postigne zadovoljavajuća točnost.

Cesto se koriste i kombinacije ovih metoda, pa integracija npr. počinje Eulerovom metodom a nastavlja se prediktor-korektor metodom.

16.10.3. Rješavanje sistema diferencijalnih jednadžbi

Metode za rješavanje diferencijalnih jednadžbi mogu se lako proširiti i na sisteme istovremenih diferencijalnih jednadžbi, tj. na veći broj diferencijalnih jednadžbi koji opisuje dati model.

Na primjeru Eulerove metode demonstrirat ćemo rješavanje sistema od dvije diferencijalne jednadžbe prvog reda:

$$\frac{dy}{dt} = y + 2, \quad (16.11)$$

$$\frac{dz}{dt} = z - \frac{1}{2}y,$$

uz početne uvjete

$$y(t=0) = 10, \quad z(t=0) = 0, \quad (16.12)$$

Uzeti ćemo korak Eulerove jednadžbe $\Delta t = 1$ i rješavat ćemo korak po korak svaku od ovih diferencijalnih jednadžbi, s time da se dobivene vrijednosti varijabli iz jedne jednadžbe koriste u rješavanju druge jednadžbe.

Primjer izvođenja nekoliko koraka rješavanja sistema jednadžbi 16.11. uz početne uvjete 16.12. Eulerovom metodom:

1. $t_0 = 0$

$$y'(0) = y(0) + 2 = 10 + 2 = 12 \\ z'(0) = z(0) - \frac{1}{2}y(0) = 0 - \frac{10}{2} = -5$$

2. $t_1 = t_0 + \Delta t = 0 + 1 = 1$

$$y(1) = y(0) + y'(0) \Delta t = 10 + 12 \cdot 1 = 22 \text{ (Eulerova integracija)}$$

$$z(1) = z(0) + z'(0) \Delta t = 0 - 5 \cdot 1 = -5$$

$$y'(1) = y(1) + 2 = 22 + 2 = 24 \quad (\text{račun derivacija za sljedeći korak})$$

$$z'(1) = z(1) - \frac{1}{2}y(1) = -5 - \frac{22}{2} = -16$$

3. $t_2 = t_1 + \Delta t = 1 + 1 = 2$

$$y(2) = y(1) + y'(1) \Delta t = 22 + 24 \cdot 1 = 46$$

$$z(2) = z(1) + z'(1) \Delta t = -5 - 16 \cdot 1 = -21$$

itd.

Tako smo pokazali da se za rješavanje diferencijalnih jednadžbi različitih tipova koje se pojavljuju u sistemskoj dinamici mogu koristiti numeričke metode razvijene za integraciju diferencijalnih jednadžbi.

16.10.4. Rješavanje diferencijalnih jednadžbi višeg reda

Dosad smo prikazali integraciju diferencijalnih jednadžbi prvog reda, tj. jednadžbi u kojima se pojavljuje samo prva derivacija varijable $y' = dy/dt$. Diferencijalne jednadžbe višeg reda, u kojima se pojavljuju više derivacije varijabli $y'' = d^2y/dt^2$, $y''' = d^3y/dt^3$ itd., mogu se svesti na sistem diferencijalnih jednadžbi prvog reda. To ćemo ovdje demonstrirati na sljedećem primjeru diferencijalne jednadžbe drugog reda:

$$y'' = g(y', y, t) \quad (16.13)$$

koja se može transformirati u sistem od dvije diferencijalne jednadžbe prvog reda

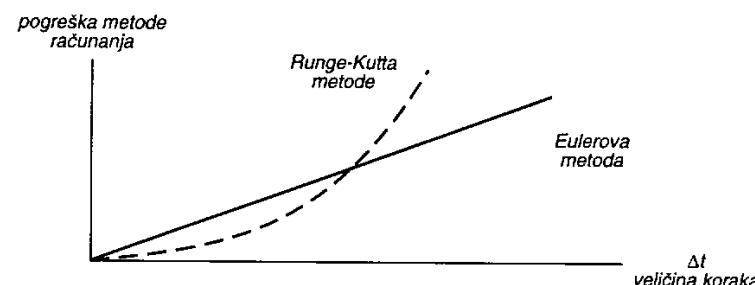
$$\begin{aligned} z' &= g(z, y, t) \\ y' &= z \end{aligned} \quad (16.14)$$

jednostavnom zamjenom prve derivacije novom varijablom.

16.10.5. Pogreške numeričkog rješavanja diferencijalnih jednadžbi

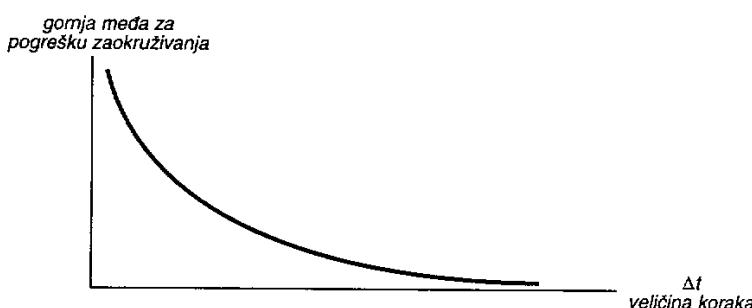
Dva su tipa pogrešaka vezanih za numeričko rješavanje diferencijalnih jednadžbi: pogreška metode računanja i pogreška zaokruživanja.

Pogreška metode računanja odnosi se na metodu kojom se izvodi numeričko rješavanje diferencijalne jednadžbe. Ta pogreška raste s veličinom koraka Δt , i to različito za razne metode rješavanja (slika 16.14).



Slika 16.14. Ovisnost pogreške metode računanja kod numeričkog rješavanja diferencijalnih jednadžbi o veličini koraka

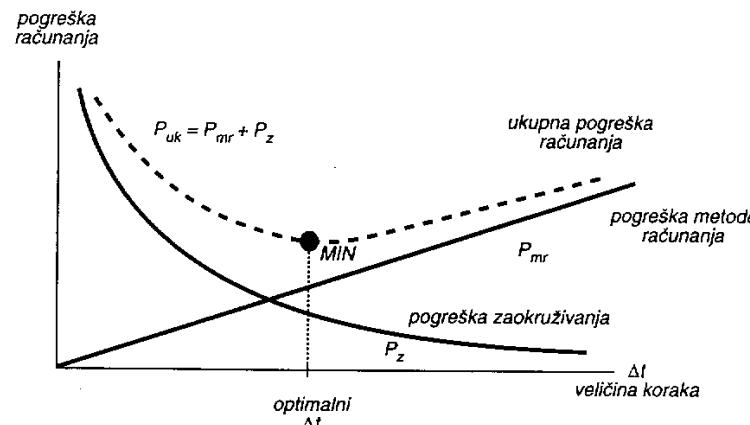
Pogreška zaokruživanja odnosi se na pogreške koje nastaju u proračunu zato što se u računalu brojevi prikazuju konačnom točnošću. Ova pogreška pada s veličinom koraka Δt , jer za proračun do nekog zadanog vremena T treba napraviti to manje izračunavanja (operacija) što je Δt veći. Gornja međa za pogrešku zaokruživanja je za sve metode oblika prikazanog na slici 16.15.



Slika 16.15. Ovisnost pogreške zaokruživanja kod numeričkog rješavanja diferencijalnih jednadžbi o veličini koraka

16.10.6. Izbor optimalnog koraka numeričkog proračuna

Optimalni korak numeričkog proračuna je onaj za koji je *ukupna pogreška računanja*, tj. zbroj pogreške metode računanja i pogreške zaokruživanja, minimalna. Grafički je to prikazano na slici 16.16.



Slika 16.16. Minimalizacija ukupne pogreške računanja kod numeričkog rješavanja diferencijalnih jednadžbi o veličini koraka

Optimalni korak Δt najčešće se traži empirički, s test- primjerima koji uspoređuju približno rješenje s točnim rješenjem što sličnije diferencijalne jednadžbe koja se može točno riješiti.

Više je različitih metoda traženja optimalnog koraka (Stoer i Bulirsch, 1980). Jedna od tih metoda kreće od nekog odabranog koraka Δt i neke pretpostavljene željene točnosti proračuna. Korak se smanjuje za neki faktor sve dok vrijednost određene funkcije koraka i željene točnosti ne padne ispod neke zadane vrijednosti. Drugi pristup ide preko korištenja dviju različitih metoda numeričkog rješavanja iste diferencijalne jednadžbe.

16.10.7. Stabilnost numeričkih rješenja diferencijalnih jednadžbi

S obzirom na značenje stabilnosti rješavanja jednadžbi sistemske dinamike, u nekoliko riječi opisat ćemo pojам stabilnosti numeričkog rješavanja diferencijalnih jednadžbi (Dorn i McCracken, 1972).

Numeričko rješenje smatramo *nestabilnim* ako dobivene numeričke vrijednosti ne konvergiraju pravom rješenju, već *sve više odstupaju* od njega tijekom izvođenja numeričkog proračuna. Pri tome razlikujemo nekoliko tipova nestabilnosti:

– *svojstvena nestabilnost* je nestabilnost (osjetljivost) same diferencijalne jednadžbe, neovisno o odabranoj numeričkoj metodi rješavanja jednadžbe. Taj tip nestabilnosti djeluje na to da se mala greška u početnim uvjetima brzo povećava pri rastućim vrijednostima varijable po kojoj se derivira (t), i to neovisno o primjenjenoj numeričkoj metodi

– *apsolutna nestabilnost* je nestabilnost numeričke metode koja uzrokuje velika odstupanja od pravog rješenja jednadžbe

– *relativna nestabilnost* je nestabilnost numeričke metode koja uzrokuje velike iznose omjera odstupanja i pravog rješenja jednadžbe

– *parcijalna nestabilnost* je nestabilnost numeričke metode primjenjene na određenu diferencijalnu jednadžbu, i to kada je metoda nestabilna za odredene vrijednosti izbora veličine koraka.

Dakle, svojstvena nestabilnost je svojstvo same diferencijalne jednadžbe, apsolutna i relativna nestabilnost su svojstvo primjenjene numeričke metode, a parcijalna nestabilnost se odnosi na specifičnu kombinaciju diferencijalne jednadžbe, primjenjene numeričke metode i odabrane vrijednosti koraka integracije.

Obrnuto, ako veličinu greške možemo održati u razumnim granicama bez obzira na to koliko se koraka proračuna izvodi, tada kažemo da je primjenjena numerička metoda stabilna.

16.10.8. Matematičko modeliranje i sistemska dinamika

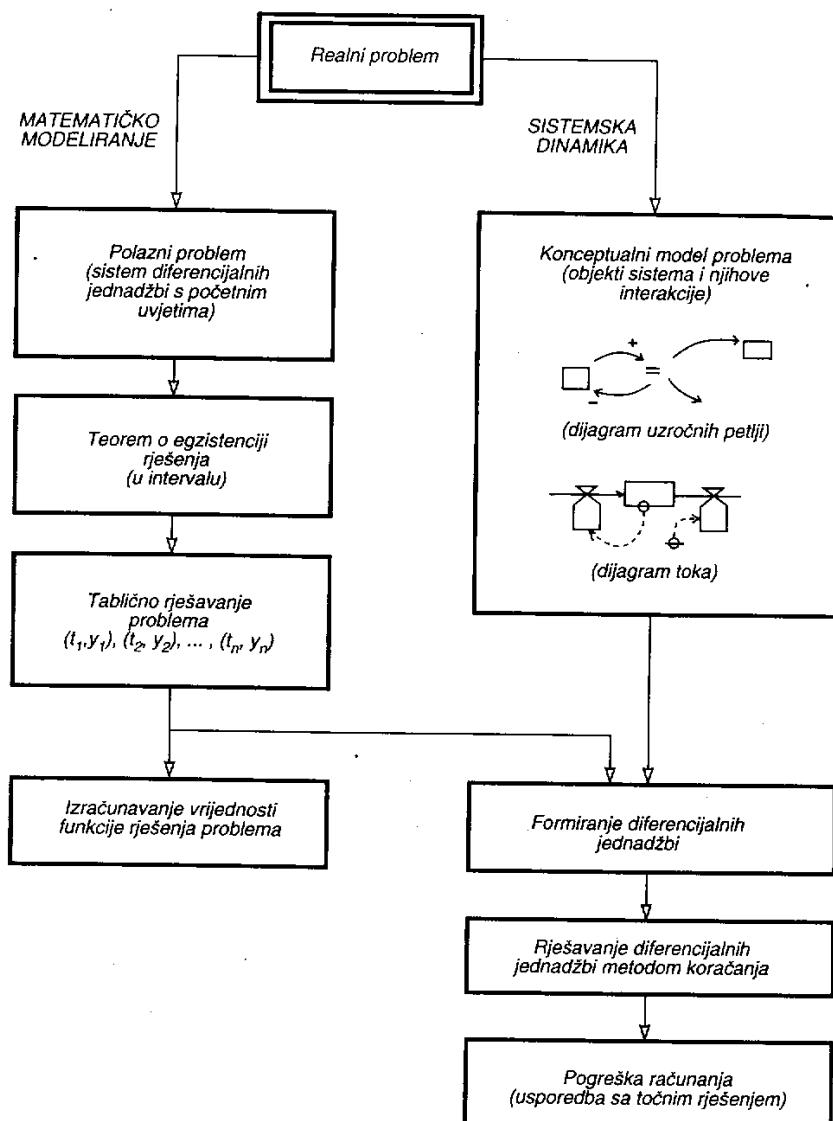
Postupak modeliranja kontinuiranih i kvazikontinuiranih problema razlikuje se u slučaju matematičkog modeliranja i modeliranja sistemskom dinamikom.

Matematičko modeliranje prikazuje problem u obliku sistema diferencijalnih jednadžbi, dokazuje egzistenciju rješenja (u određenom intervalu) i zatim traži brojčana (tablična) rješenja, tj. vrijednosti željenih varijabli u uzastopnim vremenskim trenucima $(t_1, y_1), (t_2, y_2), \dots, (t_n, y_n)$.

Ako postoji rješenje sistema diferencijalnih jednadžbi (npr. u obliku poznatih funkcija), tada se računaju vrijednosti tih funkcija u uzastopnim vremenskim trenucima. U suprotnome, postavljaju se diferencijske jednadžbe koje odgovaraju originalnim diferencijalnim jednadžbama, i one se rješavaju određenim metodama (tzv. koračanjem).

Sistemska dinamika ima drukčiji pristup. On odgovara tipu problema i znanju ljudi koji problem rješavaju. Objekti sistema preslikavaju se na elemente sistemske dinamike (nivoi, tokovi) i povezuju se u funkcionalnu cjelinu. Dobiveni konceptualni model razrađuje se detaljnije u obliku diferencijskih jednadžbi (u danom jeziku sistemske dinamike), a simulacija je eksperimentiranje modelom koje se realizira u postupcima koračanja. Dobivena rješenja preslikavaju se natrag na objekte sistema.

U oba načina rješavanja važno je ustanoviti pogreške metode rješavanja problema. Oba postupka rješavanja problema prikazana su na slici 16.17.



Slika 16.17. Pristup matematičkog modeliranja i sistemske dinamike rješavanju problema

16.11. STOHASTIČKA SISTEMSKA DINAMIKA

Sa stajališta uključivanja slučajnih varijabli sistemsko dinamika i simulacija diskretnih događaja od početka svog razvoja bitno se razlikuju. Dok je simulacija diskretnih događaja dominantno temeljena na uključivanju slučajnih varijabli u modele, sistemsko se dinamika u pravilu koristi samo determinističkim varijablama. Jedan od razloga za takvu razliku je i taj što je simulacija diskretnih događaja temeljena na dogadajima (koji su često slučajnog karaktera), dok sistemsko dinamika agregira niz događaja i tako ih uprosječe.

Ipak, i u sistemskoj se dinamici osjećala potreba da se omogući prikaz i simulacija efekata slučajnih varijabli. Razlog je što su u nekim slučajevima varijable i parametri slučajnog karaktera, a aproksimacija s determinističkim vrijednostima nije zadovoljavajuća. To npr. mogu biti procijenjeni dnevni broj kontakata u modelima širenja infekcija, vrijeme realizacije narudžbi pri poslovanju privrednih organizacija, iznosi dnevnih ulaganja u banku (odnosno koeficijenti ulaganja po jedinici kapitala banke) i sl.

Da bi se te veličine mogle mijenjati u toku izvođenja simulacijskih eksperimenata u skladu sa svojim razdiobama vjerojatnosti, potrebno je da jezik za sistemsku dinamiku ima ugrađen generator slučajnih varijabli. Tako jezik DYSMAP2 (novija verzija jezika DYSMAP) ima ugrađene generatore slučajnih varijabli s normalnom i uniformnom razdiobom vjerojatnosti.

Proširenje sistemske dinamike uključivanjem slučajnih varijabli implicira i odgovarajući tretman ulaznih podataka (statističke metode određivanja razdiobe vjerojatnosti i parametara razdiobe) te analizu rezultata simulacijskih eksperimenata.

16.12. OPTIMIZACIJA U SISTEMSKOJ DINAMICI

U novije su se vrijeme u sistemskoj dinamici počeli razvijati pristupi traženja optimalnih rješenja problema, npr. izbora takvih vrijednosti parametara sistema (ili politike rješavanja problema) koji daju optimalno rješenje sa stajališta odgovarajućih kriterija (Wolstenholme, 1990). Time se izbor najbolje kombinacije od nekoliko simuliranih kombinacija vrijednosti parametara nadomešta traženjem optimalnog rješenja. Jedan od pristupa optimizaciji je povezivanjem jezika DYNAMO sa softverom za optimizaciju (Gustafsson i Wiechowski, 1986).

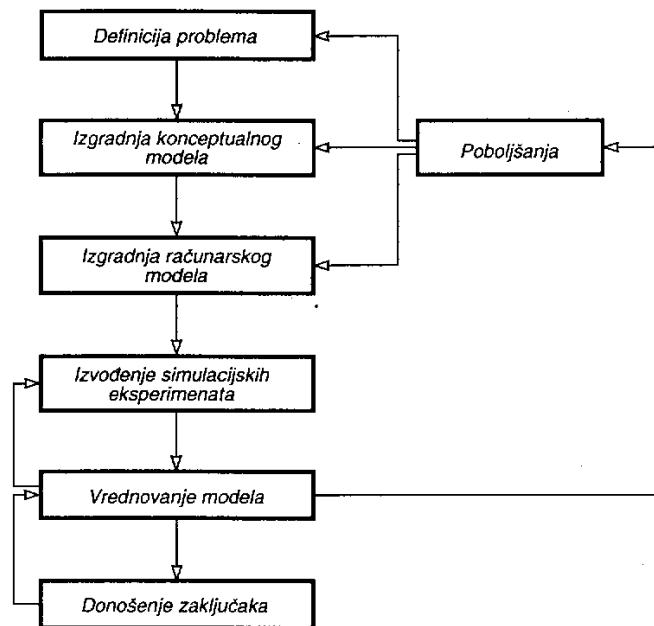
Drugi pristup ide za time da se razvije jedinstveni softver za optimizaciju simuliranih rješenja. Tako je razvijen softver DYMOD (Keloharju, 1983) temeljen na jeziku sistemske dinamike DYMAP. Taj se softver koristi rutinama za tzv. *penjanje na brežuljak* (slične onima prikazanim kod metode površine odziva u simulaciji diskretnih događaja) koje heuristički utvrđuju optimalne vrijednosti za zadani broj parametara modela uz definiranu funkciju cilja, povezanu za performanse sistema. Tijekom optimizacije naizmjenično se izvodi simulacija koja određuje vrijednost funkcije cilja za dani skup vrijednosti parametara, te optimizacija koja izabire nove vrijednosti parametara koji bi mogli poboljšati vrijednosti funkcije cilja, itd. U nizu iteracija uzašto se testiraju poboljšanja funkcije cilja za nove vrijednosti parametara, i dalje profinjuju parametri modela koji vode do rješenja s još boljom vrijednosti funkcije cilja.

Eksperimentiranja DYMOD softverom (Keloharju i Wolstenholme, 1989) pokazala su da se primjenom takva pristupa optimizaciji mogu postići znatna poboljšanja u odnosu prema kombinatoričkom pristupu mijenjanja vrijednosti parametara.

16.13. SIMULACIJSKI PROCES SISTEMSKE DINAMIKE

Može se reći da, u principu, ne bi trebalo biti bitne razlike između simulacijskog procesa u slučaju simulacije diskretnih događaja i sistemskog dinamika. Osnovna razlika je činjenica da je simulacija diskretnih događaja u pravilu stohastička a sistemskog dinamika je u pravilu deterministička. Posljedica toga je da kod determinističkih modela nema potrebe za statističkom analizom ulaznih i izlaznih podataka. Međutim, kao što smo vidjeli, i u sistemskoj dinamici su se počele koristiti slučajne varijable pa u takvima slučajevima ova razlika nestaje.

U prvom poglavlju prikazali smo strukturu i sadržaj faza simulacijskog procesa pri simulaciji diskretnih događaja. Simulacijski proces koji se obično koristi u sistemskoj dinamici nešto je jednostavniji. Tako se u knjizi (Roberts i dr., 1983) simulacijski proces sistemskog dinamike prikazuje kao na slici 16.18. gdje smo nazive prilagodili terminologiji koju smo uveli pri opisu simulacije diskretnih događaja.



Slika 16.18. Simulacijski proces sistemskog dinamike (Roberts i dr., 1983)

Postoje neke razlike u strukturi i sadržaju pojedinih faza simulacijskog procesa koje ćemo navesti uz kratku analizu simulacijskog procesa u sistemskoj dinamici. Simulacijski proces u sistemskoj dinamici analizirat ćemo prema koracima simulacijskog procesa u simulaciji diskretnih događaja (slika 1.5).

(1) Definicija cilja simulacije

i

(2) Identifikacija sistema

Obje faze imaju isti sadržaj kao i onaj kod simulacije diskretnih događaja.

(3) Skupljanje podataka o sistemu i njihova analiza

U osnovi nema razlike u načinu prikupljanja i analize podataka na temelju kojih se računaju parametri modela u sistemskoj dinamici i simulaciji diskretnih događaja. Podaci koji se koriste u sistemskoj dinamici vezani su za strukturu i način rada sistema. To su različite konstante (npr. broj radnih sati na dan) i podaci koji ovise o načinu rada elemenata sistema i njihova međudjelovanja (npr. stopa rada i vrijeme popravaka strojeva). Ako su potrebne za statističke procjene temeljene na uzorcima, koriste se iste procedure kao one opisane u poglavlju 11.

(4) Izgradnja konceptualnih modela

Dok se u simulaciji diskretnih događaja koristi izgradnja konceptualnih modela s izborom jedne od mogućih tehnika modeliranja, u sistemskoj dinamici se koriste *dvije faze* konceptualnog modeliranja – izgradnja dijagrama uzročnih petlji i dijagrama toka, koji omogućuju različitu preciznost opisa komponenata i međudjelovanja u modelu.

S druge strane izbor tipa konceptualnog modela u simulaciji diskretnih događaja je izvanredno raznovrstan. Više je različitih tipova konceptualnih modela, a unutar svakog od njih (a to vrijedi osobito za grafičke tipove modela) niz je raznovrsnih alternativnih tehnika konceptualnog modeliranja.

(5) Izgradnja simulacijskih programa

Osnovna je razlika to što sistemskog dinamika raspolaže ograničenim brojem od nekoliko simulacijskih jezika koji su međusobno veoma slični i imaju isti osnovni pristup, a u simulaciji diskretnih događaja razvijeni su deseci različitih simulacijskih jezika koji se mogu podijeliti u nekoliko bitno različitih stilova pristupa izgradnji programa.

(6) Verifikacija simulacijskih programa

i

(7) Vrednovanje simulacijskih modela

Zanimljiva je razlika u pristupu stvaranju povjerenja u simulaciji diskretnih događaja i sistemskoj dinamici. Dok se u simulaciji diskretnih događaja stvaranju povjerenja u model pridaje izvanredno velika pažnja, te gotovo da nema knjige iz tog područja u kojoj se ono ne obrađuje detaljno, sistemskog dinamika nema toliko razrađenu i opće prihvaćenu metodologiju stvaranja povjerenja u modele. Razlika u interesu za stvaranje povjerenja u modele tih dvaju tipova može se djelomično tumačiti u dominantno determinističkom karakteru modela sistemskog dinamike, što provjeru ispravnosti modela čini jednostavnijom. U objavljenim radovima iz sistemskog dinamike ipak se vidi da se provode određene procedure vrednovanja i verifikacije modela (npr. Forrester, 1961; Coyle, 1977; Taylor, 1983; Sterman, 1984; Narchal, 1988; Roberts i Dangerfield, 1990).

Neki od postupaka stvaranja povjerenja u modele sistemskog dinamika jesu:

- vrednovanje strukture modela (tj. ispitivanje u kojoj mjeri struktura modela odgovara strukturi sistema)
- vrednovanje parametara modela (da li odgovaraju parametrima sistema)
- ispitivanje konzistentnosti dimenzija (u jednadžbama modela)
- replikativno vrednovanje modela (tj. usporedba rezultata simulacije s historijskim podacima o radu sistema)
- ispitivanje osjetljivosti modela (na male promjene ključnih parametara modela)
- robustnost (zadržavanje dobrih performansi modela i kod velikih šokova nametnutih izvana na sistem).

U osnovi se za stvaranje povjerenja u modele sistemskog dinamike mogu upotrijebiti isti tipovi metoda verifikacije programa i vrednovanja konceptualnih modela kao i kod simulacije diskretnih događaja (poglavlje 8). Osnovne razlike bi mogle biti: analiza logike rada specifična je za sistemsku dinamiku; postoje dvije faze konceptualnih modela, tako da je potrebno provjeravati i korektnost prijelaza iz dijagrama uzročnih petlji na dijagrame toka modela; manuelni proračun se u sistemskoj dinamici teško može provesti (ovisno o upotrijebljenim numeričkim metodama rješavanja jednadžbi modela); statističke metode analize se koriste samo ako su modeli sistemskog dinamike stohastičkog karaktera.

(8) Planiranje i izvođenje simulacijskih eksperimenata

Sistemski se dinamika koristi različitim oblicima mijenjanja vrijednosti parametara modela, od mijenjanja jednog po jednog parametra do variranja njihovih kombinacija. Mogu se koristiti slične metode dizajna simulacijskih eksperimenata kao i kod simulacije diskretnih događaja. Ako je riječ o stohastičkim modelima sistemskog dinamike, i tehnike redukcije varijance bi također bile primjenljive.

(9) Analiza rezultata simulacijskih eksperimenata

Kod determinističkih modela sistemskog dinamike nema posebne analize izlaznih rezultata (osim regresijskih metamodela za analizu izlaza modela s različitim vrijednostima parametara modela), a kod stohastičkih modela potrebno je koristiti različite statističke metode analize uzoraka, analogno onima kod simulacije diskretnih događaja.

(10) Zaključci i preporuke

To je isti tip aktivnosti kao i kod simulacije diskretnih događaja.

16.14. SISTEMSKA DINAMIKA I DISKRETNNA SIMULACIJA: SLIČNOSTI, RAZLIKE I VEZE

Na kraju ćemo ukratko komentirati neke osnovne crte usporedbe sistemskog dinamika i diskretnne simulacije u vezi s pristupom, tipom modela i ciljem modeliranja.

Diskretna simulacija modeliranju pristupa nastojeći dosta detaljno opisati elemente sistema i njihova međudjelovanja. Pri tome se promatraju uzastopne promjene stanja svih elemenata sistema (događaji) i aktivnosti koje do njih dovode ili su im posljedica.

Model je u pravilu stohastičkog karaktera. Računarski modeli razlikuju se u ovisnosti o tome koja se strategija modeliranja primjenjuje (orientirana na događaje; aktivnosti ili procese). Koristi se detaljno razrađeni i složen proces planiranja simulacijskih eksperimenata i statističke analize izlaznih varijabli simulacijskih modela. Izvođenje simulacije daje ocjenu performansi alternativnih struktura i načina rada sistema. Posebno se detaljno dobivaju karakteristike čekanja na resurse u sistemu.

Sistemski dinamika opisuje agregirane elemente sistema, i to tako da se pojedinačni događaji aproksimiraju tokovima koji mijenjaju nivo (tj. stanja resursa u sistemu). Modeliraju se sistemi s povratnom vezom u kojima se upravlja tokovima u sistemu. Računarski modeli su u obliku diferencijskih jednadžbi (jednadžbi konačnih razlika) koje se rješavaju različitim numeričkim metodama. Modeli su u pravilu determinističkog karaktera, tako da je upotreba statističke analize i ulaznih i izlaznih varijabli modela i mnogo rjeđa. Simulacija omogućuje određivanje utjecaja upravljanja na rad sistema i na performanse sistema. Sistemski dinamika je pogodna i za određivanje posljedica dugoročnih politika na rad sistema. Koristi se najviše za modeliranje i simulaciju društveno-ekonomskih sistema.

O istraživanju moguće *kooperacije* diskretnne simulacije i sistemskog dinamika nema mnogo publiciranih radova, iako je jasno da postoji interes da se modeliraju sistemi koji se ponašaju pretežno kontinuirano ali u kojima postoje i događaji u kojima se to ponašanje drastično mijenja. Jedan takav rad je pokušaj (Coyle, 1985) da se diskretni događaji prikažu modelom sistemskog dinamike.

Ima i ideja da se *usporedi* modeli isti sistem i vrlo detaljno s diskretnom simulacijom i u agregiranom obliku pomoću sistemskog dinamika. Usporedba rezultata simulacije dobivena tim djelom metodom omogućila bi ustanovljavanje *točnosti* rada modela sistemskog dinamike (jer ovdje diskretni simulacijski model zamjenjuje realni sistem, budući da ga može detaljnije i točnije opisati).

Jedna *usporedba* konceptualnih modela sistemskog dinamike (SD) i dijagrama ciklusa aktivnosti (DCA) diskretnne simulacije (Wolstenholme, 1990) vidi analogiju varijabli brzina (u SD) s aktivnostima (u DCA), te nivoa (u SD) s repovima (u DCA).

17. NEKE KLASE PROBLEMA KOJE SE MOGU MODELIRATI SISTEMSKOM DINAMIKOM

Sistemska dinamika omogućuje modeliranje širokog spektra raznovrsnih problema iz područja ekonomije, proizvodnje, medicine, energije, okoliša i sl. Da bi čitalac stekao predodžbu o mogućnostima sistemskog modeliranja, ovdje je prikazano modeliranje nekoliko klasa važnih problema: dinamika populacije (jedna jedinstvena populacija, stratificirana populacija, populacija lovaca i plijena), dinamika zatvorenog ciklusa života, modeliranje procesa odlučivanja i zagadživanje okoline. Za svaku klasu problema dan je opis problema, dijagram uzročnih petlji i dijagram toka. Za nekoliko problema prikazan je i odgovarajući simulacijski program u jeziku DYNAMO.

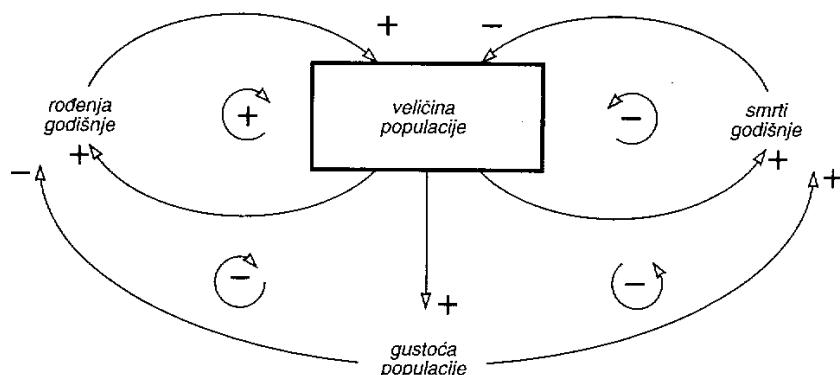
17.1. DINAMIKA POPULACIJE

Modeli dinamike populacije opisuju razvoj veličine populacije u vremenu. Njima se najčešće opisuju biološke populacije, tj. životni ciklusi razvoja životinja i biljaka u zatvorenim sredinama (otoci, visoravni, Zemlja). Može se promatrati bilo jedinstvena populacija, populacija stratificirana prema nekom atributu (životnoj dobi, stanju) bilo konkurenčnska borba više populacija.

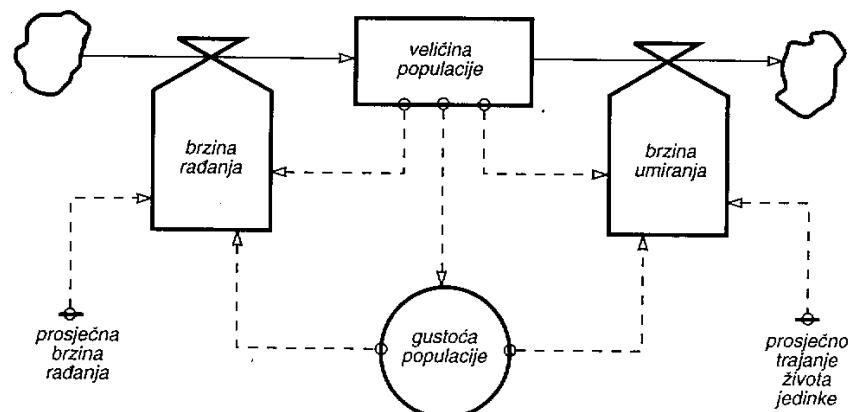
17.1.1. Jedna jedinstvena populacija

Navedeni tipovi modela opisuju populaciju jedne vrste, i pri tome se jedinke te vrste ne dijele u potklase. Sve što je populaciji potrebno za život (prostor, hrana, voda itd.) promatra se kao *nepromjenljivi resurs* koji je jedinkama populacije na raspolaganju u konstantnom *ograničenom* iznosu (neograničeni resursi nisu realno mogući i oni vode do neograničenog rasta populacije).

Promatrat ćemo primjer biološke populacije koja živi u ograničenom zatvorenom prostoru s konstantnom osiguranom količinom hrane. *Dijagram uzročnih petlji* takve populacije prikazan je na slici 17.1a. Dvije uobičajene povratne petlje su pozitivna povratna petlja vezana za rađanje i negativna povratna petlja vezana za smrtnost populacije. Osim toga, povećanje veličine populacije povećava gustoću populacije. Pretpostavlja se da će se u njoj zbog pogoršanja životnih uvjeta smanjiti brzina rađanja i povećati brzina umiranja. Time su zatvorene dvije nove negativne petlje koje dodatno stabiliziraju razvoj populacije u vremenu, i to na nižem nivou veličine populacije nego što bi to bilo bez njihova utjecaja.



(a) Dijagram uzročnih petlji



(b) Dijagram toka

Slika 17.1. Model dinamike jedinstvene populacije

Na slici 17.1b prikazan je *dijagram toka* ovog sistema. Veličina populacije je nivo, a brzine umiranja i rađanja su brzine tokova koje ulaze i izlaze iz nivoa. Prosječna brzina rađanja, prosječno trajanje života i raspoloživi prostor su konstante sistema.

Utjecaj gustoće populacije na brzinu rađanja i brzinu umiranja specifičan je za određenu vrstu i uvjete sredine u kojoj se nalazi. Da bismo napisali DYNAMO-program za taj model, specificirat ćemo parametre, ograničenja i relacije koje za njega vrijede. Neka populacija živi na ograničenom prostoru od 80 jedinica (to može biti površina ili volumen). Prosječna stopa rađanja je 0.15 jedinki po jednoj jedinki populacije godišnje, a prosječna stopa umiranja je 0.09 jedinki po jednoj jedinki populacije godišnje. Pretpostavimo da ovisnost stopi rađanja (RADJANJE.KL) o veličini populacije (POPULAC.K) i gustoći populacije (GUSTOCA.K) ima ovakav oblik:

$RADJANJE.KL=0.15*POPULAC.K*(1-0.013*GUSTOCA.K)$,
a ovisnost brzine umiranja (UMIRANJE.KL) o veličini i gustoći populacije izgleda ovako:
 $UMIRANJE.KL=0.09*POPULAC.K*(0.4+0.06*GUSTOCA.K)$.

Neka je veličina populacije na početku simulacije jednaka 5000 jedinika.

DYNAMO-program za dinamiku populacije imao bi tada ovakav oblik:

```
*-----*
* DINAMIKA JEDINSTVENE POPULACIJE
NOTE   - PROMJENA VELICINE POPULACIJE -
L      POPULAC.K=POPULAC.J+DT*(RADJANJE.JK-UMIRANJE.JK)
N      POPULAC=5000
NOTE   - BRZINA RADJANJA -
R      RADJANJE.KL=0.15*POPULAC.K*(1-0.013*GUSTOCA.K)
NOTE   - BRZINA UMIRANJA -
R      UMIRANJE.KL=0.09*POPULAC.K*(0.4+0.06*GUSTOCA.K)
NOTE   - GUSTOCA POPULACIJE -
A      GUSTOCA.K=POPULAC.K/80
*-----*
```

Takvim se modelom može analizirati dinamika razvoja populacije i naći stabilna stanja populacije te period potreban za dostizanje stabilnog stanja (iz nekog početnog stanja). Model može poslužiti i za studij ponašanja dinamike populacije u promjenjenim uvjetima (parametri, ograničenja, relacije).

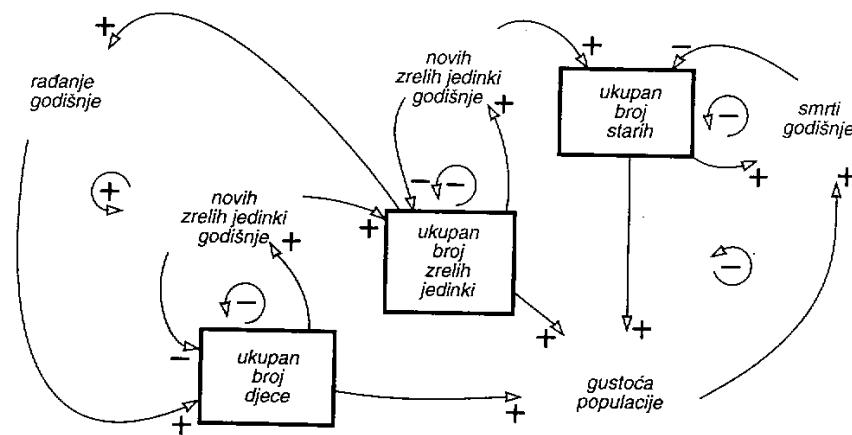
17.1.2. Stratificirana populacija

Modeli stratificirane populacije prikazuju populaciju jedne vrste čije se jedinke dijele u više kategorija. To mogu biti kategorije različite starosne dobi jedinke ili različitih stanja jedinke (npr. zdrave, zaražene i bolesne jedinke). Ovisno o različitim utjecajima u sistemu, jedinke dinamički prelaze iz kategorije u kategoriju. Opis populacije pomoću više stanja u kojima su njezine jedinke omogućuje precizniji prikaz populacije i utjecaja koji u njoj djeluju, pa se time dobiva i predviđanje ponašanja populacije koje je nemoguće dobiti pomoću jedne jedinstvene populacije. Naravno, ovakav prikaz zahtijeva i bolje kvalitativno i kvantitativno poznavanje populacije.

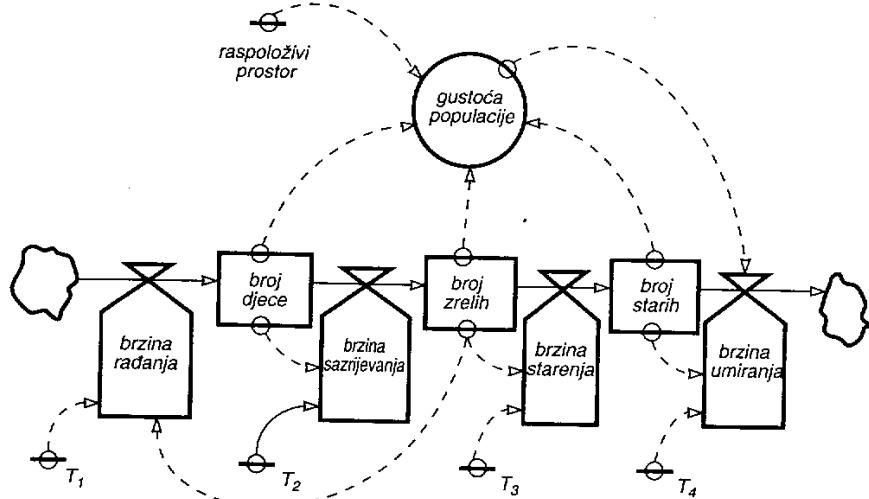
Prikazat ćemo razradu dinamike populacije iz prethodnog odjeljka, a populaciju ćemo podijeliti u tri kategorije: djeca, zrele jedinke, stare jedinke. Sve tri kategorije jedinki populacije djeluju na gustoću populacije, a samo zrele jedinke omogućuju rađanje. Pretpostavit ćemo da u ovom pojednostavljenom modelu umiru samo stare jedinke, te da gustoća populacije povećava brzinu umiranja a ne utječe na brzinu rađanja. *Dijagram uzročnih petlji* sistema prikazan je na slici 17.2a.

Na slici 17.2b prikazan je *dijagram toka sistema*. Moguća stanja jedinki populacije su nivoi, a brzine rađanja i umiranja te brzine prelaska iz stanja u stanje su brzine tokova. Konstante T_1 do T_4 su prosječna vremena koja jedinke populacije provedu u pojedinim stanjima.

Pretpostavimo da u početnom času ima 10 000 djece, 30 000 zrelih jedinki i 5 000 starih jedinki, te da konstante T_1 do T_4 imaju vrijednosti 5, 15, 10 i 30 godina.



(a) Dijagram uzročnih petlji



(b) Dijagram toka

Slika 17.2. Model stratificirane populacije

Sve su brzine proporcionalne nivoima i obrnuto proporcionalne vrijednosti konstanti T_i koje na njih djeluju. Raspoloživi prostor je veličine 160 000 jedinica. Brzina umiranja ovisi o broju starih jedinki (STARI.K) i o ukupnoj gustoći populacije (GUSTOCA.K):

$$\text{UMIRANJE.KL} = (\text{STARI.K}/T_4) * (1 - 0.022 * \text{GUSTOCA.K})$$

DYNAMO-program za dinamiku ove stratificirane populacije izgleda ovako:

* ----- DINAMIKA STRATIFICIRANE POPULACIJE ----- *	
NOTE	- PROMJENA BROJA DJECE -
L	DJEC.A.K=DJEC.A.J+DT*(RADJANJE.JK-SAZRIJEVANJE.JK)
N	DJEC.A=10 000
NOTE	- PROMJENA BROJA ZRELIH JEDINKI -
L	ZRELI.K=ZRELI.J+DT*(SAZRIJEVANJE.JK-STARENJE.JK)
N	ZRELI=30 000
NOTE	- PROMJENA BROJA STARIH JEDINKI -
L	STARI.K=STARI.J+DT*(STARENJE.JK-UMIRANJE.JK)
N	STARI=5000
NOTE	- BRZINA RADJANJA -
R	RADJANJE.KL=ZRELI.K/T1
NOTE	- BRZINA SAZRIJEVANJA -
R	SAZRIJEVANJE.KL=DJEC.A.K/T2
NOTE	- BRZINA STARENJA -
R	STARENJE.KL=ZRELI.K/T3
NOTE	- BRZINA UMIRANJA -
R	UMIRANJE.KL=(STARI.K/T4)*(1-0.022*GUSTOCA.K).
NOTE	- GUSTOCA POPULACIJE -
A	GUSTOCA.K=(DJEC.A.K+ZRELI.K+STARI.K)/160 000

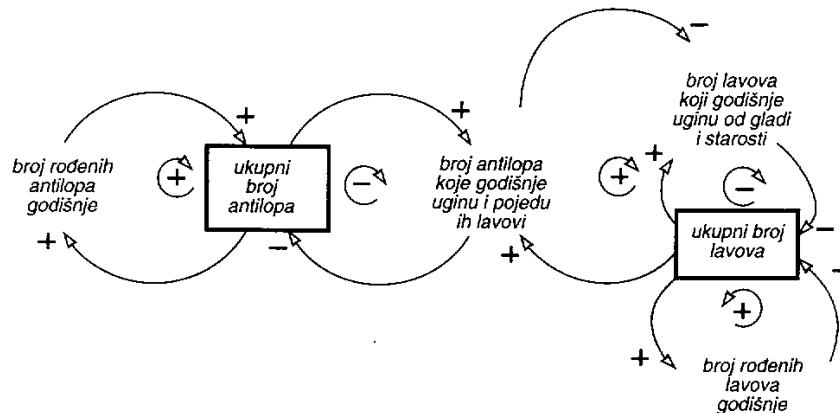
17.1.3. Međusobna ovisnost više populacija

Bioške populacije žive izmiješane u prirodi i međusobno su ovisne, bilo direktno ili indirektno. Prehrana jedne vrste jedinkama druge vrste je direktna ovisnost populacija, a utjecaj preko treće vrste (životinjske ili biljne) kojom se hrane dvije životinjske vrste indirektna je ovisnost dviju populacija. Modeliranje dinamike ovisnosti populacija omogućuje proučavanje utjecaja ovisnosti, veličina populacija, okoline itd. na ponašanje takva sistema. Čak i u sistemima sa samo dvije međusobno ovisne populacije na koje ljudi mogu utjecati nije jednostavno ispitati zašto je zbog nekih okolnosti (npr. pojave epidemije u jednoj populaciji) uslijedio određeni razvoj u vremenu (npr. promjena veličina obiju populacija). Još je teže pronaći način kojim se može postići neki željeni utjecaj na promjenu odnosa ili tendencija što vladaju u sistemu.

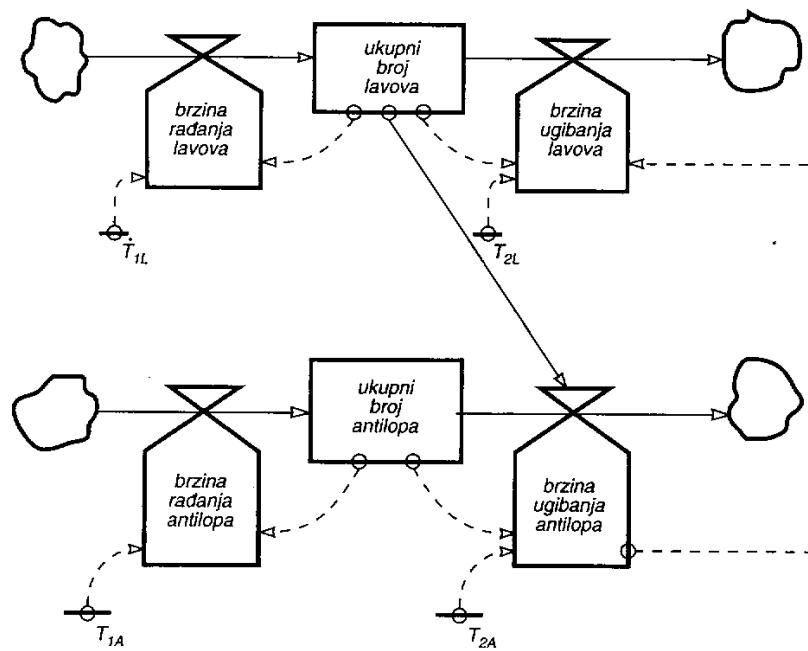
Štoviše, pokazalo se da dosta često odluke koje naizgled dovode do očekivanog efekta zapravo postižu potpuno suprotno djelovanje. To dakako nije slučaj samo u sistemima u kojima su u međusobnim ovisnostima populacije, već i u nizu drugih tipova sistema. Razlog je što čovjek nije u stanju vidjeti dinamičke efekte međusobne povezanosti većeg broja elemenata, već često intuitivno predviđa s pojednostavljenom predodžbom o utjecajima u sistemu.

Prikazat ćemo jednostavan primjer međusobne ovisnosti dviju populacija u kojem se jedinke jedne populacije hrane jedinkama druge populacije (tzv. problem lovca i plijena). Neka su to, na primjer, populacije lavova i antilopa. Na slici 17.3. prikazan je dijagram uzročnih petlji problema (Roberts i dr., 1983). Osim ubičajenih pozitivnih

i negativnih uzročnih petlji svake od dviju populacija (rođenja i smrtnost), te su dvije populacije vezane preko smanjenja populacije antilopa koje lavovi ubijaju i jedu. To dodatno smanjuje veličinu populacije antilopa a ujedno i usporava ugibanje lavova od gladi. Efekt takva razvoja je povećavanje broja lavova a smanjenje broja antilopa. Jako smanjenje veličine populacije antilopa uzrokuje međutim manji broj antilopa koje godišnje mogu pojesti



(a) Dijagram uzročnih petlji



(b) Dijagram tokova

Slika 17.3. Populacija lavova i antilopa (lovci i plijen) (Roberts i dr., 1983)

lavovi. Preko nove pozitivne povratne veze to uzrokuje veći broj lavova koji godišnje uginu od gladi, i dalje smanjenje ukupnog broja lavova. Posljedica toga je da se i dalje smanjuje broj antilopa koje pojedu lavovi. Tako sistem periodički prelazi iz jedne u drugu krajnost – iz stanja s mnogo lovaca (lavova) u stanje s puno plijena (antilopa).

Na slici 17.3b prikazan je *dijagram toka* sistema. U njemu su obje populacije prikazane posebnim tokovima koji prikazuju veličine populacija kao nivoa a brzine rada i ugibanja kao brzine. Veza među populacijama prikazana je utjecajem ukupnog broja lavova (nivo) na brzinu ugibanja antilopa (brzina toka), i utjecajem brzine ugibanja antilopa na brzinu ugibanja lavova (brzina toka).

DYNAMO-program za taj model s jednostavnim opisom međusobnih utjecaja populacija izgleda ovako:

```

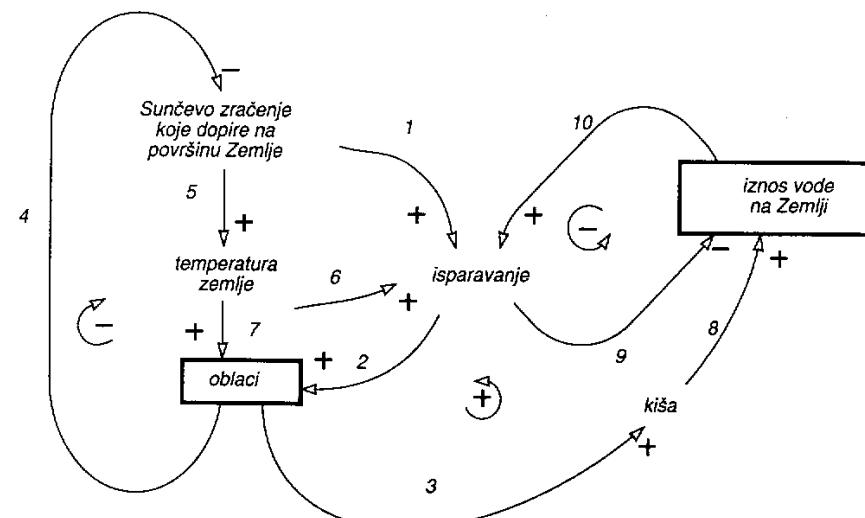
*-----*
* POPULACIJA LAVOVA I ANTILOPA (LOVCI I PLIJEN)
NOTE   - PROMJENA VELIČINE POPULACIJE LAVOVA -
L      LAVOVI.K=LAVOVI.J+DT*(LRADJANJE.JK-LUGIBANJE.JK)
N      LAVOVI=700
NOTE   - BRZINA RADJANJA I UGIBANJA LAVOVA -
R      LRADJANJE.KL=LAVOVI.K/TIL
R      LUGIBANJE.KL=(LAVOVI.K/T2L)*(1500/AUGIBANJE.KL)
NOTE   - PROMJENA VELIČINE POPULACIJE ANTILOPA -
L      ANTILOPE.K=ANTILOPE.J+DT*(ARADJANJE.JK-AUGIBANJE.JK)
N      ANTILOPE=4 500
NOTE   - BRZINA RADJANJA I UGIBANJA ANTILOPA
R      ARADJANJE.KL=ANTILOPE.K/T1A
R      AUGIBANJE.KL=(ANTILOPE.K/T2A)*(LAVOVI.K/1000)
*-----*
```

17.2. DINAMIKA ZATVORENOG CIKLUSA ŽIVOTA

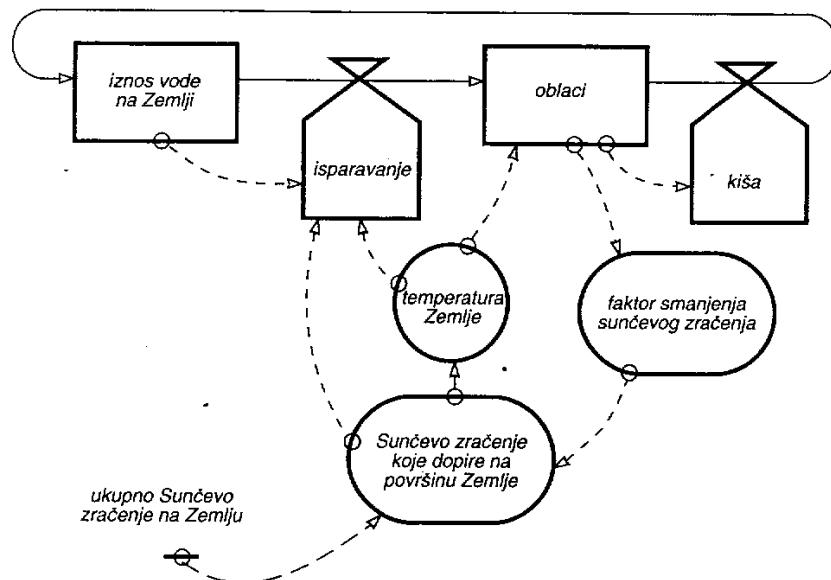
U Zemljinoj biosferi niz je zatvorenih ciklusa života koji reguliraju kružne tokove materije. Jedan od takvih kružnih ciklusa je samoregulacija količine vode koja prolazi različita agregatna stanja pod utjecajem Sunčeva zračenja i stanja atmosfere (Roberts i dr., 1983).

Na slici 17.4a prikazan je *dijagram uzročnih petlji* kružnog ciklusa vode na Zemlji. Objašnjenje veza među elementima dajemo redom koji je ujedno jedan od mogućih redoslijeda razvoja dijagrama (pa ga i čitalac može tim redom rekonstruirati). Sunčev zračenje povećava isparavanje vode na Zemlji (1), a time raste i oblačnost (2) koja povećava količine kiše (3). No, što je više oblaka to manje Sunčeva zračenja dopire na Zemlju (4). Time je zatvorena *negativna povratna petlja* : Sunčev zračenje – isparavanje – oblaci – Sunčev zračenje (1–2–4). Sada ćemo uvesti temperaturu Zemlje – što više Sunčeva zračenja dopire na Zemlju, to je viša temperatura Zemlje (5), a viša temperatura Zemlje znači i više isparavanja (6) i više oblaka (7). Konačno uvodimo i količinu vode na Zemlji – što je više kiše, više je vode na Zemlji (8), a što je veće isparavanje, manje je vode na Zemlji (9). No, što je više vode na Zemlji, to je veće isparavanje (10). Tako smo stvorili još jednu *negativnu povratnu petlju* : količina vode na Zemlji – isparavanje – količina vode na Zemlji (10–9), i jednu *pozitivnu povratnu petlju* : oblaci – kiša – količina vode na Zemlji – isparavanje – oblaci (3–8–10–2).

Na slici 17.4b prikazan je odgovarajući *dijagram toka* modela. Za nivoje je odabrana voda u svoja dva agregatna stanja – voda i oblaci – u kojima se pojavljuje u svom kružnom dinamičkom ciklusu. Isparavanje i kiša su brzine: isparavanje smanjuje količinu vode a



(a) Dijagram uzročnih petlji



(b) Dijagram tokæ

Slika 17.4. Model samoregulacije kruženja vode u biosferi Zemlje (Roberts i dr., 1983).

povećava količinu oblaka, dok kiša smanjuje količinu oblaka a povećava količinu vode. Brzina padanja kiše kontrolirana je količinom oblaka. Brzina isparavanja kontrolirana je iznosom vode na Zemlji, te dvjema veličinama prikazanim kao *pomoćne varijable* : temperaturom Zemlje i količinom Sunčeva zračenja koje dopire na površinu Zemlje. Količina Sunčeva zračenja koje dopire na površinu Zemlje ujedno utječe i na povišenje temperature Zemlje, a sama se povisuje s porastom ukupnog Sunčeva zračenja na Zemlju (*konstanta*) i smanjuje s porastom količine oblaka (to preko faktora smanjenja Sunčeva zračenja prikazanog kao *pomoćna varijabla*).

17.3. MODELI PROCESA ODLUČIVANJA

Sistemska dinamika se pokazala kao veoma prikladna metoda za modeliranje i simulaciju različitih tipova procesa odlučivanja kao što su upravljanje zalihami ili tokom novca. Standardni pristup modeliranju procesa odlučivanja ide preko postavljanja *ciljeva*, tj. željenih stanja do kojih se sistem želi dovesti. Mjerjenje *odstupanja* stvarnog stanja od željenog ciljnog stanja koristi se za *donošenje odluke* o načinu utjecaja na sistem koji omogućuje postizanje ciljnog stanja. Zato se može govoriti i o *upravljanju* sistemom prema željenom cilju.

Prikazat će moći dva primjera modeliranja donošenja odluka. Jedan je tehničke (upravljanje vodostajem u branama) a drugi ekonomski naravi (upravljanje stanjem zaliha na skladištu).

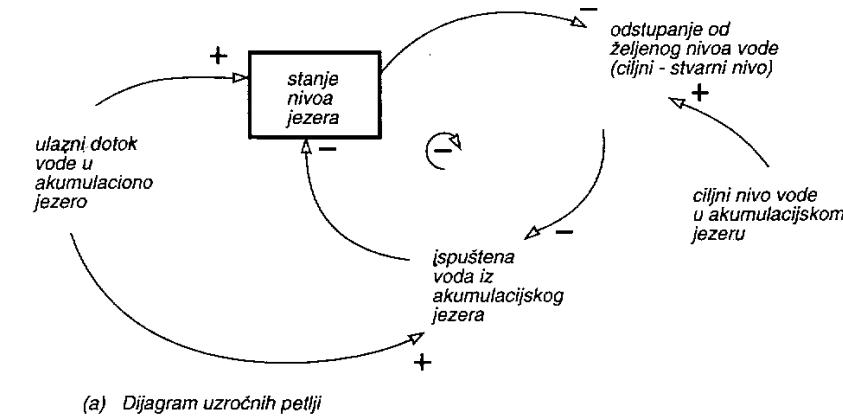
17.3.1. Upravljanje vodostajem u branama

Prikazat ćemo jednostavan sistem s jednom branom i njezinim akumulacijskim jezerom (Roberts i dr., 1983). Akumulacijsko jezero puni se dotokom vode iz rijeke čiji se intenzitet mijenja tijekom vremena. Sistemom se upravlja regulacijom ispuštanja vode iz akumulacijskog jezera. Najjednostavniji način upravljanja može biti preko cilja da se zadrži stalan nivo vode u akumulaciji. Odluka o ispuštanju količine vode temelji se na količini dotoka vode te na veličini razlike stvarnog i željenog nivoa vode. *Dijagram uzročnih petlji i dijagram toka* takva sistema prikazani su na slikama 17.5a i 17.5b. S obzirom na jednostavnost modela, ovdje ga nećemo posebno obrazlagati.

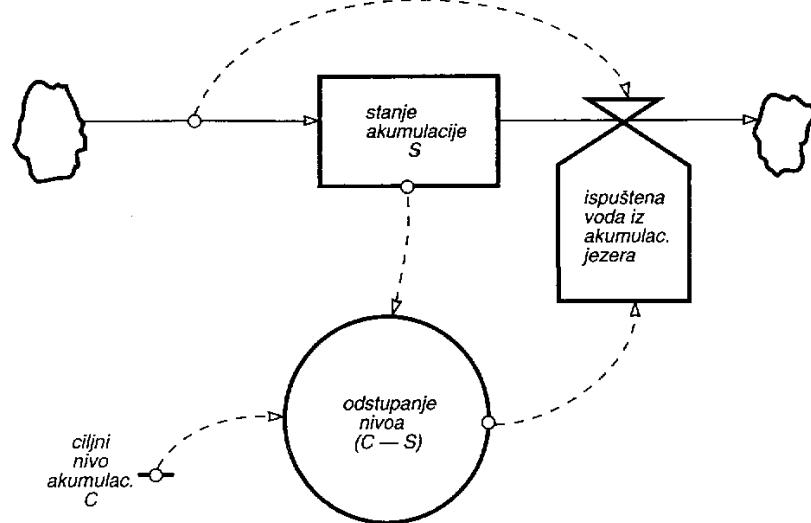
DYNAMO-program ovog jednostavnog sistema upravljanja vodostajem brane može izgledati ovako:

*-----
* UPRAVLJANJE VODOSTAJEM U BRANI
NOTE - ULAZNI TOK VODE (SKOK 10% NAKON 4 DANA) -
R DOTOK.KL=20+STEP(0.1,4)
NOTE - PROMJENA NIVOA AKUMULACIJE -
L AKUMULAC.K=AKUMULAC.J+DT*(DOTOK.JK-ISPUST.JK)
N AKUMULAC=100
NOTE - ISPUSTENI TOK -
NOTE (PROSJDOTOK = DUGOROCNI PROSJ. DOTOKA VODE)
R ISPUST.KL=PROSJDOTOK-ODSTUP.K
C PROSJDOTOK=130
NOTE - ODSTUPANJE OD CILJNOG NIVOA AKUMULACIJE -
A ODSTUP.K=CILJ-AKUMULAC.K
N CILJ=80
*

Način upravljanja može se, naravno, poboljšati, i to na različite načine. Može se, na primjer, umjesto dugoročnog prosjeka dotoka promatrati prosjek u posljednjem tjednu; umjesto ispuštanja cijele razlike suviška ciljnog iznad stvarnog nivoa može se ispuštiti samo jedan njezin dio (zbog manjih fluktuacija nivoa akumulacije) itd. Također je moguće postaviti i druge ciljeve za odlučivanje o strategiji ispuštanja vode iz akumulacije, na primjer uzimajući u obzir sezonske potrebe za vodom u području ispod brane u koje se voda ispušta. Cijeli spektar različitih pristupa ovom problemu prikazan je u knjizi (Roberts i dr., 1983).

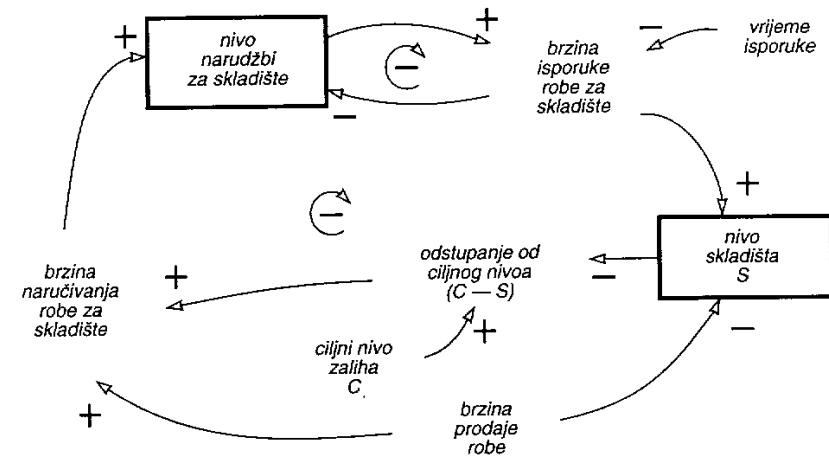


(a) Dijagram uzročnih petlji

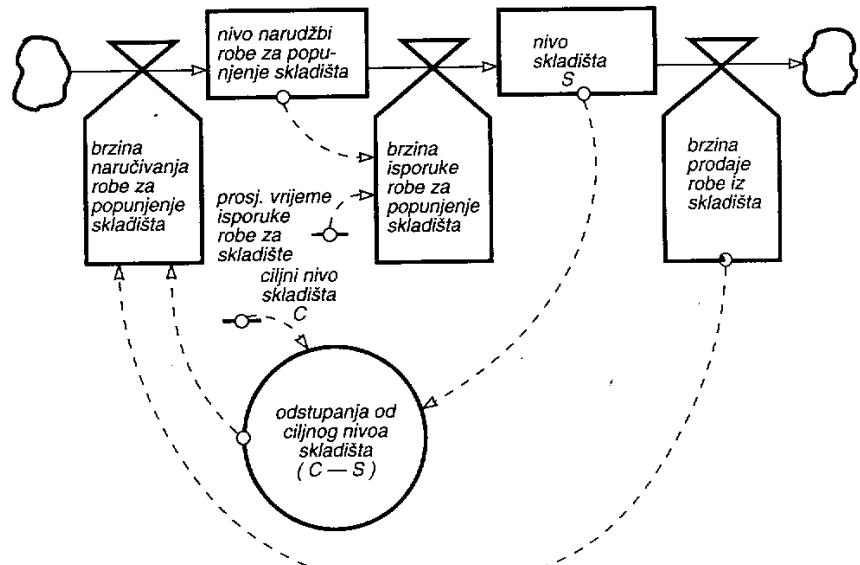


(b) Dijagram toka

Slika 17.5. Upravljanje s vodostajem na branama (Roberts i dr., 1983)



(a) Dijagram uzročnih petlji



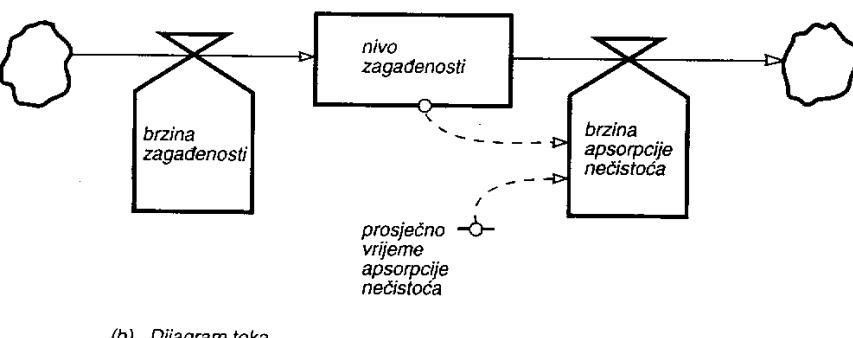
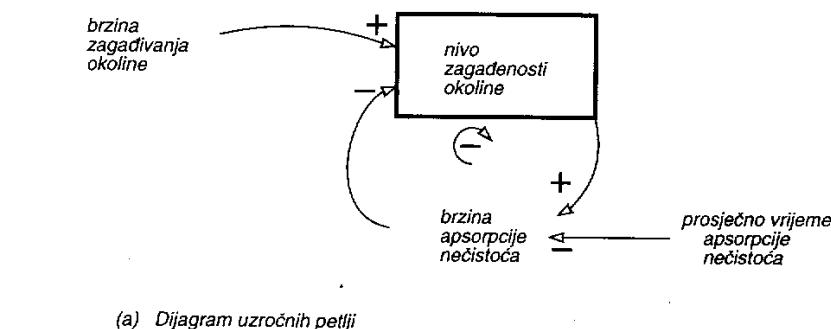
(b) Dijagram toka

Slika 17.6. Upravljanje skladištenjem robe

17.3.2. Upravljanje nivoom zaliha u skladištu

Primjer koji smo odabrali prikazuje skladište sa zalihom robe. Skladište se prazni kako se roba prodaje, a puni se u skladu s odabranim načinom odlučivanja o brzini popunjavanja skladišta. Pretpostaviti ćemo da se odluka o naručivanju robe za skladište temelji na stanju dviju veličina: na brzini prodaje robe te na razlici između ciljnog i stvarnog nivoa popunjavanja skladišta. Osim toga, pretpostaviti ćemo da u sistemu postoji samo kašnjenje vezano uz vrijeme potrebno da se roba za popunjavanje skladišta isporuči na temelju dane narudžbe.

Na slikama 17.6a i 17.6b prikazani su odgovarajući *dijagrami uzročnih petlji* i *dijagrami toka* koje će čitalac nakon prolaska dosadašnjeg teksta o sistemskoj dinamici moći bez problema sam interpretirati. Pomoću takvih modela mogu se tražiti adekvatne strategije odlučivanja koje će osigurati zadovoljavajuće ponašanje sistema u raznim uvjetima: npr. uz postojanje naglog porasta ili pada prodaje robe, uz različita vremena isporuke robe za popunjavanje skladišta i sl.



Slika 17.7. Zagadivanje okoline (prema Roberts i dr., 1983)

17.4. ZAGAĐIVANJE OKOLINE

Jedan od najtežih problema koji se pojavljuju kao posljedica razvoja tehnologije je zagađivanje okoline. Zagađuju se zrak, zemlja, rijeke, jezera, mora. Zagadivači su različiti industrijski pogoni koji različitim dinamikom ispuštaju kemikalije koje su uzrok zagađenju. Neki od uzroka zagađenosti se apsorbiraju brže, neki sporije, a neke priroda ili uopće ne može apsorbirati ili ih može apsorbirati samo do neke koncentracije. Modeliranje i simulacija procesa zagađivanja značajni su kako zbog predviđanja razvoja stupnja zagađenosti u vremenu, tako i zbog ispitivanja mogućnosti utjecaja na zagađivanje.

Opisat ćemo krajnje pojednostavljen model širenja i apsorpcije zagađenosti (Roberts i dr., 1983). Pretpostavimo da postoji izvor iz kojeg se širi uzrok zagađivanja (npr. u zrak ili rijeku). Taj izvor može ispušтati nečistoće stalno, u povremenim kratkim vremenskim intervalima i sl. Pretpostavlja se da se nečistoće mogu apsorbirati u sredini u kojoj su ispušтene, ali s nekim vremenom kašnjenja povezanim za kemijske procese koji dovode do apsorpcije nečistoća. *Dijagrami uzročnih petlji* i *dijagram toka* sistema prikazani su na slikama 17.7a i 17.7b. S obzirom na jednostavnost modela, za njega nije potrebno posebno objašnjenje.

DYNAMO-program za širenje i apsorpciju zagađenosti, ako se ona pojavljuje u obliku periodičnih udara (PULSE funkcija) i ako se apsorpcija zagađenosti odvija s eksponencijalnim kašnjenjem prvog reda, izgleda ovako:

```

* -----
* -----
* SIRENJE I APSORPCIJA ZAGADJENOSTI
NOTE - PROMJENA NIVOA NECISTOCA U VREMENU -
L  NECISTOCA.K=NECISTOCA.J+DT*(ZAGADJV.JK-APSORPC.JK)
N  NECISTOCA=50
NOTE - IZVOR ZAGADJENJA -
NOTE (SVAKIH 14 DANA PO 200 JEDINICA ZAGADJENJA)
R  ZAGADJV.KL=(1/DT)*PULSE(200,1,14)
NOTE - APSORPCIJA NECISTOCA -
R  APSORPC.KL=NECISTOCA.K/T
N  T=3
* -----

```

BIBLIOGRAFIJA

Cilj ove bibliografije jest da, osim literature citirane u knjizi, čitaocu pruži i više različitih informacija o najznačajnijim tekstovima objavljenim u području simulacijskog modeliranja, o knjigama objavljenim kod nas, te o specijaliziranim časopisima, kongresima i društvinama iz ovog područja. Time se čitaocu omogućuje lakše snalaženje u obilju literature i lakši izbor knjiga koje mu mogu biti korisne. Osim toga, ova mu bibliografija daje praktične informacije koje mu omogućuju nalaženje relevantnih časopisa, konferencija i stručnih društava.

NAJZNAČAJNIJE KNJIGE

Knjige koje su ovdje prikazane odražavaju, naravno, iskustvo i afinitete autora ovog teksta, uzevši u obzir i pristupačnost literature. Klasifikacija knjiga u nekoliko osnovnih metodoloških grupa ima za cilj da pomogne čitaocu da se lakše orientira pri izboru knjige od interesa. Knjige su mogle biti klasificirane i drukčije, ili formalnije opisane s određenim brojem atributa, ali nam se pristup koji smo napravili činio čitkijim i privlačnijim.

Diskretna simulacija i sistemska dinamika

G. Gordon, *System Simulation*, Prentice-Hall, New Jersey, 1969.

Jedna od prvih knjiga iz područja simulacije, koju je napisao autor jezika GPSS. Knjiga je još i danas preporučljiva, posebno za početnike (ali ne samo za njih). Opisana je klasifikacija modela, osnovne ideje simulacijskog modeliranja, kontinuirana simulacija, sistemska dinamika (odnosno industrijska dinamika), diskretna simulacija i osnovni statistički aspekti simulacije. Nekoliko poglavlja je posvećeno simulacijskim programskim jezicima (DYNAMO, GPSS, SIMSCRIPT, simulacija u FORTRANu). Knjiga je napisana izvanredno pregledno i čitko.

M. Pidd, *Computer Simulation in Management Science*, Wiley, Chichester, 1984.

Knjiga opisuje osnovne ideje simulacije, tipove simulacije, diskretnu simulaciju, statističke aspekte simulacije i sistemsku dinamiku. Opisane su simulacijske strategije, vrlo detaljno trofazna strategija te konceptualno modeliranje pomoću dijagrama ciklusa aktivnosti. Dan je simulacijski kod napisan u jeziku BASIC. Prikazane su osnove sistemske dinamike i demonstrirane na dva složenija primjera. Knjiga je vrlo čitka, i pisana jasnim jezikom.

Drugo izdanje ove knjige (Wiley, 1988) sadrži i dva nova poglavlja, jedno posvećeno vizualnim aspektima simulacije, a drugo vrednovanju modela. Programi opisani u knjizi mogu se naručiti i na disketama.

Miješana diskretno-kontinuirana simulacija

A. A. B. Pritsker, *Introduction to Simulation and SLAM II*, 2nd Edition, Halstead Book Press, New York, 1984.

Opisan je simulacijski jezik SLAM II koji ujedinjuje tri različita pristupa: simulacija mreža, diskretna simulacija i kontinuirana simulacija. Oni se mogu koristiti bilo kao posebne metode modeliranja, bilo u različitim kombinacijama. Upravo je mješavito diskretno-kontinuirano simulacijsko modeliranje od posebnog interesa, budući da je ono vrlo rijetko opisivano u literaturi. Svi ovi pristupi nisu prikazani kao općenite metode, već na nivou jezika SLAM II. Knjiga obiluje brojnim i raznovrsnim potpuno riješenim i programiranim primjerima koji ilustriraju svaki od pristupa i njihovih kombinacija.

Diskretna simulacija – sistemski pristup

B. P. Zeigler, *Theory of Modeling and Simulation*, Wiley, New York, 1976.

Knjiga daje pokušaj fundiranja teorije simulacije i modeliranja, temeljene na sistemskom pristupu. Opisani su osnovni elementi i ciljevi modeliranja i simulacije, te neformalni i formalni opis modela. Uveden je pojam eksperimentalnog okvira koji je imao utjecaja na daljnji razvoj modeliranja. Najveći dio knjige posvećen je diskretnoj simulaciji, ali je u jednom dijelu opisana i kontinuirana simulacija.

Diskretna simulacija – veza sa računarskim znanostima

J. B. Evans, *Structures of Discrete Event Simulation: An Introduction to the Engagement Strategy*, Ellis Horwood, Chichester, 1988.

Knjiga prikazuje ključne računarske aspekte diskretne simulacije. Detaljno su opisani mehanizmi planiranja budućih događaja i strategije diskrette simulacije. Od posebnog interesa je opis grafičkih metoda konceptualnog modeliranja i originalna klasifikacija te opis simulacijskih složenosti. Prikazana je i novorazvijena strategija angažmana. Izvanredno bogata te metodološki i tehnički sadržajna knjiga koja zahvaljuje poznavanje osnova računarskih znanosti.

Diskretna simulacija – programski aspekti

W. Kreutzer, *System Simulation: Programming Styles and Languages*, Addison-Wesley, Sydney, 1986.

Knjiga daje opsežan prikaz programskih aspekata simulacijskog modeliranja, i to u području Monte Carlo simulacije, kontinuirane simulacije, diskrette simulacije i kombinirane kontinuirano-diskrette simulacije. Prikazane su karakteristike i pristupi simulaciji u svim tim područjima, popraćene nizom kompletnih primjera koji uključuju programska rješenja. Osnovni programski jezici korišteni u knjizi su Pascal i SIMULA, a osim toga je prikazano još desetak drugih jezika. Za sve prikazane tipove simulacije dana je biblioteka baznih programskih rutina. Knjiga ima izvanrednu praktičnu vrijednost, a uz to i vrlo jasni metodološki prilaz te originalne primjere.

M. Pidd (Ed.), *Computer Modelling for Discrete Simulation*, Wiley, Chichester, 1989.

Knjiga je posvećena programskim i računarskim aspektima diskrete simulacije. Opisane su osnovne karakteristike simulacijskih jezika, veći broj softverskih okolina za podršku simulaciji, vizuelna interaktivna simulacija, korištenje koncepata i alata umjetne inteligencije u simulaciji te razvoj simulacijskog softvera u jezicima Pascal i C (popraćen i odgovarajućim programskim kodom, koji se može naručiti i na disketama). Knjiga je vrlo čitka i čini dobro organiziranu cjelinu, bez obzira na to što ju je pisalo više autora.

R. Davies i R. O'Keefe, *Simulation Modelling with Pascal*, Prentice-Hall, New York, 1989.

Ova knjiga opisuje diskretnu simulaciju u jeziku Pascal. Opisani su principi razvoja simulacije u jeziku Pascal, a uz to je dana kompletna dokumentacija i programski kod paketa Pascal-SIM. Osobito je zanimljivo što ovaj programski paket ostvaruje i vizuelni prikaz izvođenja simulacije. Pokazana je primjena ovog paketa na dva realna sistema: pacijenti u bolnici i kvarovi opreme u proizvodnim sistemima.

R. Paul i D.W. Balmer, *Simulation Modelling*, Chartwell-Bratt Student-Text Series, Lund, u tisku.

Knjiga prikazuje simulacijski paket eELSE pisan u jeziku Pascal i osnovan na prikazu modela pomoću dijagrama ciklusa aktivnosti te trofaznoj strategiji simulacije. Dan je listing modula koji čine paket eELSE i primjeri njegova korištenja. Opisano je i korištenje interaktivnog programskog generatora LANGEN koji automatski generira kod simulacijskog programa polazeci od konceptualnog modela u obliku dijagrama ciklusa aktivnosti.

B. P. Zeigler, *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, Boston, 1990.

Knjiga prikazuje realizaciju diskrette simulacije temeljenu na prikazu znanja pomoću višeslojnog programskog jezika osnovanog na objektno orientiranom pristupu i jeziku Lisp. Programski paket je primjenljiv i na višeprocesorskim računalima. Knjiga daje metodološku podlogu ovom pristupu simulaciji i demonstrira ga na izvanredno zanimljivim primjerima: studiji performansi jednostavnih i distribuiranih višeprocesorskih arhitektura, kooperativnoj organizaciji robota, modeliranju inteligentnih autonomnih robota itd. Knjiga zahtjeva dobro poznавanje osnova računarskih znanosti, prikaza znanja, sistemskog pristupa diskretnoj simulaciji i jezika Lisp.

Diskretna simulacija – programski jezik GPSS

T. J. Schriber, *Simulation Using GPSS*, Wiley, New York, 1974.

Vjerojatno najpoznatija i najviše citirana knjiga o jeziku GPSS. Opisuje unutrašnju logiku rada GPSS procesora te osnovne i napredne elemente jezika. Elementi jezika demonstrirani su na velikom broju raznovrsnih primjera.

P. A. Bobillier, B. C. Kahan i A. R. Probst, *Simulation with GPSS and GPSS V*, Prentice-Hall, Englewood Cliffs, 1976.

Vrlo lijep, sistematski i čitak prikaz jezika GPSS i mogućnosti njegove primjene u modeliranju i simulaciji realnih sistema. Osim demonstracije svih sintaktičkih elemenata prikazan je i niz manjih poučnih primjera te nekoliko složenih simulacijskih studija s kompletним GPSS-programima: automatsko skladište, podzemna željezница,

proizvodni pogon i teleprocesing sistem. Jedno poglavlje posvećeno je otkrivanju grešaka u programima.

Diskretna simulacija – statistički aspekti

To je područje s najvećim brojem publiciranih radova i knjiga. Odabrane su knjige sa sistematskim pristupom koje pokrivaju velik dio raspoloživih metoda i ilustrirane su razumljivim i prikladnim primjerima.

G. S. Fischman, *Concepts and Methods in Discrete Event Digital Simulation*, Wiley, New York, 1973.

Klasična knjiga jednog od osnivača statističkog pristupa simulaciji. Knjiga ima matematički pristup i vrlo malo primjera.

J. P. C. Kleijnen, *Statistical Techniques in Simulation*, Vol. I i II, Marcel Dekker, New York, 1974.

Također klasična knjiga o statističkom pristupu simulaciji. Vrlo opsežna i detaljna, matematički orijentirana, s vrlo velikim brojem referenci, bez primjera.

A. M. Law i W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1982.

Jedna od najboljih i najpopularnijih knjiga o statističkim aspektima simulacije. Sistematski, čitko i jasno pisana te ilustrirana nizom riješenih primjera. Uvodi terminirajući i stacionarni tip simulacije. Uključuje i simulacijski paket SIMLIB pisan u jeziku FORTRAN.

J. Banks i J. S. Carson II, *Discrete-Event System Simulation*, Prentice-Hall, Englewood Cliffs, 1984.

Opsežna knjiga koja obuhvaća sve statističke aspekte simulacije. Sadrži veoma velik broj riješenih primjera koji znatno olakšavaju razumijevanje metoda. Sadrži također dosta grafičkih ilustracija. Ponekad primjeri prethode opisu metode i otežavaju praćenje teksta.

J. P. C. Kleijnen, *Statistical Tools for Simulation Practitioners*, Marcel Dekker, New York, 1987.

Autor se ograničio na one metode za koje je, vlastitim iskustvom, ustanovio da su korisne u praktičnoj primjeni simulacije. Prikazane su tehnike za određivanje dužine izvođenja simulacije i intervala pouzdanosti, regresijska analiza i eksperimentalni dizajn. Ukratko je prikazano nekoliko realnih simulacijskih studija. Knjiga sadrži vrlo veliki broj referenci.

P. A. W. Lewis i E. J. Orav, *Simulation Methodology for Statisticians, Operations Analysts and Engineers*, Vol. I, Wadsworth and Brooks/Cole, Pacific Grove, 1989.

Knjiga opisuje simulacijsku metodologiju za probleme u diskretnoj simulaciji i matematičkoj statistici. U nju su uključene statističke metode temeljene na grafičkim tehnikama i istraživačkoj ('exploratory') analizi podataka. Izlaganje je popraćeno nizom zanimljivih primjera, riješenih problema te izvanredno velikim brojem grafičkih ilustracija. Može se naručiti i softverski paket SUPER SMTBPC, pisan u jeziku FORTRAN-77, koji omogućuje grafičku analizu višefaktorskih simulacijskih eksperimenata. Osim toga, ovaj paket sadrži i generator slučajnih brojeva, efikasne generatore često korištenih razdoba i više drugih statističkih rutina.

Sistemska dinamika

J. W. Forrester, *Industrial Dynamics*, MIT Press, Cambridge, Mass., 1961.

Klasična knjiga autora sistemske dinamike. Vrlo opširno su objašnjeni osnovni elementi sistemske dinamike i primijenjeni na modeliranje i simulaciju ponašanja niza aspekata industrijskih sistema. Temeljena na jeziku DYNAMO.

R. G. Coyle, *Management System Dynamics*, Wiley, London, 1977.

U knjizi je opsežan i detaljan opis principa i tehnika sistemske dinamike. Popraćena nizom manjih i tri veća primjera. Velik broj grafičkih ilustracija. Temeljena na jeziku DYMAP.

N. Roberts i dr., *Introduction to Computer Simulation: A System Dynamics Approach*, Addison-Wesley, 1983.

Opsežna knjiga u kojoj se veoma velik broj manjih i većih zanimljivih primjera isprepliće s objašnjavanjem metode. Velik dio primjera nije riješen. Temeljena je na jeziku DYNAMO.

M. F. Aburdene, *Computer Simulation of Dynamic Systems*, Wm. C. Brown Publishers, Dubuque, Iowa, 1988.

Knjiga opisuje kontinuiranu simulaciju i sistemsku dinamiku. Orientirana je na diferencijalne jednadžbe i njihovo numeričko rješavanje. Sadrži niz zanimljivih, riješenih problema popraćenih programskim kodom rješenja. Koristi više simulacijskih jezika i jezik Pascal.

E. F. Wolstenholme, *Systems Enquiry: A System Dynamics Approach*, Wiley, Chichester, 1990.

Izvanredan tekst o osnovama i mogućnostima primjene sistemske dinamike. Osnovne ideje i tehnike sistemske dinamike opisane su sistematično, jasno i originalno. Autor je podijelio sistemsku dinamiku u kvalitativne (konceptualni modeli) i kvantitativne metode (računarski modeli), i za obje opisao tehnike modeliranja i način analize. Opisan je koncept optimalizacije u sistemskoj dinamici i prikazana njegova primjena. Uz niz manjih primjera prikazane su i tri vrlo zanimljive simulacijske studije realnih sistema: dugoročna evolucija kompanije, upravljanje sistema transporta ugljenom i obrambena analiza. Knjiga je temeljena na jezicima DYMAP2 i STELLA.

KNJIGE OBJAVLJENE KOD NAS

Diskretna simulacija

V. Žiljak, *Simulacija računalom*, Školska knjiga, Zagreb, 1982.

Sistemska dinamika

A. Munitić, *Komputerska simulacija uz pomoć sistemske dinamike*, Brodosplit, Split, 1989.

ČASOPISI, DRUŠTVA I KONGRESI

Časopisi

- “Simulation”, mjesečnik koji izdaje The Society for Computer Simulation (SCS).
- “Transactions of the SCS”, tromjesečnik koji izdaje SCS.
- “Mathematics and Computers in Simulation”, mjesečnik koji izdaje International Association for Mathematics and Computers in Simulation (IMACS).
- “System Dynamics Review”, mjesečnik koji izdaje System Dynamics Society.
- “ACM Transactions on Modelling and Computer Simulation”, tromjesečnik koji izdaje Association for Computing Machinery (ACM).
- “EUROSIM Simulation News Europe”, četveromjesečnik koji izdaje federacija europskih simulacijskih društava EUROSIM.
- “Shimyureshen” (simulacija), tromjesečnik koji izdaje Japan Society for Simulation Technology.

Društva

- The Society for Computer Simulation (SCS), S.A.D.
- System Dynamics Society, S.A.D.
- EUROSIM, federacija europskih simulacijskih društava.
 - * ASIM, Arbeitsgemeinschaft Simulation
(Austrija, Njemačka i Švicarska)
 - * DBSS, Dutch Benelux Simulation Society
(Belgijska, Nizozemska)
 - * FRANCOSIM, Societe Francophone de Simulation
(Belgijska, Francuska)
 - * ISCS, Italian Society for Computer Simulation
(Italija)
 - * SIMS, Simulation Society of Scandinavia
(Danska, Finska, Norveška, Švedska)
 - * UKSC, United Kingdom Simulation Council
(Velika Britanija)
- CROSSIM, Hrvatsko društvo za simulacijsko modeliranje.
- International Association for Mathematics and Computers in Simulation (IMACS).
- Japan Society for Simulation Technology.
- Chinese Association for System Simulation, Kina.

Kongresi

- Winter Simulation Conference, S.A.D.
- Summer Computer Simulation Conference, S.A.D.

- Eastern Simulation Conference, S.A.D.
- Western Simulation Multiconference, S.A.D.
- European Simulation Multiconference.
- European Simulation Congress, (svake treće godine).
- UKSC Conference, Velika Britanija (svake treće godine).
- Symposium Simulationstechnik, njemačko govorno područje.
- IMACS World Congress on System Simulation and Scientific Computation, (svake treće godine).
- International Conference on System Simulation and Scientific Computation, Kina (svake treće godine).
- International Conference *Information Technology Interface* (redovita sekcija i pozvani predavači iz inozemstva iz područja simulacije).
- Konferencija iz operacijskog istraživanja KOI (redovita sekcija iz područja simulacije).

LITERATURA

- Aburdene, M.F. (1988), *Computer Simulation of Dynamic Systems*, Wm. C. Brown Publishers, Dubuque, Iowa.
- Banks, J. i J.S. Carson II (1984), *Discrete-Event System Simulation*, Prentice-Hall, Englewood Cliffs.
- Bobillier, P.A., B.C. Kahan i A.R. Probst (1976), *Simulation with GPSS and GPSS V*, Prentice-Hall, Englewood Cliffs.
- Cavana, R.Y. i R.G. Coyle (1982), *Dysmap User Manual*, University of Bradford Publications, Bradford.
- Coyle, R.G. (1977), *Management System Dynamics*, Wiley, London.
- Coyle, R.G. (1985), *Representing Discrete Events in System Dynamics Models: A Theoretical Application to Modelling Coal Production*, Journal of the Operational Research Society, Vol. 36, 307-318.
- Čerić, V. (1988), *Simulation Modelling Study for Design of the Airport Terminal Building*, Transportation Planning and Technology, Vol. 13, 43-56.
- Čerić, V. (1990), *Simulation Study of an Automated Guided-vehicle System in a Yugoslav Hospital*, Journal of the Operational Research Society, Vol. 41, 299-310.
- Čerić, V. i R. Paul (1992), *Diagrammatic Representations of the Conceptual Simulation Model for Discrete Event Systems*, Mathematics and Computers in Simulation, Vol. 34, 317-324.
- Davies, R. i R. O'Keefe (1989), *Simulation Modelling with Pascal*, Prentice-Hall, London.
- Dorn, W.S. i D.D. McCracken (1972), *Numerical Methods with Fortran V Case Studies*, Wiley, New York.
- Evans, J.B. (1988), *Structures of Discrete Event Simulation: An Introduction to the Engagement Strategy*, Ellis Horwood, Chichester.
- Forrester, J.W. (1961), *Industrial Dynamics*, MIT Press, Cambridge, Mass.
- Fujimoto, R.M. (1990), *Parallel Discrete Event Simulation*, Communications of the ACM, Vol. 33, 31-53.
- Gordon, G. (1969), *System Simulation*, Prentice-Hall, Englewood Cliffs.
- Gustafsson, L. i M. Wiechowski (1986), *Coupling DYNAMO and Optimisation Software*, System Dynamics Review, Vol. 2, No. 1.
- Graybeal, W.R. i U.W. Pooch (1980), *Simulation: Principles and Methods*, Winthrop, Cambridge, MA.
- Harrier, E., S.P. Norset i G. Wanner (1987), *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer-Verlag, Heidelberg.
- Holmes, D.S. (1970), *A Probabilistic Interpretation of the Delays of Industrial Dynamics*, Union College, Schenectady, New York.
- Hull, T.E. i A.R. Dobell (1962), *Random Number Generation*, SIAM Review, Vol. 4, 230-254.
- Hurrian, R.D. (1989), *Graphics and Interaction*, u M. Pidd (Ed.), *Computer Modelling for Discrete Simulation*, Wiley, Chichester.
- Keloharju, R. (1983), *Relativity Dynamics*, Acta Academiae Oeconomicae Helsingiensis, Helsinki School of Economics, serija A:40.

- Keloharju, R. i E.F. Wolstenholme (1989), *A Case Study in System Dynamics Optimization*, Journal of the Operational Research Society Vol. 40, 221–230.
- Kleijnen, J.P.C. (1974), *Statistical Techniques in Simulation*, Vol. I i II, Marcel Dekker, New York.
- Kleijnen, J.P.C. (1987), *Statistical Tools for Simulation Practitioners*, Marcel Dekker, New York.
- Kreutzer, W. (1986), *System Simulation: Programming Styles and Languages*, Addison-Wesley, Sydney.
- Law, A.M. i J.S. Carson (1979), *A Sequential Procedure for Determining the Length of a Steady-State Simulation*, Operations Research, Vol. 27, 1011–1025.
- Law, A.M. i W.D. Kelton (1982), *Simulation Modeling and Analysis*, McGraw-Hill, New York.
- Lehmer, D.H. (1951), *Mathematical Methods in Large-Scale Computing Units*, Ann. Comput. Labs. (Harvard Univ.), Vol. 26, 141–146.
- Lewis, P.A.W. i E.J. Orav (1989), *Simulation Methodology for Statisticians, Operations Analysts and Engineers*, Vol. I, Wadsworth and Brooks/Cole, Pacific Grove.
- Meadows, D.H. i dr. (1974), Granice rasta (prijevod), Stvarnost, Zagreb.
- Meadows, D.L. (1985), *STRATEGEM I: A Resource Planning Game*, Environmental Education Report and Newsletter, Vol. 14, 9–13.
- Montgomery, D.C. (1976), *Design and Analysis of Experiments*, Wiley, New York.
- Morecroft, J.D.W. (1988), *System Dynamics and Microworlds for Policymakers*, European Journal of Operational Research, Vol. 35, 301–320.
- Myers, R.H. (1971), *Response Surface Methodology*, Allyn and Bacon, Boston.
- Narchal, R.M. (1988), *A Simulation Model for Corporate Planning in a Steel Plant*, European Journal of Operational Research, Vol. 34, 282–296.
- Naylor i dr. (1966), *Computer Simulation Techniques*, Wiley, New York.
- Paul, R. (1989), *Artificial Intelligence and Simulation Modelling*, u M. Pidd (Ed.), *Computer Modelling for Discrete Simulation*, Wiley, Chichester.
- Paul, R. i D.W. Balmer (1991), *Simulation Modelling*, Chartwell-Bratt Student-Text Series, Lund, u tisku.
- Paul, R. i V. Čerić (1993), *Methods of Model Representation in Discrete Event Simulation: An Overview*, u pripremi.
- Pidd, M. (1984), *Computer Simulation in Management Science*, Wiley, Chichester.
- Pugh, A.L. III (1976), *DYNAMO II User's Manual*, MIT Press, Cambridge, MASS.
- Pugh-Roberts Associates (1986), *Professional DYNAMO Introductory Guide and Tutorial*, and *Professional DYNAMO Reference Manual*, Pugh-Roberts Associates, 5 Lee Street, Cambridge, MASS.
- Pritsker, A.A.B. (1984), *Introduction to Simulation and SLAM II*, 2nd Edition, Halstead Book Press, New York.
- Richmond, B.M. (1985), *A Users Guide to STELLA* (2nd printing), High Performance Systems Inc., 13 Dartmouth College Highway, Lyme, NH.
- Roberts, C. i B. Dangerfield (1990), *Modelling the Epidemiological Consequences of HIV Infection and AIDS: A Contribution from Operational Research*, Journal of the Operational Research Society, Vol. 41, 273–289.

- Roberts, N. i dr. (1983), *Introduction to Computer Simulation: A System Dynamics Approach*, Addison-Wesley.
- Schmidt, J.W. i R.E. Taylor (1970), *Simulation and Analysis of Industrial Systems*, Irwin, Homewood, ILL.
- Schriber, T.J. (1974), *Simulation Using GPSS*, Wiley, New York.
- Shannon, R.E. (1975), *System Simulation – the Art and Science*, Prentice-Hall, Englewood Cliffs.
- Sterman, J.D. (1984), *Appropriate Summary Statistics for Evaluating the Historic Fit of System Dynamics Models*, DYNAMICA, Vol. 10, pt. 1.
- Sterman, J.D. (1985), *A Behavioral Model of the Economic Long Wave*, Journal of the Economic Behaviour and Organization, Vol. 6, 17–53.
- Stoer, J. i R. Bulirsch (1980), *Introduction to Numerical Analysis*, Springer-Verlag, New York.
- Taylor, A.J. (1983), *The Verification of Dynamic Simulation Models*, Journal of the Operational Research Society, Vol. 34, 233–242.
- Vrgoč, M. i V. Čerić (1988), *Investigation and Design of Parcel Sorting Systems in Postal Centres by Simulation*, Computers in Industry, Vol. 10, 137–145.
- Watson, J.D. (1968), *The Double Helix*, Penguin Books, London.
- Wolstenholme, E.F. (1990), *Systems Enquiry: A System Dynamics Approach*, Wiley, Chichester.
- Zeigler, B.P. (1976), *Theory of Modelling and Simulation*, Wiley, New York.
- Zeigler, B.P. (1990), *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, Boston.