**1. Modeliranje i modeli, vrste modela.** Modeliranje je jedan od osnovnih procesa ljudskog uma i najvažnije je konceptualno sredstvo koje čoveku stoji na raspolaganju. To je isplativo korišćenje nečega umesto nečega drugog sa ciljem da se dođe do određenog saznanja. Rezultat modeliranja je model; koji je uprošćena i idealizovana slika stvarnosti. On sadrži objekte, atribute i određene pretpostavke o uslovima njegove validnosti. Problem validacije modela proizilazi iz nivoa apstrakcije koji smo upotrebili. Vrste: mentalni [ljudski um ih neprekidno konstruiše kako bi se povezale činjenice sa kojima se svakodnevno susrećemo], verbalni [predstavljaju izraz mentalnih modela u govornom jeziku; spadaju u grupu neformalnih modela], strukturni [stvaraju se na osnovu predstave o strukturi i logici rada sistema ili problema koji se modelira i prikazuju se u obliku čije je značenje precizno definisano], fizički [umanjeni modeli realnog sistema koji se ponašaju kao i originali], analogni [fizički objekat koji se koristi za analizu matematičkog modela drugog objekta, a sa kojim ima isti ili sličan matematički model], matematički [oni kod kojih su veze između objekata definisane matematičkim relacijama; iz klase je apstraktnih modela], simulacioni, računarski [su prikaz konceptualnih u obliku programa za računar]. Dele se na materijalne i simboličke.

**2. Neformalni i formalni modeli.** Neformalni opis modela daje osnovne pojmove o modelu, nepotpun je i neprecizan. Prilikom izgradnje ovakvog opisa vrši se podela na objekte, opisne promenljive i pravila interakcije objekata. Objekti su delovi modela, opisne promenljive su stanja objekata u pojedinim vremenskim trenucima, a pravila definišu način na koji objekti utiču jedan na drugi u cilju promene njihovog stanja. U ovakvim modelima, opis modela je nekompletan [ukoliko ne sadrži sve situacije koje mogu da nastupe], nekonzistentan [ukoliko se pravilima za jednu istu akciju dobijaju kontradiktorne akcije], nejasan [ukoliko je potrebno obaviti dve ili više akcija, a čiji redosled nije definisan]. Formalni opis treba da obezbedi veću preciznost u opisu modela, ali i da formalizuje postupak ispitivanja nekompletnosti, nekonzistentnosti i nejasnosti. Najvažnija stavka jeste usmeravanje pažnje na one karakteristike objekta koje su od najvećeg značaja. Obuhvata dva koraka razvoja modela: faza izgradnje ili formalizacija i analiza i korišćenje modela zarad efikasnijeg upravljanja realnim sistemom. Opšti koraci izgradnje modela: sistem [i sam model] moraju obuhvatati samo fenomene od interesa, modeli trebaju da sadrže samo relevantne elemente sistema, model ne sme da pojednostavi problem, razuno ga je rastaviti na više dobro definisanih modula sa tačno određenom funkcijom, predlaže se korišćenje proverenih metoda i algoritama, potrebna je provera logičke i kvantitativne ispravnosti modela i pojedinačnih modula.

**3. Računarska simulacija.** Pod simulacijom se podrazumeva proces izgradnje apstraktnih modela za neke sisteme ili podsisteme realnog sveta i obavljanje experimenata nad njima. Kada se oni vrše na računaru, onda je u pitanju računarska simulacija. Računari se koriste za razvoj modela i izvođenje proračuna na osnovu modela. Simulacija obuhvata tri elementa: realni sistem [uređen međuzavisni skup elemenata koji formiraju jedinstvenu celinu i deluju zajedno; on je izvor podataka o ponašanju, a ti podaci se javljaju u obliku zavisnosti x(t) i koriste se za specifikaciju modela], model [apstraktni prikaz sistema, daje njegovu strukturu, komponente i uzajamno delovanje; u računaru, model je skup instrukcija koje služe za generisanje ponašanja simuliranog sistema], računar [uređaj sposoban za izvršenje instrukcija modela]. Modeliranje uspostavlja veze između realnog sistema i modela, a simulacija između modela i računara. Relacija modeliranja se odnosi na validnost modela, tj. njegovu sposobnost vernog prikaza realnog sistema. Relacija simulacije odnosi se na proveru da li simulacini računar verno prenosi model na računar kao i na tačnost kojom računar izvršava instrukcije modela. Proces modeliranja i simulacije: procesom modeliranja se upravlja na osnovu ciljeva koji se generišu van granica sistema. Svaki novi cilj inicira aktivnost sinteze modela. Pri sintezi modela se koristi raspoloživo znanje iz baze modela i baze podataka. Ove baze čuvaju i organizuju znanje o relanom sistemu. Potom slede faze simulacije i validacije. Validacija vodi novom experimentisanju nad realnim sistemom i može da zahteva dodatne modifikacije ili čak odbacivanje i reinicijalizaciju prvobitnog modela. Računarska simulacija ima u osnovi model sistema. Sistem je uređaj ili proces koji postoji ili se planira.

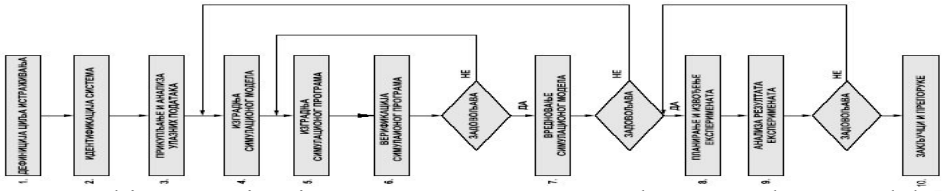
**4. Istorijski pregled razvoja simulacije.** 1600. fizičko modeliranje, 1940. pojava elektronskih računara, 1955. simulacija u avio-industriji, 1960. simulacija proizvodnih porcesa, 1970. simulacija velikih sistema uključujući ekonomske, društvene i ekološke, 1975. sistemski pristup u simulaciji, 1980. simulacija diskretnih stohastičkih

sistema i viši nivo učešća u SPO, 1990. integracija računarske simulacije, veštačke inteligencije, računarskih mreža i MM tehnologija.

**5. Karakteristike simulacionog modeliranja.** Računarska simulacija se zasniva na ideji experimentisanja sa modelom realnog sistema na računaru tokom vremena. Simulacioni modeli prikupljaju podatke o promenama stanja sistema i izlaza, fokusirajući se na ponašanje individualnih komponenti sistema. Simulacioni experimenti daju kao razultat skup tačaka, tj. vrednosti zavisnih promenljivih za pojedine vrednosti nezavisnih promenljivih. Simulacioni modeli najčešće su modeli dinamičkih sistema, ali i onih koji se ne mogu opisati niti rešavati matematičkim sredstvima. Ti modeli su najčešće dati u obliku konceptualnih i računarskih modela. Izgradnja simulacionih modela vezana je za alate i tehnike računarskih nauka i sistemske analize, operacionih istraživanja i numeričke analize. Koriste se i pristupi teroije verovatnoće i statistike.

**6. Potreba za simulacijom.** Razlozi za predstavljanje sistema modelom: experiment nad realnim sistemom može da bude skup ili nemoguć, analitički model nema analitičko rešenje, sistem može da bude složen da bi se opisao analitički, experimentisanje sa realnim sistemom je neisplativo, model može imati za cilj spoznavanje nepoznate strukture realnog sistema, realan sistem ne omogućava menjanje parametara, ponekad je potrebno simulirati razaranje nekog sistema, pri simulaciji vreme se sažima, pri realnom experimentu javlja se greška mernih uređaja, model daje mogućnost zaustavljanja odvijanja experimenta i merenja promenljivih u tom trenutku.

**7. Mogućnosti primene simulacije, prednosti i nedostaci simulacije.** Situacije u kojima je moguće uspešno primeniti simulaciju: simulacija omogućava experimentisanje koje uzima u obzir sveukupne interakcije složenog sistema, informacione i organizacione promene ili promene u okruženju mogu se simulirati, znanje stečeno tokom simulacije može poslužiti za poboljšanje sistema koji se ispituje, menjanjem ulaza a time i izlaza dolazimo do informacija o tome koje su i kako utiču najvažnije pormenljive u sistemu, može se koristiti u pedagoške svrhe, može se experimentisati sa novim politikama pr eimplementacije, može se koristiti za verifikaciju analitičkih rešenja. Prednosti: jednom izgrađen model se može koristiti višestruko, simulacione metode se mogu koristiti kao pomoć kod analize, podaci se mogu jeftinije dobiti, lakše ih je primeniti, nemaju ograničenja poput analitičkih modela, ponekad je simulacija jedino sredstvo za rešavanje problema, moguće je rešavati složene dinamičke probleme sa slučajnim promenljivim koji su nedostupni metamatičkom modeliranju. Nedostaci: modeli mogu biti skupi, potrebno je izvesti više simulacionih experimenata, pojedinačno izvođenje može zahtevati i dosta vremena i memorije računara, ne dobijaju se zavisnosti izlaza od ulaza ni optimalna rešenja, potrebno je poznavanje više različitih modela i alata, vrednovanje modela je složeno.




rnih problema pomoću simulacionog a prethodne korake. Koraci: definicija ıovih interakcija, način rada, veze sa liza, izgradnja simulacionog modela, je programa bilo pisanjem koda bilo automatskim generisanjem programa na osnovu konceptualnog modela], verifikacija simulacionog programa, vrednovanje simulacionog modela, planiranje simulacionih experimenata i njihovo izvođenje, analiza rezultata experimenata, zaključci i preporuke.

**9. Podele simulacionih modela.** Osnovna podela: prema vrsti promenljivih u modelu i prema načinu na koji se menja stanje u modelu tokom vremena. Deterministički modeli su oni kojima je novo stanje sistema određeno u potpunosti prethodnim. Stohastički su oni čije se ponašanje ne može predvideti, ali se mogu odrediti verovatnoće promene stanja sistema. U diskretnim sistemima stanje sistema se menja samo u pojedinim tačkama u vremenu. Te promene su događaji. U kontinualnim modelima promenljive stanja se menjaju kontinualno u vremenu. Na računaru se kontinualne veličine moraju aproksimirati skupom diskretnih vrednosti. Mogući su i mešoviti modeli.

**10. Vrste simulacionih modela.** Vrste se razlikuju po pristupu modeliranju, klasi porblema koji se rešava i po tehnikama modeliranja i simulacije koje su za njih razvijene. To su Monte Karlo [jedina je statička, ostale su dinamičke], kontinualna simulacija, simulacija diskretnih događaja, mešovita simulacija. Monte Karlo koristi slučajne brojeve i u rešavanju problema se koristi stvaranje uzoraka iz raspodele slučajnih promenljivih. Primene: kod determinističkih porblema koje je skupo ili teško rešavati, kod složenih fenomena koji nisu dovoljno poznati, statistički porblemi koji nemaju analitička rešenja. Kontinualna simulacija se koristi kod dinamičkih problema kod kojih se promenljive menjaju kontinualno u vremenu. Rešavaju se ili jednostavni problemi, koji su opisani detaljno i kod kojih su promene glatke i prirodno se opisuju diferencijalnim jednačinama, ili problemi koji nastaju opisom veoma složenih sistema u agregiranom obliku, u kojem se niz elemenata sistema redukuje na manji broj komponenti i u kojima se promene u sistemu aproximiraju konstantnim brzinama promene. Tipovi kontinualnih simulacionih modela: modeli koji se opisuju običnim diferencijalnim jednačinama, modeli koji se opisuju sistemima parcijalnih diferencijalnih jednačinna, modeli dinamike sistema. Simulacija diskretnih događaja je specifična metodologija simulacije koja se bavi modeliranjem sistema koji se mogu predstaviti skupom događaja. A događaj je ovde diskretna promena stanja entiteta sistema. Simulacija prati događaje i tako obrazuje pomak vremena simulacije. Sistemi koji se ovim putem modeliraju su i dinamički i stohastički. Mešovita simulacija koristi se kod sistema koji sadrže procese koji teku kontinualno i događaje koji dovode do diskontinuiteta u ponašanju sistema. Veza se postiže pomoću vremenskih i događaja stanja. Vremenske generiše mehanizam upravljanja događajima a stanja aktivira mehanizam pomaka vremena sa konstantnim prirastom, čiji je vremenski interval mali i koji je karakterističan za kontinualnu simulaciju. Osnovni zahtevi prilikom izbora tipa simulacionog modela jesu: jednostavnost i razumljivost, cena i efikasnost.

**11. Klasifikacija modela.** Klasifikacija u odnosu na promenljive - svaki model poseduje opisne promenljive; koje je moguće podeliti na one koje je moguće posmatrati i one koje nije; sve opisne pormenljive jednog modela mogu se podeliti na ulazne, izlazne i pormenljive stanja; svaka promenljiva ima skup stanja koja joj se mogu dodeliti – opseg, kao i funkciju koja opisuje promene tih stanja. Modeli koji nemaju ni jednu promenljivu stanja su modeli bez memorije, a ukoliko postoji makar jedna promenljiva stanja, tada se radi o modelu sa memorijom. Zavisno od ulaznih promenljivih, modeli mogu biti autonomni [bez ulaznih promenljivih] i neautonomni [sa ulaznim promenljivima]. Ukoliko autonomni ne sadrže izlaznu promenljivu, nazivaju se zatvoreni; u drugom slučaju su otvoreni, kao i svi neautonomni modeli koji se, uostalom, i sami dele na one sa i one bez izlaznih promenljivih. Klasifikacija u odnosu na prirodu opsega vrednosti promenljivih modela – opisne promenljive mogu uzimati vrednosti iz diskretnog ili kontinualnog skupa, pa u skladu sa tim imamo: modele sa diskretnim stanjima [sve 3 promenljive uzimaju vrednosti iz diskretnog skupa], sa kontinualnim stanjima i sa mešovitim stanjima. Klasifikacija u odnosu na prirodu opsega vrednosti promenljive vreme – taj opseg može biti diskretan i kontinualan te razlikujemo modele sa kontinualnim vremenom i kontinualnim promenama stanja i modele sa kontinualnim vremenom i diskretnim promenama stanja. U diskretnim vremenskim modelima, vreme se povećava u inkrementima koji ne moraju biti ekvidistantni. Opisna promenljiva, bilo kontinualna ili diskretna, raspoloživa je samo u tim vremenskim trenucima. U odnosu na to, razlikujemo: modele sa diskretnim vremenom i kontinualnim pormenama i modele sa diskretnim vremenom i diskretnim promenama stanja. Promenljive simulacionog modela se mogu menjati bilo kontinualno/diskretno u kontinulnom/diskretnom vremenu, dakle u te 4 kombinacije. A kakve će se promenljive koristiti zavisi od situacije koja se modelira, od svrhe modela, računara, simulatora i similacionog jezika. Klasifikacija u odnosu na vremensku zavisnost modela – ukoliko je struktura modela zavisna od vremena tada se radi o vremenski promenljivom-varijantnom modelu. U suprotnom, imamo vremenski nepromenljiv-invarijantan model. Klasifikacija u odnosu na determinizam – čime se razmatra uključivanje slučajnih promenljivih u opis modela. U determinističkim modelima, vrednosti promenljivih stanja i ulaznih promenljivih u jednom trenutku jednoznačno određuju vrednost promenljivih stanja u sledećem trenutku. Takvi modeli ne sadrže slučajne pormenljive. Dok stohastički modeli sadrže makar jednu. Klasifikacija u odnosu na predviđanje budućnosti - oni modeli koji za izračunavanje vrednosti promenljivih stanja u obzir uzimaju i buduće vrednosti ulaznih promenljivih [anticipatorski modeli]. Klasifikacija u odnosu na linearnost – linearni modeli menjaju stanja poštujući zakonitosti linearnih transformacija. Klasifikacija prema vrsti računara – analogni, digitalni i hibridni. Klasifikacija u odnosu na formalni opis modela – modeli sa kontinualnim promenama vremena se opisuju

diferencijalnim jednačinama i spadaju u kontinualne vremenske modele, kao i modeli sa kontinualnim delovima trajektorija između kojih postoji diskontinuitet. Diskontinuitet može biti doskontinuitet izvoda promenljive i diskontinuitet promenljive. Prema tome, diskontinualni modeli su oni u kojima je najmanje jedna promenljiva stanja i/ili njen izvod diskontinualan. Diskretni modeli menjaju svoje stanje u diskretnim vremenskim trenucima. Između dva uzastopna vremenska trenutka stanje modela ostaje nepromenjeno. Diskretni modeli mogu biti diskretni vremenski i kontinualni vremenski.

**12. Formalna specifikacija modela.** Formalizmi se koriste za opisivanje objekata modela, definišući na taj način parametre i ograničenja. Klase objekata su povezane tako da se te veze mogu formalizovati kao preslikavanja iz jedne klase u drugu. Apstrakcija je preslikavanje koje podrazumeva kontrolisano uključivanje detalja prilikom opisivanja modela, te je ona preslikavanje jedne klase objekata u drugu, manje složenu klasu. Izvornu klasu nazivamo konkretnom, a drugu apstraktnom. Asocijacija je preslikavanja od višeg ka nižem nivou u hijerarhiji specifikacije sistema. Specifikacija je definisanje podklasa uvođenjem novog formalizma.

**13. Ocena parametara modela.** Ocena parametara modela je postupak experimentalnog određivanja vrednosti parametara koji se pojavljuju u matematičkom opisu modela. Polazi se od pretpostavke da je struktura modela explicitno data, tj. da svi parametri imaju ne-nulte vrednosti. Ako imamo sistem definisan matematičkim modelom u prostoru stanja: $s'=f(s,u,p,t)$, $y=g(s,u,p,t,)$, $s(t_0)=s_0$, gde su s-vektor stanja dimenzije n, u-vektor ulaza dimenzije m, y-vektor izlaza dimenzije k, p-vektor sastavljen od $n_p$ nepoznatih parametara, f i g odgovarajuće vektorske funkcije a $s_0$ početni uslovi. Potrebno je za dati model i skup ulazno-izlaznih parametara odrediti vektor nepoznatih parametara p. Za ocenu parametara pojedinih klasa modela koriste se algoritmi čija struktura zavisi od: 1. formalizma modela (kontinualno-vremenski, diskretno vremenski, linearni ili nelinearni, deterministički/stohastički), 2. konteksta modeliranja (tip promenljivih sistema, apriorno znanje, svrha modela), 3. filozofija procene (kriterijumi, numerička procedura, prilaz izračunavanju). Ocene parametara determinističkog modela – mogu se uvesti definicije: 1. za skalarni parametar $p_j$ kaže se da je identifikabilan na intervalu $[t_0, T]$ ako postoji konačni broj rešenja za $p_j$ koja proizilaze iz relacija datog modela. U suprotnom je neidentifikabilan. 2. opis modela je sistemski identifikabilan ako su svi parametri identifikabilni. I obratno. 3. skalarni parametar $p_j$ je jedinstveno identifikabilan na intervalu $[t_0, T]$ ako postoji jedinstveno rešenje za $p_j$ koje proizilazi iz relacija modela sa datim ulazom. 4. specifikacija modela je parametarski identifikabilna na intervalu $[t_0, T]$ ako su svi parametri jedinstveno identifikabilni. a identifikabilnost sistema zavisi od specifičnosti primenjenog ulaza i početnih uslova i strukture jednačina i ograničenja. Ocene parametara modela stohastičkih sistema – identifikabilnost zavisi od: strukture (koja parametre stavlja u relaciju jedne s drugima), ulaza (od kojih neki mogu biti suviše prosti) i procedure procene (koje treba da konvergiraju ka stvarnim vrednostima parametara).

**14. Statistički pristup proceni parametara modela.** Statističke ocene parametara statističkih modela se mogu realizovati u prostoj sekvencijalnoj ili rekurzivnoj formi. Sekvencijalne statističke metode se primenjuju kada se podaci prikupljaju i memorišu na nekom medijumu, dok se rekurzivna forma koristi kada memorisanje nije moguće i kada postoje zahtevi za prikupljenjem podataka u realnom vremenu. Prednost rekurzivne forme jeste u poznavanju tekuće vrednosti statističkih pokazatelja. Ocena srednje vrednosti slučajne promenljive – sekvencijalni metod: u slučaju kada su svi rezultati merenja na realnom sistemu dostupni u vreme računanja, procedura je: $y_N^{nad}=\sum_{i=1}^{N} y_i/N$. Rekurzivni metod: za njega je karakteristično da se baza podataka proširuje tokom računanja; međurezultati za određeni broj uzoraka su takođe dostupni i teže sekvencijalnom rešenju, kako ozračunavanje odmiče. Procedura je: $a_0\hat{}=0$, $a_k\hat{}=a_{k-1}\hat{}-(a_{k-1}\hat{}-y_k)/k$.

**15. Ocena nepoznatog parametra po metodi najmanjih kvadrata.** Ako imamo statički model kod koga je veza između parametara modela i rezultata experimenta data relacijom: $y_i=x_i a+e_i$, $i=1,N$; gde su $x_i$ nepoznate vrednosti a $e_i$ greške merenja. Sekvencijalno rešenje: cilj je minimizirati odstupanje između podataka koje geeriše realni sistem i onih koji se dobijaju na osnovu modela, tj. da se minimizira kvadrat greške: $J=\sum_{i=1}^{N}[y_i-x_i a]^2=\sum_{i=1}^{N} e_i^2$. Ovde imamo jedan nepoznati parametar: $\partial J/\partial a=0$, tj. $2\sum_{i=1}^{N}[y_i-x_i a](-x_i)=0 \Rightarrow [\sum_{i=1}^{N} x_i^2]a=\sum_{i=1}^{N} x_i y_i$ iz čega određujemo parametar a. Rekurzivno rešenje: uvedimo oznake $a_k\hat{}=p_k b_k$, gde $p_k=[\sum_{i=1}^{k} x_i^2]^{-1}$, $b_k=\sum_{i=1}^{k} x_i y_i$. u rekurzivnom obliku, može se zapisati kao: $p_k=p_{k-1}-p_k p_{k-1} x_k^2$,

$b_k=b_{k-1}+x_k y_k$. Ako $k_k$ definišemo u obliku $p_{k-1}x_k[1+p_{k-1}x_k^2]^{-1}$ tada se ocena parametra može izraziti kao $a_k\hat{}=a_{k-1}\hat{}-k_k\hat{}(a_{k-1}\hat{}-y_k)$.

**16. Procena k nepoznatih parametara.** Ako je veza između k parametara modela i rezultata nekog eksperimenta data relacijom: $y=a_1x_1+a_2x_2+...+a_kx_k$ gde su $a_k$ nepoznati parametri. Rezultate n experimenata pokazaćemo tabelom:

| Rb | $x_1$ | $x_2$ | ... | $x_k$ | y |
|---|---|---|---|---|---|
| 1 | $x_{11}$ | $x_{12}$ | ... | $x_{1k}$ | $y_1$ |
| 2 | $x_{21}$ | $x_{22}$ | ... | $x_{2k}$ | $y_2$ |
| ... | ... | ... | ... | ... | ... |
| j | $x_{i1}$ | $x_{i2}$ | ... | $x_{ik}$ | $y_i$ |
| ... | ... | ... | ... | ... | ... |
| n | $x_{n1}$ | $x_{n2}$ | ... | $x_{nk}$ | $y_n$ |

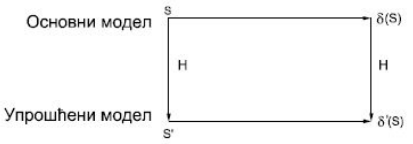
gde $y_j$ predstavlja rezultate merenja. I uvek će postojati određena odstupanja od gornje relacije, pa će se ta odstupanja obeležiti sa $e_j$. Tako dobijamo sistem jednačina: $y_1-(a_1x_{11}+a_2x_{12}+...+a_kx_{1k})=e_1$, $y_2-(a_1x_{21}+a_2x_{22}+...+a_kx_{2k})=e_2$,..., $y_j-(a_1x_{j1}+a_2x_{j2}+...+a_kx_{jk})=e_j$,..., $y_n-(a_1x_{n1}+a_2x_{n2}+...+a_kx_{nk})=e_n$. Ocene nepoznatih parametara $a_1...a_k$ određuju se minimizacijom kriterijumske funkcije: $J=\sum_{i=1}^{n}e_i^2=\sum_{i=1}^{k}[y_i-(a_1x_{i1}+a_2x_{i2}+...+a_kx_{ik})]^2$, te što je n [broj experimenata] veće, to je i tačnija određenost parametara $a_1...a_k$. Minimum funkcionala J dobija se izjednačavanjem njegovih parcijalnih izvoda po parametrima $a_k$ sa nulom, tj. $\sum_{i=1}^{k}[y_i-(a_1x_{i1}+a_2x_{i2}+...+a_kx_{ik})](x_{ij})=0$, j=1,k. Ukoliko se iskoristimo matričnim računom i stavimo da je: $y=[y_1\ y_2\ ...\ y_n]$, $a=[a_1\ a_2\ ...\ a_n]$, $X=[x_{11}\ x_{12}\ ...\ x_{1k};x_{21}\ x_{22}\ ...\ x_{2k};...;x_{n1}\ x_{n2}\ ...\ x_{nk}]$, onda $J=(y-Xa)^T(y-Xa)$. Primenom pravila diferenciranja matrica i vektora imamo $\partial J/\partial a=2X^T(y-Xa)=0\Rightarrow X^TXa=X^Ty\Rightarrow a=(X^TX)^{-1}X^Ty$.

**17. Validacija i verifikacija modela.** Postupak kojim se ispituje koliko verno i precizno jedan model predstavlja realni sistem, može se posmatrati kroz dva povezana koraka: verifikaciju, tj. proveru da li je simulacioni program bez grešaka i konzistentan sa modelom, i validaciju, tj. postupak određivanja da li je model precizna reprezentacija realnog sistema. Faze izgradnje modela: prva faza je posmatranje realnog sistema i prikupljanje podataka; potom formiranje konceptualnog modela, skupa pretpostavki za komponente i strukturu ka i hipoteza za vrednosti parametara; treća faza je pisanje računarskih programa.

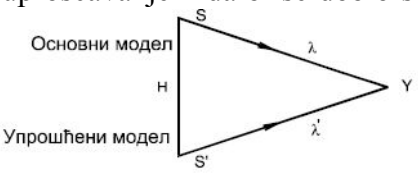
**18. Validacija simulacionih modela.** Problem validacije nastaje usled aproximacija realnog sistema. Aproximacije mogu biti: funkcionalne, aproximacije raspodele, nezavisnosti, agregacije (vremenske, među-sektorske, agregacije pomoćnih sredstava) i stacionarnosti. Cilj procesa validacije: da proizvede model koji predstavlja ponašanje realnog sistema i koji muje dovolno blizak i da pouzdanost modela poveća na prihvatljiv nivo. Konceptualni model se poredi sa realnim sistemom i ukoliko postoje neslaganja vrši se ispravka pa se tako dobijeni model ponovo poredi i po potrebi se vrše nove ispravke. Iterativni proces se nastavlja dok se ne postigne zadovoljavajuća tačnost modela. Poređenje modela sa realnim sistemom se vrši pomoću subjektivnih [učešće ljudi koji poseduju znanje o sistemu i koji donose zaključke] i objektivnih testova [zahtevaju podatke koje proizvodi model i podatke o ponašanju sistema; ti podaci se porede statističkim testovima]. Faze praktičnog pristupa problemu validacije: izgraditi model koji verno predstavlja realni sistem, potvrditi pretpostavke modela, uporediti ulazno-izlazne transformacije modela sa ulazno-izlaznim transformacijama sistema. Za proveru validnosti modela koristi se analiza osetljivosti. Testira se osetljivost modela na različite pretpostavke i promene ulaznih veličina. Promene izlaznog ponašanja modela, koje nastaju usped promene ulaznih veličina, mogu pružiti korisne informacije. Stoga je potrebno odabrati najkritičnije ulazne veličine pri testiranju. Pretpostavke modela se mogu svrstati u dve grupe – pretpostavke o strukturi [kako sistem funkcioniše] i podacima [zasnovane na pozudanim podacima i ispravnoj statističkoj analizi tih podataka]. Statistička analiza se sastoji iz: identifikacije odgovarajućih raspodela dobijenih podataka, procene parametara izabrane raspodele i validacije pretpostavljenog statističkog modela nekim testom. Jedini objektivni test modela kao celine je provera sposobnosti modela da predvidi buduće ponašanje realnog sistema, kada ulazni podaci modela odgovaraju ulaznim podacima sistema i kada je implementirana "politika" modela takođe

implementirana u nekom realnom sistemu. Za potpunu validaciju neophodno je da već postoji realan sistem koji se proučava.

**19. Formalni kriterijum za utvrđivanje validnosti modela.** Homomorfizam predstavlja formalni kriterijum za utvrđivanje validnosti modela za date experimentalne uslove, u odnosu na osnovni model. Cilj je pronaći preslikavanje H pomoću koga je moguće iz svakog stanja osnovnog modela s preći u odgovarajuće stanje uprošćenog modela s'. Pritom moraju biti ispunjeni uslovi: očuvanje funkcije nastupanja vremena [stanja osnovnog i uprošćenog modela se moraju menjati u istim trenucima vremena], očuvanje funkcije prelaza stanja




[s i s' moraju uvek biti u identičnim stanjima; ], očuvanje izlazne funkcije [neka je s i neka je moguće primenom preslikavanja H preći u s'; primena izlazne funkcije osnovnog modela, za one experimentalne uslove za koje je izvršeno uprošćavanje λ da bi se dobio s' za stanje s, daje isti izlaz Y

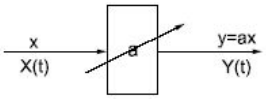



kao i primena izlazne funkcije s', λ', na stanje s'; ]. s' je validan samo ukoliko zadovoljava sve uslove.

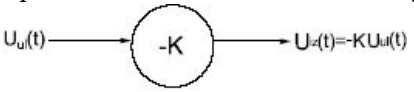
**20. Verifikacija simulacionih modela.** Verifikacija treba da pokaže koliko se slažu konceptualni i računarski kod. Potrebno je izvršiti provere: ručna verifikacija logičke ispravnosti [poređenje ručno i računarski dobijenih rezultata], modularno testiranje [testiranje svakog modula], provera u odnosu na poznata rešenja, testiranje osetljivosti, testiranje na poremećaje.

**21. Analogni računar kao sredstvo za simulaciju.** Ideja analognih računara sastoji se u iznalaženju takvih fizičkih objekata koji će moći da se koriste za analizu matematičkih modela pomoću kojih su ti objekti matematički opisani, pa je sam princip rada analognih računara zasnovan na fizičkim zakonima fizičkog objekta koji se koristi. Analogni računari se dele na dve grupe: specijalni [za analizu jednog ili manjeg broja specifičnih matematičkih modela] i univerzalni [za razne matematičke modele]. Univerzalni se dele na repetitivne [velika brzina rada, rešenje se ponavlja od 20-100 puta u sec; moguće vizuelno praćenje rešenja] i spore [rade u realnom vremenu, rešenje se dobija samo u jednom ciklusu rada]. Savremeni analogni računari mogu da rade i kombinovano. Matematički modeli koji se najčešće rešavaju su sistemi diferencijalnih jednačina, pa se često nazivaju i diferencijalnim analizatorima. Računarski elementi savremenih analognih računara su: množenje promenljive veličine konstantom, algebarsko sabiranje više promenljivih veličina, integracija jedne ili više funkcija, množenje i deljenje promenljivih veličina, generisanje funkcija, logičke operacije.

**22. Osnovni elementi elektronskog analognog računara.** Naponski izvor – konstanta. Potenciometar - ulaz množi sa konstantom<1 tj. y(t)=ax(t), gde je a konstanta a x(t) promenljiva. Realizuje se kao omski otpornik sa klizačem na čije se krajeve dovodi promenljivi napon x(t) a sa klizača koji može da se postavi u ma koji položaj




između krajeva potenciometra, odvodi se napon y(t). Ali mora biti y(t)≤x(t). Potenciometri su snabdeveni skalama od 0 do 1. Problem je što se potenciometar nikada ne može postaviti idealno tačno pa postavljanje klizača na neku dužinu l unosi i grešku Δl. Greška je manja što je dužina otpornika veća. Potenciometar se kod analognih računara koristi za dobijanje konstantnih napona zarad postavljanja početnih uslova i za množenje promenljivog napona sa pozitivnom konstantom<1. Pojačavač - ulaz množi sa konstantom>1. Realizuje sa u formi integrisanih kola.




Pojačava

ulazni signal: $U_{iz}=-kU_{ul}$, k>>1. On ima veliku ulaznu imalu izlaznu otpornost. U zavisnosti sa kojim pasivnim računarskim elementom je spregnut, nosi naziv sabirač [$y(t)=-\sum_{i=1}^{n}a_i x_i(t)$; množi ulazni napon sa konstantom ili algebarski sabira više ulaznih napona], integrator [$y(t)=-a\int_0^1 x(t)dt$;obično je u obliku koncenzatora i za ovu operaciju se koristi zakon fizike o odnosu između struje, napona i kapaciteta kondenzatora; sve operacije se izvršavaju pod određenim ograničenjima koje pojačavač ima, a te greške mogu biti usled konačnog pojačanja, šetanja nule, konačne ulazne struje, usled kašnjenja na višim frekvencijama, ili slučajne greške], množač [$z(t)=x(t)y(t)$;dele se na one koji koriste elektromehaničke metode tzv. servomnožači (elektromehanički uređaj koji može da pomera klizač potenciometra u zavisnosti od neke druge promenljive veličine) i oni koji koriste elektronske metode tzv. elektronski množači (npr. diodni, koji generiše fiksne funkcije). Integrator - izlazni napon je integral ulaznog. Razni diodni ograničavači - nelinearni elementi. Diodni generatori linearnih funkcija – diodni elementi koji rade u sprezi sa računskim jednosmernim pojačivačima. U zavisnosti od odnosa električnog potencijala na anodi i katodi, dioda se može ponašati i kao provodnik [anoda>katoda] i kao izolator [katoda>anoda]. Postupak: formulisanje matematičkog modela, sastavljanje bloka dijagrama na osnovu matematičkog modela, povezivanje računarskih elemenata pomoću električnih vodova.

**23. Digitalni računar kao sredstvo za simulaciju.** Isti može biti veoma tačan pri rešavanju simulacionih problema ali vreme određivanja rešenja može biti veoma dugo. Hardverski noviteti značajni za razvoj simulacije i njenog delokruga primene su: vektorski procesori, računari sa više procesora, jeftine i brze dinamičke i masovne memorije, brzi grafički procesori. Takođe sa softverske strane, razvoju simulacija su najviše doprineli: AI, algoritmi paralelnog procesiranja, računarske mreže, MM tehnologije. Budući pravci kretanja: specifikacija modela u obliku govornog jezika i prevođenje u formalni model uz pomoć expertnog sistema, implementacija simulacionog model ana višeprocesorskim računarima povezanih u mreže, analiza rezultata korišćenjem baza znanja.

**24. Hibridni računar kao sredstvo za simulaciju.** Spoj digitalnog i analognog računara. Analogni izvršava implicitne operacije, a digitalni algebarske funkcije. Predstavlja kombinaciju hibridne i sekvencijalne obrade podataka što radi hibridni procesor. Hibridni procesor je hardversko-softverski sistem. Sastoji se od paralelnog i jednog ili više sekvencijalnih procesora. Time se kombinuje tačnost i memorisanje podataka digitalnih računara sa velikom brzinom izvođenja operacija analognih računara. Delovi hibridnog računarskog sistema: samostalni digitalni računar sa svom standardnom perifernom opremom, jedan analogni računar odgovarajuće veličine i interfejs koji ih povezuje. Delovi hibridnog procesora: mikroprocesor upravlja rešavanjem hibridnog zadatka, zadaje režime analognim računskim i jedinicama paralelne logike, aktivira AD i DA konverziju i upravlja radom perifernih jedinica; analogni deo utiče na efikasnost hibridnih računarskih sistema; kontroler služi za povezivanje hibridnog procesora sa perifernim jedinicama ali i sa digitalnim računarom; sistem za spregu omogućava povezivanje analognog i digitalnog dela hibridnog računskog sistema u cilju razmenjivanja numeričkih podataka i kontrolnih funkcija.

**25. Simulacija kontinualnih sistema, formalni model.** SKS se odnosi na experimentisanje sa modelima sistema čija se stanja menjaju kontinualno. To su uglavnom dinamički sistemi i mogu biti deterministički i stohastički. Predstavljaju se diferencijalnim j-ma, koje se rešavaju numeričkim putem, kako bi se odredilo ponašanje sistema. Formalni model se može predstaviti kao: $M=(U,Y,S,\delta,\lambda,S_0)$, U-skup ulaza, Y-skup izlaza, S-skup promenljivih stanja, δ-funkcija prenosa $\delta:UxS\rightarrow S$,λ-funkcija izlaza $\lambda:UxS\rightarrow Y$, $S_0$-skup početnih stanja. Proces simulacije kontinualnih sistema može se pokazati j-ma kontinualnog automata: $S(t):=IxA_1\{U(t),S(t)\}$, $Y(t):=A_2\{U(t),S(t)\}$, oba A su algebarske funkcije, a I je integrator. $A_1\{U(t),S(t)\}$ i $A_2\{U(t),S(t)\}$ se računaju u prvoj fazi a zatim se integracijom ili funkcijom vremenskog kašnjenja dobija novi skup vrednosti promenljivih stanja. Razlika između analogne i digitalne simulacije kontinualnih sistema je u proceduri – analogni je sposoban za rešavanje rekurzivnih relacija direktno kao rezultat dinamičkih procesa čije je stabilno stanje zahtevana funkcija, dok digitalni primenjuje iterativnu funkciju. Još jedna razlika je i u tome što se sve operacije predstavljene operatorima izvršavaju istovremeno u analognim računarima. Izlazne promenljive su podskup skupa promenljivih stanja. Ako razložimo operator $A_1$ na primitivne funkcije, dobijamo detaljnu strukturu procesa simulacije kontinualnog sistema. Blok se može predstaviti trojkom $b=(\phi,X,Y)$, gde je φ funkcija preslikavanja nepraznog skupa X zvanog domen u neprazan skup Y zvanog opseg. Neka je V skup

promenljivih, t vreme simulacije, i B skup blokova; tada je KSS=(B,V,t), tj. apstraktni kontinualni simulacioni sistem. Dalje, ako je X skup svih ulaza jednog bloka a Y skup svih izlaza tog bloka, tad ase skup svih promenljivih V može podeliti u tri disjuktna podskupa i to skup ulaza U=X-Y, skup izlaza U=Y-X i skup veza C=X∩Y.

**26. Simulacija kontinualnih sistema pomoću analognog računara.** Ideja analognih računara sastoji se u iznalaženju takvih fizičkih objekata koji će moći da se koriste za analizu matematičkih modela pomoću kojih su ti objekti matematički opisani, pa je sam princip rada analognih računara zasnovan na fizičkim zakonima fizičkog objekta koji se koristi. Prednosti korišćenja analognih računara za simulaciju kontinualnih sistema: mogućnost pristupa bilo kom funkcionalnom delu analognog programa i mogućnost obavljanja velikog broja ponavljajućih operacija kombinovanih sa veoma velikom brzinom izračunavanja. Nedostaci: ne postoji ekvivalent reprezentaciji podataka u pokretnom zarezu i ne mogu se izbeći različita tehnička ograničenja.

**27. Simulacija kontinualnih sistema pomoću digitalnog računara.** Digitalni računar obavlja osnovne računarske operacije i određene logičke operacije. Sve ostale obavlja koristeći numeričke metode. Prednost mu je i veliki nivo apstrakcije kod pisanja programa i tačnost rešenja. Da bi se izvršila digitalna simulacija sve veličine moraju biti diskretizovane, tj. sve funkcije nezavisne promenljive t treba da budu predstavljene sekvencom diskretnih brojeva u tačkama $t_n=t_0+nh$, n=0,N, gde veličina koraka $h=t_n-t_{n-1}$ ne mora biti konstantna. Na početku simulacije potrebno je označiti numeričke vrednosti svakog ulaza i svake promenljive. To su zapravo početni uslovi. Kompletno izračunavanje sekvenci vrednosti za sve promenljive stanja i ulaza, a za određene početne uslove, nazivamo izvršavanje simulacije. Ono se sastoji iz faze inicijalizacije i faze računanja. U inicijalizaciji, postavljaju se početne vrednosti i uslovi, parametri ali i izračunavanja kao priprema za numeričku integraciju. U simulacionom algoritmu n-ti korak se sastoji od izračunavanja vrednosti funkcije svih algebarskih blokova, na osnovu skupa ulaza i promenljivih stanja datih u $t_n$ i izračunavanja svih proenljivih stanja za naredni korak $t_{n+1}$ u skladu sa osnovnim j-ma: $S(t_{n+1})=I^*xA_1\{U(t_n),S(t_n)\}$, $Y(t_n)=A_2\{U(t_n),S(t_n)\}$, n=0,N. Procedura simulacije se ponavlja dok se ne ispuni $t_n=t_N$.

**28. Integracija u simulaciji kontinualnih sistema.** Integracione metode se mogu svrstati u jedno- i višekoračne ili u metode sa konstantnom ili promenljivom veličinom koraka. Problem numeričke integracije sastoji se u definisanju nekog operatera F tako da $s_{n+1}=F(s_n,...,s_{n-k+1};s_{n+1}',...,s_{n-k+1}')$ aproksimira na datom intervalu $(t_n,t_n+h)$ integral $s_{n+1}=\int_{tn}^{tn+h}\varphi(S,U,t)dt=\int_{tn}^{tn+h}s'dt$, gde se koristi skraćena notacija $t_n=t_0+nh$, $s_n=s_n(t_n)$, $u_n=u_n(t_n)$, s'=ds/dt a F je k-koračna integraciona formula. Ova jednačina se može nazvati diskretnim blokom za integraciju. U jednokoračne integracione metode spadaju Ojlerova, metod trapeza i Runge-Kuta drugog reda. Višekoračne numeričke metoda integracije zasnivaju se na zameni funkcije nekim interpolacionim polinomom u izabranim čvorovima interpolacije, koji se potom integrali. Opšta formula u tom slučaju ima oblik: $s_{n+1}=a_0s_n+a_1s_{n-1}+...+a_rs_{n-r}+h(b_{-1}s_{n+1}'+b_0s_n'+b_1s_{n-1}'+...+b_rs_{n-r}')$.

**29. Određivanje intervala integracije.** Funkcija integracije se mora prikazati u skladu sa pravilima modela sa diskretnim vremenom. Poznato je da važi: $dY(t)/dt=\lim_{\Delta T\to 0}(Y(t+\Delta T)-Y(t))/\Delta T=X(t)$. Ako je ΔT dovoljno malo tada približno važi $Y(t+\Delta T)\cong Y(t)+\Delta TX(t)$. Što je manji interval ΔT to je tačnost aproksimacije veća ali je vreme računara duže. Cilj je da se dabere takav interval integracije da se kako smanji greška računanja tako i smanji vreme izračunavanja integrala. Jedan od načina određivanja intervala integracije este preko diskretizovane Laplasove transformacije tzv. 'Z transformacija'. Međutim ona je nepogodna jer zavisi i od numeričke metode ali i od modela koji simuliramo. Moramo ga stoga utvrditi experimentom. Preporuka je da pri prvom experimentu interval integracije bude bar 10 puta manji od najbrže vremenske konstante u sistemu. Pri velikim intervalima integracie gubi se uvid u ponašanje sistema za vreme tog intervala. S fruge strane, suviše mali interval integracije dovodi do nagomilavanja greške usled velikog broja računskih operacija i zaokruživanja. Jedan od osnovnih problema koji se javljaju prilikom prevođenja kontinualnog modela u model sa diskretnim vremenom jeste taj da ma koliko mali da se uzme interval integracije nikada se ne zna da li se između dva računanja nije dogodila neka nagla promena koja bi mogla da izazove veliku grešku u izračunavanju naredne vrednosti. Drugi problem jeste nepoznavanje ponašanja modela između dva uzastopna trenutka vremena u kojima se vrši računanje.

**30. Izvođenje implicitnih operacija.** Ako matematički model simulacije kontinualnog sistema nije uredljiv onda on sadrži jednu ili više funkcija u kojoj je vrednost funkcije arguement iste funkcije, tj. y=f(y). Algebarskom petljom nazivamo slučaj kada su jedan ili više algebarskih blokova povezani u zatvorenu petlju koja ne sadrži memorijske elemente tipa integratora ili jediničnog kašnjenja. Algebarska petlja se može ukloniti algebarskom manipulacijom na originalnom sistemu jednačina. Time se dobija kanonički oblik sistema. U CSMPu ovo se rešava iterativnim rešavanjem jednačinne y=f(y). zato ubacujemo dodatni blok za implicitne elemente u petlju i dobijamo slučaj koji se može opisati sa: y=f(x), x=IMP(y). Početnu vrednost $x_0(t_0)$ određuje korisnik. Početne vrednosti za iterativni proces pri svakom koraku iteracije određuje se po formuli $x_0(t_n)=x(t_{n-1})$. Pri svakoj iteraciji $y_i(t_n)=f(x_i(t_n))$ se koristi za izračunavanje korigovane vrednosti za svako x po formuli $x_{i+1}(t_n)=IMP(y_j(t_n))$. Iteracija se ponavlja sve dok greška $\varepsilon=x_i(t_n)-y_i(t_n)$ ne bude manja od unapred zadate vrednosti. Problem sa ovom metodom je znatno produžavanje vremena izračunavanja.

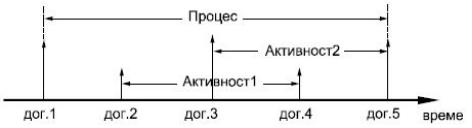
**31. Računarska realizacija simulatora.** Svaki sistem za digitalnu simulaciju kontinualnih procesa sastoji se od simulacionog jezika, procesora i skupa funkcija. Simulacioni jezik služi za opis matematičkih modela, parametara, kontrolnih komandi, u obliku razumljivim za korisnika i računar. Procesor je program koji: učitava, prevodi i memoriše matematički model, sortira matematički model, učitava, prevodi i memoriše parametre i upravljačke komande, izvršava simulaciju matematičkog modela. Pri specifikaciji simulacionih jezika treba poštovati: format ulaznih podataka treba da bude što flexibilniji, matematička notacija za funkcije treba da bude blok-orijentisana, izbegavati nepotrebna ograničenja, za identifikaciju blokova i veza između njih koristiti simbolička imena, jezik bi trebao biti lak za učenje. Sam procesor se sastoji iz više modula. Prvi prevodi opis modela u interni jezik i naziva se prevodilac. On generiše mašinski kod ili internu tabelu. Drugi modul izvršava simulaciju. On vrši inicijalizaciju, izračunava vrednosti svih algebarskih funkcija, vrši integraciju svih promenljivih, vodi računa o broju ponavljanja i završava simulaciju us kladu sa datim uslovima. Naziva se simulator. Kontrolu toka simulacije vrši modul monitor. Pored njega postoji još i interpreter.

**32. Procesor simulatora.** Sam procesor se sastoji iz više modula. Prvi prevodi opis modela u interni jezik i naziva se prevodilac. On geneiše mašinski kod ili internu tabelu. Drugi modul izvršava simulaciju. On vrši inicijalizaciju, izračunava vrednosti svih algebarskih funkcija, vrši integraciju svih promenljivih, vodi računa o broju ponavljanja i završava simulaciju us kladu sa datim uslovima. Naziva se simulator. Kontrolu toka simulacije vrši modul monitor. Pored njega postoji još i interpreter.

**33. Simulacija diskretnih događaja, formalni model.** SDD je metod simulacionog modeliranja sistema kod kojih se diskretne promene stanja u sistemu ili njegovom okruženju događaju diskontinualno u vremenu. Koristi se za analizu dinamičkih sistema sa stohastičkim karakteristikama. Događaj, koji je diskretna promena stanja entiteta sistema, nastupa u određenom trenutku vremena. Takvi su sistemi masovnog opsluživanja. Veličine u ovakvim sistemima su slučajne prirode, te se z anjihov opis mogu koristiti različite raspodele verovatnoće. Formalni opis: $M_d=\langle U,S,Y,\delta,\lambda,\tau\rangle$ , gde je U-skup externih događaja, S-skup sekvencijalnih stanja diskretnih događaja, Y-skup izlaza, δ-kvazi prenosna funkcija ($\delta^{\varphi}$, preslikava S→S i pokazuje u koje će stanje preći sistem s ukoliko ne nastupi ni jedan externi događaj; $\delta^{ex}$, preslikava UxSxT→S i pokazuje stanje u koje će preći sistem s kada nastupi događaj u u trenutku t), λ-izlazna funkcija, S→Y, τ-funkcija nastupanja vremena, preslikava $S\rightarrow R^{+}_{0,\infty}$, tj. pokazuje koliko će dugo ostati u stanju s, pre nego što nastupi naredna promena stanja, pod pretpostavkom da neće nastupiti ni jedan externi događaj.

**34. Događaj, aktivnost i proces.** Ova tri opisuju dinamiku. Događaj predstavlja diskretnu promenu stanja netiteta u sistemu ili njegovom okruženju. Između dva uzastopna događaja stanje u sistemu se ne menja. Nastaje usled: ulaka/izlaska privremenog entiteta u/iz sistema ili usled promene atributa pojedinih objekata sistema. Uslovni događaji mogu da nastupe tek kada je ispunjen uslov njihovog nastupanja [obično zauzimanjem nekog resursa]. Bezuslovni događaji su oni čiji je jedini uslov da tekuće vreme simulacije bude jednako vremenu njegovog nastupanja, što je uglavnom planirano vreme završetka neke aktivnosti. Aktivnost je skup događaja koji menjaju stanje jednog ili više entiteta. Trajanje se može unapred definisati ili vezati za ispunjenje određenih uslova. Proces je niz uzastopnih logički povezanih događaja kroz koje prolazi neki

privremeni objekat, tj. to je uređena sekvenca događaja koji opisuju neku pojavu od nastanka do terminiranja. Tako se SDD može opisati kao: entiteti koji se opisuju atributima i uzajamno deluju učestvujući u aktivnostima pod određenim uslovima stvaraju događaje koji menjaju stanja sistema.




**35. Razvoj simulacije diskretnih događaja.** Ključni elementi: mehanizam pomaka vremena, pristupi generisanju događaja, osnovne strategije simulacije. Mehanizam pomaka vremena – osnovni mehanizmi: pomak vremena za konstantni prirast i pomak vremena na naredni događaj. Pomak vremena za konstantni priraštaj – vreme u simulacionom modelu se menja tako da se uvek dodaje konstantni priraštaj. Nakon svakog pomaka vremena ispituje se da li je u prethodnom intervalu vremena trebalo da dođe do nastupanja nekh događaja. Ukoliko jeste, tada se ti događaji planiraju za kraj intervala. Mana mu je ta što pomeranjem događaja na kraj vremenskog intervala u kojem bi oni trebalo da nastupe, uvodi se greška u izvođenje simulacije. Potom događaji koji u stvarnosti nisu istovremeni se ovde prikazuju kao takvi. Takođe postoji problem sa redosledom izvođenja. Smanjenjem vremenskog priraštaja te se greške smanjuju ali se zato povećava vreme koje se troši na izvođenje simulacije. Pomak vremena na naredni događaj – ovde se simulacioni sat pomera na vreme u kojem će nastupiti prvi naredni događaj. U tom trenutku se događaj izvede i napravi se odgovarajuća promena stanja sistema. Simulacija se završava kada nema više događaja ili kada je zadovoljen neki unapred definisan uslov. Ovime se izbegava greška u vremenu izvođenja događaja i preskaču se intervali bez događaja. Generisanje događaja – događaj se opisuje sa više atributa koji formiraju slog događaja, koji se memoriše u listama događaja. Događaji se generišu ili definisanjem unapred ili pristupu zasnovanom na narednom događaju. Događaje možemo svrstati prema mestu nastanka u eksterne [ne zavise od modela] i interne [zavise i generišu se u modelu]. Za osnovne strategije simulacije, pogledati 36., 37. i 38. pitanje.

**36. Strategija raspoređivanja događaj u SDD.** Podrazumeva da se događaji planiraju unapred i drže u listi budućih događaja sortirani po vrmenu nastupanja i prioritetu. Procedura: generiše se slog događaja. Dodele se vrednosti atributima, koji mogu biti identifikatori tipa dogajaja, vreme nastupanja, prioritet... zati se događaj stavlja u listu budućih događaja. Potom se sa liste uzima prvi događaj. Tada se ažurira simulacioni sat na vreme njegovog nastupanja. Na osnovu tipa događaja poziva se određena procedura koja izvršava sva ažuriranja u modelu i simulatoru. Kada se izvrše svi događaji koji imaju isto vreme nastupanja, siulacioni sat se žurira na vreme sledećeg događaja.

**37. Procedura skaniranja aktivnosti u SDD.** Događaji se implicitno raspoređuju tako da se promena stanja izvršava preko funkcija koje se nazivaju aktivnosti. Svaka aktivnost ima uslov i akciju. Za svaki vremenski korak traži se ona aktivnost koja prva ima zadovoljen uslov i tada se izvršava programski segment koji specificira aktivciju za datu aktivnost. Skaniranje se nastavlja sve dotle dok sve aktivnosti ne budu blokirane. Tek onda se simulacioni sat ažurira za sledeći vremenski korak. Za izvršenje aktivnosti u realnom sistemu potrebno je određeno vreme dok se pri simulaciji nije potrebno pratiti kretanje aktivnosti od početka do završetka. Samo se prelazi sa jedne aktivnosti na drugu definišu explicitno i oni jedini troše izvesno računarsko vreme. Ovaj postupak ima prednosti nad raspoređivanjem događaja kada je broj aktivnosti u modelu mali a broj događaj u okviru aktivnosti veliki.

**38. Strategija interakcije procesa u SDD.** Kombinacija je prethodna dva metoda. Proces je skup međusobno isključivih aktivnosti, gde terminiranje jedne aktivnosti dovoljava inicijalizaciju neke druge aktivnosti iz skupa aktivnosti procesa. Jedini problem koji se javlja jeste sinhronizacija procesa što se rešava uvođenjem naredbi ‘wait’ [wait until <uslov>; ukoliko je uslov zadovoljen, proces koji čeka na taj uslov nastaviće svoj tok] i ‘delay’ [advance <broj vremenskih jedinica>; nastavak procesa se odlaže za određeni broj vremenskih jedinica]. U okviru procesora održavaju se dve liste događaja: lista tekućih [uslovni događaji, sortirani po prioritetu] i lista budućih događaja [koji tek treba da nastupe, sortirani po nastupanju]. Planiranje događaj proceor vrši po strategiji narednog događaja. Pri svakom premeštanju događaja iz LBD u LTD, planira se njegov naslednik koji se stavlja u LBD. Pre početka simulacije za svaki proces generiše se prvi događaj i stavlja

se u LBD. Zatim se vreme simulacije ažurira na vreme događaj sa vrha LBD. Rad procesora odvija se u dve faze: faza ažuriranja vremena simulacije i faza skeniranja LTD.

**39. Veštačka inteligencija i simulacija.** Veštačka inteligencija proistekla je iz istraživanja iz oblasti kognitivne psihologije i matematičke logike i usled potrebe da se razume ponašanje dinamičkih, složenih sistema iz okruženja. Definicija: to je naučna disciplina posvećena opremanju računara intelektualnim sposobnostima – opažanje, učenje, rezonovanje. Razlike između VI i simulacije – modeli VI kriste heuristički pristup a predmet interesovanja je inteligentno ponašanje. Ujedno, simulacija zahteva aktivnosti koje i dalje moraju da rade ljudi: formulisanje problema i analiza rezultata simulacije. Na tim mestima na red dolazi VI.

**40. Istorijski razvoj VI.** Koreni datiraju još iz 1843. kada je kćerka lorda Bajrona, Ada, postavila pitanje može li analitička mašina Čarlsa Babidža misliti. Po njoj je kasnije nazvan i programski jezik Ministarstva odbrane SAD, ADA. Počeli su se potom rešavati problemi tipa šaha i slagalica od strane Bara, Fajgenbauma i Koena. Norbert Vajner i Dzon Njuman povezali su principe kibernetike sa realizacijom složeih odluka i kontrolnim funkcijama na mašini. Prekretnica nastaje 1956. kada je održana Darthmouth konferencija gde VI postaje posaebna disciplina. Osnovu današnjih istraživanja postavili su Novel i Sajmon. Usled velike složenosti simuliranja moždanih porcesa, odlučilo se da se pokuša sa nižim nivoom složenosti. Prvi projekti pojavljuju se 1972. [HPS] i 1963. [EPAM] koji su automatski prevodili prirodne jezike i automatski dokazivali teoreme. Sedamdesetih godina napušta se ideja simuliranja misaonih procesa, a okreće se formalizaciji prilaza problemima. Tada se pojavljuje plejada programa – Vinogradov ŠRDLU za razumevanje prirodnih jezika; Walcov program za razumevanje crteža, itd. Krajem sedamdesetih je preovladalo mišljenje da za praktične rezultate neohodno VI programe snabdeti explicitnim i extenzivnim znanjem iz date oblasti.

**41. Osnovni koncepti VI.** Simboličko umesto numeričko izračunavanje, nelogaritamski pristup rešavanju problema, zaključivanje zasnovano na znanju, primenljivost kod loše struktuiranih problema i podataka. Razlike između VI i konvebcionalnog programiranja: simbolička/numerička obrada, heuristika/algoritmi, upravljačke strukture odvojene od znanja/informacije i upravljanje integrisani, jednostavno modifikovanje, ažuriranje i proširivanje/teško, tolerišu se pogrešni odgovori/neophodni su tačni.

**42. Osnovni elementi VI.** Heurističko pretraživanje – omogućava da se polazni problem ograniči na razumnu veličinu, da ograniče prostor stanja rešenja. Predstavljanje znanja – znanje se organizuje u formu da VI program može direktno da ga iskoristi u procesu odlučivanja, planiranja, prepoznavanju objekata i situacija, izvođenju zaključaka, itd. Šeme za predstavljanje znanja mogu biti deklarativne [predstavljanje činjenicai tvrdnji] i proceduralne [akcije koje treba preduzeti]. Logičko zaključivanje – sprovodi se dokazivanjem teorema. Najpopularniji metod je procedura rezolucije; on određuje da li teorema proizilazi iz postavljenog skupa premisa. Prvo se zaključci stavljaju u formu rečenice kako bi se potvrdili a potom se negira zaključak koji se proverava. Ponovo se izvode zaključci i ako dođe do kontradikcije, teorema je dokazana. Jezici i alati VI – na njihov razvoj najviše je uticala neophodnost VI u pogledu dinamičke alokacije memorije i nemogućnošću da se predvide strukture i forme koje bi dobijali podaci tokom izvršenja programa. Ujedno, alati i jezici su morali da pruže podršku rekurzivnoj obradi, radi lakšeg pisanja programa.

**43. Osnovna područja VI.** To su expertni sistemi, prirodni jezici i robotika. Expertni sistemi koriste proces zaključivanja nalik ljudskom kod rešavanja problem. Ljudsko, expertsko znanje kodirano je u programu u vidu posebne strukture, baze znanja. Poseban mehanizam koristi to znanje. Sistemi prirodnih jezika prepoznaju prirodni jezik korisnika. Sistemi za percepciju vida, govora i dodira mogu da interpretiraju vizuelne scene ili da donose zaključke o kvalitetu ili fizičkoj orijentaciji.

**44. Expertni sistemi, definisanje.** To su inteligentni programi u koje je ugrađena velika količina visokokvalitetnog znanja iz nekog domena ljudske aktivnosti a koji mogu da procesiraju to znanje u cilju uspešnog rešavanja određenog problema. Inteligencija expertnog sistema samo zavisi od količine znanja koja se u njemu nalazi. Rad današnjih expertnih sistema zasniva se na simboličkom predstavljanju i procesiranju urođenog znanja. Znanje se predstavlja preko formalnih simbola i pogodnih struktura podataka iskazanih u

nekom programskom jeziku, a problemi se rešavaju izvođenjem induktivnih i deduktivnih zaključaka putem manipulacije simbolima i strukturama. Znanje ESa je uvek dostupno i nije podložno degradaciji kao ljudsko. Uže posmatrano, definicije ESa se odnose na njihove tehnike, a šire te tehnike predstavljaju samo prvi korak procesa koji će transformisati način računarske obrade podataka. Same definicije ESa se mogu svrstati u dve grupe: prve su one koje objašnjavaju kako su ES implementirani a druge potenciraju aspekt ljudskog znanja. Unatoč velikom broju definicija ESa, može se reći da su ESi računarski programi razvijeni sa ciljem da posluže kao konsultanti u rešavanju složenijih problema, koji zahtevaju ljudsku expertizu.

**45. Experti i expertni sistemi.** Ekspert je stručnjak u nekoj oblasti, poseduje određeno znanje iz t eoblasti, poseduje razumevanje problema i zadataka iz te oblasti, kao i veštine i iskustva. Efikasno koristi svoje znanje, mogu brzo da dođu do rešenja. Poseduju i sposobnost da u konkretnom probelmu koji rešavaju prepoznaju tipski zadatak/problem zi te oblasti; poseduju, ujedno i neke ličen osobine, snalažljivost i osećaj, što čini njeogov privatno znanje. Ono se još naziva i heurističkim i na osnovu njega expert može da prepozna na koji će način najbrže da dođe do rešenja, da oseti kada je pristup rešavanju ispravan, i može da se snađe i kada su podaci kojima raspolaže nekompletni. Razlozi zašto se naglasak stavlja na heurističko znanje, leži u činjenicama da: za mnoge probleme ne postoje jedinstvena i prihvatljiva logaritamska rešenja, mnogi matematički i drugi logaritmi ne mogu da predstavljaju znanje potrebno za izvođenje zaključaka i na kraju, znanje stručnjaka je nešto što ima cenu i vrednost. Postupak prikupljanja znanje je sledeći: inženjer znanja nastoji da od experta dobije heurističko znanje, da ga kodira i unese u ES. Korisnik sa ESom komunicira preko terminala. Danas se ESi iz pojedinih oblasti povezuju čineći na taj način bazu znanja šire namene.

**46. Expertni sistemi i konvencionalni programi.** Konvencionalni programi se uglavnom upotrebljavaju za obradu velikih količina podataka numeričkog tipa. Obrada istih se vrši prema unapred definisanim algoritmima. ESi se ponašaju drugačije – manipulišu simboličkim podacima i ne rade po unapred zadatim algoritmima. Problemi koje ES rešava su slabo struktuirani i ne podležu matematičkom modeliranju i formalizmu. Zato se pristupa upotrebi heuristika. Heuristika je skup empirijskih i svrsišodnih poteza koji u svojoj ukupnosti, oportunistički primenjeni, vode rešenju. Heuristika zahteva: razbijanje problema na manje, definisanje ocena karakteristika vezanih za datu oblast primene, rekurzivne metode, kao i stanje iz prethodnog koraka koje se koristi za prelazak na naredni. Razlike između konvencionalnih programa i ESa: algoritmi/heuristike, predstavljanje i korišćenje podataka/predstavljanje i korišćenje znanja, ciklički procesi/procesi zaključivanja, znanje i metode znanja su pomešani/odvojeni model, znanje organizovano u obliku podataka i programa/u obliku podataka, baze znanja i upravljačke strukture, novo znanje zahteva reprogramiranje/novo znanje se dodaje bez reprogramiranja.

**47. Struktura expertnih sistema.** Osnovni elementi su baza znanja, mehanizam zaključivanja, radna memorija


i interfejs prema korisniku. Tu se još mogu naći i pomoćni moduli: podsistemi za prikupljanje znanja, posebni interfejsi, sistem za objašnjenja. Baza znanja je specijalizovana i jedinstvena za konkretni ES i sadrži znanje experata iz određene ooblasti koje je uneto putem sistema za prikupljanje znanja i ne menja se tokom vremena. Radna memorija sadrži trenutne podatke o problemu kji se rešaa. Oni su prmenljivi i odražavaju trenutno stanje u procesu rešavanja. Mehanizam zaključivanja na osnovu tih promenljivh podataka i fixnog znanja iz baze znanja rešava problem. Preko interfejsa prema korisniku odvija s ekomunikacija.

**48. Predstavljanje znanja.** Dva osnovna pristupa: deklarativni i proceduralni. Deklarativni podrazumeva predstavljanje znanja u obliku nezavisnih modularnih celina, iskaza, činjenica i struktura podataka. Ti elementi znanja su pasivni te ne predstavljaju programske celine koje u sebi sadrže explicitan niz programskih komandi i explicitan redosled izvršavanja tih komandi. Jedini program koji koristi te lemente jeste mehaničko zaključivanje. Prednost ovog pristupa leži u modularnosti znanja. Time su nadogradnja i izmena olakšane. Ali sve to utiče negativno na brzinu izvšavanja. Proceduralni pristup se odnosi na direktno unošenje znanja u

programske procedure u vidu explicitnog programskog koda. Mehanizam zaključivanja poziva te procedure i koristi ugrađeno znanje. Brzina izvršavanja je veća, ali su mogucnosti izmena i unošenja novog znanja male. Tehnike predstavljanja znanja: produkciona pravila, semnatičke mreže, okviri. Sve tri zadovoljavaju kriterijume predstavljanja znanja: mogućnost proširivanja, jednostavnost i explicitnost. Produkciona pravila – svi ESi kod kojih je znanje predstavljeno u obliku pravila nazivaju se produkcioni sistemi. Pravila su elementi znanja. Pravilo je logična relacija zimeđu elemenata problemskog područja oblika: "ako" X "tada" Y, što znači da ako važi neka premisa tada se može izvesti zaključak ili se može preduzeti neka akcija. Zaključci se izvode poređenjem grupe pravila sa skupom činjenica ili znanja o trenutnoj situaciji. Osnovna ideja je da se na problem iterativno primenjuju pravila iz baze znanja. Mehanizam zaključivanja sadrži poseban program, interpretator pravila koji je zadužen za procesiranje i interpretaciju pravila u toku rada sistema. Produkciona pravila služe i da se prikaže neizvesnost u donošenju odluka. To se postiže korišćenjem faktora izvesnosti, koji je zapravo broj koji predstavlja stepen izvesnosti sa kojim se određene činjenice mogu smatrati važećim, odnosno verovatnoću da važe određeni zaključci u slučaju da su prisutni samo neophodni faktori koji ih podržavaju. Semantičke mreže – su grupe objekata koji su povezani orijentisanim lukovima koji predstavljaju binarne relacije između objekata stavljajući ih time u određeni odnos. Čvorovi i grane su označeni imenima objekata i time je određena semantika mreže. Čvorovi mogu da predstavljaju ljude, objekte, zgrade... a grane različite vrste odnosa između tih objekata. Trojke oblika objekat-atribut-vrednost mogu se posmatrati kao zasebni oblici ovih mreža sa samo tri vrste čvorova i dve vrste grana ['jeste' (između atribut-vrednost) i 'ima' (objekat-atribut)]. Okviri – je komplexna struktura pomoću koje može da se predstavi neki pojam ili objekat. Svaki okvir pripada jednom objektu i sadrži proizvoljan broj označenih polja u koje se smeštaju činjenice/atributi od značaja za taj objekat. Svaki atribut ima niz svojstava. Svojstvo ne mora da bude pasivni epitet, već i neka procedura. Polja mogu biti i prazna ili ukazivati na drugi okvir. Polja možemo popuniti deklarativnim predstavljanjem [jednostavnim navođenjem tvrdnji koje se smatraju tačnim], proceduralnim predstavljanjem [skupom instrukcija koje po izvršenju daju rezultat konzistentan sa početnom činjenicom] i proceduralnim pridruživanjem [polje sadži niz instrukcija za određivanje ulaza u polje]. Okviri mogu da sadrže i podokvire koji mogu da imaju i specifične atribute i koji se povezuju sa nadređenim okvirom sa atributima "je (ste)" ili "vrsta". Okviri su pogodni za kodiranje one vrste znanja koja se odnosi na stereotipne karakteristike pojedinih klasa.

**49. Proces zaključivanja.** O ovom procesu mehanizam zaključivanja, putem inicijalnih podataka o problemu u radnoj memoriji i znanja iz baze znanja, pokušava da pronađe rešenje problema. Korisnik unosi početne uslove u radnu memoriju a mehanizam zaključivanja potom primenjuje znanje iz baze znanja na te podatke pritom generišući nove podatke u radnoj memoriji. Novo stanje u radnoj memoriji može da bude dovoljno za rešavanje problema i time se proces završava. Ako to nije slučaj, proces se iterativno ponavlja dok se ne dođe do rešenja ili dok se ne zaključi da rešenje nije moguće dobiti. Konsultativni ESi mogu zahtevati od korisnika da tokom tog procesa unese dodatne podatke. U sistemima zasnovanim na pravilima interpretator pravila mehanizma zaključivanja radi na sledeći način: inicijalni podaci u radnoj memoriji se tretiraju kao pojedinačni uzorci opštijih tvrđenja definisanih u premisama i zaključcima pravila iz baze znanja. Interpretator pretražuje bazu znanja da bi pronašao ona pravila čijim klauzulama odgovaraju podaci u radnoj memoriji. Ovo pretraživanje se naziva prepoznavanje oblika. Interpretator koristi neku strategiju za efikasnije pretraživanje baze znanja sa ciljem da se ispitaju samo relevantna pravila. Najpoznatije strategije su ulančavanje unapred i ulančavanje unazad. Ulančavanje unapred – polazi od premisa pravila. U svakom ciklusu interpretator pravila ispituje vrednosti premisa relevantnih pravila poredeći ih sa podacima u radnoj memoriji i određuje koja su pravila zadovoljena. Pravilo je zadovoljeno kada svaka njegova premisa odgovara nekom podatku u radnoj memoriji. Ako se pokaže da ni jedno nije zadovoljeno, sistem zaključuje da nema dovoljno podataka za rešenje problema. Ako dođe do konfliktnog slučaja kada je zadovoljeno više pravila, mora se odabrati samo jedno od njih kako bi se primenilo. Za tu odluku se primenjuje heuristički postupak za razrešavanje konflikta. Primenom odabranog pravila dolazi do promene stanja u radnoj memoriji – ako se time rešava problem, korisnik se obaveštava, a ukoliko ne, započinje se sa novim ciklusom zaključivanja. Pravila iz skupa zadovoljenih konfliktnih pravila koja nisu odabrana u prethodnom ciklusu ostaju u tom skupu dok na njih ne dođe red u narednim ciklusima, ili dok se promenom stanja u radnoj memoriji ne stvore uslovi po kojima neko do konfliktnih pravila više nije zadovoljeno. U skupu konfliktnih pravila se nalaze konkretni uzorci zadovoljenih pravila. Ulančavanje unazad – mehanizam zaključivanja pretpostavlja da važi neko rešenje problema iz skupa mogućih rešenja i pronalazi

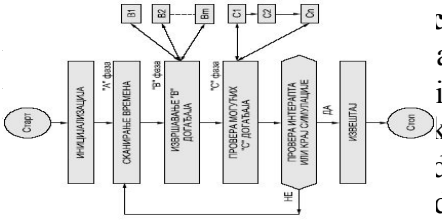
pravilo čije THEN-klauzule označavaju to rešenje. Pretpostavljeno rešenje se naziva tekuća hipoteza. Za pronađeno pravilo interpretator nastoji da proveri da li je zadovoljeno i to putem provere zadovoljenja IF-klauzula tog pravila i to na taj način što se svaka premisa pravila proglašava za novu tekuću hipotezu i postupak se rekurzivno ponavlja. Time se dekomponuje tekući cilj na podciljeve. U slučaju da se ne nađe ni jedno pravilo koje potvrđuje pretpostavljeno rešenje, uzima se novo rešenje i postavlja za tekuću hipotezu. Ovakvi mehanizmi se nazivaju "mehanizmi vođeni ciljevima" za razliku od mehanizama sa pretraživanjem unapred, koji su "vođeni podacima". Korisnik može tražiti od ESa objašnjenje zašto ili kako se došlo do nekog objašnjenja, gde će ES prikazati primenjena pravila i prethodne tekuće ciljeve. U mehanizam zaključivanja mogu da budu uključeni i elementi kao što su prioriteti podataka, kada će se u razmatranje prvo uzeti oni podaci koji se smatraju najvažnijim za proces zaključivanja. Mehanizam zaključivanja može da vodi statistiku o korišćenju pojedinih pravila. Upravljanje procesom zaključivanja može da bude u izvesnoj meri pod kontrolom inženjera znanja preko određenih elemenata baze znanja. Već u samom načinu na koji je znanje prevedeno u skup pravila nalaze se implicitno elementi upravljanja. Za to se koriste metapravila koja predstavljaju upravljačko znanje o načinu primenjivanja ostalih pravila u procesu zaključivanja.

**50. Simulacija zasnovana na znanju.** U tradicionalnom pristupu, analitičar bi kreirao model sistema koristeći neki programerski jezik. Nakon validacije i verifikacije pristupa se experimentisanju sa modelom. Rezultati toga se analiziraju i donosi se odluka. Ovakav način je pogodan za probleme koji su striktno vezani za konvencionalno modeliranje i simulaciju, tj. "simulacija u malom". "Simulacija u velikom" podrazumeva i sve one aktivnosti koje se na prvi pogled nisu mogle dovesti u vezu sa simulacijom, ali se u toku izvođenja experimenta ipak nisu mogle izbeći. Komplexnost problema omogućava korišćenje VI u simulaciji na 2 načina: kreativnost simulacionog modeliranja poboljšati dodavanjem softverskih aplikacija VI i moguće je delove životnog ciklusa simulacije modelirati pomoću VI aplikacija. Time proces simulacije, pored davanja odgovoa na pitanje "šta-ako" treba da omogući i metode zaključivanja, odlučivanja i pretraživanja. Ovakav pristup se naziva simulacijom zasnovanom na znanju.

**51. Simulacioni proces i expertni sistemi.** Moguće primene ESa u simulacionom procesu – izgradnja modela: tri su načina kako ES može da poboljša performanse simulacije u ovoj fazi: expertska pomoć novim korisnicima [kod specifikacije modela i izbora najboljeg], poboljšanje formi predstavljanja realnih sistema [problem koji se javlja jeste da simulacioni modeli ne mogu na adekvatan način da predstave složeno odlučivanje i neizvesne/nekompletne modele; ugrađivanjem ES modula u simulaciju, ovi problemi se otklanjaju], smanjenje obima izračunavanja u simulaciji [kroz zamenu kvalitativnom simulacijom]. Procena parametara:/ Validacija i verifikacija modela: ES otklanjaju greške, daju savete, automatski testiraju model, analiziraju pogrešne rezultate, ukazuju na najčešće izvore grešaka. Planiranje simulacionih experimenata: ESi mogu da pomognu kroz: savete zasnovane na opštim statističkim i simulacionim principima, savete zasnovane na znanju iz posmatranog domena i savete zasnovane na prethodnom iskustvu sa odabranim modelom. Planiranje simulacionih experimenata zavisi od njihove svrhe, koja može biti: evaluacija performansi, poređenje, predviđanje, analiza osetljivosti, optimizacija, uspostavljanje funkcionalnih veza, itd. specifikacija simulacionog modela zahteva detaljno znane, pa kako bi se izbegle greške, ES u ovom domenu predstavlja rešenje. Analiza rezultata: ES može odrediti redosled kojim će se parametri modela menjati kako bi se što pre dostigli postavljeni ciljevi.

**52. Razvoj expertnog sistema za specifikaciju simulacionog modela.** Potrebno je pre svega odrediti atribute i događaje a potom i aktivnosti koje prevode sistem iz jednog u drugo stanje. Savremene metode insistiraju na uspostavljanju baze za definisanje sistema. Osnovna filozofija ovih metoda zasniva se na konceptu odvojenih funkcionalnih zahteva modela koji reprezentuju domen problema. Model se definiše događajima, atributima i skupom procesa koji opisuju redosled tih događaja. Model služi kao baza svih funkcionalnih informacija o stanju okoline. Pravilo se sastoji od dva dela: premise i akcije. Premisa je spoj iskaza a akcija proizvodi jedan ili više zaključaka koji se izvode ako su premise zadovoljene. Predstavljanje znanja u obliku pravila ima prednosti: pravila obezbeđuju prirodni mehanizam za opisivanje problema, pravilo se može posmatrati kao modularni segment znanja unutar domena problema, pravila mogu biti upotrebljena za definisanje znanja ili informacija o drugim pravilima. Specifikacija modela ima tri glavne oblasti u kojima se pravila mogu primeniti:

oblast akcija, atributa i izvođenja entiteta. Oblast akcija entiteta – entitet modela simulira entitet u realnosti. Svaka akcija uključuje dva stanja: početno i završno. Validnost promene stanja definisana je strukturom entiteta. Promena stanja entiteta se može prikazati kao: početno stanje→akcija→odredišno stanje. Pravila definišu skup dozvoljenih akcija za određeno stanje entiteta što se još naziva i "pravilima integriteta entiteta" (EIR). Pokušaj promene stanja entiteta a koje nije sadržano u okviru EIRa posmatra se kao narušavanje ograničenja entiteta. Međutim, postoje situacije u kojima moraju da se zadovolje određeni uslovi, koji se moraju ispitati pre nego se započne sa izvršenjem akcije. Provera tih uslova se može predstaviti u obliku pravila, što se još naziva "pre-akciona uslovna pravila". Mogu se zapisati kao: pre <akcija> uslov <uslov>. Potom, imamo slučaj kada je potrebno izvršiti određene funkcije nakon završetka nekih drugih akcija. To se realizuje "post-akcionim okidačkim pravilima" što se predstavlja kao: nakon <akcija> and <uslov> call <funkcija(argument)>. Oblast atributa entiteta – atribute je potrebno zaštiti od nekonzistentnih vrednosti, kao što su nekonzistentan tip i opseg vrednosti. Nekonzistentnost tipa se odnosi na dodeljivanje nedozvoljenog tipa vrednosti atributu. Pridruživanje opsega vrednosti podrazumeva dodeljivanje vrednosti iz specificiranog opsega određenom atributu. Ovime se opis atributa diže na nivo razumljivom od strane korisnika. Pravilo o tipu atributa definiše tip atributa, čime se atributu mogu dodeliti samo vrednosti propisanog tipa. Pravilo o opsegu vrednosti atributa definiše opseg dozvoljenih vrednosti atributa. Pravilo o zavisnosti atributa ograničava opseg dozovljenih vrednosti atributa ali se opseg vrednosti određuje na osnovu uslova koji ga stavlja u zavisnost od vrednosti drugog atributa. Oblast izvođenja veza – definiše pravila uslovnih reakcija i vrše izračunavanja u procesu simulacije. Pravilo zaključivanja se predsatvlja kao: if <uslov> then <proces>. Ovime se uslovnom relacijom udružuju dva atributa unutar jednog pravila i tako nastaje uslovno pravilo. Pravilo izračunavanja povezuje atribut preko aritmetičkih formula. Jedan atribut koji se nalazi na više mesta menja svoju vrednost izmenom funkcije koja mu dodeljuje novu vrednost.




**cije kao produkcioni sistem.** Metodologiju interakcije procesa moguće je ažuriranje sistema (A), izvršavanje bezuslovnih događaja (B) i izvršavanje ikovito može predstaviti kroz interakciju procesa kao produkcionog sistema: koja opisuju opšte znanje o posmatranom problemu. Pravila se sastoje iz dva događaji), koji predstavljaju završetke aktivnosti i koji se označavaju kao deo, već samo deo koji obuhvata akciju i ovakvo pravilo biva ispaljeno kada nastupi odgovarajući B događaj; i uslovnih događaja (C događaja) koji predstavljaju "ako-tada" produkciona pravila. Kod njih se akcija nakon then-dela pravila izvršava tek kada se zadovolji uslov iz if-dela. Radna memorija produkcionog sistema čuva tekuće znanje o sistemu. Strukturama podataka iz radne memorije upravlja mehanizam zaključivanja preko poziva bibilioteke simulacionih modula. Ova biblioteka kreira strukture podataka i dovoljava njihovo brisanje i/ili ažuriranje. Osnovni deo radne memorije su činjenice i ciljevi. Činjenice su stanja redova i entiteta. Ciljevi su elementi mehanizma koji upravlja vremenom i koji predstavlja plan nastupanja određenih događaja. Mehanizam zaključivanja kontroliše vreme, terminiranje, pozive određenih B događaja i vrši testiranje svih produkcionih pravila C događaja. Produkcioni sistem prvo pronađe sva pravila koja mogu biti zadovoljena tekućim sadržajem podataka u radnoj memoriji a potom strategijom selekcije određuje ono koje će da primeni. Metapravila A faze ažuriraju vreme simulacije kada se odabere novi događaj za izvršenje; metapravila prekida proveravaju da li je ispunjen uslov za prekid simulacije; metapravila B faze izvršavaju B događaje identifikovane metapravilima A faze; metapravila C faze testiraju produkciona pravila za svaki C događaj po redu i ispaljuju ona pravila kod kojih se uslovni deo pravila slaže sa tekućim podacima u radnoj memoriji. Mehanizam zaključivanja koristi strategiju ulančavanja unapred prilikom ispitivanja/izvršavanja pravila C događaja.

**54. Jezici za simulaciju kontinualnih sistema.** Za izradu simulacionog programa na raspolaanju je neki od brojnih simulacionih jezika specijalne namene, neki od opštih jezika viseg nivoa i neki generički model vođen podacima. Postoji podela na jezike za simulaciju kontinualnih, diskretnih i hibridnih modela. Jezici za simulaciju kontinulanih sistema se sastoje od simulacionog jezika, procesora i skupa blokova. Modeli sa kontinualnim promenama sačinjeni su od niza diferencijalnih i diferencnih jednačina. Naziv kontinualni se koristi zato što je nezavisna promenljiva u jednačinama vreme. Diferencijalne jednačine u ovakvim sistemima mogu da sadrže slučajne promenljive za predstavljanje šuma ali i probleme numeričke analize. Jezici za

kontinualnu simulaciju bazirani su na pristupima: 1. definisanje sistema vrši se pomoću podprograma, 2. unošenje jednačina modela u prostoru stanja vrši se pomoću specijalno projektovnog komandnog jezika, 3. vizuelno opisivanje modela sistema pomoću blok dijagrama. Za programiranje se najčešće koriste (u zagradama su dati brojevi pristupa koji se koriste): CSMP (3), DINAMO, ACSL (1), CSSL (1), TUTSIM (1), SIMULINK (3).

**55. CSMP – osnovni koncepti.** Spada u klasu programa za rešavanje sistema diferencijalnih jednačina kod kojeg se sistem jednačina specifira preko blok orijentisanog programa. Svaki element u konfiguraciji ima svoj identifikacioni i grafički simbol. Rad programa zasniva se na algoritmu za sortiranje rednih brojeva blokova. On daje redolse dkojim je potrebno izračunati vrednosti izlaza blokova tako da pri određivanju vrednosti izlaza bilo kof bloka unutar konfiguracije vrednosti izlaza blokova na koji su vezani ulazi tog bloka moraju biti prethodno određene. Po završetku svakog intervala integracije vrednosti konstanti i memorisjkih elemenata su poznati. Koristi se numerička analiza za određivanje vrednosti izlaza integratora. Sistem diferencijalnih jednačina se obrađuje kao vektorska jednačina. Pri svakoj polovini intervala, pomoću odgovarajućih podprograma se određuju vrednosti izlaza modela. Integracija se vrši metodom Runge-Kuta II reda.

**56. Jezici za simulaciju diskretnih događaja.** Digitalna simulacija diskretnih događaja bavi se modeliranjem sistema na digitalnim računarima gde se promenljive stanja mogu predstaviti skupom diskretnih događaja. Ovde se promene stanja sistema dešavaju kada se dogodi događaj. Tehnika intervala zavisi od prirode intervala odabiranja. Vremenski intervali mogu da budu slučajni ili deterministički. Razlikujemo dve vrste simulacija: sinhronu i asinhronu. Strategija upravljanja tokom simulacije su algoritmi ili zasnovani na događajima, aktivnostima i procesima. Najpoznatiji jezici su GPSS, SIMULA, SIMSCRIPT. Prilaz zasnovan na narednom događaju – se sastoji u tome da se odredi naredni događaj i da se izvrše sve promene zavisne od tog događaja. Trofazni prilaz – karakterišu 3 faze izvršavanja: pomeranje simulacionog sata na vreme narednog raspoređenog događaja, izvršavanje svih planiranih događaja čije je vremem nastupanja jednako sadašnjem trenutku, testiranje svih uslovnih dogajaja i izvršavanje onih čiji su uslovi zadovoljeni. Ova metoda dobra je kod modeliranja sistema sa složenijim uslovima ali je relativno neefikasna ukoliko se poveća broj uslovnih događaja. Prilaz zasnovan na procesima – je kombinacija strategije raspoređivanja događaja i skaniranja aktivnosti. Pogodan je za sisteme masovnog opsluživanja. Osnova je za GPSS. Objektno orijentisani simulacioni jezici – su usredsređeni na individualne objekte koji se povezuju sa drugim objektima komunikacijom preko poruka i koji imaju mogućnost da nasleđuju osobine drugih objekata. Jezici za hibridnu simulaciju – se koriste ukoliko model sadrži kontinualne i diskretne promenljive.

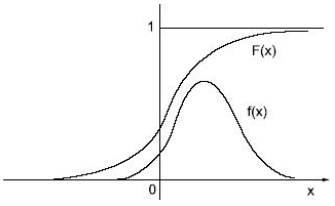
**57. GPSS – osnovni koncepti.** GPSS je interpreterski jezik za simulaciju diskretnih-stohastičkih sistema. Pomoću njega se definiše struktura modela i vrši simulacija. Orijentisan je na procese. Model se predstalja u obliku dijagrama toka. Objekte modela možemo podeliti na dinamičke [transakcija; poseduje niz karakteristika - parametara], statički [resursi na koje se dejstvuje transakćama], statistički [redovi i tabele], i entiteti operacija [prižaju logiku sistemu, daju instrukcije transakcijama]. Transakciju generiše blok GENERATE. Ona se kreće kroz model dok ne naiđe na blok koji nema uslova da je primi ili dok ne naiđe na blok TERMINATE koji je uklanja iz modela. Transakcije se čuvaju u dvema listama – LTD i LBD. Transkacije iz LTD se kreću kroz blok dijagram. Blok dijagram su opercije koje se vrše unutar sistema. Transakcije iz LBD nisu spremne za kretanje. U svakoj tački simulacionog vremena GPSS skanira sve transakcije u LTD. Svaka transakcija koja ima ispunjen uslov daljeg kretanja to i čini dok se ili ne uništi, ili je blok odbije primiti ili ako uđe u blok koji je zadržava određeno vreme. GPSS poseduje 3 vrste naredbi: deklaracione, blok naredbe i kontrolne.

**58. Generisanje slučajnih promenljivih.** Generisanje slučajnih uzoraka se izvršava u dva koraka: najpre se vrednost izvlači iz generatora slučajnih brojeva koji daju brojeve uniformno raspodeljene u otvorenom intervalu (0,1) a zatim se uniformno raspoređeni slučajni uzorak preslikava u ekvivalentnu vrednost iz ciljne raspodele. Kao podrška, potrebno je da imamo izvor slučajnih brojeva uniformno raspodeljenih na intervalu 0-1. U GPSSu postoji osam identičnih tokova generatora slučajnih brojeva i svaki do njih ima neidentičnu podrazumevanu početnu tačku (seme). U GPSSu se koristi jedan algoritam za dobijanje uniformnih slučajnih brojeva na intervalu 0-1. Algoritam je deterministički, te brojevi nisu stvarno slučajni, već pseudoslučajni, što znači da se

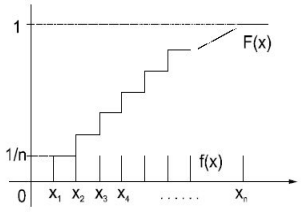
ponavljaju. Stvaranje pseudoslučajnih brojeva može biti prednost pri planiranju statističkih experimenata. Transparentno uzimanje uzoraka u GPSSu se ostvaruje blokvima GENERATE, ADVANCE i TRANSFER. Exponencijalna slučajna promenljiva koja je kontinualna uzima sve vrednosti veće do nule. Koristi se za modeliranje vremena između dolazaka kada ne postoji kontrola nad procesom dolazaka. Ova raspodela je potpuno definisana sa njenom očekivanom vrednošću. Preslikavanje se vrši pomoću jednačine: $t_e=\mu[-\ln(RN_j)]$. Exponencijalna raspodela ima visok stepen varijabilnosti. Kada su vremena između dolazaka exponencijalno raspodeljena, tada iznos dolazaka povezan sa njim sledi Pausonovu raspodelu, pri čemu moraju biti zadovoljene pretpostavke: verovatnoća da se dolazak dešava za vreme malog vremenskog intervala proporcionalna je veličini intervala, verovatnoća dva ili više dolazaka koj se dešavaju za vreme dovoljno malog vremenskog intervala je neznatno mala, vremena između dolazaka su nezavisna od svih drugih. Potom uzorak iz Erlangove raspodele može se dobiti sabiranjem dva ili više uzoraka iz exponencijalne raspodele. Koristi se za modeliranje vremena opsluge. Karakterišu je dva parametra: redom i srednjim vremenom opsluživanja. Exponencijalna raspodela je Erlangova raspodela prvog reda. Jedan od načina da uzmemo uzorak iz Erlangove raspodele k-tog reda sa očekivanom vrednošću $\mu$ jeste da formiramo sumu od k uzoraka iz exponencialne raspodele sa očekivanom vrednošću $\mu/k$. Normalna slučajna promenljiva koja je kontinualna i simetrična, sa osvrtom na njenu očekivanu vrednost, uzima vrednosti u opsegu od $-\infty$ do $+\infty$. Normalna raspodela je u potpunosti okarakterisana sa očekivanom vrednošću $\mu$ i njenom standardnom devijacijom $\sigma$. Normalna slučajna promenljiva sa očekivanom vrednošću 0.0 i standardnom devjacijom 1.0 naziva se standardizovana normalna slučajna promenljiva. Normalnoj raspodeli podvrgavaju se vremena ospluge. Metod koji se koristi za uzmanje uzoraka iz nestandardne normalne raspodele je da konvertujemo uzorak iz uniformen raspodele 0-1 u vrednost iz standardne raspodele a onda da konverujemo vrednost iz standardne raspodele u ekvivalentnu vrednost iz nestandardne normalne populacije na osnovu izraza: $x=\sigma z+\mu$.

**59. Slučajni događaji, verovatnoća, slučajne promenljive.** Slučajni događaji su oni koji se u datoj pojavi mogu, ali i ne moraju ostvariti. Pojave koje preko njih izučavamo nazivamo experimentima, a skup mogućih išoda experimenata zovemo skup elementarnih događaja koji može biti ili beskonačan i prebrojiv ili beskonačan i neprebrojiv. Karakteristike slučajnih događaja: slučajni događaj je podskup skupa elementarnih događaja, nemoguć događaj je podskup skupa elementarnih događaja koji nema nijedan element i označava se sa $\varnothing$, ako je dat događaj A onda je njemu suprotan događaj $A^{nad}$ koji se ostvari onda i samo onda ako se A ne ostvari, suprotan događaj nemogućeg događaja je siguran događaj, ukoliko se svaki put kada se ostvari događaj A ostvari i događaj B onda $[A\Rightarrow B]$, za dva slučajna događaj A i B kažemo da su jednaki ako $[A\Rightarrow B \text{ i } B\Rightarrow A]$, unija događaj A i B je onaj događa koji se ostvari onda i samo onda kada se ostvari bar jedan od događaj A i B i označava se sa $A\cup B$, presek događaja A i B je slučajan događaj koji se ostvaruje onda i samo onda kada se ostvare oba događaja i piše se kao $A\cap B$, za dva događaja A i B kažemo da se međusobno isključuju ako je njihov presek nemoguć događa tj. $A\cap B=\varnothing$, ako se dva događaj međusobno isključuju tada je njihova unija zbir tih događaja tj. $A\cup B=A+B$, za operacije unije i preseka važe osobine komutativnosti, asocijativnosti i distributivnosti. Verovatnoća slučajnih događaja je svaka funkcija P definisana nad slučajnim događajima koja preslikava slučajne događaje u realne brojeve i koja ima osobine: nenegativnosti, normiranosti, aditivnosti, a potom i: verovatnoća suprotnog događaj jednaka je $P(A)=1-P(A^{nad})$, verovatnoća nemogućeg događaj jenaka je nuli, verovatnoća bilo kog slučajnog događaja je broj koji se nalazi između $0<P(A)<1$. Uslovna verovatnoća je verovatnoća događaja A ako se događaj B desio i obeležava se kao $P(A/B)=P(AB)/P(B)$. Slučajna promenljiva $X^{\sim}$ može se definisati kao promenljiva koja predstavlja išod experimenta, tj. ona uzima vrednost iz skupa mogućih rezultata experimenta i to na slučajan način. Postoje dve kategorije slučajnih promenljivih, diskretne [ukoliko vrednosti koje ona može uzeti obrazuju konačan ili prebrojiv niz realnih brojeva a uzimanje svake od ovih vrednosti je slučajan događaj sa određenom verovatnoćom] i kontinualne [ako uzimanje bilo koje vrednosti iz intervala $[-\infty,+\infty]$ predstavlja slučajan događaj i ako postoji funkcija f(x) takva da je verovatnoća da će se X naći u proizvoljno maloj okolini tačke x u intervalu $[x-\Delta x/2;x+\Delta x/2]$.

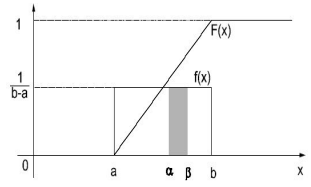
**60. Funkcija gustine raspodele i funkcija raspodele.** Funkcija f(x) naziva se funkcija gustine raspodele slučajne promenljive $X^{\sim}$ i ona mora da zadovolji dva uslova: verovatnoća da se slučajna promenljiva nađe u nekom intervalu $(x_1,x_2]$ je: $P(x_1<X<x_2)=\int_{x1}^{x2} f(x)dx$, i verovatnoća da slučajna promenljiva $X^{\sim}$ uzme vrednost ne manju od x pri čemu je x unapred određen broj iz intervala $(-\infty,+\infty)$. Funkcija raspodele slučajne promenljive

X~ je funkcija F(x) koja za svako x iz (-∞,+∞) predstavlja verovatnoću da slučajna pormenljiva X neće uzeti vrednost veću od z, tj. F(x)=P(X<z). 

Osobine: funkcija raspodele ima vrednosti između nula i jedan, F(-∞)=0, F(+∞)=1, funkcija F(x) je monotono neopadajuća funkcija, kod slučajne pormenljive X~ diskretnog tipa funkcija raspodele F(x) je stepenasta funkcija koja u tačkama x1,x2,... ima skokove jednake odgovarajućim verovatnoćama, ako je X~ neprekidna slučajna promenljiva funkcijom gustine raspodele f(x) tada je za svako $x \in (-\infty,+\infty) \Rightarrow F(x)=\int_{-\infty}^{x} f(x)dx$.

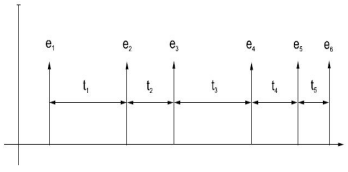
**61. Diskretna i kontinualna uniformna raspodela.** Diskretna uniformna raspodela slučajne promenljive X može se predstaviti kao: 

Slučajna promenljiva X~ uzima vrednosti iz skupa {x1,...xn}. Verovatnoća da slučajna promenljiva uzima vrednosti $x_i$ je $P(X=x_i)=1/n$, 1=1,n pri čemu je matematičko očekivanje $\mu=\sum_{i=1}^{n} x_i/n$, a varijansa $\sigma^2=\sum_{i=1}^{n}(x_i-\mu)^2$. Kontinualna uniformna raspodela može se grafički prikazati:



gde je x∈X=[a,b] interval u R, i f(x)={1/(b-a), a≤x<b; 0, drugačije. Verovatnoća da će se slučajna promenljiva X~ naći u intervalu {α,β} jednaka je P(α≤x<β)=(β-α)/(b-a).

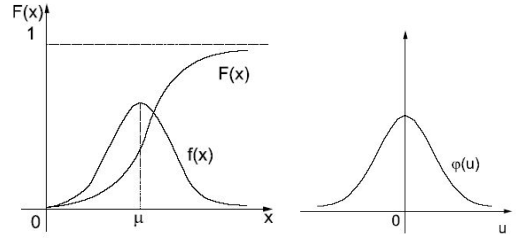
**62. Exponencijalna raspodela.** Događaji se u okviru ove raspodele mogu predstaviti kao:



Važe pretpostavke: vreme nastupanja događaja ne zavisi od prethodnog događaja, verovatnoća da događaj nastupi u malom intervalu vremena proprocionalna je dužini tog intervala: $P(t\sim<t+\Delta t/t\sim\geq t)=P(t\leq t\sim<t+\Delta t)/P(t\sim\geq t)=(F(t+\Delta t)-F(t))/(1-F(t))=\lambda\Delta t$, gde je P(t~<t)=F(t) tražena raspodela za t~. Ako $\Delta t\rightarrow 0 \Rightarrow F'=\lambda(1-F(t))$, t≥0 pa je tražena raspodela $F(t)=\{1-e^{-\lambda t}$, t≥0; 0, t<0, dok je funkcija gustine raspodele $f(t)=\{\lambda e^{-\lambda t}$, t≥0; 0, t<0.

**63. Poisson-ova raspodela.** Za svaku slučajnu promenljivu X~ kažemo da ima Poisson-ovu raspodelu ako može uzeti vrednost k iz niza nenegativnih celih brojeva sa verovatnoćom $P(k)=P(x=k)\lambda^k e^{-\lambda}/k!$, pri čemu je λ>0, realan broj i predstavlja parametar raspodele. Ako intervali između dva susedna događaja imaju exponencijalnu raspodelu, tada je verovatnoća da se n dogajaja desi na intervalu T: $P(n\sim=n)=(\lambda T)^n e^{-\lambda T}/n!$, n=1,2...

**64. Normalna raspodela.** Koristi se za predstavljanje stohastičkih pojava unošenje šuma u determinističke promenljive. Definisana je sa dva parametra: μ-srednja vrednost i σ-standardna devijacija; $P(x<X)=\Phi((x-\mu)/\sigma)$ i

$\Phi(z)=\int_{-\infty}^{x}\xi d\xi$. Funkcija standardizovane normalne raspodele za koju važi da je $\mu=0$ i $\sigma=1$ ima oblik: $\phi(z)=e^{-u\hat{}2/2}/\sqrt{(2\pi)}$. 

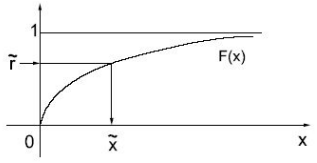
(za $\mu=0$ i $\sigma=1$).

**65. Generisanje slučajnih brojeva.** Slučajnu promenljivu sa uniformnom raspodelom moguće je dobiti jednom do sledećih metoda: manuelne metode [malog opsega, gotovo neprimenjive u simulaciji], tabele slučajnih brojeva [serije slučajnih brojeva dobijenih pomoću fizičkog izbora ili nekom do preostalih metoda], metode za rad sa analognim [ove metode daju prave slučajne brojeve, ali nepodesne za rad jer se pri novom slučaju za iste parametre dobijaju novi slučajni brojevi] i metode za rad sa digitalnim računarima [mogući postupci: korišćenje fizičkih izvora +A/D konverzija, korišćenje tabela slučajnih brojeva na diskvima računara, korišćenje algoritma za dobijanje pseudoslučajnih brojeva].

**66. Linearni kongruenti GSB.** Kongruentni brojevi su brojevi deljivi bez ostatka. Na osnovu te osobine mogu se definisati sledeći generatori: multiplikativni $Z_i=(Z_{i-1}a)\text{mod } m$, mešoviti $Z_i=(Z_{i-1}a+c)\text{mod } m$, aditivni $Z_i=(Z_{i-1}a+Z_{i-k}b)\text{mod } m$, gde $Z_i$-pseudoslučajni broj, a-multiplikator, b,c-konstante, m-modulus. Linearni multiplikativni kongrunetni generator slučajnih brojeva generiše slučajne brojeve po sledećoj formuli: $Z_i=(Z_{i-1}a+b)\text{mod } m$, $m=2^{b-1}$. Maximalna moguća dužina brojeva bez ponavljanja je m-1. Kvalitet ovog generatora zavisi od broja bitova slučajne porenljive b. da bi se on dostigao početna vrednost sekvence pseudoslučajnih brojeva mora da ispunjava uslove: $a=8j\pm3$, $j=1,2,3...$, $Z_0$1,3,5,7...

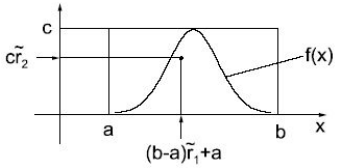
**67. Testovi za proveru GSBa.** K-test: ovaj test služi za ispitivanje da li neki skup generisanih slučajnih brojeva ima pretpostavljeni raspored. Neki skup od n elemenata može se grupisati prema nekoj osobini u r grupa. Neka se u i-toj grupi nađe $f_1$ elemenata tako da je $f_1+f_2+...+f_r=n$. Verovatnocja da cje se element x nacji u grupi $f_i$ je $p_i$. Poznato je da broj elemenata koji prema teorijskoj raspodeli treba da se nadju u i-tom intervalu iznosi $np_i$. Da bi se pretpostavka ($H_0$) da dati skup podlezhe odredjenoj raspodeli bila u vazhnosti, treba da se nadje vrednost izraza $2(f_i-np_i)$ i da se ona uporedi sa vrednoshcju k koja se nalazi u tablicama za r-1 stepene slobode i za verovatnocju koja odgovara zadatom nivou znachajnosti. Ako je $k<k_0$ smatra se da skup podlezhe pretpostavljenoj raspodeli. Kolmogorov-Smirnov test: omogucjava poredjenje da li ispitivani skup sluchajnih brojeva podlezhe pretpostavljenoj raspodeli. Neka je generisano n sluchajnih brojeva. Treba prvo obaviti njihovo sredjivanje po velichini tj. $x_1<x_2<...<x_n$. Kod uniformne raspodele verovatnocja pojavljivanja svakog od njih iznosi 1/n. Kumulativna funkcija raspodele mozhe se konstruisati na taj nachin shto cje kod svakog x docji do porasta funkcije za 1/n. Poshto je kumulativna funkcija uniformne raspodele Fn(x) poznata, to se mozhe izrachunati izrazom: $D_n=\max_{-\infty<z<\infty}|F(z)-F_e(z)|$. Ukoliko je $D_n(x)<$od kritichne vrednosti $d_0$ za prag sluchajnosti $\alpha$, hipoteza da se radi o uniformnoj raspodeli bicje prihvacjena.

**68. Metoda inverzne transformacije.** Uvodi se smena $X\sim=h(r\sim)$ takva da sluchajnoj promenljivoj X~ odgovara zheljena funkcija raspodele F(X). $F(X)=P(X\sim<X)=P(r\sim<r)=P(r\sim<h^{-1}(X))=U(h^{-1}(X))=h^{-1}(X)$, $h(r\sim)=F^{-1}(r\sim)$. Primena je ogranichena na funkcije raspodele koje imaju inverznu funkciju i monotone su.



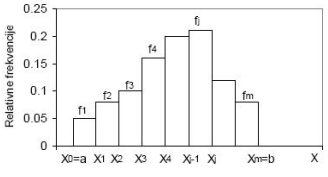

**69. Metoda odbacivanja.** Uslov za primenu je ogranichen opseg definisanosti fukcije raspodele: $f(X)\approx0$ za $X\notin[a,b]$. Ako se na povrshini (b-a)*c generishe niz tachaka sa uniformnom raspodelom i ako se odbace sve

tachke koje padaju iznad krive f(X) preostale tachke imacje funkciju gustine raspodele f(X). Algoritam: repeat

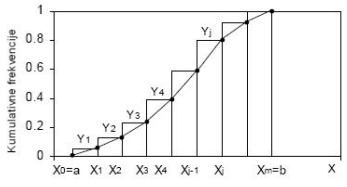



generishi $r_1$ i $r_2$;$X_1$:=(b-a)*$r_1$+a; until $r_2c \leq f(X_1)$ X:=$X_1$.

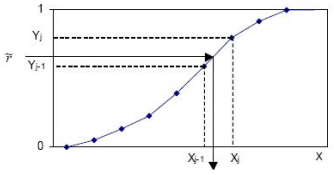
**70. Metoda pravougaone transformacije.** Koristi se za generisanje empirijskih raspodela kada je funkcija raspodele monotona i zadata parovima tachaka. Verovatnocja, da cje se desiti dogadjaj, izrazhena u vidu empirijskih podataka, grupishe se u proizvoljan broj frekvencijskih klasa (m) koje sachinjavaju histogram. Svaka klasa oznachena je pravougaonikom chija se donja i gornja granica mogu oznachiti sa $X_{j-1}$, $X_j$, j=1,m. Ukupan broj realizacija sluchajne promenljive u okviru jedne klase se naziva apsolutna frekvencija i oznachava se sa $n_j$, j=1,m. Relativna frekvencija svake klase $f_j=1/n_j$ predstavlja verovatnocju da cje realizacija sluchajne promenljive X~ uzeti vrednost iz klase j tj. $X_{j-1} \leq X\sim \leq X_j$.




(histogram raspodele relativnih frekvencija) zbir relativnih frekvencija je $\sum_{j=1,m} f_j=1$. Na osnovu histograma, mozhe se konstruisati funkcija raspodele, tj. izrachunati kumulativna suma prethodnih frekvencija: $Y_1=f_1$ $Y_2=f_1+f_2$ ... $Y_m=f_1+f_2+...+f_m=1$. Kumulativna suma predstavlja verovatnocju da sluchajna vrednost X~ ne prelazi vrednost $X_j$. Potom, potrebno je aproksimativno odrediti funkciju raspodele, tako shto se kroz kumulativni histogram provlachi kontinualna kriva




za dobijanje odgovarajucje vrednosti X~ potrebno je generisati uniformno raspodeljen sluchajni broj r~∈[0,1) i koristi jednachinu za linearnu interpolaciju: $X\sim=X_{j-1}+[(r\sim-Y_{j-1})/(Y_j-Y_{j-1})](X_j-X_{j-1})$ shto se mozhe ilustrovati:




Klasa koja odgovara sluchajnoj velichini r~ se odredjuje pretrazhivanjem, koje startuje od krajnje leve klase i sukcesivnim poredjenjem vrednosti r~ sa vrednostima $Y_j$. Trazhena klasa je prva klasa za koju je $r\sim \leq Y_j$.

**71. Metoda sumiranja.** Koristi se za generisanje normalne raspodele. zasniva se na principu centralne granichne teoreme, koja se formulishe kao: Neka je $X_1\sim,X_2\sim,...,X_n\sim$ niz nezavisnih sluchajnih promenljivih sa jednakim verovatnocjama, svaka sa srednjom vrednoshcju $\mu_x$ i konachnom varijansom $\sigma_x^2$. Njihova srednja vrednost je: $X^{nad}=\sum_{j=1,m} X_j\sim/n$. Tada promenljiva $Y=(X\sim-\mu_x)/\sigma_x/\sqrt{n}$ tj. za dovoljno veliko n razlika izmedju promenljive Y i standardizovane normalne promenljive mozhe se zanemariti. Da bi generisali uzorak iz standardne normalne raspodele mozhemo uzeti n nezavisnih uniformno raspodeljenih sluchajnih brojeva $r_i\sim\in[0,1)$ sa srednjom vrednoshcjuE$(r_i\sim)=1/2$ i varijansom $E(r_i\sim-\mu)^2=1/12$. Sluchajna promenljiva $Z=[\sum_{j=1,n} r_i\sim/n-1/2]/\sqrt{[(1/12)/\sqrt{n}]}=[\sum_{j=1,n} r_i\sim-n/2]/\sqrt{(n/12)}$ ima normalnu raspodelu sa srednjom vrednoshcju 0 i varijansom 1 kada $n\to\infty$. Zadovoljavajucji rezultati aproksimacije dobijaju se za n=12.

**72. Box-Muller-ov metod.** Ovo je egzaktan metod koji koristi dve nezavisne pseudo-sluchajne promenljive $r_1$ i $r_2$ koje se koriste za generisanje standardizovanih normalno raspodeljenih sluchajnih promenljivih korishcjenjem oba ili jednog od sledecjih izraza: $Z_1=\cos(2\pi r_1)\sqrt{(-2\ln(r_2))}$ i $Z_2=\sin(2\pi r_1)\sqrt{(-2\ln(r_2))}$. Potrebno je za svako Z imati samo po dve vrednosti $r_1\sim$.