

---

## РАЧУНАРСКА СИМУЛАЦИЈА

Божидар Раденковић  
Милорад Станојевић  
Александар Марковић

---

❖ **Рецензенти:**

Проф. др Милош Рајков, дипл. инж.  
Проф. др Срђан Станковић, дипл. инж.  
Проф. др Добривоје Јовановић, дипл. инж.

-----  
На основу одлуке Издавачког одбора Факултета организационих наука Универзитета у Београду 04-07 број 61/10 од 07. 09. 1999. године, одобрава се за употребу у настави као основни уџбеник

На основу одлуке Уређивачког одбора Саобраћајног факултета Универзитета у Београду број 392/1 од 06. 07. 1999. године, одобрава се за употребу у настави као основни уџбеник  
-----

❖ **Главни и одговорни уредници:**

Др Наход Вуковић, редовни професор  
Др Стево Ерор, редовни професор

❖ **Корице:**

Вида Милошевић

❖ **Лектор:**

Љубица Урошевић

❖ **Издавачи:**

Факултет организационих наука, Београд, Јове Илића 154  
Саобраћајни факултет, Београд, Војводе Степе 305

❖ **Штампа:**

Служба за издавачку делатност Саобраћајног факултета

❖ **Тираж:**

300 примерака

YUISBN 86-7395-074-0

---

Издавачи задржавају сва права. Репродукција појединих делова или целине ове публикације није дозвољена

---

## САДРЖАЈ

<b>1. МОДЕЛИРАЊЕ И СИМУЛАЦИЈА</b>	<b>1</b>
1.1 Моделирање и модели	1
1.1.1 Врсте модела	3
1.1.2 Неформални и формални модели	5
1.2 Рачунарска симулација	8
1.2.1 Моделирање и симулација	9
1.2.2 Предмет моделирања и симулације	11
1.3 Историјски преглед развоја симулације	13
1.4 Карактеристике симулационог моделирања	14
1.5 Потреба за симулацијом	16
1.5.1 Када је могуће експериментисати на систему?	17
1.5.2 Када није могуће експериментисати на систему ?	17
1.6 Могућности примене симулације	18
1.7 Предности и недостаци симулације	19
1.8 Популарност симулације	20
1.9 Симулациони процес	21
1.10 Поделе симулационих модела	24
1.10.1 Детерминистички и стохастички модели	24
1.10.2 Дискретни и континуални модели	26
1.11 Врсте симулационих модела	27
1.11.1 Монте Карло симулација	28
1.11.2 Континуална симулација	29
1.11.3 Симулација дискретних догађаја	31
1.11.4 Мешовита симулација	31
1.12 Избор типа симулационог модела	32
<b>2. КЛАСИФИКАЦИЈЕ МОДЕЛА</b>	<b>33</b>
2.1 Класификација модела	33
2.1.1 Класификација у односу на променљиве	33
2.1.2 Класификација у односу на природу опсега вредности променљивих модела	35

2.1.3	Класификација у односу на природу опсега вредности променљиве "време"	35
2.1.4	Класификација у односу на временску зависност модела	36
2.1.5	Класификација у односу на детерминизам	36
2.1.6	Класификација у односу на предвиђање будућности	37
2.1.7	Класификација у односу на линеарност	38
2.1.8	Класификација према врсти рачунара	38
2.1.9	Класификација у односу на формални опис модела	39
2.2	Формална спецификација модела	41
2.2.1	Апстракција	41
2.2.2	Асоцијација	41
2.2.3	Спецификација	42
2.3	Формални модел улазно-излазног система	42
<b>3.</b>	<b>ОЦЕНА ПАРАМЕТАРА МОДЕЛА</b>	<b>47</b>
3.1	Оцене параметара детерминистичког модела	49
3.2	Оцене параметара модела стохастичких система	50
3.3	Статистички приступ процени параметара статичких модела	51
3.3.1	Оцена средње вредности случајне променљиве	52
3.3.1.1	Секвенцијални метод	52
3.3.1.2	Рекурзивни метод	52
3.4	Оцена непознатог параметра по методи најмањих квадрата	53
3.4.1	Секвенцијално решење	53
3.4.2	Рекурзивно решење	54
3.5	Процена к непознатих параметара	55
<b>4.</b>	<b>ВАЛИДАЦИЈА И ВЕРИФИКАЦИЈА</b>	<b>59</b>
4.1	Валидација симулационих модела	61
4.1.1	Практичан приступ процесу валидације	64
4.1.1.1	Процена валидности модела	64
4.1.1.2	Валидација претпоставки модела	65

4.1.1.3 Валидација улазно – излазних трансформација	66
4.1.2 Формални критеријум за утврђивање валидности модела	67
4.2 Верификација симулационих модела	69
<b>5. СРЕДСТВА ЗА СИМУЛАЦИЈУ</b>	<b>71</b>
5.1 Аналогни рачунар	71
5.1.1 Потенциометар	75
5.1.2 Појачавач	76
5.1.3 Сабирач	77
5.1.4 Интегратор	78
5.1.5 Множач	79
5.1.6 Сервомножач	80
5.1.7 Диодни множачи	80
5.1.8 Генератори функција	81
5.1.9 Поступци при раду са аналогним рачунаром	83
5.2 Дигитални рачунар	83
5.3 Хибридни рачунар	85
<b>6. СИМУЛАЦИЈА КОНТИНУАЛНИХ СИСТЕМА</b>	<b>89</b>
6.1 Формални модел	90
6.2 Функција једне или више променљивих – блок	92
6.3 Апстрактни континуални симулациони систем	92
6.3.1 Подскупови КСС-а	93
6.4 Функције стања, алгебарске функције, променљиве стања	93
6.5 Уредљивост	93
6.6 Симулација континуалних система помоћу аналогног рачунара	94
6.7 Симулација континуалних система помоћу дигиталног рачунара	96
6.8 Интеграција у симулацији континуалних система	98
6.9 Одређивање интервала интеграције	101
6.10 Извођење имплицитних операција	103
6.11 Рачунарска реализација симулатора континуалних система	104

6.12	Симулациони језик	105
6.13	Процесор	106
<b>7.</b>	<b>СИМУЛАЦИЈА ДИСКРЕТНИХ ДОГАЂАЈА</b>	<b>111</b>
7.1	Формални опис система са дискретним догађајима	112
7.2	Догађај, активност и процес	113
7.3	Развој симулације дискретних догађаја	115
7.3.1	Механизам помака времена	115
7.3.1.1	Помак времена за константан прираштај	115
7.3.1.2	Помак времена на наредни догађај	116
7.3.2	Генерисање догађаја	117
7.4	Стратегије извођење симулације	119
7.4.1	Распоређивање догађаја	119
7.4.2	Сканирање активности	121
7.4.3	Интеракција процеса	122
<b>8.</b>	<b>ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И СИМУЛАЦИЈА</b>	<b>125</b>
8.1	Историјски развој вештачке интелигенције	130
8.2	Основни концепти вештачке интелигенције	132
8.3	Основни елементи вештачке интелигенције	134
8.3.1	Хеуристичко претраживање	134
8.3.2	Представљање знања	136
8.3.3	Логичко закључивање	136
8.3.4	Језици и алати вештачке интелигенције	137
8.4	Основна подручја вештачке интелигенције	137
8.5	Експертни системи	139
8.6	Дефинисање експертних система	140
8.7	Експерти и експертни системи	142
8.8	Експертни системи и конвенционални програми	144
8.9	Структура експертних система	146
8.10	Представљање знања	148
8.10.1	Продукциона правила	150
8.10.2	Семантичке мреже	152
8.10.3	Оквири	153
8.11	Процес закључивања	154
8.11.1	Уланчавање унапред	156
8.11.2	Уланчавање уназад	157

8.12	Симулација заснована на знању	160
8.13	Аспекти интеграције ЕС и симулационих модела	161
8.14	Симулациони процес и експертни системи	162
8.14.1	Изградња модела	163
8.14.2	Процена параметара	164
8.14.3	Валидација и верификација модела	164
8.14.4	Планирање симулационих експеримената	165
8.14.5	Анализа резултата	165
8.15	Спецификација симулационог модела преко ЕС	166
8.16	Развој експертног система за спецификацију симулационог модела	168
8.16.1	Уграђивање правила	168
8.16.2	Област акција ентитета	169
8.16.2.1	Правила интегритета ентитета	169
8.16.2.2	Пре-акциона условна правила	171
8.16.2.3	Пост-акциона окидачка правила	171
8.16.3	Област атрибута ентитета	172
8.16.3.1	Правило о типу атрибута	173
8.16.3.2	Правило о опсегу вредности атрибута	173
8.16.3.3	Правило о зависности атрибута	173
8.16.4	Област извођења веза	174
8.16.4.1	Правило закључивања	174
8.16.4.2	Правило израчунавања	175
8.17	Стратегија трофазне симулације као продукциони систем	175
8.18	Језици вештачке интелигенције у симулацији	179
<b>9.</b>	<b>ЈЕЗИЦИ ЗА СИМУЛАЦИЈУ КОНТИНУАЛНИХ СИСТЕМА</b>	<b>181</b>
9.1	Језици за симулацију континуалних система	182
9.2	TUTSIM	185
9.2.1	Решење система “вешања” аутомобила на језику TUTSIM	186
9.3	SIMULINK	187
9.3.1	Решење система “вешања” аутомобила на SIMULINK-у	189
9.4	Симулациони језик CSMP	190
9.4.1	Начин рада CSMP-а	191

9.4.2	Симулациони језик CSMP-W95/NT	192
9.4.2.1	Дефинисање конфигурације	193
9.4.2.2	Дефинисање вредности од значаја за симулацију	196
<b>10.</b>	<b>ЈЕЗИЦИ ЗА СИМУЛАЦИЈУ ДИСКРЕТНИХ ДОГАЂАЈА</b>	<b>199</b>
10.1	Прилаз заснован на наредном догађају	199
10.1.1	SIMSCRIPT	200
10.2	Трофазни прилаз	200
10.2.1	ECSL	201
10.3	Прилаз заснован на процесима	201
10.4	Објектно орјентисани симулациони језици	202
10.5	Језици за хибридну симулацију	202
10.6	Симулација на језику GPSS	203
10.6.1	Динамички ентитети	204
10.6.2	Статички ентитети	205
10.6.3	Статистички ентитети	205
10.6.4	Ентитети операција	205
10.6.5	Основни концепти GPSS језика	206
10.6.6	Врсте наредби у GPSS-у	207
10.6.7	Основни скуп наредби GPSS језика	208
10.6.7.1	Генерисање трансакција	209
10.6.7.2	Временско задржавање трансакција	210
10.6.7.3	Статистичко гранање трансакције	211
10.6.7.4	Безусловни скок	212
10.6.7.5	Уклањање трансакције из система	213
10.6.7.6	Стартовање симулатора	214
10.6.7.7	Дефинисање почетка и краја GPSS програма	215
10.6.8	Основни перманентни ентитети	217
10.6.8.1	Уређаји	217
10.6.8.2	Складишта	221
10.6.8.3	Логички прекидачи	224
10.6.8.4	Редови	225
10.6.8.5	Табеле (хистограми)	226
10.6.8.6	Блок MARK	227
10.6.9	Генерисање случајних променљивих	229

10.6.9.1 Узимање узорака из популације	229
10.6.9.2 Генератори униформних случајних бројева	230
10.6.9.3 Транспарентно узимање узорака у блоковима GENERATE, ADVANCE и TRANSFER	230
10.6.9.4 Експоненцијална расподела	235
10.6.9.5 Релације између експоненцијалних и Пуасонових случајних променљивих	236
10.6.9.6 Узимање узорака из Ерлангове расподеле	237
10.6.9.7 Нормална расподела	238
10.6.9.8 Статистичка независност у симулацији	240
10.6.9.9 Остале наредбе GPSS језика	244
10.6.9.10 Симулација телефонске централе на GPSS –у	245
<b>11. ДИНАМИКА СИСТЕМА И ПРОГРАМСКИ ЈЕЗИК SDS</b>	<b>251</b>
11.1 Историјски преглед динамике система	251
11.2 Системи са и системи без повратног дејства	252
11.3 Поларитет кола повратног дејства	253
11.4 Основни појмови система са повратним дејством	257
11.5 Концептуални модели динамике система	258
11.5.1 Дијаграми узрочних петљи	258
11.5.2 Дијаграми тока	260
11.6 Рачунарски модели у динамици система	262
11.6.1 Описивање промене времена у једначинама модела	263
11.7 Програмски пакет SDS	265
11.7.1 Пример програма у SDS –у.	266
<b>П ВЕРОВАТНОЋА И СТАТИСТИКА У СИМУЛАЦИЈИ</b>	<b>271</b>
П.1 Случајни догађаји	271
П.2 Вероватноћа	273
П.2.1 Условна вероватноћа	274
П.3 Случајне променљиве	274
П.4 Функција расподеле и функција густине расподеле	275



П.4.1	Трансформација случајне променљиве	277
П.4.2	Нумеричке карактеристике функције расподеле	277
П.5	Стандардизована случајна променљива	280
П.6	Типичне расподеле случајних променљивих	281
П.6.1	Дискретна униформна расподела	281
П.6.2	Континуална униформна расподела	282
П.6.3	Експоненцијална расподела	283
П.6.4	Poisson-ова расподела	284
П.6.5	Нормална (Gauss-ова) расподела	285
П.7	Генерисање случајних бројева	286
П.7.1	Мануелне методе	287
П.7.2	Табеле случајних бројева	287
П.7.3	Методе за генерисање СБ на аналогном рачунару	287
П.7.4	Методе за генерисање СБ на дигиталном рачунару	288
П.7.5	Линеарни конгруентни ГСБ	288
П.8	Тестови за проверу ГСБ	291
П.8.1	$\chi^2$ -тест	291
П.8.2	Колмогоров-Смирнов тест	292
П.9	Генерисање случајне променљиве са задатом расподелом	293
П.9.1	Метода инверзне трансформације	293
П.9.2	Метода одбацивања	294
П.9.3	Метода правоугаоне апроксимације	295
П.9.4	Метода сумирања	299
П.9.5	Box-Muller-ов метод	300
<b>ЛИТЕРАТУРА</b>		<b>303</b>

## ПРЕДГОВОР

Уџбеник под називом “Рачунарска симулација” намењен је студентима Факултета организационих наука и Саобраћајног факултета за предмете из области рачунарске симулације.

Симулација је савремена научна дисциплина, која у свету, тек сада, развојем информационих технологија, а нарочито мултимедија, мултипроцесорских система, вештачке интелигенције и рачунарских мрежа, добија прави значај и примену.

Циљ овог уџбеника је да читаоца упозна са тренутним стањем метода и техника које се користе у рачунарској симулацији и да укаже на основне правце будућег развоја ове научне дисциплине.

У првој глави дати су основни појмови везани за изградњу модела, планирање експеримента, извођење симулације и анализу резултата. Друга глава се односи на класификацију и формалну спецификацију модела коришћењем теорије скупова. У трећој глави се описују методе за оцену параметара модела. Четврта глава се бави проблемима одређивања валидности модела и верификације симулатора. У петој глави дат је опис аналогних, дигиталних и хибридних рачунара који се користе као средства у симулацији. Шеста глава се бави проблемима симулације континуалних система на дигиталним рачунарима. У седмој глави су разматране методе за симулацију система описаних помоћу дискретних догађаја. Примена вештачке интелигенције по фазама животног циклуса симулационог процеса описана је у осмој глави. Девета глава се бави симулационим језицима за континуалну симулацију, док је десета глава посвећена језицима за симулацију система описаних помоћу дискретних догађаја. У једанаестој глави разматрани су основни концепти везани за динамику система и коришћење програмског језика SDS.

У прилогу уџбеника је дат кратак преглед статистичких метода који се користе у симулацији.

С обзиром да на српском језику до сада није било свеобухватне литературе из ове области, узор при писању ове књиге били су радови најпознатијих аутора који се баве теоријом моделирања и симулације.

Захваљујемо мр Драгану Вукмировићу и дипл. инж. Владимиру Вујину на сугестијама и помоћи при изради ове књиге.

Такође захваљујемо бројним студентима Факултета организационих наука и Саобраћајног факултета који су својим семинарским и дипломским радовима допринели квалитету и изгледу ове књиге.

Све сугестије у вези са изложеним градивом су добродошле.

Аутори

## МОДЕЛИРАЊЕ И СИМУЛАЦИЈА

### 1.1 Моделирање и модели

Моделирање представља један од основних процеса људског ума. Оно је уско везано за начин људског размишљања и решавања проблема. Као резултат процеса који називамо интелигентно људско понашање, моделирање представља свакодневну активност и велики део онога што нас чини људским (интелигентним) бићима. Моделирање изражава нашу способност да мислимо и замишљамо, да користимо симболе и језике, да комуницирамо, да вршимо генерализације на основу искуства, да се суочавамо са неочекиваним. Оно нам омогућава да уочавамо обрасце, да процењујемо и предвиђамо, да управљамо процесима и објектима, да излажемо значење и сврху. Управо зато, моделирање се најчешће посматра као најзначајније концептуално средство које човеку стоји на располагању (*Rothenberg, 1989*).

Међутим, у пракси се често дешава да се користе неадекватни модели, што с једне стране умањује ефикасност, а с друге стране може имати катастрофалне последице било у технолошком, социјалном или историјском погледу. Имајући у виду чињеницу да процес моделирања "лежи" у основи сваке рачунарске симулације, неопходно је дефинисати овај процес са више пажње.

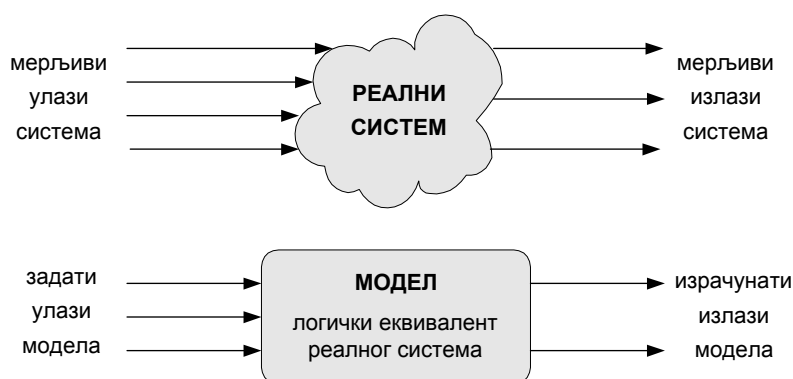
У најширем смислу, моделирање представља исплативо (у смислу трошкова) коришћење нечега (модел) уместо нечега другог (реални систем) са циљем да се дође до одређеног сазнања. Резултат моделирања је модел. Модел је апстракција реалности у смислу да он не може да обухвати све њене аспекте. Модел је упрошћена и идеализована слика реалности. Он нам омогућава да се суочимо са реалним светом (системом) на поједностављен начин, избегавајући његову комплексност и иреверзибилност, као и све опасности (у најширем смислу те речи) које могу проистећи из експеримента над самим реалним системом. Другим речима, модел је опис

реалног система са свим оним карактеристикама које су релевантне из нашег угла посматрања. То заправо значи да у процесу моделирања морамо извршити избор између оних елемената и карактеристика система које су од значаја за наше истраживање и које ће бити обухваћене моделом и преосталих, за нас ирелевантних, које наш модел неће садржати. Стога и кажемо да модел представља упрошћену слику реалног система, те као такав не садржи само објекте и атрибуте реалног система, већ и одређене претпоставке о условима његове валидности. Циљ модела није, наравно, да прецизно репродукује стварност у свој њеној сложености. Његов је циљ да уобличи на видљив, често формалан начин, оно што је суштинско за разумевање неког аспекта његове структуре или понашања.

Модели су увек апстракције реалног система, због тога задржавају само оне карактеристике оригинала које су битне за сврху његовог изучавања. Било какав модел мора да остави по страни читав низ детаља који су иначе саставни део појаве која се анализира. Тако, на пример, један идеалан модел система набавке требало би да узима у обзир и чињеницу да је потражња, између осталог, и функција времена (у метеоролошком смислу), које је опет функција активности сунца. Тешко је међутим претпоставити да би било који економски користан и оправдан модел узео у обзир и ову чињеницу (*Bratley, Fox & Scharge, 1987*).

Ниво апстракције у процесу моделирања утиче на валидност модела, односно на успешност представљања реалног система моделом. Проблем валидације модела јавља се у сваком процесу моделирања, а проистиче из чињенице да је модел увек поједностављени поглед на реални систем који је предмет посматрања. Сувише сложени или савршени модели који имају способност да за исти скуп улазних величина производе исте излазне вредности као и реални системи, чак иако су оствариви, по правилу су прескупи и неадекватни за експериментисање. С друге стране, сувише поједностављени модели не одсликавају на прави начин посматрани систем, а резултати који се добијају њиховом применом могу да буду неадекватни и погрешни. Стога, опредељујући се за ниво апстракције у посматрању реалног система, потребно је у одређеном тренутку повући границу у реалном систему и то тако да резултујући модел што верније одсликава посматрани систем, али и да, с друге стране, његова сложеност и цена не буду ограничавајући фактори.

Посматрајући однос реалног система и једног његовог модела са слике 1.1, можемо извести закључак да приказани модел није савршен у смислу како је то претходно објашњено. Са слике се јасно запажа граница која је "повучена" приликом креирања модела (нису узети у обзир сви мерљиви улази референтног система; модел даје излазе који се разликују од излаза реалног система).



Слика 1.1 Реални систем и модел посматрани као "црна кутија"

### 1.1.1 Врсте модела

За представљање система користе се различити модели, као што су: ментални (мисаони), вербални, структурни, физички, аналогни, математички, симулациони, рачунарски и разни други модели. Често их делимо на материјалне (модел хемијске структуре молекула или модел авиона) и симболичке моделе (математички, концептуални, рачунарски, симулациони идр).

Ментални модели су структуре које људски мозак непрекидно конструише како би био у стању да повеже низ чињеница са којима се човек сусреће, а потом на основу тога делује. Такви модели омогућују, на пример, разумевање физичког света, комуникацију међу људима и планирање акција. Истраживање природе и развој технологије довели су до стварања опипљивијих и формалнијих врста модела, првенствено због потребе да се омогући описивање сложених феномена како би се они могли прецизније решавати. У менталним процесима, различити концепти које јединка поседује

постављају се у различите нове односе. Концепт који јединка носи нису реалан свет, а закључци које ствара на основу њих су закључци формулисани помоћу модела.

Вербални модели су директна последица менталних модела и представљају њихов израз у говорном језику, а уобичајено се представљају у писаном облику. Вербални модели спадају у класу неформалних модела.

Физички модели представљају умањене моделе реалног система, који се понашају на исти начин као и њихови оригинали. Углавном се праве на основу теорије сличности или, у бољем случају, на основу физичких закона сличности.

Уколико су везе између објеката модела описане математичким (нумеричким) релацијама, тада се ради о математичким моделима. Математички модел је из класе апстрактних. Код формулисања математичког модела полази се од вербалног модела који се трансформисањем доводи у стање које се може описати математичким језиком. Ова класа модела има широку примену, нарочито у науци и инжењерским дисциплинама.

Одавно је уочена чињеница да се различити физички објекти могу описивати истим математичким моделом. Између два физичка модела који имају исте математичке моделе, каже се да постоји математичка аналогија (*Пејовић и Парезановић, 1981*). Истоветност математичких модела двају објеката пружа могућност да један од физичких објеката буде коришћен за анализу математичког модела другог објекта. Физички објекат који се користи за анализу математичког модела другог објекта, а са којим има исти или сличан математички модел, назива се аналогни модел. Према томе, између физичког објекта који се испитује и аналогног модела постоји математичка аналогија (аналогија у понашању).

Концептуални модели се стварају на основу представе о структури и логици рада система или проблема који се моделира и приказују се у облику чије је значење прецизно дефинисано (на пример, дијаграми са тачно дефинисаним симболима). Њихова посебна важност проистиче из чињенице да они представљају основу за израду рачунарских модела. Ови се модели често називају и структурни, пошто у графичком облику указују на структуру (релативно стабилан однос елемената) посматраног система. Такав приказ омогућује да се модели визуализују и на тај начин постају згодно средство за комуникацију међу људима који са њима раде.

Поред тога, графички приказ модела обезбеђује релативно једноставан приказ сложених система и несумњиво значи важан корак ка бољем разумевању система који се моделира.

Рачунарски (симулациони) модели су приказ концептуалних модела у облику програма за рачунар. У том облику модели постају средство којим се може ефикасно анализирати рад модела у различитим спољним условима и са различитим унутрашњим параметрима и тако добити увид у понашање система који модел описује. Као средство за изражавање, рачунарски модели користе програмске језике и стога су блиско везани за развој рачунарских наука.

### 1.1.2 Неформални и формални модели

Неформални опис модела даје основне појмове о моделу и, мада се тежи његовој потпуности и прецизности, он то најчешће није. Приликом изградње неформалног описа, управо ради елиминисања поменутих недостатака, врши се подела на објекте, описне променљиве и правила интеракције објеката (*Zeigler, 1976*).

Објекти су делови из којих је модел изграђен; описне променљиве (преко вредности које узимају) описују стања у којима се објекти налазе у одређеним временским тренуцима (параметри помоћу којих се описују константне карактеристике модела су такође укључени у описне променљиве); правила интеракције објеката дефинишу како објекти модела утичу један на други у циљу промене њиховог стања. Треба нагласити да не постоји правило за избор објеката, описних променљивих и правила интеракције. Тај избор је препуштен ономе ко описује и изграђује модел и треба га вршити тако да се добије валидан модел.

Неформални опис модела припрема се доста брзо и лако, али он најчешће није конзистентан и јасан, нарочито када су у питању сложени модели. Аномалије које се јављају приликом неформалног описа модела најчешће се могу описати на следећи начин (*Раденковић и Марковић, 1992*):

- ◆ Некомплетан опис модела
- ◆ Неконзистентан опис модела
- ◆ Нејасан опис модела



Уколико модел не садржи све ситуације које могу да наступе, тада је опис некомплетан. Уколико су у опису модела за исту ситуацију предвиђена два или више правила чијом се применом добијају контрадикторне акције, тада је опис неконзистентан. На крају, ако у једној ситуацији треба обавити две или више акција, а при томе није дефинисан њихов редослед, тада је опис модела нејасан.

С друге стране, савремена методологија моделирања у великој мери се ослања на одређене конвенције у комуницирању, зване формализми. Формализам специфицира класу посматраних објеката на недвосмислен и генералан начин, коришћењем конвенција и правила.

Формални опис модела треба да обезбеди већу прецизност и потпуност у описивању модела, а понекад омогућава и да се формализује поступак испитивања некомплетности, неконзистентности и нејасности. Оно што је, ипак, најзначајније јесте чињеница да увођење формализама у методологију моделирања омогућава да сву своју пажњу усмеримо на оне карактеристике објеката које су од највећег значаја за наше истраживање, дакле да користимо апстракције.

Ако засебно посматрамо област научног рада, може се лако запазити да се он у великој мери, а вероватно и у целини, састоји од формализација и изградњи модела. Формални модел или апстрактно представљање често се спомиње као кључ успеха интеракције човек - реални свет која се заснива на "научно-инжињерском" приступу (*Spriet i Vansteenkiste 1982*). Важно је напоменути да овако посматрани тип интеракције човек-реални систем обухвата два различита корака (слика 1.2). Прво, постоји увек фаза изградње модела или формализација, где се као резултат јавља модел реалног система. У другој фази, формални модел се анализира и користи, а на основу добијених резултата доносе се одлуке које обезбеђују ефикасније управљање самим реалним системом.

На крају, треба нагласити и чињеницу да не постоје стриктна правила везана за процес израде модела, већ велико значење има здрав разум, способност апстракције, систематичност и искуство. Дуго искуство великог броја људи који су се бавили моделирањем довело је до неких општих препорука при изради модела (*Gordon, 1969*):

1. Граница система са околином мора бити одабрана тако да систем, односно његов модел, обухвата само феномене од интереса. Околина система моделира се тако да се не описују детаљи феномена и узрочна веза међу њима, већ се даје само њихов сажети приказ.
2. Модел не смеју бити сувише сложени нити детаљни, већ треба да садрже само релевантне елементе система - сувише сложене и детаљне моделе готово није могуће вредновати ни разумети.



Слика 1.2 Интеракција између моделара, рачунара и реалног система

3. Модел не сме сувише да поједностави проблем, нпр. избацавањем важних променљивих потребних за адекватан

опис система или сувише великим степеном агрегације компоненти система.

4. Модел је разумно раставити на више добро дефинисаних и једноставних модула с тачно одређеном функцијом, које је лакше и изградити и проверити.
5. У развоју модела препоручује се коришћење неке од проверених метода за развој алгоритама и програма. Тако се нпр. у методи *top-down* полази од прве верзије модела која се састоји од неколико модула са више функција, затим се у следећем кораку ти модули растављају на неколико једноставнијих модула, итд., све док се не достигне жељени степен детаљности модела. При томе је могуће разумети модел и његове модуле у свим фазама развоја модела.
6. Потребна је провера логичке и квантитативне исправности модела, и то како појединачних модула тако и целог модела. Код модела који укључују случајне променљиве то значи и примену одговарајућих статистичких техника.

## 1.2 Рачунарска симулација

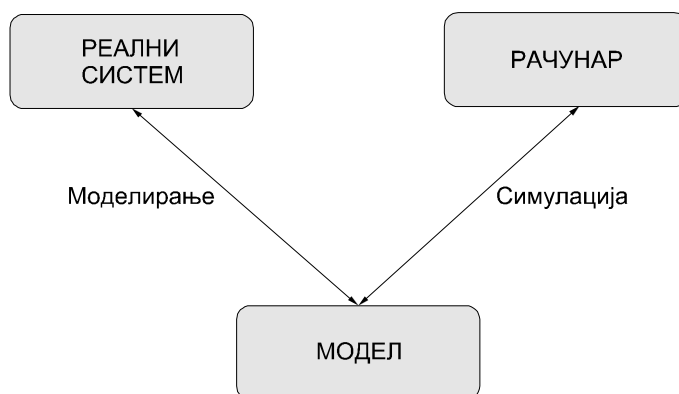
Да би модел био користан, од суштинске је важности то да се за дати ограничени скуп његових описних променљивих, његово понашање може одредити на практичан начин: аналитички, нумерички или путем експеримента, где се за извесне, углавном случајне улазе, посматрају одговарајући излази. Овај последњи процес назива се симулација.

Реч симулација у свакодневној употреби може да значи већи број различитих активности, као на пример: сложене видео игре, испитивање утицаја бројних фактора на лет нових модела авиона, део експеримента у социо-психолошким истраживањима итд. Када реч користе рачунарски стручњаци, организатори, менаџери или статистичари, обично под симулацијом подразумевају процес изградње апстрактних модела за неке системе или подсистеме реалног света и обављање већег броја експеримената над њима. Посебно нас интересује случај када се ти експерименти одвијају на рачунару. Тада говоримо о рачунарском моделирању и симулацији.

### 1.2.1 Моделирање и симулација

Савремено моделирање незамисливо је без рачунара. У моделирању рачунари се користе у две сврхе: у развоју модела и у извођењу прорачуна на основу створеног модела (нпр. прорачуна понашања модела у времену код симулационих модела). На тај начин, моделирање помоћу рачунара постаје дисциплина којом се могу адекватно и ефикасно приказивати сложени системи и обликовати и испитивати њихово понашање (Черић, 1993).

Израз моделирање и симулација изражава сложену активност која укључује три елемента: реални систем, модел и рачунар. Ова се активност на упрошћен начин може представити дијаграмом на слици 1.3 (Zeigler, 1976).



Слика 1.3 Релације моделирања и симулације

Под реалним системом подразумевамо уређен, међузависан скуп елемената који формирају јединствену целину и делују заједнички како би остварили задати циљ или функцију, без обзира да ли се ради о природном или вештачком систему, и такође, без обзира да ли тај систем у посматраном тренутку постоји или се његово постојање планира у будућности. Реални систем је извор података о понашању, а ови се подаци јављају у облику зависности  $X(t)$ , где је  $X$  било која променљива која интересује истраживача, а  $t$  је време мерено у одговарајућим јединицама. Другим речима, реални систем се може посматрати као извор података за спецификацију модела.

Модел, као и сваки реални систем, има своје објекте који се описују атрибутима или променљивим. Он је апстрактни приказ система и даје његову структуру, његове компоненте и њихово узајамно деловање. С обзиром да се за симулацију најчешће користи рачунар (разматрамо само тај случај), то се под моделом може подразумевати скуп инструкција (програм) који служи да се генерише понашање симулираног система (временска серија вредности променљивих симулираног система). Понашање модела не мора да буде у потпуности једнако понашању симулираног система, већ само у оном домену који је од интереса.

Рачунар као трећа компонента ове активности, представља уређај способан за извршење инструкција модела, које на бази улазних података генеришу развој модела у времену. Рачунари, уз различите методе и програмске алате, омогућавају погодан амбијент за стварање сложених модела и ефикасан рад над њима.

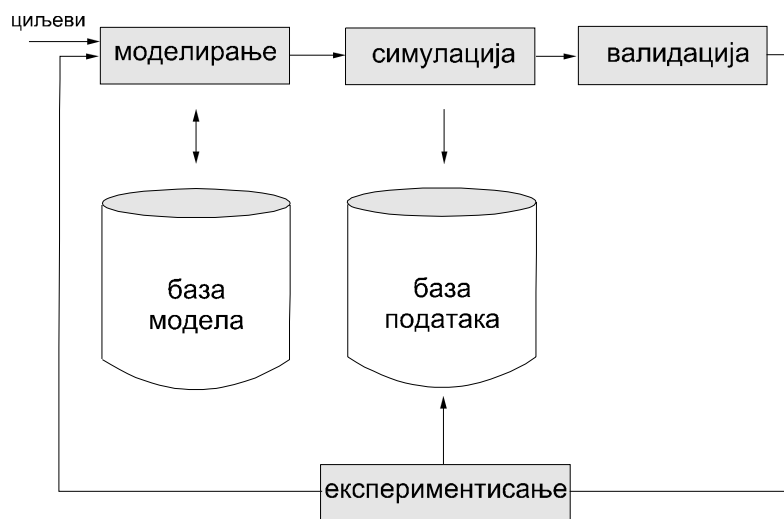
Међутим, поред ових елемената, пажњу треба усмерити и на откривање и дефинисање релација које постоје између њих. Моделирање је процес којим се успоставља веза између реалног система и модела, док је симулација процес који успоставља релацију између модела и рачунара.

Релација моделирања односи се на валидност модела. Валидност или ваљаност модела описује колико верно један модел представља симулирани систем. Процес утврђивања степена слагања података о реалном систему са подацима модела назива се валидација модела. Процес валидације је веома значајан, јер се на основу њега доносе одлуке о употребљивости резултата симулације, измени модела, измени података (улазних променљивих, параметара), даљем наставку симулације, понављању симулације, итд.

Релација симулације односи се на проверу да ли симулациони програм верно преноси модел на рачунар као и на тачност којом рачунар извршава инструкције модела. Пре поређења стварних података са подацима које генерише рачунар (симулатор), мора се утврдити тачност, односно коректност симулатора. Процес процене коректности симулатора назива се верификација.

На слици 1.4 приказане су активности процеса моделирања и симулације са базом модела као централним објектом (*Zeigler, 1984*). Процесом моделирања се управља на основу циљева који се генеришу ван граница система. Сваки нови циљ иницира активност

синтезе модела. При синтези модела се користи расположиво знање из базе модела и базе података. Ове базе чувају и организују прикупљене податке о реалном систему. Фазе симулације (експериментисање са моделом) и валидације следе фазу изградње модела.



Слика 1.4 Процес моделирања и симулације

Валидација води новом експериментисању над реалним системом и може да захтева додатне модификације или чак одбацивање и реиницијализацију првобитног модела. У том процесу, као резултат недостатка података у бази знања могу се формулисати нови или изменити постојећи циљеви. На крају, као резултат јавља се један или више модела који воде ка испуњењу екстерних циљева (уколико процес не "упадне" у петљу из које не може да изађе). Креиране моделе користи доносилац одлуке. Поред тога, они се могу меморисати у бази модела и користити у некој наредној фази активности.

### 1.2.2 Предмет моделирања и симулације

Рачунарска симулација има у основи модел система. Систем је уређај или процес који постоји или се планира. На пример, као што су:

- ♦ Производни погон са машинама, људима, транспортним уређајима, покретним тракама и складишним простором.
- ♦ Банка, пошта или нека слична услужна организација са различитим категоријама клијената, службеника и уређајима као што су шалтери, аутомати за новац, клупе за чекање, сигурносни сефови.
- ♦ Дистрибуциона мрежа која се састоји од фабрика (произвођача), складишта и транспортних средстава.
- ♦ Служба за хитне интервенције у болници са особљем, собама, инструментима, помоћним прибором и помагалима за транспорт болесника.
- ♦ Сервисна служба са потенцијалним клијентима који су географски дислоцирани, техничарима различитих квалификација, камионима са опремом и резервним деловима и централним магацином са одговарајућом службом.
- ♦ Рачунарска мрежа са серверима, клијентима, дисковима, принтерима, мрежним карактеристикама и операторима.
- ♦ Саобраћајни систем са сегментима саобраћајница, раскрсницама, сигнализацијом, возилима и пешацима.
- ♦ Предузеће за осигурање где стиже обимна документација која се сортира, прегледа, допуњава, копира и шаље.
- ♦ Систем правосуђа са судовима и судницама, судијама, помоћним особљем, референтима, адвокатима исл.
- ♦ Фабрика хемијске индустрије са танкерима за складиштење, цевоводима, реакторима, возилима за превоз.
- ♦ Ресторан “брзе хране” са радницима различитих задужења, гостима, опремом и снабдевањем.
- ♦ Самопослуга са производима, продавцима, купцима, магацином, поруџбинама, допремом робе.

### 1.3 Историјски преглед развоја симулације

Физичко моделирање људи су несвесно вршили још од постанка људске врсте, односно од тренутка када је човеков ум почео да схвата сву сложеност појава и ствари које су га у природи окруживале.

Моделирање и симулација добијају научни смисао тек појавом првих дигиталних рачунара. Тада се процес моделирања и процес симулације формализују. Рачунар се том приликом користи као помоћно средство како у фази изградње модела тако и у самом симулационом експерименту. Данас, када улога рачунара постаје све значајнија у многим областима људског деловања, отварају се нове перспективе за примену метода симулације и јачање њене научне основе.

Историјску перспективу развоја моделирања и симулације можемо укратко приказати табелом 1.1 (*Spriet i Vansteenkiste, 1982*):

Табела 1.1 Историјски преглед развоја симулације

1600.	Физичко моделирање
1940.	Појава електронских рачунара
1955.	Симулација у авио – индустрији
1960.	Симулација производних процеса
1970.	Симулација великих система укључујући економске, друштвене и еколошке
1975.	Системски приступ у симулацији
1980.	Симулација дискретних стохастичких система и виши ниво учешћа у системима за подршку одлучивању
1990.	Интеграција рачунарске симулације, вештачке интелигенције, рачунарских мрежа и мултимедијалних технологија



Реч симулација, у садашњем значењу, води порекло из радова *Von Neumann-a* и *Ulam-a* с краја четрдесетих година. Они су решавајући одређене проблеме веће сложености, установили да се резултати не могу добити аналитичким путем, а извођење непосредних експеримената било би веома скупо, те су приступили коришћењу Монте Карло технике. На тај начин, долазили су до математичких решења детерминистичких проблема симулацијом стохастичких процеса, који су имали расподелу вероватноће која задовољава математичке релације датог детерминистичког проблема.

Модерни токови у симулацији окренути су ка коришћењу формализама дискретних догађаја као преовлађујућем формализму моделирања последњих деценија двадесетог века. При томе се инсистира на томе да интеракција корисника и симулационог система буде једноставна, а спецификација модела користи концепте за које постоји одговарајући еквивалент у реалном систему (*Banks i Carson, 1984*).

## 1.4 Карактеристике симулационог моделирања

Рачунарска симулација је процес решавања проблема који се тиче предвиђања и одређивања будућих стања реалног система на основу проучавања рачунарског модела тог система (*Widman i Loparo, 1989*). Другим речима, рачунарска симулација се заснива на идеји експериментисања са моделом реалног система на рачунару, током времена.

Симулациони експерименти најчешће се изводе са циљем да се прикупе одређене информације, чије би добијање путем експеримента над самим реалним системом било непрактично или сувише скупо. Те информације касније се трансформишу у одлуке значајне за управљање реалним системом који је предмет симулационог моделирања. Циљ симулације јесте тај да проучимо понашање система који симулирамо, али и да установимо како би се исти систем понашао када би на њега деловао неки други скуп променљивих околности (улазних величина и параметара).

За разлику од аналитичких модела који свеукупно понашање система третирају директно, симулациони модели прикупљају податке о променама стања система и излаза, фокусирајући се на понашање индивидуалних компоненти система (*Schmidt, 1984*). Значај симулационих модела проистиче из чињенице да се само

мали број комплексних реалних система може адекватно описати преко аналитичких једначина.

Код примене симулационог моделирања, не може се добити решење у аналитичком облику, у којем су зависне променљиве функције независних променљивих, већ се решење проблема добија експериментисањем над моделом. При томе, симулациони експерименти дају као резултат скуп тачака, тј. вредности зависних променљивих за поједине вредности независних променљивих (време). Због случајног карактера променљивих модела, добија се чак и више различитих вредности зависних променљивих за исту вредност независних променљивих, тј. експерименти дају одређени узорак вредности зависних променљивих. При томе, планирање и анализа симулационих експеримената захтевају статистички приступ.

Симулациони модели најчешће су модели динамичких система, тј. система који се мењају у времену, с обзиром на то да су истраживачи, у већини случајева, заинтересовани за симулацију модела динамичких система. Ови су модели углавном дати у облику концептуалних (структурних) и рачунарских модела. Постоји, међутим, и група стационарних проблема за чије је решавање симулација такође интересантна. Најчешће, али не и генерално, симулациони модели су модели система који се не могу описати нити решавати математичким средствима. Њихове типичне примене су у областима инжењерства, менаџмента и економије, али све више и у медицини, биологији и другим природним наукама.

Изградња и коришћење симулационих модела је процес који захтева знатну вештину и добро познавање бројних научних дисциплина. По својој природи он је блиско везан за алате и технике рачунарских наука и системске анализе. Такође, процес симулације се ослања и на методе операционих истраживања и нумеричке анализе. Због постојања случајних променљивих у симулационим моделима, често се користе и приступи теорије вероватноће и статистике. Експериментална природа метода решавања проблема у симулационом моделирању има аналогije и са емпиријском природом истраживања у природним наукама, будући да се у оба подручја користи планирање, извођење и анализа експеримената ради разумевања рада система који се испитује, односно провере постављених хипотеза. Но, без обзира на свој мултидисциплинарни приступ, симулација данас представља кохерентну и добро развијену област и с правом носи назив засебне научне дисциплине.

## 1.5 Потреба за симулацијом

Може се поставити питање због чега се уопште један систем (симулирани систем) замењује моделом, а затим врши симулација. Постоји више разлога, али су најважнији следећи (*Naylor i dr., 1966*):

- ◆ Експеримент над реалним системом може да буде скуп или чак немогућ (на пример, у економским системима или хемијским постројењима).
- ◆ Аналитички модел нема аналитичко решење (нпр., сложенији модели масовног опслуживања)
- ◆ Систем може да буде сувише сложен да би се описао аналитички (на пример, системом диференцијалних једначина).

Поред ових, навешћемо још неколико значајних разлога:

- ◆ Експериментисање са реалним системом, чак и ако се занемаре други аспекти, углавном је неисплативо или сувише сложено. Моделирање, с друге стране, може да укаже на то да ли је даље улагање у експеримент економски оправдано или не.
- ◆ Изградња модела и симулација понекад имају за циљ да се схвати функционисање постојећег система чија је структура непозната и не може јој се прићи.
- ◆ Приликом изналажења оптималног функционисања неког система, уобичајено је да се мењају разни параметри. Често је то неизводљиво са реалним системом, било зато што таквог система уопште нема (тек га треба градити) или зато што би такав експеримент био прескуп. Тада су градња модела и његова симулација могуће решење.
- ◆ Понекад треба симулирати услове под којима наступа разарање система. Наравно, разарање реалног система најчешће није допустиво. У таквим приликама, симулација представља право решење.

- ♦ Време може да буде врло јак разлог да се прибегне симулацији. При симулацији, време се може сажети. То је значајно код симулације дуготрајних процеса. У другим случајевима, време се може знатно продужити. На пример, са циљем да се прати постепено одвијање врло брзих процеса, што је у реалном ситему немогуће.
- ♦ Када се врши реални експеримент, увек постоји извесна грешка при мерењу услед несавршености мерних уређаја. При симулацији ове грешке нема. Постоји само грешка "заокруживања" услед коначне дужине речи у рачунару, али се она са мало труда може учинити занемарљивом.
- ♦ Понекад је пожељно зауставити даље одвијање експеримента, како би се испитале вредности свих променљивих у том тренутку. Ово је тешко могуће у реалном систему.

#### 1.5.1 Када је могуће експериментисати на систему?

- ♦ Неки градови имају на улазним саобраћајницама инсталирану светлосну сигнализацију и могуће је експериментисати са сигналним плановима како би се систем подесио, тако да проток саобраћаја буде што већи и безбеднији у време јутарњих или поподневних шпицева.
- ♦ Менаџер самопослуге може да испроба различите начине управљања набавком и расподелом задатака запосленима како би дошао до комбинације која пружа најбољу услугу и доноси највећи профит.
- ♦ У рачунарској мрежи могуће је експериментисати са различитим мрежним параметрима и приоритетима за *job* – ове да би се сагледало како они утичу на искоришћеност уређаја и брзину рада.

#### 1.5.2 Када није могуће експериментисати на систему ?

- ♦ Немогуће је експериментисати са алтернативним производним програмом фабрике која не постоји.

- ♦ Чак и у случају да фабрика постоји, било би веома скупо прећи на неки експериментални производни програм који можда неће дати добре резултате.
- ♦ Тешко је “угурати” у банку или пошту два пута више клијената него што је уобичајено како би се испитало шта се дешава у систему када се број клијената приближи граници функционисања система.
- ♦ Увођење нове неиспитане процедуре за предају пртљага на аеродрому могло би изазвати такве гужве, да велики број путника пропусти своје летове.
- ♦ Истраживање ефеката примене нових процедура код пријема хитних случајева у болницу потпуно је немогуће.

## 1.6 Могућности примене симулације

Велике могућности модерних рачунара, снижење трошкова по операцији, напредак у методологији симулације и њено профилисање као засебне научне дисциплине, затим савремени, уско специјализовани симулациони језици, значајно су проширили подручје примене симулације. Питање које из тога проистиче, односи се на могућност примене симулације. Навешћемо неколико ситуација када се симулација може успешно применити (*Naylor i dr., 1966*):

- ♦ Симулација омогућава проучавање и експериментисање које узима у обзир свеукупне интеракције сложеног система или подсистема унутар сложеног система.
- ♦ Информационе и организационе промене или промене у окружењу могу се симулирати, а уједно се могу посматрати ефекти тих промена на понашање модела.
- ♦ Знање стечено у процесу изградње модела и симулације може бити од великог значаја код побољшања система који се испитује.
- ♦ Мењањем симулационих улаза и посматрањем резултујућих излаза, долазимо до важног сазнања о томе које су

променљиве система најважније и како променљиве утичу једна на другу.

- ♦ Симулација се може користити и као педагошко средство са циљем да побољшава методологије аналитичких решења.
- ♦ Симулација се може користити за експериментисање са новим концепцијама или политикама пре него што се изврши њихова имплементација.
- ♦ Симулација се може користити за верификацију аналитичких решења.

## 1.7 Предности и недостаци симулације

Бављење симулационим методама и техникама захтева, између осталог, и познавање њених предности и недостатака. Као основне предности коришћења симулације наводе се следеће (*Schmidt i Taylor, 1970*):

- ♦ Једном изграђени модел може се вишеструко користити за анализу предложених планова или политика.
- ♦ Симулационе методе могу се користити као помоћ код анализе, чак иако су улазни подаци на неки начин непотпуни.
- ♦ Чест је случај да се симулациони подаци могу много јефтиније добити од сличних података из реалног система.
- ♦ Симулационе методе лакше је применити него аналитичке методе. Стога је круг потенцијалних корисника симулационих метода знатно шири.
- ♦ Аналитички модели углавном захтевају више поједностављујућих претпоставки које их чине математички прилагодљивим. Симулациони модели таква ограничења немају. Са аналитичким моделима, најчешће се може израчунати једино ограничени број мерљивих карактеристика система, док код симулационих модела генерисани подаци могу да се користе за процену било које схватљиве и мерљиве карактеристике.

- ♦ У неким случајевима, симулација је једино средство за решавање одговарајућег проблема.
- ♦ Могуће је описати и решавати сложене динамичке проблеме са случајним променљивим који су недоступни математичком моделирању.

У основне недостатке коришћења симулације убрајају се следећи (*Law i Kelton, 1982; Schmidt i Taylor, 1970*):

- ♦ Симулациони модели за дигиталне рачунаре могу бити скупи и могу захтевати значајно време за изградњу и валидацију.
- ♦ Због статистичког карактера симулације потребно је извођење већег броја симулационих експеримената како би се добио одговарајући узорак резултата симулације, а већ и појединачно извођење експеримента може захтевати доста времена и меморије рачунара.
- ♦ Не добијају се зависности излазних променљивих од улазних променљивих модела нити оптимална решења.
- ♦ За исправно коришћење симулационог моделирања потребно је познавање више различитих метода и алата.
- ♦ Вредновање модела је доста сложено и захтева додатне експерименте.

## 1.8 Популарност симулације

- ♦ *Rasmunssen i George (1978)* анкетирали су свршене постдипломце Case Western универзитета (смер за операциона истраживања) о методама које су користили. Прве четири методе по резултатима биле су статистичка анализа, предвиђање, систем анализа и информациони системи, све четири веома широке и опште категорије. На петом месту била је симулација, рангирана далеко испред већине традиционалних метода операционих истраживања (линеарно програмирање, теорија масовног опслуживања идр).

- ♦ *Thomas i DaCosta (1979)* спровели су истраживање међу аналитичарима 137 великих фирми. На листи која је садржала алате и методе, требало је означити оне које користе. Статистичка анализа била је на првом месту (користе је у 93% фирми), а на другом месту се нашла симулација (84% фирми).
- ♦ *Shannon, Long i Buckles (1980)* анкетирали су чланове одељења за операциона истраживања америчког Института индустријских инжењера и закључили да се од свих понуђених алата, симулација рангира на првом месту према интересовању и коришћењу, а на другом месту према приступачности (иза линеарног програмирања).
- ♦ *Forgionne (1983); Harpell, Lane i Mansour (1989); Lane, Mansour i Harpell (1993)* саопштавају да је по коришћености метода, међу аналитичарима у великим корпорацијама, статистичка анализа на првом, а симулација на другом месту.
- ♦ *Morgan (1989)* је извео већи број истраживања овог типа и закључује да се готово свуда запажа изузетно велико коришћење симулације; чак и у делу индустрије са најмањим учешћем метода операционих истраживања, симулација је на првом месту.

## 1.9 Симулациони процес

Симулациони процес је структура решавања стварних проблема помоћу симулационог моделирања. Он се може приказати у облику низа корака који описују поједине фазе решавања проблема овом методом (животни циклус симулације). Структура симулационог процеса није строго секвенцијална, већ је могућ и повратак на претходне кораке процеса, зависно од резултата добијених у појединим фазама процеса. Број фаза и редослед њиховог обављања зависи од сваке конкретне ситуације, али је ипак могуће навести један општи, уређен скуп процедура (слика 1.5).

Основни кораци симулационог процеса су следећи (*Law i Kelton, 1982*):



### *1. Дефиниција циља симулационе студије*

Дефиниција жељеног циља и сврхе студије: проблем који треба решити (обликовање система, анализа система и сл.), границе систем/околина, ниво детаљности.

### *2. Идентификација система*

Опис компоненти система, интеракција компоненти, начин рада, везе с околином, формални приказ система.

### *3. Прикупљање података о систему и њихова анализа*

Прикупљање и мерење релевантних података о систему, анализа тих података (избор расподела независних случајних променљивих, оцена вредности параметара расподела).

### *4. Изградња симулационог модела*

Стварање концептуалног модела који адекватно описује систем и омогућава решавање задатог проблема.

### *5. Изградња симулационог програма*

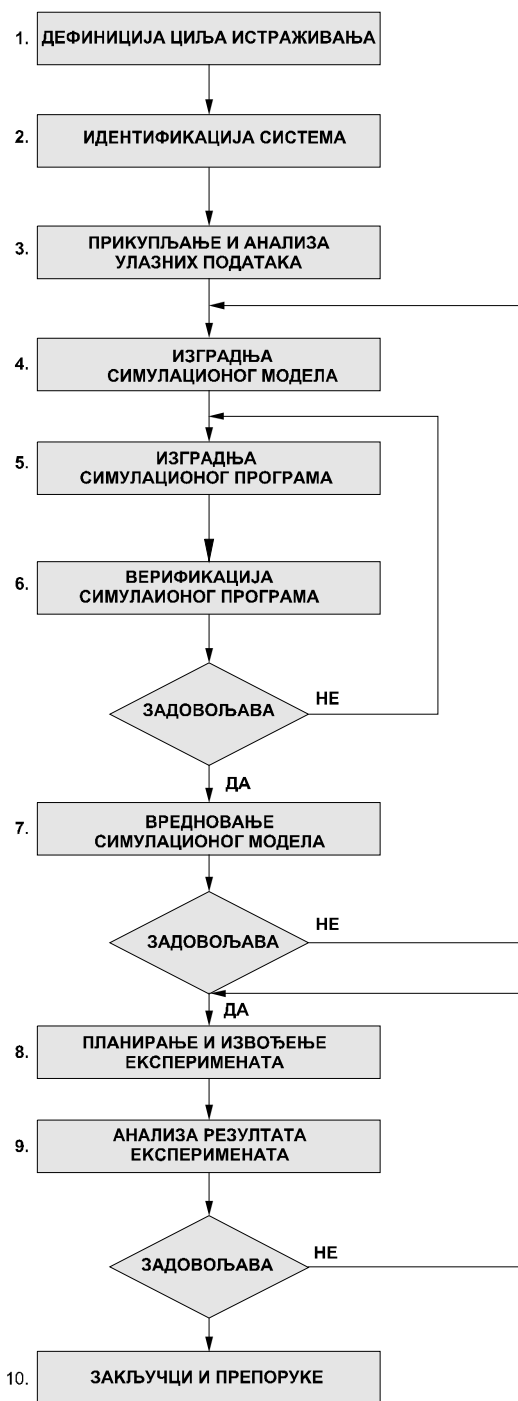
Избор програмског језика или пакета и стварање симулационог програма било писањем програма, било аутоматским генерисањем програма на основу концептуалног модела.

### *6. Верификација симулационог програма*

Тестирање симулационог програма према поставкама симулационог модела (појединачне процедуре, генерисање случајних променљивих, повезивање процедура). Уколико верификација програма није дала задовољавајуће резултате, потребан је повратак на корак 5.

### *7. Вредновање (валидација) симулационог модела*

Испитивање да ли симулациони модел адекватно представља стварни систем (испитивањем подударности излаза модела и реалног система, анализом резултата од стране експерата, анализом осетљивости). Уколико вредновање модела није успешно, потребно је вратити се на тачку 4 (измене у симулационом моделу).



Слика 1.5 Дијаграм тока симулационог процеса

### 8. Планирање симулационих експеримената и њихово извођење

Планирање симулационих експеримената који омогућују испуњење циља студије (план промене параметара модела, понављање експеримента због анализе утицаја случајних променљивих, идр). Извођење симулационих експеримената према усвојеном плану.

### 9. Анализа резултата експеримената

Статистичка анализа резултата симулационих експеримената. Током анализе резултата може се показати потреба за допуном корака 8 (извођење додатних експеримената).

### 10. Закључци и препоруке

Презентација релевантних резултата на основу којих се могу донети одговарајуће одлуке (избор конфигурације система, измене у систему, идр.).

## 1.10 Поделе симулационих модела

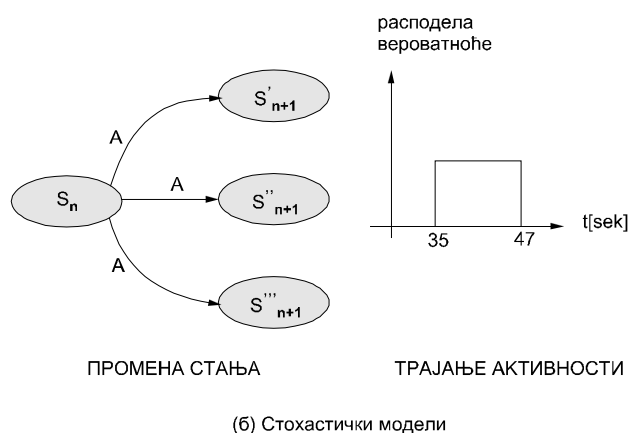
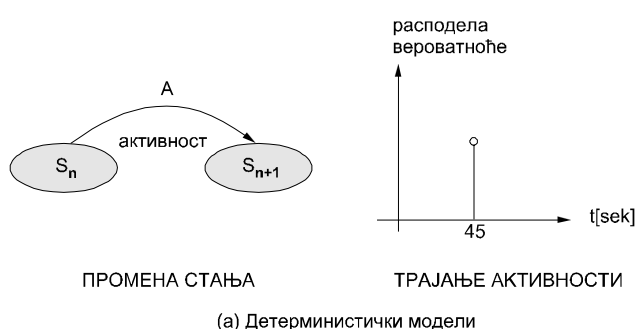
Разликујемо два основна начина поделе симулационих модела: први, према врсти променљивих у моделу и други, према начину на који се стање у моделу мења у времену.

### 1.10.1 Детерминистички и стохастички модели

Детерминистички модели су они чије се понашање може предвидети, односно у којима је ново стање система у потпуности одређено претходним стањем. На слици 1.6-а приказан је детерминистички модел у којем се стање система  $S_n$  мења под утицајем активности  $A$  у стање  $S_{n+1}$  (Graybeal i Pooch, 1980). Приказана је такође и активност  $A$  са детерминистичким трајањем од 45 секунди, нпр. трајање прегледа једне компоненте одређеног производа.

Стохастички модели су они чије се понашање не може унапред предвидети, али се могу одредити вероватноће промена стања система. Дакле, за стохастичке моделе је карактеристично случајно

понашање, односно постојање случајних променљивих у систему. На слици 1.6-б приказан је стохастички модел у коме се стање система  $S_n$  може променити у једно од стања  $S'_{n+1}$ ,  $S''_{n+1}$  или  $S'''_{n+1}$  под утицајем активности А. Тако на пример, након извршеног прегледа, компонента може бити одбачена или се може уградити у финални производ, при чему свако од тих стања има одређену вероватноћу.



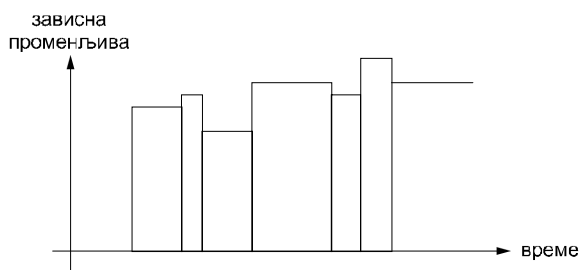
Слика 1.6 Детерминистички (а) и стохастички (б) модели

Други извор случајности, приказан на истој слици, је активност са случајним трајањем, односно трајањем које се одређује из униформне расподеле вероватноће. Пример такве активности је точење горива на бензинској пумпи, где се мерењем одређеног узорка точења горива возилима добијају подаци из којих се може

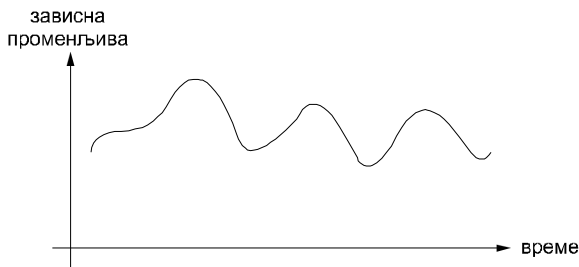
одредити расподела вероватноћа трајања те активности (Черић, 1993).

### 1.10.2 Дискретни и континуални модели

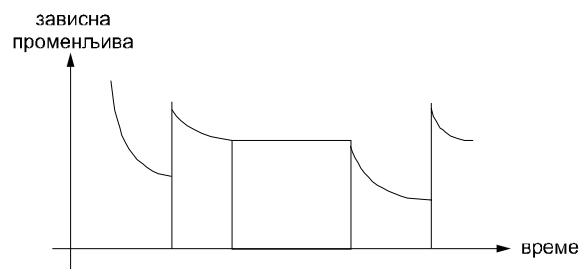
У дискретним моделима стање система се мења само у појединим тачкама у времену (нема континуалне промене стања). Такве промене се називају догађаји. Дискретна (дисконтинуална) промена стања приказана је на слици 1.7-а (Pritsker, 1984).



(а) Дискретни модел



(б) Континуални модел



(в) Мешовито континуално-дискретни модел

Слика 1.7 Дискретни, континуални и мешовити симулациони модели

На пример, у моделу самопослуге могу постојати променљиве које описују број купаца у редовима пред касама, а тај се број може мењати само у тренутку доласка купаца у ред и у тренутку почетка опслуживања на каси.

У континуалним моделима променљиве стања се мењају континуално у времену, као што је приказано на слици 1.7-б. Пример континуалне промене је лет авиона чији се положај и брзина мењају континуално у времену. Треба имати у виду да се на дигиталним рачунарима не могу изводити континуалне промене величина, већ се оне морају апроксимирати скупом дискретних вредности.

Могући су и мешовити, континуално-дискретни модели који садрже и континуалне и дискретне променљиве. Промена стања таквог модела приказана је на слици 1.7-в.

## 1.11 Врсте симулационих модела

Приказане поделе симулационих модела довеле су до формулисања четири основне врсте симулационих модела, који се разликују, с једне стране, по приступу моделирању и класи проблема који се решава, и с друге стране, по техникама моделирања и симулације које су за њих развијене (*Law i Kelton, 1982; Evans, 1988*). То су:

- ◆ Монте Карло (*Monte Carlo*) симулација
- ◆ Континуална симулација
- ◆ Симулација дискретних догађаја
- ◆ Мешовита, континуално-дискретна симулација.

Осим Монте Карло симулације, која је статичка, све остале набројане врсте су динамичке. Монте Карло симулација споменута је овде из разлога што су у њој третман случајних догађаја и генерисање случајних вредности блиски оном из симулације дискретних догађаја.

У наставку су укратко описане све набројане врсте симулационих модела, док су модели континуалне симулације и модели са дискретним догађајима, детаљније разматрани у посебним поглављима.

### 1.11.1 Монте Карло симулација

Монте Карло симулација (статистичка симулација), као што јој и име каже, повезана је са случајним феноменима. Ова метода је развијена у САД током Другог светског рата, али и данас још увек нема потпуне сагласности о коришћењу тог термина. Неки аутори Монте Карло симулацијама називају било коју врсту програма који се користе случајним бројевима. С друге стране, у већини литературе о симулационом моделирању, овај термин се употребљава само за статичке типове симулације код којих се у решавању проблема користи стварање узорака из расподела случајних променљивих. При томе, проблеми могу бити било детерминистичког, било стохастичког карактера.

Разликујемо следеће типове примене Монте Карло симулације (Kleijnen, 1974).

#### 1. *Детерминистички проблеми које је тешко или скупо решавати*

Типичан пример овога типа је рачунање вредности одређених интеграла који се не могу решити аналитички, тј. чија је подинтегрална функција таква да се не може наћи решење у облику аналитичког израза. Монте Карло симулација приступа прорачуну интеграла тако што се генерише низ случајних тачака  $(x_j, y_j)$  са једнаким вероватноћама унутар одређеног правоугаоника и потом испитује колико је генерисаних тачака унутар површине која одговара интегралу. Овакав приступ, који се заснива на генерисању случајних бројева, аналоган је оном који се користи код симулације система са дискретним догађајима.

#### 2. *Сложени феномени који нису довољно познати*

Ово је друга класа проблема који се решавају Монте Карло симулацијом. За њих је карактеристично да није познат начин узајамног деловања између елемената већ су познате само вероватноће његовог исхода, које се користе за извођење низа експеримената који дају узорке могућих стања зависних променљивих. Статистичком анализом таквих узорака добија се расподела вероватноћа зависних променљивих које су од интереса. Овакав приступ се најчешће примењује код анализирања друштвених или економских феномена.

### 3. Статистички проблеми који немају аналитичка решења

Статистички проблеми без аналитичког решења (нпр. процене критичних вредности или тестирање нових хипотеза) су једна специфична класа проблема који се решавају Монте Карло симулацијом. Приликом решавања таквих проблема такође се користи генерисање случајних бројева и променљивих.

#### 1.11.2 Континуална симулација

Континуална симулација се користи за динамичке проблеме код којих се променљиве стања мењају континуално у времену. Постоје две основне класе проблема који се решавају овом методом.

У првој класи су релативно једноставни проблеми који су описани детаљно и код којих су промене "глатке" и природно се описују диференцијалним једначинама. То су типично проблеми из физике, биологије и инжењерства. У другој класи су проблеми који настају описом веома сложених система у агрегираном облику, у којем се низ елемената система редукује на мањи број компоненти и у којима се промене у систему апроксимирају константним брзинама промене. То су најчешће проблеми из подручја економије и друштвених наука (Черић, 1993).

Разликујемо три основна типа континуалних симулационих модела (Aburdene, 1988).

##### 1. Модели који се описују обичним диференцијалним једначинама (системи обичних диференцијалних једначина)

Проблеми у вези са разним процесима као што су, на пример, разни облици кретања и многи физички, хемијски, биолошки и други процеси где се ради о једној непознатој функцији ( $y = y(t)$ ) једне независно променљиве ( $t$ ), изражавају се математички једначинама у којима се, поред независно променљиве и непознате функције, јављају (обавезно) и изводи те функције ( $dy/dt$ ). Тим једначинама се у сваком конкретном случају успоставља веза између непознате функције ( $y$ ), њене реалне променљиве ( $t$ ) и њених извода ( $y'$ ) и такве се једначине називају обичне диференцијалне једначине (Дајовић и Вујчић, 1981).



Модели који се описују обичним диференцијалним једначинама или њиховим системима могу се у одређеним случајевима решити аналитички, али је то доста ретко код стварних проблема (уобичајено је да ове једначине узимају облик сложених нелинеарних диференцијалних једначина које се по правилу не могу решити аналитички). Због тога се за такве једначине морају користити нумеричке методе. Нумеричке методе могу се применити и на аналитички решиве једначине, али се то у пракси избегава, јер је у већини случајева за такве једначине једноставније аналитичко решење (Пејовић, 1983). Нумеричке методе које служе за решавање диференцијалних једначина код којих се као независна променљива јавља време, уобичајено се називају методе континуалне симулације.

## 2. *Модели који се описују системима парцијалних диференцијалних једначина*

Парцијалне диференцијалне једначине садрже више од једне независне променљиве ( $x_j$ ) по којима се траже изводи зависне променљиве. Типични примери проблема који се могу описати парцијалним диференцијалним једначинама су проблеми аеродинамике, хидродинамике и метеорологије.

## 3. *Модели динамике система (System Dynamics)*

Динамика система је методологија истраживања, моделирања и симулације сложених динамичких система. Системи са повратном везом су основни тип система који се моделирају динамиком система. Повратна веза може бити позитивна или негативна. Модел са повратном везом користе се најчешће за моделирање инжењерских, биолошких, друштвених и економских система.

Динамика система приказује системе као повезане управљачке петље. При томе су појединачни догађаји јако агрегирани, а последица тога је да се могу описивати као континуални токови описани диференцијалним једначинама (тј. коначним разликама, а не бесконачно малим величинама). Ново стање система у наредном тренутку времена рачуна се на основу стања у претходном тренутку времена и разлике улазних и излазних токова за то стање у претходном интервалу времена. Да би прорачун могао почети, неопходно је да буду задате почетне

вредности величина као и вредности свих константи и параметара које модел садржи.

### 1.11.3 Симулација дискретних догађаја

Симулација дискретних догађаја је специфична методологија симулације која се бави моделирањем система који се могу представити скупом догађаја. Под догађајем овде се подразумева дискретна промена стања ентитета система. Догађај наступа у одређеном тренутку времена, односно промене стања ентитета се дешавају дисконтинуално у времену, тј. само у неким временским тренуцима (када наступи догађај). Симулација описује сваки дискретни догађај, крећући се од једног догађаја до другог при чему настаје помак (прираст) времена симулације. Између два узастопна догађаја, стање система се не мења. Системи који се моделирају на овај начин су динамички и готово редовно стохастички (*Davies i O'Keefe, 1989*). У поглављу десет дат је шири опис методе симулације дискретних догађаја.

### 1.11.4 Мешовита симулација

Код појединих врста система, континуална симулација као и симулација дискретних догађаја, не могу у потпуности да опишу начин рада система. То су они системи који садрже процесе који теку континуално и догађаје који доводе до дисконтинуитета у понашању система. Да би се такви системи моделирали и симулирали, развијена је мешовита симулација која омогућава интегрисање континуалних и дискретних елемената система (*Pritsker, 1984*).

Веза између дискретног и континуалног приступа постиже се увођењем два типа догађаја. Временски догађаји су догађаји које генерише механизам управљања догађајима, какав постоји у симулацији дискретних догађаја. Они могу да изазову тренутну промену стања континуалне променљиве. С друге стране, догађаји стања су они догађаји које активира механизам помака времена са константним прирастом, чији је временски интервал мали, а који је карактеристичан за континуалну симулацију. Ови догађаји могу да активирају догађаје дискретног дела модела. Познатији симулациони језици за мешовиту симулацију су GASP и SLAM.

### 1.12 Избор типа симулационог модела

Тип симулационог модела најчешће се одабира тако да буде једнак типу оригиналног система. То ипак не значи да увек морамо дискретни ситем приказати дискретним моделом, односно континуални систем континуалним моделом. Избор типа симулационог модела зависи највише од циља симулационе студије. Поред тога, важну улогу има и способност моделара да пронађе адекватну апстракцију система за решавање одређеног проблема.

Најважније при избору симулационог модела је да модел буде што једноставнији и разумљивији, како због његовог лакшег развоја и модификације, тако и због потребе да га корисник што лакше разуме. Осим тога, значајан фактор при избору модела увек мора да буде и његова цена, као и ефикасност у погледу утрошка ресурса рачунара, како би се могао ефикасно користити за решавање проблема.

## КЛАСИФИКАЦИЈЕ МОДЕЛА

За почетак, укратко ћемо се осврнути на класификацију модела као битан предуслов за касније увођење и разликовање појединих специфичних формализама моделирања.

### 2.1 Класификација модела

Модели могу да се класификују према разним класификационим критеријумима. Овде ћемо указати на неке најважније.

#### 2.1.1 Класификација у односу на променљиве

Код сваког модела могуће је идентификовати описне променљиве значајне за његово разумевање, опис и управљање. Скуп описних променљивих се може поделити на подскуп оних које је могуће и подскуп оних које није могуће посматрати (оне које је могуће мерити неким расположивим средствима и друге које није могуће мерити). Све описне променљиве једног модела могу се поделити на улазне, излазне и променљиве стања. Свака променљива има свој опсег или домен - скуп вредности које јој се могу доделити, као и једну функцију којом се описују промене тих вредности у времену. Зависно од постојања ове три врсте променљивих, може се извршити подела модела као што је приказано у табели 2.1.

Модели који немају ни једну променљиву стања називају се модели без меморије или тренутне функције. Њихове излазне променљиве зависе од вредности улазних променљивих у тренутку посматрања.

Уколико постоји макар једна променљива стања, тада се ради о моделу са меморијом. Зависно од улазних променљивих, модели могу бити аутономни (без улазних променљивих) и неаутономни (са улазним променљивама).

Аутономни модел који не садржи излазну променљиву назива се затворени. С друге стране, аутономни модели који имају излазне променљиве и сви неаутономни модели називају се отворени модели.

Табела 2.1. Класификација модела у односу на променљиве

			Постојање описних променљивих		
			стања	улазне	излазне
Модели без меморије (тренутне функције)			Не	Да	Да
Модели са меморијом	Аутономни модели	Без излаза-затворен	Да	Не	Не
		Са излазом	Да	Не	Да
	Неаутономни модели	Без излаза	Да	Да	Не
		Са излазом	Да	Да	Да

Неаутономни модели се такође деле на две групе: са и без излазних променљивих.

### 2.1.2 Класификација у односу на природу опсега вредности променљивих модела

Под опсегом променљивих се подразумева скуп свих вредности које променљива може да узме. Описне променљиве могу узимати вредности из пребројивог (дискретног) или небројивог, односно континуалног скупа. У складу са тим разликујемо три класе модела:

#### 1. *Модели са дискретним стањима*

Код ових модела све три врсте описних променљивих (улазне, излазне и променљиве стања) узимају вредности из скупова чији су елементи дискретне вредности.

#### 2. *Модели са континуалним стањима*

Код ових модела све три врсте описних променљивих узимају вредности из подскупова реалних бројева.

#### 3. *Модели са мешовитим стањима*

Поједине променљиве узимају вредности из разних подскупова чији су елементи дискретне вредности, а остале променљиве из различитих подскупова реалних бројева.

### 2.1.3 Класификација у односу на природу опсега вредности променљиве "време"

Скуп вредности које се додељују променљивој "време" може бити пребројив или небројив. Стога разликујемо моделе са континуалним временом и моделе са дискретним временом.

Разликујемо две подкласе ових модела:

- ◆ моделе са континуалним временом и континуалним променама стања и
- ◆ моделе са континуалним временом и дискретним променама стања (иако време тече континуално, промене стања се могу дешавати само у дискретним скоковима).

У дискретним временским моделима, време се повећава у инкрементима који не морају бити еквидистантни. Описна променљива, било да је континуална или дискретна, расположива је (израчунава се) само у тим временским тренуцима. И овде разликујемо две подкласе модела :

- ♦ моделе са дискретним временом и континуалним променама стања и
- ♦ моделе са дискретним временом и дискретним променама стања.

Променљиве симулационог модела се могу мењати на било који од четири наведена начина: континуално у континуалном времену, дискретно у континуалном времену, дискретно у дискретном времену и континуално у дискретном времену (слика 2.1). Какве ће се променљиве користити зависи од ситуације која се моделира, од сврхе модела, рачунара (аналогни, дигитални или хибридни), симулатора и самог симулационог језика. Код сложених модела, у већој мери се користе променљиве које узимају вредности из дискретног скупа и у дискретној временској основи. За моделе код којих се промене вредности променљивих стања дешавају у дискретним временским тренуцима, трајање симулације зависи од изабраног временског корака, тј. од изабране временске јединице. Временски корак може бити константан или променљив.

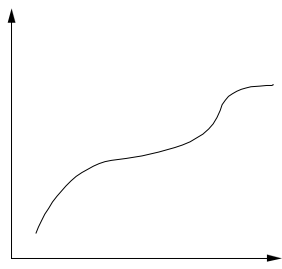
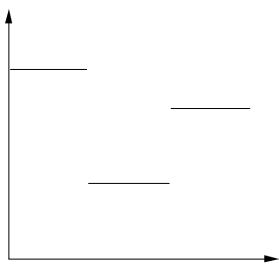
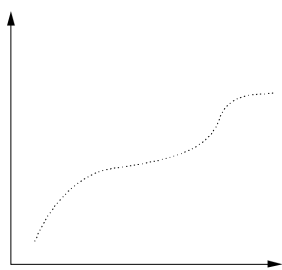
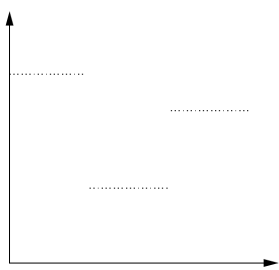
#### 2.1.4 Класификација у односу на временску зависност модела

Ова класификација третира питање експлицитне зависности правила интеракције модела од времена. Уколико је структура модела (правила интеракције између објеката модела) зависна од времена, тада се ради о временски променљивом - варијантном моделу. У супротном, када је структура модела независна од времена, модел се назива временски непроменљив - инваријантан. Такав модел се још назива и стационарни модел.

#### 2.1.5 Класификација у односу на детерминизам

Ова класификација разматра укључивање случајних променљивих у опис модела. У детерминистичким моделима, вредности

променљивих стања и улазних променљивих у једном тренутку једнозначно одређују вредности променљивих стања у следећем тренутку. Такви модели не садрже случајне променљиве. У недетерминистичким (стохастичким) моделима постоји бар једна случајна променљива. Већина проблема у реалном свету поседује особине стохастичности.

		Типови модела према опсегу вредности опсних променљивих	
		Модели са континуалним стањима	Модели са дискретним стањима
Типови према опсегу променљиве време	Континуални временски модел		
	Дискретни временски модел		

Слика 2.1 Врсте модела према опсегу променљиве време  
и према типу описних променљивих

### 2.1.6 Класификација у односу на предвиђање будућности

Постоје модели који за израчунавање вредности променљивих стања узимају у обзир и будуће вредности улазних променљивих (антиципаторски модели). Уколико то није случај, модел је неантиципаторски. Код модела који служе за планирање будућности



(производња, инвестиције, становништво) неопходно је узети у обзир будуће вредности улазних променљивих.

Код антиципаторских модела постоји посебан модул помоћу кога се генеришу будуће вредности улазних променљивих, а затим се симулација понавља уз коришћење претходно генерисаних вредности.

### 2.1.7 Класификација у односу на линеарност

Линеарни модели мењају стања и дају излазе поштујући законитости линеарних трансформација. Једна линеарна трансформација  $L : U \rightarrow Y$  задовољава принцип суперпозиције, ако за  $u_1, u_2 \in U$  и  $c_1, c_2$  важи:

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

Другим речима, ако су улази у систем  $u_1(t)$  и  $u_2(t)$ , излази из система  $y_1(t)$  и  $y_2(t)$ , а  $c_1$  и  $c_2$  скалари, тада важи:

$$\begin{aligned} u(t) &= c_1 u_1(t) + c_2 u_2(t) \\ y(t) &= c_1 y_1(t) + c_2 y_2(t) \end{aligned}$$

### 2.1.8 Класификација према врсти рачунара

Три врсте рачунара се могу користити за симулацију:

- ◆ аналогни
- ◆ дигитални
- ◆ хибридни

Скоро сви модели се могу симулирати на дигиталним и хибридним рачунарима. На аналогним рачунарима се могу симулирати само континуални модели са континуалним временом.

### 2.1.9 Класификација у односу на формални опис модела

Ова класификација је посебно значајна за схватање специфичних формализама моделирања који ће бити објашњени у овом поглављу.

Табела 2.2 Класификација у односу на формални опис модела

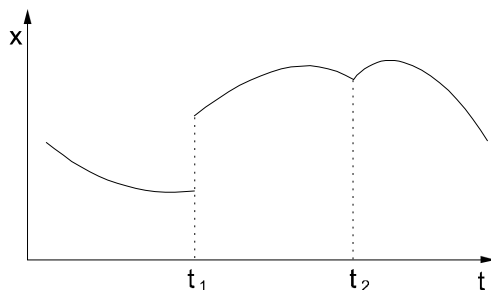
	Трајекторије описних променљивих модела	Време	Формални опис модела	Опсег променљивих	
				Континуални	Дискретни
Комбиновани модели	Континуални и Дисконтинуални модели	Континуални временски модели	Парцијалне диференцијалне једначине	Да	Не
	Дискретни модели	Дискретни временски модели	Диференцијалне једначине	Да	Не
			Диференцне једначине	Да	Да
			Сканирање активности	Да	Да
			Коначни аутомати	Не	Да
			Марковљеви ланци	Не	Да
		Континуални временски модели	Дискретни догађаји	Да	Да
			Интеракција процеса	Да	Да

Под формалним описом модела се подразумева прецизан, математички опис модела. Полазећи од различитих техника за формални опис модела и узимајући у обзир класификације из поглавља 2.1.2 и 2.1.3, везане за природу описних променљивих и променљиве "време", урађена је табела 2.2.

Модели са континуалним променама времена се описују диференцијалним једначинама и спадају у континуалне временске моделе. Као што се види на слици 2.1, континуални временски модели не морају да мењају своје стање континуално. На пример, модели са дискретним догађајима су континуални у односу на промену времена и дискретни у односу на промену стања. Модел са континуалним деловима трајекторија између којих постоји дисконтинуитет такође се описују диференцијалним једначинама и могу имати две врсте дисконтинуитета и то: дисконтинуитет извода променљиве и дисконтинуитет променљиве. Тачка  $t_1$  на слици 2.2 односи се на дисконтинуитет променљиве, а  $t_2$  на дисконтинуитет извода променљиве.

Према томе, дисконтинуални модели су они модели у којима су најмање једна променљива стања и/или њен извод дисконтинуални.

Дискретни модели мењају своје стање у дискретним временским тренуцима. Између два узастопна временска тренутка, стање модела остаје непромењено. Дискретни модели могу бити дискретни временски и континуални временски модели.



Слика 2.2 Дисконтинуитет променљиве и њеног извода

Поред наведених, постоје и друге класификације модела у односу на низ других релевантних фактора, које се овде неће спомињати.

## 2.2 Формална спецификација модела

Теорија скупова омогућава конструисање формализама који се користе за описивање објеката модела. Свака класа објеката, може

се представити одговарајућим формализмом који дефинише њене параметре и ограничења. Да би се дефинисао посебан објект неке класе у оквиру неког формализма, параметрима формализма додељују се вредности које задовољавају ограничења.

Један објект из класе задате формализмом, дефинише се на тај начин што се параметрима додељују конкретне вредности које задовољавају ограничења. Структура формализма односи се како на параметре, тако и на ограничења.

Класе објеката су најчешће повезане тако да се те везе могу формализовати као пресликавања из једне класе у другу. Посебно су интересантне три врсте таквих пресликавања: апстракција, асоцијација и спецификација.

### 2.2.1 Апстракција

Апстракција је пресликавање које подразумева контролисано укључивање детаља приликом описивања модела. То је процес којим се остављају по страни поједини детаљи објеката, односно врши се раздвајање битних особина од небитних, како би се указало на суштину неког објекта. Апстракција се стога може окарактерисати као пресликавање објекта једне класе у другу, мање сложену класу. Изворну класу називамо "конкретном", а ону другу, циљну, "апстрактном". Значај апстракције је у томе што је могуће већи број изворних објеката представити истим одредишним апстракцијама.

### 2.2.2 Асоцијација

Асоцијација је врста пресликавања од вишег ка нижем нивоу у хијерархији спецификације система. Инверзно пресликавање називамо реализацијом или имплементацијом.

### 2.2.3 Спецификација

За дату класу објеката могуће је дефинисати подкласе, увођењем новог формализма за сваку од подкласа. Објекте подкласа могуће је, затим, користити унутар нових формализама.

Међутим, да би успоставили релације између објеката различитих сродних подкласа, потребно је дефинисати пресликавање које преводи један посебан формализам у други, генералнији. Уколико постоји такво пресликавање, сви концепти и акције који се могу применити над објектима дефинисаним у генералном формализму, применљиви су и над објектима дефинисаним у посебном формализму.

Формалне дефиниције које су дате у овом поглављу, детаљније су разрађене у раду (Zeigler, 1984).

## 2.3 Формални модел улазно-излазног система

Посматрајмо улазно-излазни систем као скуповну структуру:

$$M = \langle T, U, \Omega, S, Y, \delta, \lambda \rangle$$

где су :

$T$ -	Временска база
$U$ -	Скуп улаза
$\Omega$ -	Скуп улазних сегмената
$S$ -	Скуп интерних стања
$Y$ -	Скуп излаза
$\delta$ -	Функција прелаза стања
$\lambda$ -	Функција излаза

Временска база ( $T$ ) је скуп вредности времена у моделу. На основу вредности из  $T$ , планира се редослед догађаја. Избор домена за  $T$  је област целих бројева  $I^+$ , или област реалних бројева  $R$ . Рационални бројеви, као и цели бројеви, могу да се користе за опсег дефинисаности временске базе у симулационим моделима, а да се задржи способност коришћења дигиталног рачунара у симулацији.

Скуп улаза ( $U$ ) је део интерфејса преко кога окружење утиче на систем. Избор домена за  $U$  најчешће је  $R^n$  за неко  $n \in I^+$ , које представља број улазних променљивих. Друга могућност избора домена за  $U$  је:  $U_m \cup [\emptyset]$ , где  $U_m$  представља скуп екстерних догађаја, а  $\emptyset$  је празан скуп.

Скуп улазних сегмената  $(\Omega)$  описује облик улаза у систем за неки временски период. Улазни сегмент је одређен окружењем система. Улазни сегмент је дефинисан пресликавањем облика

$$\omega: \langle t_0, t_1 \rangle \rightarrow U$$

где је:  $\langle t_0, t_1 \rangle$  - интервал временске базе између почетног и крајњег тренутка. Скуп свих таквих улазних сегмената назива се  $(U, T)$ . Скуп улазних сегмената  $\Omega$  је подскуп од  $(U, T)$ .

У пракси је често  $\Omega$  скуп континуалних сегмената  $T \in R$  и  $U \in R^n$ . Друга могућност је скуп дискретних сегмената  $U \in U_m$  и  $T \in R$ . Сегмент дискретних улазних догађаја је пресликавање:

$$\omega: \langle t_0, t_1 \rangle \rightarrow U_m \cup \{\emptyset\}$$

такво да  $\omega(t) = \emptyset$  искључује могућност ограниченог скупа догађаја

$$\{\tau_1, \tau_2, \dots, \tau_n\} \subseteq \langle t_0, t_1 \rangle.$$

Када је  $T \in I^+$ ,  $\Omega$  представља скуп коначних секвенци.

Скуп интерних стања  $(S)$  представља меморију система, тј. утицај предхотних догађаја на његов садашњи и будући одзив. Од избора интерних стања и њиховог карактера зависи структура модела.

Функција прелаза стања  $(\delta)$  је пресликавање

$$\delta: S \times \Omega \rightarrow S$$

Интерпретација функције прелаза стања огледа се у следећем: када се на систем са стањем  $S$  у времену  $t_0$  доведе улазни сегмент

$$\omega: \langle t_0, t_1 \rangle \rightarrow U,$$

тада  $\delta(S, \omega)$  пресликава стање система са временом  $t_1$ . Према томе, интерно стање у почетном тренутку и улазни сегмент од тог тренутка, надаље јединствено одређују стање система на крају сегмента.

За свако  $s \in S$  и  $\omega \in \Omega$  и  $t$  у области  $\omega$  важи:

$$\delta(S, \omega) = \delta[\delta(S, \omega_{t>}), \omega_{t<}] \quad (\text{аксиома семи-групе})$$

где су:

$$\omega_{t>} = \omega|_{\langle t_0, t \rangle} \text{ - део } \omega \text{ између } t_0 \text{ и } t_1$$

$$\omega_{t<} = \omega|_{\langle t, t_1 \rangle} \text{ - део } \omega \text{ између } t \text{ и } t_1.$$

Ово захтева да стање  $S_t = \delta(S, \omega_{t>})$  које припада било којем тренутку  $t$ , сумира потребне претходне догађаје, тако да настављање експеримента од тог стања резултује истим крајњим стањем.

Избор скупа стања заснован на датој дефиницији није јединствен. Чак ни његове димензије нису фиксирани. Због тога поступак проналажења одговарајућег и корисног простора стања представља срж процеса моделирања.

Скуп излаза ( $Y$ ) представља део интерфејса којим систем утиче на окружење. Дефиниција је иста као и за скуп улаза с тим што је правац деловања супротан. У оквиру једног вишекомпонентног система, излаз једне компоненте система представља улаз у другу компоненту система.

У најпростијем облику функција излаза ( $\lambda$ ) представља пресликавање  $\lambda: S \rightarrow Y$  које повезује претпостављено стање система са утицајем система на његову околину. Такво пресликавање ипак не дозвољава директан утицај улаза на излаз. Општија функција излаза је пресликавање  $\lambda: S \times U \times T \rightarrow Y$ . Често  $\lambda$  није једнозначно пресликавање, тако да се из окружења, стање система не може директно посматрати.

На основу дефинисаног формализма могуће је одредити појам понашања система. Понашање система је спољашња

манифестација његове интерне структуре, па је релација помоћу које се може одредити понашање система задата са  $(U, T) \times (Y, T)$ .

Ова релација се одређује на следећи начин. Сваком стању  $s \in S$  и улазном сегменту  $\omega: \langle t_0, t_1 \rangle \rightarrow U$  у  $\Omega$  придружена је јединствена трајекторија стања

$$Straj_{s, \omega}: \langle t_0, t_1 \rangle \rightarrow S$$

тако да је:

$$Straj_{s, \omega}(t_0) = s$$

$$Straj_{s, \omega}(t) = \delta(s, \omega_{t>}) \quad , \quad t \in \langle t_0, t_1 \rangle$$

Таква трајекторија стања резултат је аналитичког решења, или је израчуната у току извршења симулације на рачунару. Осмотрива пројекција ове трајекторије је трајекторија излаза која се дефинише заједно са  $y \in Y$  и  $\omega \in \Omega$  на следећи начин:

$$Otraj_{s, \omega}: \langle t_0, t_1 \rangle \rightarrow Y$$

У случају просте функције излаза  $\lambda(S)$ ,  $Otraj$  добија облик

$$Otraj_{s, \omega}(t) = \lambda(Straj_{s, \omega}(t))$$

Тада се понашање система дефинише на основу улазно-излазне релације  $R_s$ :

$$R_s = \{(\omega, \rho) \mid \omega \in \Omega, \rho = Otraj_{s, \omega} \text{ за } s \in S\}$$

Сваки елемент  $(\omega, \rho) \in R_s$  назива се пар улазно-излазног сегмента и представља резултат посматрања или експеримента на систему у коме је  $\omega$  улаз у систем, док је  $\rho$  посматрани излазни сегмент. С обзиром да се систем може наћи у било ком почетном стању, за један улазни сегмент  $\omega$  може постојати више излазних сегмената  $\rho$ .

Оваква дефиниција модела је довољно општа да се њоме може дефинисати модел сваког система који је од интереса за градиво овог



предмета, али и довољно проста да би се њоме једноставно објаснили специфични формализми моделирања.

## ОЦЕНА ПАРАМЕТАРА МОДЕЛА

Оцена параметара модела представља поступак експерименталног одређивања вредности параметара који се појављују у математичком опису модела. Полази се од претпоставке да је структура модела, односно везе између променљивих модела и параметара, експлицитно дата. У пракси је тешко "повући црту" између структуре модела и одговарајућег скупа параметара. Тако, на пример, промена вредности параметра из не-нулте у нулту вредност, може довести до поједностављене структуре модела. Зато се полази од претпоставке да сви параметри имају не-нулте вредности, тако да се добија јасно дефинисана структура модела и његових параметара, чије вредности треба одредити.

Посматрајмо систем дефинисан математичким моделом у простору стања:

$$\begin{aligned}s' &= f(s, u, p, t) \\ y &= g(s, u, p, t) \\ s(t_0) &= s_0\end{aligned}$$

где су:

$s$  - вектор стања димензије  $n$ ,  
 $u$  - вектор улаза димензије  $m$ ,  
 $y$  - вектор излаза димензије  $k$ ,  
 $p$  - вектор састављен од  $n_p$  непознатих параметара,  
 $f$  и  $g$  су одговарајуће векторске функције,  
 $s_0$  - почетни услови.

Потребно је за дати модел и скуп улазних и излазних података, одредити вектор непознатих параметара  $p$ . При томе почетни

услови морају бити познати. Питање које се овде поставља односи се на то да ли се параметри могу идентификовати, односно да ли их је могуће математички одредити.

Постоји више различитих приступа проблему оцене параметара. Класичан приступ је чисто статистички и његова примена захтева испуњење одговарајућих услова. Могуће је проблем одређивања параметара посматрати и као оптимизацију погодно дефинисане функције циља, што захтева мање почетних претпоставки. Јединствени приступ оцени параметара модела није могуће реализовати. Стога се за оцену параметара за поједине класе модела користе алгоритми чија структура зависи од:

1. формализма модела:

- ◆ континуално-временски,
- ◆ дискретно-временски,
- ◆ линеарни или нелинеарни,
- ◆ детерминистички, стохастички.

2. контекста моделирања:

- ◆ тип променљивих система,
- ◆ априорно знање,
- ◆ сврха модела.

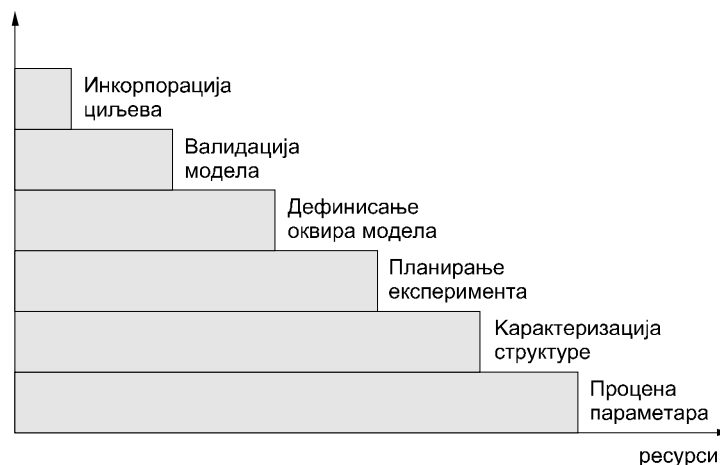
3. филозофија процене:

- ◆ критеријуми,
- ◆ нумеричка процедура,
- ◆ прилаз израчунавању.

Поступак процене параметара није нимало лак и једноставан посао. Напротив, то је фаза у процесу изградње модела где је потребно уложити највише истраживачког напора, времена и новца, како би се реализовао валидан модел, о чему илустративно говори и слика 3.1.

Постоји већи број метода и техника за процену параметара. Неке од њих, као на пример оне из шездесетих година овог века, углавном су се користиле одређеним "триковима". Најстарије технике процене односе се на проблем процене параметара код модела где је могуће лако извршити мерење, а где је мали ниво поремећаја.

Споменимо неке од њих: технике одзива у временском домену, као што је добро познати метод одзива на јединичну-одскачну функцију. Користе се и методе које се заснивају на одзиву система у фреквенцијском домену. Оне су углавном познате и о њима овде неће бити речи.



Слика 3.1. Ангажовање ресурса по фазама изградње модела

Овде ће бити изложене методе оцене параметара модела у временском домену. Оне се могу дати у виду блок дијаграми за *off-line* израчунавање или као рекурзивни алгоритми погодни за *on-line* обраду (Spriet & Vansteenkiste, 1982).

### 3.1 Оцене параметара детерминистичког модела

За модел са познатом структуром могу се увести следеће дефиниције:

1. За скаларни параметар  $p_j$  каже се да је идентификабилан на интервалу  $[t_0, T]$ , ако постоји коначни број решења за  $p_j$  која произилазе из релација датог модела. Скаларни параметар је неидентификабилан ако постоји бесконачан број решења за  $p_j$ , која произилазе из релација модела.

2. Опис модела је системски идентификабилан, ако су сви параметри идентификабилни. У супротном, ако је један параметар неидентификабилан, опис модела је системски неидентификабилан.
3. Скаларни параметар  $p_j$  је јединствено идентификабилан на интервалу  $[t_0, T]$ , ако постоји јединствено решење за  $p_j$  које произилази из релација модела са датим улазом.
4. Спецификација модела је параметарски идентификабилна на интервалу  $[t_0, T]$  ако су сви параметри јединствено идентификабилни.

Наведене дефиниције, иако су изложене у терминима континуалних упрошћених параметарских модела, примењиве су и на дискретне временске моделе. Оне су уједно довољне да опишу све услове идентификабилности. Међутим, оне не показују како се идентификабилност може проверити.

Битно је схватити да идентификабилност система и параметара зависи од два фактора:

1. Специфичности примењеног улаза и почетних услова.

Почетни услови су битни јер идентификабилност система и параметара може зависити од њихових вредности. Кад је у питању улаз, може се догодити да се за дати систем улазни сегмент може поделити у више посебних категорија. Неке од њих могу обухватити тип идентификабилности, док друге могу изазвати различите проблеме неидентификабилности.

2. Структуре једначина и ограничења.

Неке структуре се могу идентификовати ако се примене одговарајући улази, док су друге неидентификабилне, без обзира на то какав се улаз примени.

### 3.2 Оцене параметара модела стохастичких система

Код стохастичких система, излазне секвенце су стохастички процеси, тако да се проблем усложњава захтевом за укључивање

параметара појединих реализација процеса. Према *Ljung-u (1979)*, проблем идентификабилности се може поделити у три дела, односно идентификабилност зависи од следећа три фактора:

- ♦ *Структуре* - која параметре ставља у релацију једне с другима као и са улазима и излазима. Као и у детерминистичком случају, неидентификабилност може бити узрокована специфичним структуралним односима компоненти система.
- ♦ *Улаза* - од којих неки могу бити сувише "прости" да би се постигла идентификабилност.
- ♦ *Процедуре процене* - које треба да буду конзистентне у смислу да конвергирају ка стварним вредностима параметара. Идентификабилност система захтева постојање конзистентног естиматора чија процена конвергира једном од коначних бројева допустивих вектора параметара. Идентификабилност параметара захтева исто, с тим што је допустиви вектор параметара јединствен.

Примери који илуструју идентификабилност параметара за детерминистички или стохастички случај, дати су у радовима (*Spriet & Vansteenkiste, 1982; Ljung, 1976; Ljung, 1979*).

### 3.3 Статистички приступ процени параметара статичких модела

Статистичке оцене параметара статичких модела се могу реализовати у простој секвенцијалној форми или рекурзивној форми.

За идентификацију параметара система за које су подаци о њиховом понашању прикупљени и меморисани на неком медујуму, најједноставније је применити секвенцијалне статистичке методе. Међутим, за прикупљање података у реалном времену, када је број података такав да се не може меморисати, треба користити одговарајуће статистичке формуле у рекурзивној форми. За рачунање у рекурзивној форми је најчешће потребно само неколико променљивих које се ажурирају у току експеримента. На тај начин се штеди у меморијским ресурсима. Предност рекурзивног приступа се огледа и у томе што су приликом сваког ажурирања статистичких показатеља познате њихове текуће вредности.

### 3.3.1 Оцена средње вредности случајне променљиве

Као илустрација за процену средње вредности случајне променљиве за  $N$  узорака, користимо секвенцијални метод и рекурзивни метод:

#### 3.3.1.1 Секвенцијални метод

Уз претпоставку да су сви резултати мерења на реалном систему доступни у време израчунавања, процедура за процену средње вредности може се дефинисати на следећи начин:

$$\bar{y}_N = \frac{1}{N} \sum_{i=1}^N y_i$$

Дакле, сва мерења се сабирају и резултат дели са бројем узорака.

#### 3.3.1.2 Рекурзивни метод

За рекурзивно израчунавање је карактеристично следеће:

- ♦ база података (резултати мерења) се проширује током рачунања,
- ♦ међурезултати за одређени број узорака су такође доступни и они теже секвенцијалном решењу како израчунавање одмиче.

Рачунање се одвија по следећој процедури:

$$\begin{aligned}\hat{a}_0 &= 0 \\ \hat{a}_k &= \hat{a}_{k-1} - \frac{1}{k}(\hat{a}_{k-1} - y_k)\end{aligned}$$

где је:

$y_k$  - текући узорак ( $k=1,2,\dots,N$ )

У литератури се може наћи доказ да је за дато  $N$  секвенцијално решење истоветно са рекурзивним.

### 3.4 Оцена непознатог параметра по методи најмањих квадрата

Посматрајмо статички модел код кога је веза између параметра модела и резултата експеримента дата релацијом:

$$y_i = x_i a + e_i \quad i = 1, 2, \dots, N$$

Претпоставке су:

$x_i$  - познате вредности

$e_i$  - грешке мерења

Потребно је проценити непознати параметар  $a$  користећи доступне резултате мерења.

#### 3.4.1 Секвенцијално решење

Циљ је да се минимизира одступање између података које генерише реални систем и података који се добијају на основу модела, односно да се минимизира квадрат грешке:

$$J = \sum_{i=1}^N [y_i - x_i a]^2 = \sum_{i=1}^N e_i^2$$

Ова метода је позната као метода најмањих квадрата (МНК) и представља једну од најпознатијих и најкоришћенијих метода за процену параметара модела. Основна предност ове методе лежи у њеној једноставности.

За критеријумску функцију која се минимизира, узима се квадратна функција грешке, јер се њен минимум лако налази изједначавањем парцијалних извода по непознатим параметрима са нулом.

У горњем случају имамо један непознати параметар, односно



$$\frac{\partial J}{\partial a} = 0, \quad t_j.$$

$$2 \sum_{i=1}^N [y_i - x_i a] \cdot (-x_i) = 0$$

одакле следи:

$$\left[ \sum_{i=1}^N x_i^2 \right] \cdot a = \sum_{i=1}^N x_i y_i$$

из чега се директно може одредити непознати параметар  $a$ .

### 3.4.2 Рекурзивно решење

Уведимо следеће ознаке:

$$\hat{a}_k = p_k b_k$$

где је:

$$p_k = \left[ \sum_{i=1}^k x_i^2 \right]^{-1}$$

$$b_k = \sum_{i=1}^k x_i y_i$$

У рекурзивној форми ове формуле можемо написати:

$$p_k = p_{k-1} - p_k p_{k-1} x_k^2$$

$$b_k = b_{k-1} + x_k y_k$$

Ако дефинишемо  $k_k$  у облику:

$$k_k = p_{k-1} x_k \left[ I + p_{k-1} x_k^2 \right]^{-1}.$$

тада се оцена параметра  $a$  може изразити као:

$$\hat{a}_k = \hat{a}_{k-1} - k_k (\hat{a}_{k-1} - y_k)$$

Описани изрази чине рекурзивни алгоритам методе најмањих квадрата (МНК) за процену једног непознатог параметра. Ограничења и детаљан опис за примену ове методе могу се наћи у (Ljung, 1976; Ljung, 1979).

### 3.5 Процена $k$ непознатих параметара

Посматрајмо случај где је веза између  $k$  параметара модела и резултата неког експеримента дата следећом релацијом:

$$y = a_1 x_1 + a_2 x_2 + \dots + a_k x_k,$$

где су параметри  $a_k$  непознати. Претпоставимо да смо  $n$  пута обавили експеримент на систему, чији се резултати могу приказати следећом табелом:

Табела 3.1. Општи облик резултата за  $n$  експеримената

Редни број експеримента	$x_1$	$x_2$	.....	$x_k$	$y$
1	$x_{11}$	$x_{12}$	.....	$x_{1k}$	$y_1$
2	$x_{21}$	$x_{22}$	.....	$x_{2k}$	$y_2$
⋮	⋮	⋮	⋮	⋮	⋮
$j$	$x_{j1}$	$x_{j2}$		$x_{jk}$	$y_j$
⋮	⋮	⋮	⋮	⋮	⋮
$n$	$x_{n1}$	$x_{n2}$	.....	$x_{nk}$	$y_n$

Табела показује да, ако су у  $j$ -ом експерименту реализована мерења  $x_{j1}, x_{j2}, \dots, x_{jk}$ , тада се као резултат добија  $y_j$ . С обзиром да у већини случајева није могуће извршити тачна мерења, а с друге стране, често усвојени модел није адекватан, експериментални

результати неће задовољавати горњу релацију, већ ће постојати одређена одступања (грешке) која ћемо обележити са  $e_j$ .

На основу изложеног, за  $n$  експеримената и  $k$  променљивих добија се ситем једначина облика:

$$\begin{aligned} y_1 - (a_1 x_{11} + a_2 x_{12} + \dots + a_k x_{1k}) &= e_1 \\ y_2 - (a_1 x_{21} + a_2 x_{22} + \dots + a_k x_{2k}) &= e_2 \\ &\vdots \\ y_j - (a_1 x_{j1} + a_2 x_{j2} + \dots + a_k x_{jk}) &= e_j \\ &\vdots \\ y_n - (a_1 x_{n1} + a_2 x_{n2} + \dots + a_k x_{nk}) &= e_n. \end{aligned}$$

Оцене непознатих параметара  $a_1, a_2, \dots, a_k$  одређују се минимизацијом критеријумске функције

$$J = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n [y_i - (a_1 x_{i1} + a_2 x_{i2} + \dots + a_k x_{ik})]^2.$$

Уколико је  $n$  веће, односно уколико је већи број експеримената (узорака) утолико ће се тачније одредити вредности параметара  $a_1, a_2, \dots, a_k$ .

Минимум функционала  $J$  добија се изједначавањем његових парцијалних извода по параметрима  $a_1, a_2, \dots, a_k$  са нулом, односно решавањем система од  $k$  једначина.

$$\sum_{i=1}^n [y_i - (a_1 x_{i1} + a_2 x_{i2} + \dots + a_k x_{ik})] \cdot (x_{ij}) = 0, \quad j = 1, 2, \dots, k$$

Међутим, на поступак методе најмањих квадрата у овом случају је знатно једноставније применити матрични рачун. Обележимо са  $\mathbf{y}$   $n$ -димензиони вектор резултата експеримента, са  $\mathbf{a}$   $k$ -димензиони вектор непознатих параметара, а са  $\mathbf{X}$  матрицу мерења улаза експеримента димензије  $(n \times k)$ .

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}$$

Критеријумска функција се тада може представити у облику

$$J = (\mathbf{y} - \mathbf{Xa})^T (\mathbf{y} - \mathbf{Xa})$$

Применом правила диференцирања за матрице и векторе добија се

$$\frac{\partial J}{\partial \mathbf{a}} = 2 \mathbf{X}^T (\mathbf{y} - \mathbf{Xa}) = 0$$

одакле се добија

$$\mathbf{X}^T \mathbf{Xa} = \mathbf{X}^T \mathbf{y}$$

односно

$$\mathbf{a} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

је релација која даје непознате параметре модела.

За процену  $k$  непознатих параметара, могуће је формирати рекурзивни алгоритам, на врло сличан начин као и код рекурзивног решења МНК за модел са једним непознатим параметром. Детаљи се могу наћи у литератури (*Ljung, 1979*).

## ВАЛИДАЦИЈА И ВЕРИФИКАЦИЈА

Један од најзначајнијих и најтежих задатака са којим се среће моделар у поступку изградње модела су *верификација* и *валидација* симулационих модела. Корисници модела, инжењери и аналитичари који користе излазе (резултате) модела као помоћ за своје предлоге при пројектовању и менаџери који доносе одлуке засноване на тим предлозима, увек гледају на модел са извесном дозом скептицизма у погледу његове валидности. Да би смањио постојећи скептицизам и повећао поверење у свој модел, моделар мора да сарађује са крајњим корисницима у процесу изградње и валидације модела.

Проблем валидације модела проистиче из чињенице да је модел увек упрошћена слика реалног система који представља, односно поједностављени поглед на реални систем који се истражује. Поред тога, у модел се готово увек уносе одређене апроксимације реалности, што процес валидације чини још важнијим. Из свега тога проистиче да се модел мора тестирати како би се установило да ли је он поуздан, без грешака и да ли је довољно уверљив за све оне који ће га користити. Поступак којим испитујемо колико верно и прецизно један модел представља реални систем, најчешће се може посматрати кроз два повезана корака:

### 1. Верификација.

Односи се на проверу да ли је симулациони програм, којим се имплементира модел, без грешака и конзистентан са моделом. То је у ствари поређење концептуалног модела са рачунарским кодом којим се таква концепција имплементира.

### 2. Валидација.

Односи се на поступак одређивања да ли је модел прецизна репрезентација реалног система. Најчешће то је једна итеративна процедура у којој се понашање модела пореди са

понашањем реалног система и уочена неслагања и разлике користе за доградњу и исправку модела. Поступак побољшања модела се наставља, све док се не одлучи да добијена тачност модела задовољава.

Иако се верификација и валидација модела концептуално разликују, најчешће се симултано спроводе од стране моделара. Разлог за то је што се ова два процеса у пракси добрим делом поклапају. Наиме, уколико један симулациони програм производи бесмислице, није увек јасно колико су криве грешке у концептуалном моделу, колико грешке у програмирању, а колико коришћење погрешних података. Зато се често каже да су изградња модела, верификација и валидација у динамичкој повратној спрези. На то указује и слика 4.1.



Слика 4.1. Изградња модела, верификација и валидација

Први корак у изградњи модела обухвата посматрање реалног система и интеракција између његових различитих компоненти, као и прикупљање података о понашању система. Међутим, у том поступку моделар не сме да се ослони само на своја запажања, пошто то често води погрешном схватању самог система и његовог

понашања, већ је неопходно ангажовање стручњака који добро познају систем или неке његове делове. Специфична знања о појединим аспектима система, која ови стручњаци различитих профила поседују, од великог су значаја за моделара и будући ток изградње модела.

Други корак у изградњи модела је формирање концептуалног модела, скупа претпоставки за компоненте модела и структуру система, као и хипотеза за вредности параметара модела. Као што се може видети са слике 4.1, концептуална валидација је поређење реалног система и концептуалног модела.

Трећи корак у изградњи модела је писање рачунарских програма за симулационе моделе.

Треба напоменути да се ове три фазе изградње модела вишеструко преплићу и да то није једноставан линеарни процес који се састоји из три корака; наиме, моделар се враћа на сваку од ове три фазе више пута у току изградње, верификације и валидације модела.

## 4.1 Валидација симулационих модела

Проблем валидације модела настаје пошто се у фази изградње модела уносе многе апроксимације реалног система. На тај начин, ми ограничавамо модел (успостављамо границе у односу на окружење), игноришући све оно изван њега што није експлицитни улаз и одбацујемо факторе за које сматрамо да су неважни. Према *Bartley, Fox & Scharge, (1987)*, апроксимације које се најчешће примењују су:

### 1. Функционална апроксимација

Изразито нелинеарне функције често се апроксимирају неким једноставнијим функцијама, на пример линеарним. Важна претпоставка је да једноставнија функција треба да буде приближна "оригиналној" функцији у области где ће систем вероватно функционисати. Ако програм мора да функционише и у области где је поклапање функција слабо, потребно је обезбедити да он штампа поруку упозорења.

### 2. Апроксимација расподеле

Реалне вероватноће расподела које су и саме познате једино апроксимативно, често се апроксимирају једноставнијим расподелама, као што су нормална или експоненцијална. Најекстремнији пример апроксимације је онај када се случајна променљива замени константом.

### 3. Апроксимација независности

Модел се често поједностављује тако што се претпоставља да су различите компоненте (описане случајним променљивим) статистички независне.

### 4. Апроксимација агрегације

Веома чест тип апроксимације је агрегација. Под агрегацијом подразумевамо ситуацију када више елемената посматрамо као једну целину. Неки типични примери агрегација су:

#### а) Временска агрегација

Сви дискретно-временски модели врше временску агрегацију. Интервал времена као што је дан, третира се као један појединачни период. За све догађаје који се десе током дана, претпоставља се да су се десили истовремено.

#### б) Међу-секторска агрегација

Више одељења, фирми, производних линија, итд. посматра се као једна целина.

#### с) Агрегација помоћних средстава

Овде се више помоћних средстава посматра као једно. На пример, ако поседујемо рачунарску систем са два централна процесора у паралелној вези, након агрегације ћемо сматрати да поседујемо један, два пута бржи централни процесор.

### 5. Апроксимација стационарности

Ствар поједностављује чињеница да се параметри и друге карактеристике система не мењају у времену. За извесне физичке процесе, као што су неке астрономске појаве, ово може бити прихватљива почетна апроксимација. Међутим, у политичким, економским, организационим и социјалним системима, свакодневно искуство указује да је већина појава



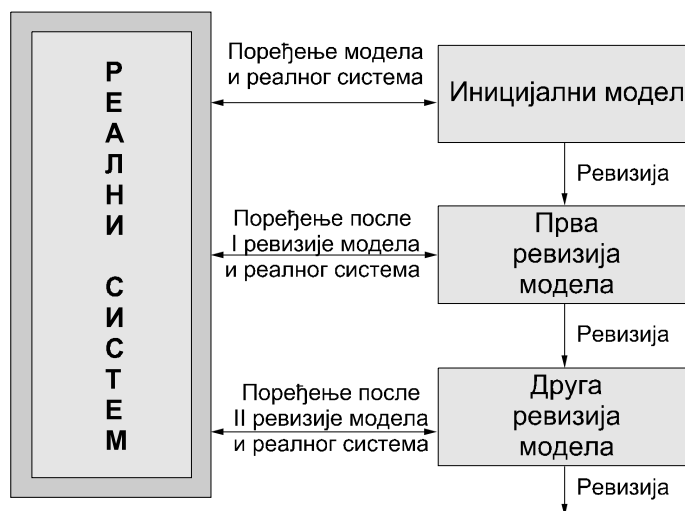
нестационарне природе и да је као последица тога претпоставка постојане структуре неодржива.

Све претпоставке и апроксимације које уносимо у модел могу, на извесан начин, да доведу у питање валидност тог модела. Из тог разлога ми и приступамо одређивању валидности модела, односно поредимо модел и његово понашање са реалним системом и његовим понашањем.

Циљ процеса валидације је двојак:

- ◆ Да произведе модел који представља понашање реалног система и који је довољно близак реалном систему, тако да се може користити за обављање експеримента.
- ◆ Да поузданост модела повећа на прихватљив ниво, тако да модел могу користити различити доносиоци одлука.

На слици 4.2 приказан је итеративни поступак поређења модела и реалног система у поступку валидације.



Слика 4.2 Итеративни процес валидације модела

Процес валидације модела не треба посматрати као изоловани скуп мера и процедура које следе након изградње модела, већ као интегрални и незамењиви део развоја модела. Већ смо поменули да је процес валидације итеративни процес. Концептуални модел се

пореди са реалним системом и уколико постоје неслагања и разлике за које се сматра да нису оправдане, врши се исправка (или чак значајнија промена) на моделу, затим се тако добијени модел поново пореди са реалним системом и врше нове исправке, уколико је то потребно итд. Итеративни процес се наставља све док се не постигне задовољавајућа тачност модела.

Поређење модела са реалним системом врши се помоћу више различитих тестова; неки од њих су субјективне природе, док су други објективни. Субјективни тестови најчешће подразумевају учешће људи који добро познају систем или неке његове аспекте и који, примењујући посебна знања о систему, доносе закључке о моделу и његовом излазу. Објективни тестови увек захтевају неке податке о понашању система, као и податке које производи модел. Прикупљени подаци се обрађују различитим статистичким тестовима који пореде одређене аспекте скупа података система и скупа података модела. Најчешће, један скуп података није довољан за коначну валидацију модела, већ се "финална" валидација врши уз помоћ неког другог, додатног скупа података.

#### 4.1.1 Практични приступ процесу валидације

*Naylor* и *Finger* (1976) формулисали су практичан приступ проблему валидације, који се састоји се из три фазе:

1. Изградити модел који верно представља реални систем
2. Потврдити претпоставке модела
3. Упоредити улазно-излазне трансформације модела са одговарајућим улазно-излазним трансформацијама реалног система.

##### 4.1.1.1 Процена валидности модела

Први циљ моделара је да изгради модел који ће изгледати разумно корисницима или другима који поседују знања о реалном систему који се симулира. Потребно је да потенцијални корисници модела буду укључени у процес изградње модела од почетка, дакле од концептуализације, па све до имплементације модела, како би се осигурало да се у модел "угради" висок степен реализма кроз

разумне претпоставке које се односе на структуру система и поуздане податке. Такође је посебно важно да потенцијални корисници, али и други "људи од знања", буду укључени у процес итеративног побољшавања модела, који се спроводи на основу уочених неслагања и одступања у поступку поређења модела и реалног система. Посебна предност укључивања корисника огледа се у томе што се на тај начин повећава поверење у модел, без кога менаџери и други доносиоци одлука сигурно не би били вољни да верују резултатима симулације, у процесу доношења одлука.

За проверу валидности модела, може се користити и *анализа осетљивости*. То је поступак тестирања осетљивости модела на различите претпоставке и промене улазних величина. Промене излазног понашања модела, као последица промена улазних величина, могу пружити корисне информације о моделу и кориснику и моделару. Код већине симулационих модела, постоји велики број улазних величина, па самим тим и велики број могућих тестова осетљивости. Зато је потребно да моделар одабере оне улазне величине за које сматра да су "најкритичније" и њих тестира, пошто је врло вероватно да време и финансијска средства неће дозволити да се испита осетљивост модела на све улазне променљиве. Уколико подаци о реалном систему омогућавају барем две измене улазних параметара, тестови осетљивости се могу допунити коришћењем одговарајућих статистичких техника.

#### 4.1.1.2 Валидација претпоставки модела

Претпоставке модела је могуће сврстати у две генералне категорије: претпоставке о *структури* и претпоставке о *подацима*. Претпоставке о структури се тичу питања како функционише систем и често обухватају поједностављења и апстракције реалног система. Претпоставке о подацима треба да буду засноване на скупу поузданих података и исправној статистичкој анализи тих података. Статистичка анализа углавном се састоји из три корака:

1. Идентификација одговарајућих расподела добијених података
2. Процена параметара изабране расподеле
3. Валидација претпостављеног статистичког модела неким тестом (нпр. Хи-квадрат или Колмогоров-Смирнов тест).

#### 4.1.1.3 Валидација улазно - излазних трансформација

Једини објективни тест модела као целине је провера способности модела да предвиди будуће понашање реалног система, када улазни подаци модела одговарају улазним подацима реалног система и када је "политика" која је имплементирана у моделу, имплементирана негде и у реалном систему. Осим тога, ако се ниво неке улазне величине повећава или смањује, модел треба прецизно да предвиди шта ће се десити у реалном систему под истим околностима. Другим речима, структура модела треба да буде довољно прецизна да модел даје добра предвиђања и то не само за један скуп улазних података већ за цео опсег скупова улазних података који су од интереса.

У овој фази валидације, модел се посматра као улазно-излазна трансформација, тј. модел прихвата вредности улаза и трансформише ове улазе у излазе који указују на понашање модела.

Уместо валидације улазно-излазне трансформације модела преко предвиђања будућности, моделар може користити и старе историјске податке, који су сачувани искључиво за процес валидације (подаци који нису коришћени у фази изградње и доградње модела).

Да би се извршила валидација улазно-излазне трансформације, неопходан услов је постојање система који се проучава, како би било могуће прикупити податке о систему за барем један скуп улазних података. Уколико је систем у фази планирања и није могуће прикупити податке, потпуна валидација улазно-излазне трансформације није могућа. У неким случајевима могу постојати одређени подсистеми планираног система, па је могуће спровести само делимичну валидацију модела.

Валидација симулационих модела је битан и одговоран посао, јер се бројне и важне одлуке заснивају на резултатима експериментисања са моделом. Зато је обимна и свестрана анализа модела и резултата добијених симулацијом тог модела од велике важности за квалитет симулације и могућности примене добијених резултата у процесу одлучивања.

Најчешће је сувише тешко, сувише скупо или временски немогуће користити све могуће технике валидације за сваки модел који се развија. Стога је веома важан посао сваког моделара да одабере

оне технике валидације које највише одговарају у конкретном случају и да на тај начин обезбеди тачност и поузданост модела.

#### 4.1.2 Формални критеријум за утврђивање валидности модела

У складу са изнесеним поставкама, реалан систем треба посматрати као систем описан на вишем нивоу, док модел за који треба утврдити валидност треба да буде на нижем нивоу описа. У овом поглављу приказаћемо преглед већ раније изложене материје са становишта утврђивања валидности модела.

*Хомоморфизам* (грчка реч: *homo*-сличан, *morph*-структура) представља формални критеријум за утврђивање валидности упрошћеног модела за дате експерименталне услове, у односу на основни модел чије су карактеристике објашњене раније. Ради једноставнијег излагања, ограничићемо се на доказивање валидности дискретних временских модела.

Суштина формалног поступка за проверу валидности је у томе да се нађе пресликавање  $H$  помоћу кога је могуће из сваког стања основног модела (нпр.  $s$ ) прећи у одговарајуће стање упрошћеног модела (нпр.  $s'$ ). Ако овакво пресликавање постоји, тада се закључује да су улазно-излазна понашања основног и упрошћеног модела иста за дате експерименталне услове. Да би се утврдио хомоморфизам, морају бити испуњени следећи услови:

##### 1. Очување функције наступања времена

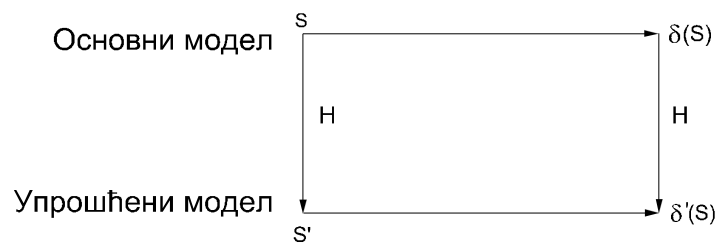
Стање у основном моделу ( $s$ ) и њему одговарајуће стање у упрошћеном моделу ( $s'$ ) морају имати исту функцију наступања времена. То значи да ће се стања основног модела и стања упрошћеног модела мењати у истим тренуцима времена.

##### 2. Очување функције прелаза стања

Ако стању  $s$  основног модела одговара стање  $s'$  упрошћеног модела (тј. применом пресликавања  $H$  на стање  $s$  добија се стање  $s'$ ), тада и следећа стања у која ће модел прећи у наредниом тренутку посматрања морају одговарати једно другом. Следеће стање основног модела ће бити  $\delta(s)$ , а

упрошћеног модела  $\delta'(s')$ , што значи да применом пресликавања  $H$  на стање  $\delta(s)$  треба да се добије стање  $\delta'(s')$ .

Ово се графички може приказати преко тзв. комутативних дијаграма (слика 4.3).

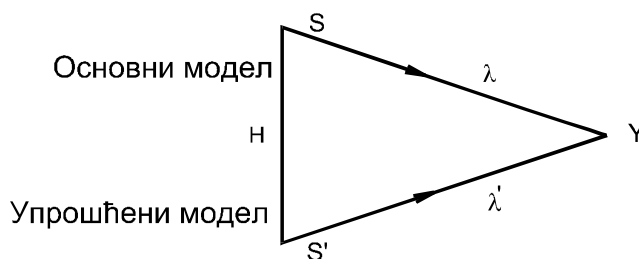


Слика 4.3. Очување функције прелаза стања

Дакле, полазећи од стања  $s$  основног модела може се прво вршити пресликавање  $H$  да би се дошло у стање  $s'$  упрошћеног модела, а затим применити функција прелаза стања упрошћеног модела да би се дошло у стање  $\delta'(s')$ ; исто тако могуће је прво применити функцију прелаза стања основног модела (прелаз из стања  $s$  у  $\delta(s)$ ), а затим пресликавање  $H$ . У оба случаја долази се у исто стање. На основу изложеног може се закључити да пресликавање  $H$  које важи између стања основног и упрошћеног модела мора важити и између њихових будућих стања која се добијају на основу функције прелаза стања.

### 3. Очување излазне функције

Нека је  $s$  стање основног модела и нека је могуће применом пресликавања  $H$  прећи у стање  $s'$  упрошћеног модела. Примена излазне функције основног модела (за оне експерименталне услове за које је извршено упрошћавање  $(\lambda)$  да би се добио упрошћени модел за стање  $s$ ), даје исти излаз  $Y$  као и примена излазне функције упрошћеног модела  $(\lambda')$  на стање  $s'$ . Одговарајући комутативни дијаграм дат је на слици 4.4. Овде се јасно уочава да се валидност упрошћеног модела утврђује и у односу на задате експерименталне услове. Уколико је пресликавање  $H$  типа 1:1, у питању је *изоморфизам*.



Слика 4.4. Очување излазне функције

Ако су задовољени сви претходно дефинисани услови, онда је упрошћени модел *валидан* у односу на основни модел, за дате експерименталне услове. Даље, ако је раније већ било утврђено да је основни модел валидан у односу на реални систем (симулирани систем), тада је и упрошћени модел валидан у односу на реални систем.

Целокупан процес валидације захтева одређено време, труд и новчана средства. Ови се фактори свакако морају узети у обзир у поступку валидације и доношењу одлуке колико ће се далеко отићи у процесу ревизије модела. Најчешће, моделар одређује максимални прихватљиви ниво одступања модела од реалног система. Уколико се овај ниво одступања не може постићи уз улагање ограничених буџетских средстава, намењених процесу валидације, моделар или смањује очекивану прецизност модела или одбацује дати модел.

## 4.2 Верификација симулационих модела

Иако у досадашњим разматрањима није било речи о конкретној имплементацији симулационог модела неким програмским језиком, тј. о рачунарском кодирању модела, овде ће укратко бити указано на неке најважније карактеристике процеса верификације модела, који, као што знамо, полази баш од рачунарског кода модела. Тим пре што процес тестирања исправности једног модела, најчешће преплиће процесе верификације и валидације, односно моделар, као што смо већ напоменули, та два испитивања најчешће врши симултано. Дакле, процес *верификације* треба да покаже да ли је и у којој мери, концептуални модел на одговарајући начин

представљен рачунарским кодом, односно у којој се мери слажу концептуални (претпоставке за компоненте система и структуру; вредности параметара; апстракције и поједностављења у моделу) и рачунарски код.

У поступку верификације нема стандардног рецепта. Зато је потребно извршити више различитих провера:

- ◆ Ручна верификација логичке исправности: модел се извесно време пропушта на рачунару и ручно, а потом пореде добијени резултати.
- ◆ Модуларно тестирање: појединачно тестирање сваког модула како би се установило да ли даје разумне излазе за све могуће улазе.
- ◆ Провера у односу на позната решења: подесимо модел тако да представља систем чија су решења позната и упоређујемо их са резултатима модела.
- ◆ Тестирање осетљивости: варирамо један параметар, док остали остају непромењени и проверавамо да ли је понашање модела осетљиво на промену тог параметра.
- ◆ Тестирање на поремећаје: постављамо параметре модела на неприродне вредности и проверавамо да ли се модел понаша на несхватљив начин. На тај начин се могу открити грешке у програму које је врло тешко уочити на други начин.

Осим наведеног, могу се применити и неки други познати поступци који се користе при отклањању грешака у рачунарским програмима.



## СРЕДСТВА ЗА СИМУЛАЦИЈУ

Као средства за симулацију користе се три врсте рачунара:

- ◆ аналогни рачунар,
- ◆ дигитални рачунар,
- ◆ хибридни рачунар.

### 5.1 Аналогни рачунар

Давно је уочена чињеница да се различити физички објекти могу описивати истим математичким моделом. Између два физичка модела А и Б, који имају исте математичке моделе, каже се да постоји математичка аналогија. Истоветност математичких модела објеката А и Б пружа могућност да један од физичких објеката буде коришћен за анализу математичког модела другог објекта. Физички објект А, који се користи за анализу математичког модела објекта Б, са којим има сличан математички модел, назива се аналогни модел.

Према томе, између физичког објекта који се испитује и аналогног модела постоји математичка аналогија (аналогија у понашању).

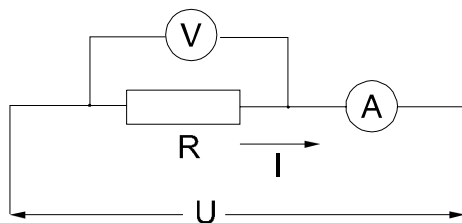
Идеја аналогних рачунара (односно свих аналогних рачунских средстава) састоји се у изналажењу таквих физичких објеката који ће моћи да се користе за анализу математичких модела помоћу којих су ти објекти математички описани. Према томе, принципи рада аналогних рачунара засновани су на физичким законима који важе за онај физички објект који се користи у рачунске сврхе. Следећи пример илуструје аналогни начин рачунања (*Пејовић и Парезановић, 1972*).

*Пример:* Претпоставимо да је потребно да се изврши множење две величине  $x$  и  $y$ . Према томе, задати математички модел је

$$z = xy \quad (5.1)$$

За електрично коло приказано на слици 5.1. важи Омов закон, који даје везу између напона ( $U$ ), струје ( $I$ ) и отпора ( $R$ ), тако да је

$$U = RI \quad (5.2)$$



Слика 5.1. Електрично коло

Како електрично коло на слици 5.1. има математички модел (5.2) који је идентичан са математичким моделом (5.1), само што ознаке у моделу (5.1) представљају математичке величине, а у моделу (5.2) физичке величине, то електрично коло на слици 5.1. може да представља аналогни уређај за множење.

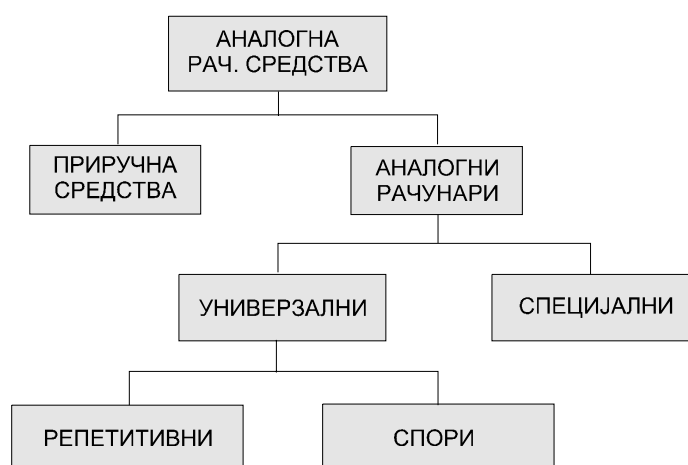
Из овог примера се јасно виде основне особине аналогног рачунања. Наиме, за дати математички модел који је предмет рачунања, налази се одговарајући физички објект, чије се понашање описује истим или сличним математичким моделом, као што је и онај што се изучава. Свака математичка величина у математичком моделу има своју представу у аналогном моделу, као одговарајућа физичка величина. Према томе, неке од физичких величина одговарају познатим величинама, а неке непознатим величинама у математичком моделу. Давањем одговарајућих вредности првима, посредством физичких закона, добијају се друге физичке величине, које уствари представљају тражена решења математичког модела.

У аналогна рачунска средства спадају приручна средства, као што су, на пример, логаритмар и аналогни рачунари. Аналогни рачунари (АР) деле се најчешће у две групе и то (слика 5.2):

- ♦ специјални аналогни рачунари
- ♦ универзални аналогни рачунари.

Специјални аналогни рачунари граде се у одређене сврхе и служе за анализу једног или евентуално мањег броја специфичних математичких модела.

Универзални аналогни рачунари служе за анализу разних математичких модела. Они се састоје из више рачунских компоненти, при чему свака од њих може да обави једну или више математичких операција.



Слика 5.2. Подела аналогних рачунских средстава

Универзални аналогни рачунари граде се као (слика 5.2):

- ♦ репетитивни
- ♦ спори.

Репетитивни АР раде великом брзином и на њима се решење обично понавља 20 до 100 (код савременијих и до 1000) пута у секунди. Велика учестаност решења омогућава његово визуелно праћење на екрану осцилоскопа (уређаја који на екрану показује промену напона у времену) у виду криве линије.

Спори АР раде у тзв. реалном времену, тј. тако што се решење добија само у једном циклусу рада рачунара. Да би решење било трајно записано, обично се црта на специјалном XY писачу.

Савременији АР могу да раде комбиновано - било као спори, било као репетитивни.

Савремени аналогни рачунари граде се искључиво као електронски аналогни рачунари и као такви за нас су интересантни са становишта симулације континуалних система, односно решавања диференцијалних једначина.

Наглим развојем електронике и појавом електронских цеви, четрдесетих година овог века, а нарочито појавом транзистора и технологије интегрисаних кола, омогућени су услови за развој аналогних рачунара који свој рад заснивају на аналогiji између понашања електронских кола која свој рад заснивају на физичким законима и њиховим описом помоћу диференцијалних једначина. Уколико је потребно симулирати неку појаву која се може описати диференцијалном једначином, тада се приступа представљању те диференцијалне једначине помоћу електронских склопова, док се решења диференцијалне једначине добијају на основу мерења напона и струја у тим електронским склоповима.

Електронски аналогни рачунари (ЕАР) располажу са одговарајућим бројем електронских рачунских компоненти које могу да обављају рачунске операције сабирања, множења, интеграције и слично. За сваки математички модел, ове компоненте морају на одговарајући начин бити повезане. Посао који се састоји у повезивању рачунарских компонената у циљу постављања конкретног математичког модела на рачунару, назива се програмирање. Математички модели који се најчешће решавају на аналогним рачунарима су системи диференцијалних једначина, па се зато ови рачунари називају често диференцијални анализатори.

Рачунски елементи којима су опремљени савремени аналогни рачунари деле се у шест група. Подела је извршена према функцији рачунских елемената:

1. Множење променљиве величине константом
2. Алгебарско сабирање више променљивих величина
3. Интеграција једне или више функција
4. Множење и дељење променљивих величина
5. Генерисање функција
6. Логичке операције.

Главни елементи електронског аналогног рачунара су:

1. Напонски извор (константа)
2. Потенциометар (улаз множи са константом мањом од 1)
3. Појачивач (улаз множи са константом већом од 1)
4. Интегратор (излазни напон је интеграл улазног)
5. Разни диодни ограничавачи (нелинеарни елементи)
6. Диодни генератори нелинеарних функција.

### 5.1.1 Потенциометар

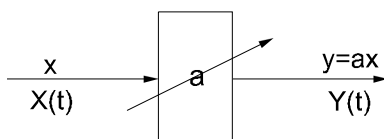
Потенциометар је рачунски елемент ЕАР који омогућава да се реализује математичка операција множења променљиве величине са константом, тј.

$$y(t) = ax(t) \quad (5.3)$$

где је  $a$  константа, а  $x(t)$  променљива. Потенциометар се реализује као омски отпорник са клизачем, на чије се крајеве доводи променљиви напон  $X(t)$ , а са клизача, који може да се постави у ма који положај између крајева потенциометра, одводи се напон  $Y(t)$ . По природи овог уређаја мора да буде

$$Y(t) \leq X(t)$$

Према томе, потенциометар омогућује множење променљивог улазног напона  $X(t)$  са константом мањом од јединице. Вредност константе одређена је положајем клизача на скали потенциометра.



Слика 5.3 Шематски приказ потенциометра

Савремени електронски аналогни рачунари имају потенциометре који су снабдевени скалама од 0 до 1. Према овој скали лако се бира жељени положај клизача (вредност константе), али се

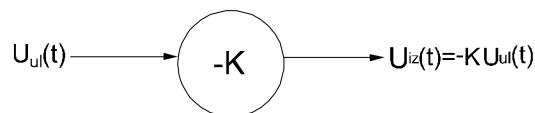
потенциометар не може никад подесити идеално тачно, јер свако постављање клизача на неку дужину  $l$  уноси извесну грешку  $\Delta l$ . Закључујемо да је грешка код подешавања потенциометра мања, што је дужина отпорника већа.

Потенциометар се код аналогних рачунара користи у две сврхе:

- ♦ за добијање константних напона у циљу постављања почетних услова у рачунарским моделима
- ♦ за множење променљивог напона са позитивном константом мањом од јединице.

### 5.1.2 Појачавач

Код електронских аналогних рачунара најважнију улогу има појачавач, који се назива још и операциони или рачунски појачавач. Појава овог елемента је омогућила настанак електронских аналогних рачунара и зато се с правом он третира као главни рачунски елемент аналогних рачунара. Најпре се реализовао у цевној технологији применом диференцијалних појачивача, да би касније са развојем електронике овакав вид реализације заменила реализација у транзисторској технологији. Данас се операциони појачивачи реализују искључиво у форми интегрисаних кола.



Слика 5.4 Шематски приказ појачавача

Операциони појачавач је у ствари електронски уређај који има врло велико појачање, односно појачава улазни сигнал (улазни напон), тако да се на излазу добија напон знатно појачан у односу на онај на улазу.

$$U_{iz} = -k U_{ul}, \quad k \gg 1 \quad (5.4)$$

Поред тога, операциони појачавач има велику улазну, а малу излазну отпорност. Помоћу њега се могу обављати разне математичке операције и у зависности од тога са којим пасивним

рачунским елементом је спрегнут, он носи назив сабирач, интегратор, итд.

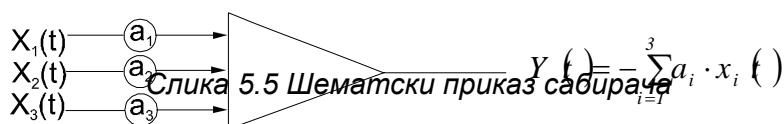
### 5.1.3 Сабирач

Математичка операција алгебарског сабирања  $n$  променљивих величина коју желимо да извршимо, дата је изразом:

$$y(t) = -\sum_{i=1}^n a_i x_i(t) \quad (5.5)$$

где су  $a_i$  константе којима се множи одговарајућа променљива величина  $x_i(t)$ .

Рачунски елемент који омогућава реализацију математичке операције (5.5) назива се сабирач. Физичке величине  $X_i(t)$ , ( $i = 1, 2, \dots, n$ ) које одговарају променљивим  $x_i(t)$  ( $i = 1, 2, \dots, n$ ) су улазни напони сабирача. Напон  $Y(t)$  као излазна величина из сабирача одговара математичкој величини  $y(t)$ . Сабирач се реализује коришћењем одређеног броја отпорника и увођењем негативне повратне спреге.



Сабирач код електронских аналогних рачунара може да обави две математичке операције:

- ♦ множење улазног напона са константом и
- ♦ алгебарско сабирање више улазних напона.

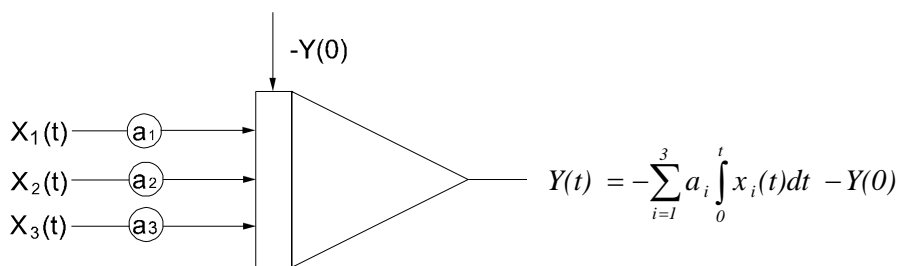
Сабирач је могуће користити и као инвертор, тако што ће се довести само један улаз са константом  $k_I = 1$ .

### 5.1.4 Интегратор

Математичка операција која се жели реализовати помоћу интегратора има облик:

$$y(t) = -a \int_0^t x(t) dt \quad (5.6)$$

где је  $a$  константа (иста ознака  $t$  употребљена је за горњу границу интеграла и независну променљиву у подинтегралној функцији, јер је време  $t$  једина независна променљива на рачунару).



Слика 5.6 Шематски приказ интегратора

Физички елемент који се код аналогних рачунара користи за обављање операције интеграције је кондензатор. При овоме се користи познати закон физике о односу између струје, напона и капацитета кондензатора који гласи: ако је на крајевима кондензатора капацитета  $C$  доведен напон  $U$ , онда је струја  $I$  кроз кондензатор одређена односом:

$$I(t) = C \frac{dU(t)}{dt} \quad (5.7)$$

Интегратор код аналогних електронских рачунара реализује се помоћу отпорника, кондензатора и операционог појачивача са повратном спрегом. Почетни услов за интегратор реализује се тако што кондензатор у повратној спрези интегратора у тренутку  $t = 0$  мора бити напуњен количином електрицитета  $Q_0$ , тако да на његовим крајевима постоји напон:

$$Y(0) = \frac{Q(0)}{C} \quad (5.8)$$

Постављање интеграционе константе код аналогних рачунара може се вршити на два начина:



- ♦ пуњењем кондензатора у повратној спрези интегратора пре почетка интеграције и
- ♦ додавањем интеграционе константе после извршене интеграције.

Као што смо видели, операционим појачивачем се могу вршити разне линеарне математичке операције. Свака таква операција важи под извесним ограничењима које појачивач као електронски уређај има. Ова ограничења најчешће уносе одређене грешке у рачун, које се могу сврстати у следеће групе:

- ♦ грешке услед коначног појачања
- ♦ грешке услед шетања нуле (дрифта)
- ♦ грешке услед коначне улазне струје
- ♦ грешке услед кашњења на вишим фреквенцијама
- ♦ случајне грешке.

У овом поглављу нећемо се бавити анализом наведених грешака, пошто та материја по свом обиму превазилази оквире ове књиге, а заинтересовани читалац упућује се на рад (Пејовић и Парезановић, 1972).

### 5.1.5 Множач

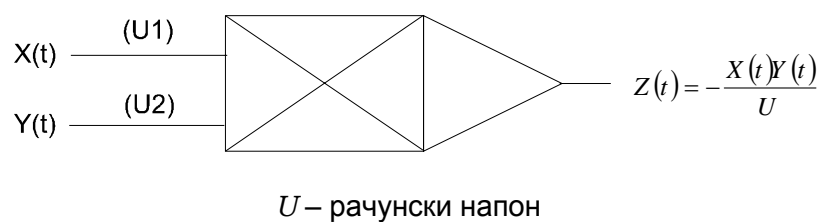
Нека је математичка операција множења две променљиве дата изразом

$$z(t) = x(t)y(t) \quad (5.9)$$

где су  $x(t)$  и  $y(t)$  променљиве величине са временом. Рачунски елемент на електронском аналогном рачунару који може да обавља математичку операцију (5.9) назива се множач. Физичке величине  $X(t)$  и  $Y(t)$ , које одговарају променљивим величинама  $x(t)$  и  $y(t)$ , називају се улазни напони у множач, а физичка величина  $Z(t)$  која се јавља на излазу множача, а која одговара променљивој  $z(t)$ , назива се излазни напон.

Код аналогних електронских рачунара, множење две променљиве величине може се обављати на више начина, али се сви типови множака могу поделити у две главне групе и то:

- ◆ множачи који користе електромеханичке методе и који се називају сервомножачи и
- ◆ множачи који користе електронске методе и који се називају електронски множачи.



Слика 5.7 Шематски приказ множака

### 5.1.6 Сервомножач

Идеја сервомножака потекла је од особине потенциометра да улазни напон множи са неком константом мањом од јединице. За реализацију множака било је потребно направити електромеханички уређај, који може да помера клизач потенциометра у зависности од неке друге променљиве величине, у нашем случају  $Y(t)$ . Код ове врсте множака за покретање клизача потенциометра се користи тзв. позициони сервомеханизам, па отуда и име сервомножач.

Позициони сервомеханизам се састоји од дискриминатора, појачивача, електромотора, механичког редуктора и потенциометра.

### 5.1.7 Диодни множачи

Диодни множачи припадају групи електронских множака, а састављени су од диодних генератора неких фиксних функција и рачунских појачивача. Најчешће се користе диодни генератори функција који дају следеће фиксне функције:

$$F_1(X) = X^2$$

$$F_2(X) = \ln X$$

$$F_3(X) = e^X$$

Први тип диодног електронског множача базира се на основној једначини:

$$\frac{1}{4}[(x+y)^2 - (x-y)^2] \equiv xy \quad (5.10)$$

Према томе, да бисмо остварили множење две величине  $x$  и  $y$ , можемо остварити разлику квадрата збира и разлике те две величине и добијену вредност поделити са четири.

Други тип електронског диодног множача базира се на основној једначини:

$$xy = e^z \quad (5.11)$$

где је  $z = \ln x + \ln y$ . Десна страна ове релације може се остварити ако се располаже са два генератора логаритамских функција и једним генератором  $e^x$  функције.

### 5.1.8 Генератори функција

Често се у математичким проблемима јавља потреба да се нека величина, рецимо  $y$ , изрази као функција неке друге величине, рецимо  $x$ . У математичкој нотацији та зависност се може изразити на следећи начин:

$$y = f(x) \quad (5.12)$$

где оператор  $f$  означава закон по коме величина  $y$  зависи од величине  $x$ , а  $x$  и  $y$  могу бити и функције времена. У аналогној рачунарској техници, физички уређај који може да оствари математичку операцију (5.12) назива се генератор функција. Према томе, математички модел који треба да има генератор функција биће:

$$Y = f(X) \quad (5.13)$$

Оператор  $f$  може се изразити било у математичком облику, као нека аналитичка функционална зависност, било као нека графичка зависност (крива линија) или као табела парова дискретних вредности за  $x$  и  $y$ . У аналогној рачунској техници данас се користе две групе генератора функција. Прву групу представљају генератори специјалних, фиксних функција, код којих се жељена функција поставља једном за свагда. Друга група су универзални генератори функција који се могу по жељи подешавати на ону функцију за коју се тренутно укаже потреба приликом решавања одређеног проблема.

Генератори функција раде тако да жељену функцију (5.12) апроксимирају на неки одређени начин:

$$\bar{Y} = \bar{f}(X) \quad (5.14)$$

Један тип генератора функцију (5.12) апроксимира помоћу праволинијских одсечака тако да се уместо дате функције јавља полигонална изломљена линија. Други тип генератора апроксимира функцију (5.12) помоћу степенасте линије.

Диодни генератор функција састоји се од диодних елемената, који најчешће раде у спрези са рачунским једносмерним појачивачима. Диода се, у ствари, понаша као нека врста вентила и служи да се генератор пребаци из једног радног стања у друго. Диода је састављена од две електроде, аноде и катодe. Када је електрични потенцијал на аноди диоде већи од потенцијала на катоди, онда диода проводи струју и понаша се као проводник. Ако је пак потенцијал катодe виши од потенцијала аноде, онда диода не проводи струју, већ се понаша као изолатор. Ова се особина на адекватан начин користи при конструисању елемента за генерисање функција.

Наведени елементи представљају основне компоненте електронског аналогног рачунара. Сваки овакав рачунски елемент, као што смо видели, конструисан је тако да изврши одређену елементарну математичку операцију. Решавање сложених математичких задатака обавља се одговарајућим повезивањем потребног броја рачунских елемената, а у зависности од облика математичког модела који се жели анализирати помоћу рачунара.

### 5.1.9 Поступци при раду са аналогним рачунаром

Први корак у аналогном моделовању и симулацији је да се коректно формулише математички модел физичког објекта. На основу математичког модела, програмер саставља блок дијаграм. На аналогном рачунару се могу решавати само они проблеми који имају одређене конкретне вредности свих математичких величина које улазе у математички модел. Када је реч о диференцијалним једначинама, то значи да сви почетни услови, као и сви коефицијенти, морају бити одређени нумерички и постављени на аналогном рачунару.

Код аналогних рачунара, на основу блок шеме, врши се повезивање рачунарских елемената помоћу електричних водова. Електронски аналогни рачунари конструисани су тако да су све улазне и излазне везе са рачунарским елементима изведене на једној централној табли. На ову таблу се поставља покретна програмска плоча, на којој су помоћу проводника успостављене одговарајуће везе између рачунских елемената.

Добра страна овакве представе је да се решења нарочито за алгебарске једначине добијају готово тренутно. Лоша страна је да је максимална тачност решења ограничена тачношћу реализације електронских елемената, тачношћу мерних инструмената и мерним методама на основу којих добијамо решења. Због наведених разлога сматра се да је максимална тачност аналогног рачунара ограничена грешком од приближно 1% .

## 5.2 Дигитални рачунар

Структура класичног дигиталног рачунара који се састоји од процесора, меморије и улазно-излазних уређаја је добро позната и обрађена у литератури (*Spriet & Vansteenkiste, 1982*). Дигитални рачунари имају велику тачност рачунања (која у пракси иде до 20 значајних цифара) али и незгодну особину да се за решавање алгебарских једначина користе итерационе методе које у већини случајева траже доста рачунарског времена. Процес интеграције мора да се реализује преко неке од алгебарских шема, што такође троши доста рачунарског времена. Укратко, дигитални рачунар може бити доста тачан при решавању симулационих

проблема, али у неким случајевима време одређивања резултата може бити сувише дуго.

Структура симулационих модела који се симулирају на дигиталном рачунару може бити врло сложена и састављена од низа паралелних процеса који међусобно координирају и синхронизују своје акције. Симулација таквих модела на класичном једнопроцесорском рачунару захтева врло сложене софтверске концепте за синхронизацију и координацију, па је заступљеност симулационих метода тиме била ограничена. Да би се наведена ограничења премостила, развојем технологије хардвера и софтвера уведени су нови концепти који су омогућили смањење трошкова изградње и експлоатације симулационих модела и омогућили брз развој симулације.

Хардверски новитети који су значајни за развој симулације и проширење делокруга примене су:

- ◆ Векторски процесори
- ◆ Рачунари са више процесора
- ◆ Јефтине и брзе динамичке меморије
- ◆ Јефтине и брзе масовне меморије
- ◆ Појава брзих графичких процесора.

Развој софтверских компоненти које имају значајан утицај на развој симулације су у следећим областима:

- ◆ Развој вештачке интелигенције
- ◆ Развој алгоритама паралелног процесирања
- ◆ Рачунарске мреже (оперативни системи за мреже)
- ◆ Развој мултимедијалних технологија

Развој симулационих метода и техника као и делокруг примене симулације данас је већ увелико одређен и очекује се да ће се и убудуће кретати у следећим правцима:

1. Спецификација модела у облику говорног језика и његово превођење у формални модел уз помоћ експертног система. При спецификацији модела користе се базе података и базе знања на локалном, државном или глобалном нивоу.
2. Имплементација симулационог модела на вишепроцесорским рачунарима који су међусобно повезани у рачунарске мреже.

Надзор дистрибуције ресурса врше оперативни системи који свој рад заснивају на размени порука. При томе се интензивно користе алгоритми за паралелно процесирање. Ток симулације се прати на графичким терминалима као слика реалног система. Ток симулације моделар прати интерактивно и интервенише кад год је потребно.

3. Анализа резултата симулације врши се уз помоћ експертних система коришћењем одговарајућих база знања. Добијени резултати приказују се на графичким терминалима у облику слика стања реалног система док се статистички показатељи анализе дају у облику дијаграма. Резултати симулације се дају као пречишћене информације о могућим пословним одлукама и њиховим последицама.

Развој локалних рачунарских мрежа и наметнути IBM-PC стандард у хардверској и софтверској реализацији малих рачунара већ данас омогућавају јефтину и ефикасну примену симулације у малим и средњим предузећима. Масовна појава јефтиних PC-компатибилних рачунара има изузетан значај за образовање јер омогућава просечном студенту коришћење готово свих познатих симулационих језика као и алата вештачке интелигенције уз прихватљиве трошкове, што је донедавно била привилегија само великих фирми и института.

Могуће је очекивати у ближој будућности даље приближавање метода вештачке интелигенције и симулације и реализацију нових хардверских и софтверских концепата који би то омогућили (нпр: хардверска реализација неуронских мрежа) који би били довољно распрострањени и комерцијални за употребу у симулацији пословних система.

### 5.3 Хибридни рачунар

Хибридни рачунар је спој аналогног и дигиталног рачунара изведен на такав начин да се предности сваког од њих користе у потпуности док се недостаци међусобно анулирају. Код хибридног рачунара аналогни рачунар изводи имплицитне операције и извршава интеграљење док дигитални рачунар извршава алгебарске функције.

Због високе цене хибридних рачунара они се користе углавном за симулацију у авионској индустрији и управљању летелицама, где је од изузетне важности одређивање решења диференцијалне једначине у кратком временском интервалу.

При избору конкретног рачунарског средства за симулацију моделованог система потребно је да се води рачуна о оптималном односу брзине решавања и цене рачунарских система. Такође је од значаја и тачност резултата симулације коју је могуће остварити одређеним средством за симулацију. Из тих разлога, често се за решавање симулационих задатака, поред универзалних рачунских система, равноправно користе и специјализована средства рачунске технике. Област у којој постоје знатне могућности за смањење трошкова и повећање брзине моделовања и симулације система, а уједно и добијање довољно прецизних решења симулације, је област хибридне рачунарске технике.

Хибридна обрада података представља комбинацију паралелне и секвенцијалне обраде података. Таквом начину обраде одговара тзв. хибридни процесор. Хибридни процесор је хардверско-софтверски систем који може да врши и паралелну и секвенцијалну обраду. Састоји се од паралелног (аналогног) процесора и једног или више секвенцијалних процесора. Циљ оваквог повезивања је да се добре стране оба начина обраде података повежу и да се, колико је то могуће, елиминишу ограничења која извесно сваки од наведена два начина обраде поседује. Другим речима, хибридна обрада је покушај да се комбинују, с једне стране, тачност и могућност меморисања података дигиталног рачунара и с друге стране, велика брзина извођења операција аналогног рачунара.

За време паралелне обраде, обрада свих података почиње и завршава се у истом тренутку. Ова се обрада обавља на аналогном процесору, који сачињава више оперативно независних подпроцесора.

Секвенцијална обрада може се дефинисати као обрада коначног броја података у низу (један за другим) и обавља се најчешће на само једном дигиталном процесору.

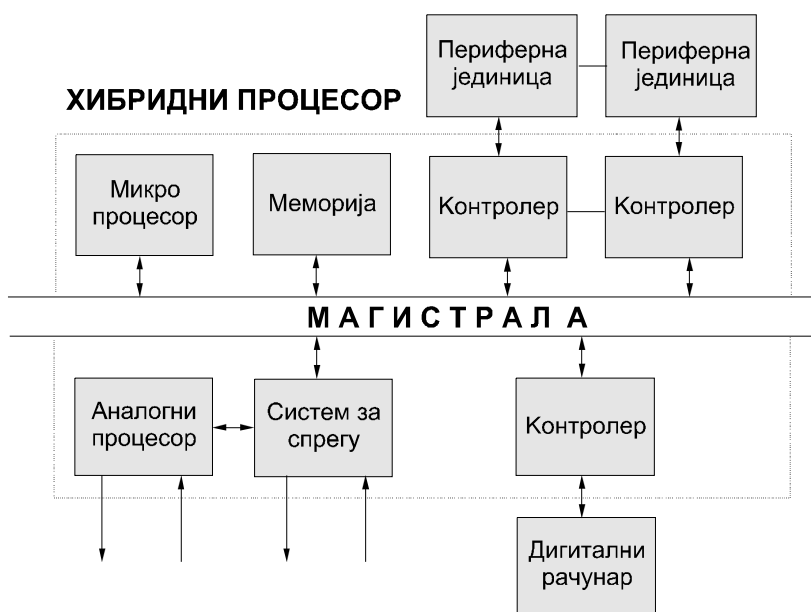
Постоји мноштво начина за повезивање аналогних и дигиталних принципа обраде и конструисање хибридних система. Типичан хибридни рачунарски систем састоји се од три повезана дела:



1. Самосталног дигиталног рачунара са свом стандардном периферном опремом коју једна таква инсталација подразумева,
2. Једног аналогног рачунара одговарајуће величине и
3. Интерфејса који повезује та два рачунара.

Интерфејс је неопходан због различитог начина функционисања и различитог начина представљања података у оба рачунара.

Хибридни рачунски систем са паралелним аналогним процесором као централном јединицом и помоћним дигиталним мини- или микропроцесором може се дефинисати као аналогно-оријентисани хибридни рачунски систем или *хибридни процесор*. Организација једног таквог хибридног процесора приказана је на слици 5.8.



Слика 5.8 Организација хибридног процесора

Микропроцесор управља решавањем хибридног задатка, задаје режиме аналогним рачунским јединицама и јединицама паралелне логике, активира AD и DA конверзију и управља радом периферних јединица. Осим тога, микропроцесор може да обавља и нумеричка израчунавања што рачунару даје карактер хибридног система

способног за решавање, у реалном времену, симулационих задатака средњих размера.

Што се тиче аналогног дела хибридног процесора, од њега у великој мери зависи ефикасност коришћења хибридних рачунарских система. У хибридним процесорима, проблем аутоматизације повезивања аналогних рачунских јединица може се задовољавајуће решити, што није био случај код ранијих хибридних рачунарских система. Елиминацијом ручног програмирања аналогног дела, створене су нове могућности за примену хибридних рачунара.

Хибридни процесор може бити повезан са већим дигиталним рачунаром помоћу посебног контролера. На овај начин добија се мултипроцесорска структура у којој систем за спрегу и аналогни део могу бити управљани од стране микропроцесора и од стране дигиталног рачунара.

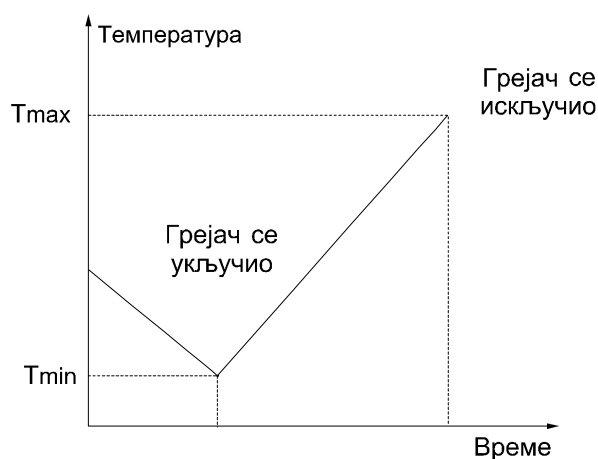
Систем за спрегу омогућава повезивање аналогног и дигиталног дела хибридног рачунског система у циљу размењивања нумеричких података и контролних функција. У односу на систем за спрегу, размена информација може бити у оба смера.

Хибридни рачунски системи представљају веома ефикасно средство за решавање проблема симулације како у погледу цене, тако и у погледу времена потребног за решавање и тачности добијених резултата. Технолошки напредак у овој области омогућава развој хибридних рачунских система четврте генерације који, с обзиром на модуларну организацију, могу решавати симулационе задатке неколико десетина пута ефикасније него постојећи дигитални рачунари. Ови системи могу решавати и проблеме симулације који се због своје сложености уопште не могу решавати на дигиталним рачунарима.

## СИМУЛАЦИЈА КОНТИНУАЛНИХ СИСТЕМА

Симулација континуалних система (СКС) односи се на експериментисање са моделима система чија се стања мењају континуално. Ови системи су већином динамички и могу бити детерминистички или стохастички. Најчешће се представљају одређеним бројем диференцијалних једначина, које се решавају нумеричким путем, како би се одредило понашање модела. Време је најчешћа независна променљива у овим моделима. У принципу, могуће је решавати и детерминистичке и стохастичке диференцијалне једначине (моделе) коришћењем језика за континуалну симулацију.

Посматрајмо као пример рад електричног грејача са термостатом. Када се укључи, он загрева просторију све док се не достигне одређена температура, подешена на термостату. Када се та температура постигне, термостат искључује грејач. Слика 6.1 приказује промену температуре просторије у времену.



Слика 6.1. Функционисање грејача са термостатом

Повећање температуре просторије и њено постепено смањивање које настаје након искључивања грејача, представља континуални процес који се може представити одговарајућим диференцијалним једначинама и потом симулирати.

## 6.1 Формални модел

Математички модел за симулацију континуалних система може се представити шесторком :

$$M = (U, Y, S, \delta, \lambda, S_0)$$

где су:

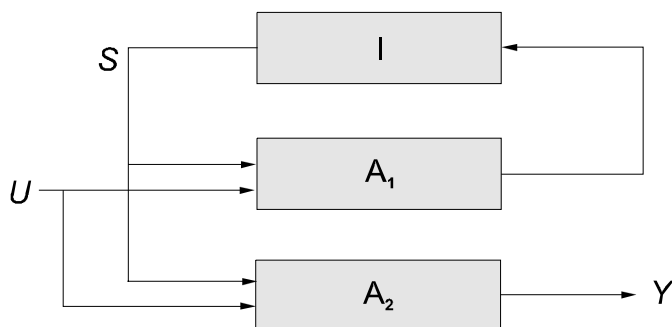
- $U$  - скуп улаза
- $Y$  - скуп излаза
- $S$  - скуп променљивих стања
- $\delta$  - функција преноса  $\delta : U \times S \rightarrow S$
- $\lambda$  - функција излаза  $\lambda : U \times S \rightarrow Y$
- $S_0$  - скуп почетних стања (почетних услова)

Процес симулације континуалних система може се описати једначинама *континуалног аутомата* :

$$\begin{aligned} S(t) &:= I \times A_1 \cdot \{U(t), S(t)\} \\ Y(t) &:= A_2 \cdot \{U(t), S(t)\} \end{aligned}$$

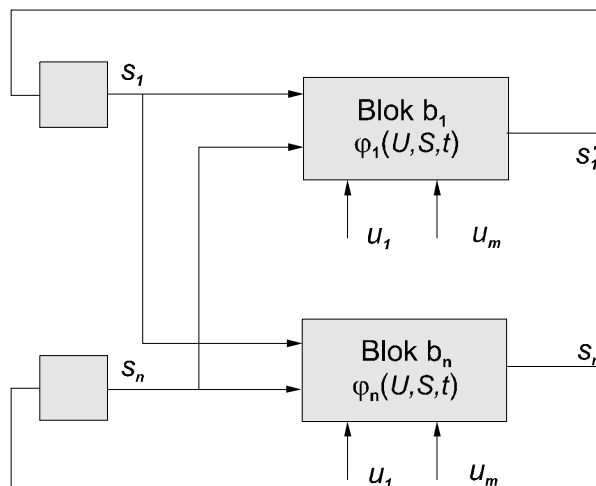
Преносна функција, у горњим једначинама, раздељена је у две врсте оператора: оператор  $A_1$  и  $A_2$  су алгебарске функције, док оператор  $I$  представља интеграцију.

Приказ рада континуалног аутомата дат је на слици 6.2. Скуп алгебарских операција датих изразом  $A_1 \{U(t), S(t)\}$  и  $A_2 \{U(t), S(t)\}$  се рачунају у првој фази, а затим се интеграцијом или применом функције временског кашњења, добија нови скуп вредности променљивих стања. Потом се цео процес обнавља за нови корак времена.



Слика 6.2. Структура коначног аутомата СКС

Основна разлика између аналогне и дигиталне симулације континуалних система није у структури већ у процедури. Аналогни рачунар је способан за решавање рекурзивних релација датих горњом једначином, директно као резултат динамичких процеса чије је "стабилно стање" захтевана функција, док дигитални рачунар мора применити итеративну процедуру. Друга разлика настаје из чињенице да се у аналогном рачунару, све операције представљене операторима  $I$ ,  $A_1$  и  $A_2$  извршавају истовремено; док дигитални рачунар најпре мора да изврши све алгебарске операције, а потом све интеграције, једну за другом.



Слика 6.3. Блок дијаграм детаљне структуре СКС

Посматране излазне променљиве често су подскуп променљивих стања, односно  $Y \subset S$ . Ако разложимо оператор  $A_I$  на његове елементарне или примитивне функције, које могу да буду представљене алгебарским блоковима, добијамо детаљнију структуру процеса симулације континуалног система (слика 6.3). Скуп улаза и скуп променљивих стања ( $U$  и  $S$ ), су одређени векторским функцијама:  $U = [y_1(t), \dots, y_m(t)]$  и  $S = [s_1(t), \dots, s_n(t)]$ . Овакав модел представља систем диференцијалних једначина првог реда.

## 6.2 Функција једне или више променљивих - блок

Функција једне променљиве  $\varphi$  је пресликавање непразног скупа  $X$ , променљивих  $x$ , званог домен, у непразан скуп  $Y$ , променљивих  $y$ , званог опсег (или кодомен, скуп вредности функције). Симболички се представља са:

$$\varphi : X \rightarrow Y$$

Функција више променљивих  $\varphi$  је пресликавање облика

$$\varphi : X \times X \times X \times \dots \times X \rightarrow Y.$$

Блок се може представити уређеном тројком

$$b = (\varphi, X, Y)$$

За различите блокове који имају исту ф-ју  $\varphi$ , каже се да припадају истом типу. Свако  $x \in X$  назива се *улаз блока*, док се  $y \in Y$  назива *излаз блока*.

## 6.3 Апстрактни континуални симулациони систем

Нека је  $V$  скуп променљивих,  $t$  време симулације и  $B$  скуп блокова. Тада се тројка

$$KSS = (B, V, t)$$

назива *апстрактни континуални симулациони систем (KSS)*

### 6.3.1 Подскупови KSS-а

Нека су:  $X$  скуп свих улаза једног блока и  $Y$  скуп свих излаза тога блока. Тада се скуп свих променљивих  $V$  може поделити у три дисјунктна подскупа и то:

$$\begin{array}{ll} \text{скуп улаза} & U = X - Y \\ \text{скуп излаза} & R = Y - X \\ \text{скуп веза} & C = X \cap Y . \end{array}$$

### 6.4 Функције стања, алгебарске функције, променљиве стања

Две следеће функције:

$$\begin{aligned} INT: \quad y(t_{n+1}) &= y(t_n) + \int_{t_n}^{t_{n+1}} x(t) dt \\ DELAY: y(t_n) &= x(t - \tau), \quad t_n > \tau \end{aligned}$$

називају се *функције стања*.

Све друге функције (осим горе наведених) називају се *алгебарске функције*. Променљиве које представљају излазе блокова за претходно наведене функције, називају се *променљиве стања*. Скуп свих променљивих стања  $S$  је

$$S = \{y \in Y \mid y = INT(x) \vee y = Delay(x)\}.$$

Променљиве стања могу бити из скупа веза ( $C$ ) или скупа излаза ( $R$ ), тј.

$$S \subset C \cup R$$

### 6.5 Уредљивост

Најосновније својство било ког нетривијалног математичког модела КСС је *повратна спрега*. Модел има повратну спрегу ако је немогуће означити све блокове  $b \in B$  тако да, ако је излаз блока

$b_i$  везан на улаз блока  $b_j$ , буде  $i < j$ . Појава затворене повратне спреге намеће неопходан услов за израчунљивост математичког модела, који се назива *услов уредљивости*, а дефинише се на следећи начин:

Скуп  $A$  свих пребројивих алгебарских блокова (блокова који одговарају алгебарским функцијама) модела  $M$  назива се *уредљивим* ако сви чланови  $a_j \in A$  могу бити уређени (сортирани) у линеарну листу такву да су улази сваког члана  $a_j$ , елементи неког од следећих скупова:

1. скупа улаза  $U$
2. скупа стања  $S$
3. подскупа  $C' \subset C$  који је дефинисан као

$$C' = \{c \in C \mid \forall i < j, \forall a_i = (\varphi_i, X_i, Y_i): c \in X_i\}$$

(тј. скуп свих повезаних променљивих које су улази алгебарских блокова који претходе члану  $a_j$  у листи).

## 6.6 Симулација континуалних система помоћу аналогног рачунара

Постоје две основне класе електронских рачунара које се у основи разликују по начину обраде. То су *аналогни* и *дигитални* рачунари.

Аналогни рачунари заснивају свој рад на чињеници да се различити физички системи описују истим математичким моделима. Такав математички опис физичких појава се назива физички закон. То значи да, за велики број система различите природе, њихово понашање можемо описати истим једначинама, користећи принцип аналогije у понашању.

Код електронских аналогних рачунара, поједине математичке операције, као на пример сабирање, одузимање, множење, дељење као и интеграцију или генерисање функција, обављамо користећи физичке законе који описују односе који постоје у једном електричном систему (нпр. однос између напона и струје у мрежи отпорника итд.). Променљиве величине у математичким операцијама које ми "обављамо" на електронском аналогном



рачунару, могу представљати физичке величине система чије понашање испитујемо. У том случају каже се да симулирамо такав систем.

Аналогни рачунари могу симулирати било који систем за који се физички систем, чији је математички опис аналоган систему који симулирамо, може поставити. По својој природи, физички системи су континуални системи, па се речи *аналогни* и *континуални* често користе као синоними.

Аналогни рачунари су практично изборили своје право на постојање највише због чињенице да су они недостижни као "машине за решавање обичних диференцијалних једначина", где су у великој предности у односу на дигиталне рачунаре. Суштина њихове супериорности у овој области лежи у чињеници да они све операције обављају симултано. Рекурзију изражену кроз једначине:

$$\begin{aligned} S(t) &:= I \times A_1 \cdot \{U(t), S(t)\} \\ Y(t) &:= A_2 \cdot \{U(t), S(t)\} \end{aligned}$$

које описују процес симулације континуалних система аналогни рачунар решава "директно" тј. користећи континуалну рекурзију, за разлику од дигиталног рачунара који мора да "обави" итеративну процедуру нумеричке апроксимације.

Симулација континуалних система помоћу аналогних рачунара своје предности заснива на следећем:

- ♦ могућност директног приступа било ком функционалном делу аналогног програма и
- ♦ могућност обављања великог броја понављајућих операција комбинованих са веома великом брзином израчунавања.

Недостаци оваквог начина симулације огледају се у следећем:

- ♦ Не постоји еквивалент репрезентацији података у покретном зарезу (најчешћи начин представљања код дигиталних рачунара), те се мора водити рачуна да се избегну могућа прекорачења и грешке сведу у подношљиве оквири, па се подаци скалирају.

- ♦ Не могу се избећи различита техничка ограничења, а онај који рукује аналогним рачунаром мора да поседује одређена знања о компонентама рачунара и руковању њиме.

## 6.7 Симулација континуалних система помоћу дигиталног рачунара

Основна разлика између аналогног и дигиталног рачунара није у начину представљања података, већ у принципу функционисања. Дигитални рачунар обавља основне рачунске операције и одређене логичке операције. Све друге математичке операције, изузимајући оне основне, дигитални рачунар обавља користећи одговарајуће нумеричке методе. Основна карактеристика дигиталног рачунара је секвенцијално обављање операција (једна за другом, нема симултаног обављања операција као код аналогног рачунара). Сваки податак или инструкција дигиталног рачунара састоји се, у основи, од бинарних цифара које представљају једно од два могућа стања електричног сигнала.

Предности дигиталног рачунара огледају се у великом нивоу апстракције код писања програма (програмер не мора да поседује знање о електричним законитостима) и у великој тачности решења, док је брзина извођења операција знатно мања него код аналогног рачунара. Проблем дигиталне симулације континуалних система није толико у брзини, колико је у могућој нестабилности нумеричке интеграције.

Да би се извршила дигитална симулација континуалних система, све величине морају бити дискретизоване, односно све функције независно променљиве  $t$  треба да буду представљене секвенцом дискретних бројева у тачкама  $t_n = t_0 + nh$ ,  $n = 0, 1, \dots, N$ . Величина корака  $h = t_n - t_{n-1}$  може, али не мора бити константна.

Дигитална симулација континуалних система је дискретан алгоритамски процес, који се извршава корак по корак, по процедури која је дата у дефиницији СКС. Резултат процедуре симулације је листа дискретних бројева за свако стање променљиве и сваки излаз.

Вредности променљиве  $t$  за коју се одређују вредности улаза, променљивих стања и излаза, монотono расту од почетне вредности  $t_0$ , до вредности  $t_N = t_0 + Nh$ .

На почетку процедуре симулације, потребно је означити нумеричке вредности сваког улаза и сваке променљиве. Те вредности називамо почетним условима.

Комплетно израчунавање секвенци вредности за све променљиве стања и улазе, а за извесни скуп вредности параметара и почетних услова, називамо извршавање симулације. Оно се састоји из фазе иницијализације и фазе рачунања.

У фази иницијализације постављају се почетне вредности и почетни услови, затим параметри, али и врше нека израчунавања везана за припрему нумеричке интеграције.

У симулационом алгоритму  $n$ -ти корак се састоји од израчунавања вредности функције свих алгебарских блокова, на основу скупа улаза и променљивих стања датих у  $t_n$  и израчунавања свих променљивих стања за наредни корак  $t_{n+1}$  у складу са основним једначинама:

$$\begin{aligned} S(t_{n+1}) &= I^* \times A_1 \cdot \{U(t_n), S(t_n)\} \\ Y(t_n) &= A_2 \cdot \{U(t_n), S(t_n)\}; \quad n = 0, 1, \dots, N \end{aligned}$$

Прва једначина означава да се крећемо кроз уређену листу алгебарских функција, и израчунавамо по датом редоследу, за сваки блок, вредност његовог излаза. Потом променљиве стања (или саме алгебарске функције) нумерички интегралимо ( $I^*$ ), зависно од тренутне и неке претходне вредности променљиве стања  $s_i(t_{n-r})$  и њеног првог извода  $s'_i(t_{n-r})$ ,  $r = 0, 1, \dots, k$ . Величина  $s_i(t_{n+1})$  треба да буду израчунате неком од нумеричких метода тако да функција

$$s_i(t_{n+1}) = s_i(t_n) + \int_{t_n}^{t_{n+1}} s'_i(t) dt$$

буде апроксимирана онолико тачно колико је потребно. После извршења свих корака интеграције, понавља се циклус операција, сада за  $n + 1 \rightarrow n$ .

Процедура симулације се понавља по затвореним циклусима док се не испуни услов  $t_n = t_N$  (или неки други услов), када се симулација завршава.

Ако са  $T_{cik}$  означимо време извршења једног циклуса, а са  $h$  корак израчунавања (инкремент), услов  $h \geq T_{cik}$ , обезбеђује симулацију у *реалном времену*. Међутим, ако желимо да постигнемо већу тачност, корак  $h$  мора да буде мањи од неке дате вредности која, када је мања од  $T_{cik}$ , онемогућава извршавање симулације у реалном времену.

Дигитални рачунар, као средство за симулацију континуалних система, готово у потпуности је потиснуо аналогне рачунаре, који се углавном сматрају застарелим и примењују искључиво у неким специфичним областима где је потребна велика брзина интеграљења и тачност реда процента. Више детаља о овој проблематици може се наћи у (Giloi, 1975).

## 6.8 Интеграција у симулацији континуалних система

У аналогној симулацији основни проблеми настају због ниског нивоа апстракције у програмирању, техничке преосетљивости рачунара, а нарочито због потребе за скалирањем.

Основни проблем дигиталне симулације везан је за нумеричку интеграцију. Оно што нас посебно интересује су интеграционе методе које се код симулације континуалних система могу "убацити" као интеграциони блокови, без потребе да се води рачуна о специфичностима самог система.

Интеграционе методе такве врсте могу се класификовати у две групе:

- ♦ једнокорачне интеграционе методе и
- ♦ вишекорачне интеграционе методе

или према другој подели:

- ♦ методе са константном величином корака и
- ♦ методе са променљивом величином корака.

Свака таква метода је једна апроксимација стварне интеграције дискретном алгебарском шемом, те је стога најважније питање, питање тачности и стабилности изабране методе. Посебну тешкоћу овде представља чињеница да је појава грешке у тесној вези са изабраним континуалним системом за симулацију, а не само у вези са одабраном методом за решавање.

Проблем нумеричке интеграције састоји се у томе да се дефинише неки оператор  $F$  тако да једначина

$$s_{n+1} = F(s_n, \dots, s_{n-k+1}; s'_{n+1}, \dots, s'_{n-k+1})$$

апроксимира на датом интервалу  $(t_n, t_n + h)$  интеграл

$$s_{n+1} = \int_{t_n}^{t_n+h} \varphi(S, U, t) dt = \int_{t_n}^{t_n+h} s' dt ,$$

где се користи скраћена нотација  $t_n = t_0 + nh$ ,  $s_n = s(t_n)$ ,  $u_n = u(t_n)$ ,  $s' = ds / dt$ , а  $F$  називамо  $k$ -корачном интеграционом формулом.

Наведена једначина може се назвати *дискретни блок за интеграцију* и може се представити сликом 6.4.



Слика 6.4 Дискретни блок за интеграцију

Величина интервала  $h$  не мора бити константна, већ се може мењати.

У једнокорачне интеграционе методе спадају:

1. Ојлерова (*Eulier*) метода

$$s_{n+1} = s_n + h s'_n + O(h^2)$$

2. Метода трапеза

$$s_{n+1} = s_n + (h/2)(s'_n + s'_{n+1}) + O(h^3)$$

3. Метода Рунге-Куте (*Runge-Kutta*) другог реда

$$\begin{aligned}\tilde{s}_{n+1} &= s_n + h s'_n \\ s_{n+1} &= s_n + \frac{h}{2}(s'_n + \tilde{s}'_{n+1}) + O(h^3)\end{aligned}$$

Свака од ових метода формира диференцну једначину, помоћу које се може израчунати низ  $\{s_n\} = \{s_0, s_1, \dots, s_n\}$ , чији елементи представљају приближне вредности решења  $s(t_0), s(t_1), \dots$  где је  $t_n = t_0 + nh$ , и ако је то диференцна једначина првог реда, онда она представља једнокорачну нумеричку методу (на основу вредности  $s_n$  израчунавамо  $s_{n+1}$ , тако да не користимо претходне вредности).

Вишекорачне нумеричке методе интеграције заснивају се на замени функције неким интерполационим полиномом, у изабраним чворовима интерполације, који се потом интегрални. Тако се формира диференцна једначина вишег реда помоћу које се вредност  $s_{n+1}$  траженог решења израчунава не само на основу претходне  $s_n$ , већ и на основу вредности  $s_{n-1}, s_{n-2}, \dots$  итд.

Општа формула за вишекорачну методу интеграције има облик:

$$s_{n+1} = a_0 s_n + a_1 s_{n-1} + \dots + a_r s_{n-r} + h(b_{-1} s'_{n+1} + b_0 s'_n + b_1 s'_{n-1} + \dots + b_r s'_{n-r})$$

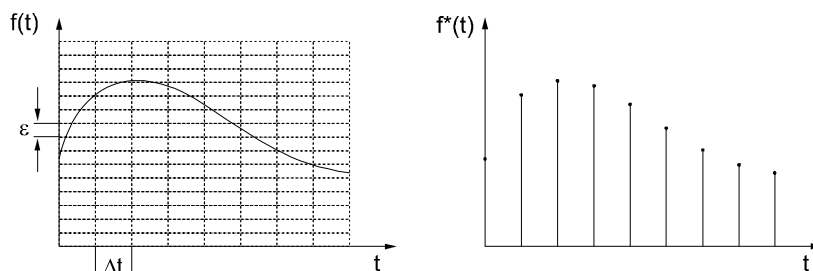
За симулацију континуалног система, при израчунавању вредности  $s_{n+1}$  није доступна вредност  $s'_{n+1}$ , те коефицијент  $b_{-1}$  мора бити

нула. Интеграциона шема горњег облика има уграђену потенцијалну нестабилност.

Више детаља о овој области могу се наћи у (Пејовић, 1983).

## 6.9 Одређивање интервала интеграције

Процес симулације на дигиталном рачунару је дискретан процес. То значи да се вредности променљивих у моделу рачунају само у одређеним тренуцима времена, а вредности континуалних променљивих познате су само у тим тренуцима. Из овога произилази да функције независне променљиве  $t$ , треба да се представе низом дискретних бројева у тачкама  $t_n = t_0 + nh$ ,  $n = 0, 1, \dots, N$ . Коначна, ненулта вредност  $h = t_{n+1} - t_n$  назива се корак интеграције. Корак интеграције може али не мора бити константан. Слика 6.5 приказује како се континуална функција "памти" у дигиталном рачунару.



Слика 6.5 Представљање континуалне  $f$ -је у дигиталном рачунару

- ◆  $f^*(t)$ -дискретизована функција  $f(t)$  (по времену и по нивоима  $\varepsilon$ )
- ◆  $\varepsilon$ -најмања вредност са којом може да се престаи промена
- ◆  $\Delta t \equiv T \equiv h$  -корак интеграције

Функција интеграције (INT), мора се приказати у складу са правилима модела са дискретним временом. Познато је да важи:

$$\frac{dY(t)}{dt} = \lim_{\Delta T \rightarrow 0} \frac{Y(t + \Delta T) - Y(t)}{\Delta T} = X(t)$$

Ако је  $\Delta T$  ( $T$  или  $h$ ) довољно мало, тада приближно важи

$$Y(t + \Delta T) \cong Y(t) + \Delta T X(t)$$

Што је мањи интервал  $\Delta T$ , то је тачност апроксимације већа, али је и време рада рачунара дуже. Циљ је да се одабере такав интервал интеграције (корак), да се с једне стране што је могуће више смањи грешка рачунања, а да се с друге стране време израчунавања интеграла (време симулације) не повећа превише.

Један од могућих начина за одређивање интервала интеграције је преко дискретизоване Лапласове трансформације, познатије под називом "*Z* трансформација" (*Петровић 1987*). Одређивање интервала интеграције применом *Z* трансформације најчешће није практично, пошто не само да зависи од изабране нумеричке методе, већ зависи и од модела који симулирамо.

Према томе, величину интервала интеграције морамо утврдити експериментом. Препоручује се да при првом експерименту интервал интеграције буде бар десет пута мањи од најбрже временске константе у систему. Оно што се не сме изгубити из вида је чињеница да се при великим интервалима интеграције губи увид о понашању система за време интервала интеграције, а такође је могућа и појава нумеричке нестабилности. Насупрот томе, при сувише малим интервалима интеграције може доћи до нагомилавања грешки услед великог броја рачунских операција и заокруживања до којих долази услед представљања реалних бројева у форми рационалних бројева на дигиталном рачунару, а и сам процес симулације знатно дуже траје.

Превођење континуалног модела у модел са дискретним временом (услов за извршавање симулације на дигиталном рачунару) само по себи представља један од основних проблема симулације континуалних система, тако да ма колико мали да се узме интервал интеграције, никада се не зна да ли се између два тренутка рачунања није догодила нека нагла промена, која би могла да изазове велику грешку у израчунавању наредне вредности. У случају да таква нагла промена постоји у неком интервалу рачунања, рецимо  $\{t_j, t_{j+h}\}$ , нису од користи ни вредности из прошлости, ни вредности из будућности којима се служи нека сложенија интеграциона метода. Други велики недостатак симулације континуалних система, који такође произилази из потребе за превођењем континуалног модела у модел са дискретним временом, састоји се у чињеници да нам понашање



модела није познато између два узастопна тренутка времена у којима се врши рачунање, односно није регистровано, иако би могло да буде сасвим различито од онога што се очекује (*Жикић, 1989*).

## 6.10 Извођење имплицитних операција

Ако математички модел симулације континуалног система није уредљив, онда он садржи једну или више функција у којој је вредност функције аргумент исте функције:

$$y = f(y)$$

или мање уочљив случај код скупа једначина:

$$\begin{aligned} y_1 &= f_1(x_1) \\ x_1 &= f_2(y_1) \end{aligned}$$

*Алгебарском петљом* називамо случај када су један или више алгебарских блокова повезани у затворену петљу која не садржи меморијске елементе типа интегратора или јединичног кашњења.

Врло често алгебарска петља се може отклонити алгебарским манипулацијама на оригиналном систему једначина. Тиме се добија тзв. канонички облик система. Међутим, има случајева када се систем не може довести у канонички облик и не може се извести симулација. У неким системима за симулацију, међу којима је и CSMP, наведени проблем се решава итеративним решавањем једначине  $y = f(y)$ .

У ту сврху убацујемо додатни блок за имплицитне елементе (IMP) у петљу и добијамо случај са слике 6.6 који се може описати као

$$\begin{aligned} y &= f(x) \\ x &= IMP(y) \end{aligned}$$

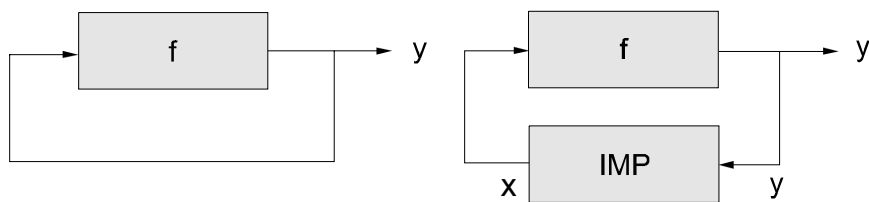
Први део репетитивне петље алгоритма симулације мора се извршити више пута. Почетну вредност  $x_0(t_0)$  одређује корисник. Почетне вредности за итеративни процес при сваком кораку итерације одређују се по формули  $x_0(t_n) = x(t_{n-1})$ . При свакој

итерацији,  $y_i(t_n) = f(x_i(t_n))$  се користи за израчунавање кориговане вредности за  $x$  по формули

$$x_{i+1}(t_n) = IMP(y_i(t_n)).$$

Итерације се понављају све док грешка  $\varepsilon = x_i(t_n) - y_i(t_n)$  не буде мања од унапред задате вредности. Функцију IMP треба тако дефинисати да конвергенција итерационог процеса буде што бржа. Није у свим случајевима унапред гарантовано да итерациони процес аутоматски конвергира.

Овај метод се употребљава у случају када нема других начина за решавање имплицитно задатих операција. Недостатак овог метода је знатно продужавање времена извршења сваког интервала интеграције.



Слика 6.6 Алгебарска петља (лево) и убацивање IMP блока (десно)

Више детаља о решавању алгебарских једначина може се наћи у (Пејовић, 1983).

### 6.11 Рачунарска реализација симулатора континуалних система

Сваки систем за дигиталну симулацију континуалних процеса састоји се од симулационог језика, процесора и скупа функција тј. блокова.

Симулациони језик служи за опис:

- ◆ математичких модела

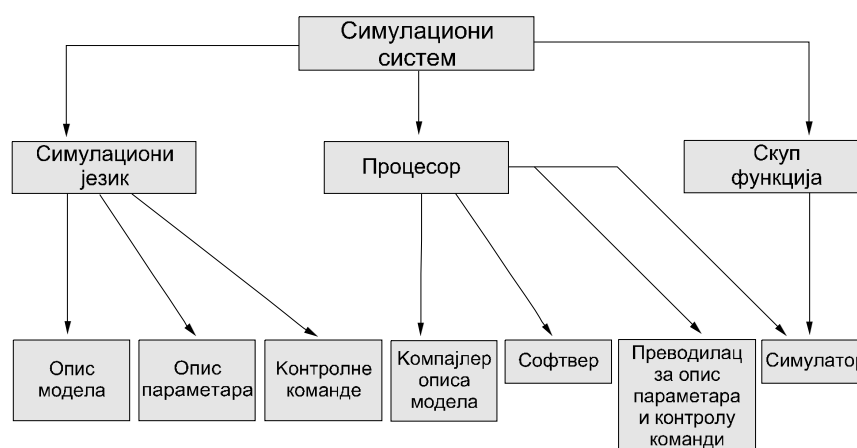
- ♦ параметара
- ♦ контролних (управљачких) команди

и то у облику који је разумљив како за корисника тако и за рачунар.

Процесор је програм који извршава следеће задатке:

- ♦ учитава, преводи и меморише математички модел
- ♦ сортира математички модел
- ♦ учитава, преводи и меморише параметре и управљачке команде
- ♦ извршава симулацију математичког модела.

Слика 6.7 шематски приказује описани симулациони систем и релације између његових делова.



Слика 6.7 Компоненте система за симулацију континуалних система

## 6.12 Симулациони језик

За симулацију континуалних система могу се користити програмски језици опште намене као што су нпр. *PASCAL*, *FORTRAN*, *ALGOL*, *PL/I*, итд. Међутим, њихова примена је ограничена јер захтевају добро знање рачунарске технике и нумеричке математике. Због тога се у симулацији континуалних система углавном користе специјализовани језици.

Језици за симулацију континуалних система су програмски језици вишег нивоа, чија структура језика подржава концепте симулације континуалних система. Иако већином једноставнији и више специјализовани од општих, њима се може користити просечан корисник рачунара без претходног образовања из области рачунарских наука и нумеричке математике.

При спецификацији симулационих језика треба поштовати следећа правила:

1. Формат улазних података треба да буде што флексибилнији, пожељно је да то буде формат сличан оном у *FORTRAN*-у или *PASCAL*-у.
2. Математичка нотација за функције или блокове треба да буде блок - оријентисана без нејасних математичких релација.
3. Избегавање непотребних ограничења. Функцијски блокови симулационог система не би требало да подлежу ограничењима, наметнутим из техничких разлога. То се посебно односи на симулацију на аналогном рачунару.
4. За идентификацију блокова и веза између њих треба користити симболичка имена, пре него бројеве.
5. Језик треба да буде такав да се лако учи, чак кад је у питању и неискусан програмер. То се постиже блок-оријентисаним описом математичког модела, уместо алгоритамског приступа, итд.

### 6.13 Процесор

*Процесор* је рачунарски програм. Састоји се из више модула. Први модул преводи опис модела у интерни језик и назива се преводилац. Он генерише машински код, посредни језик за превођење у машински код, или интерну табелу која се користи за време симулације.

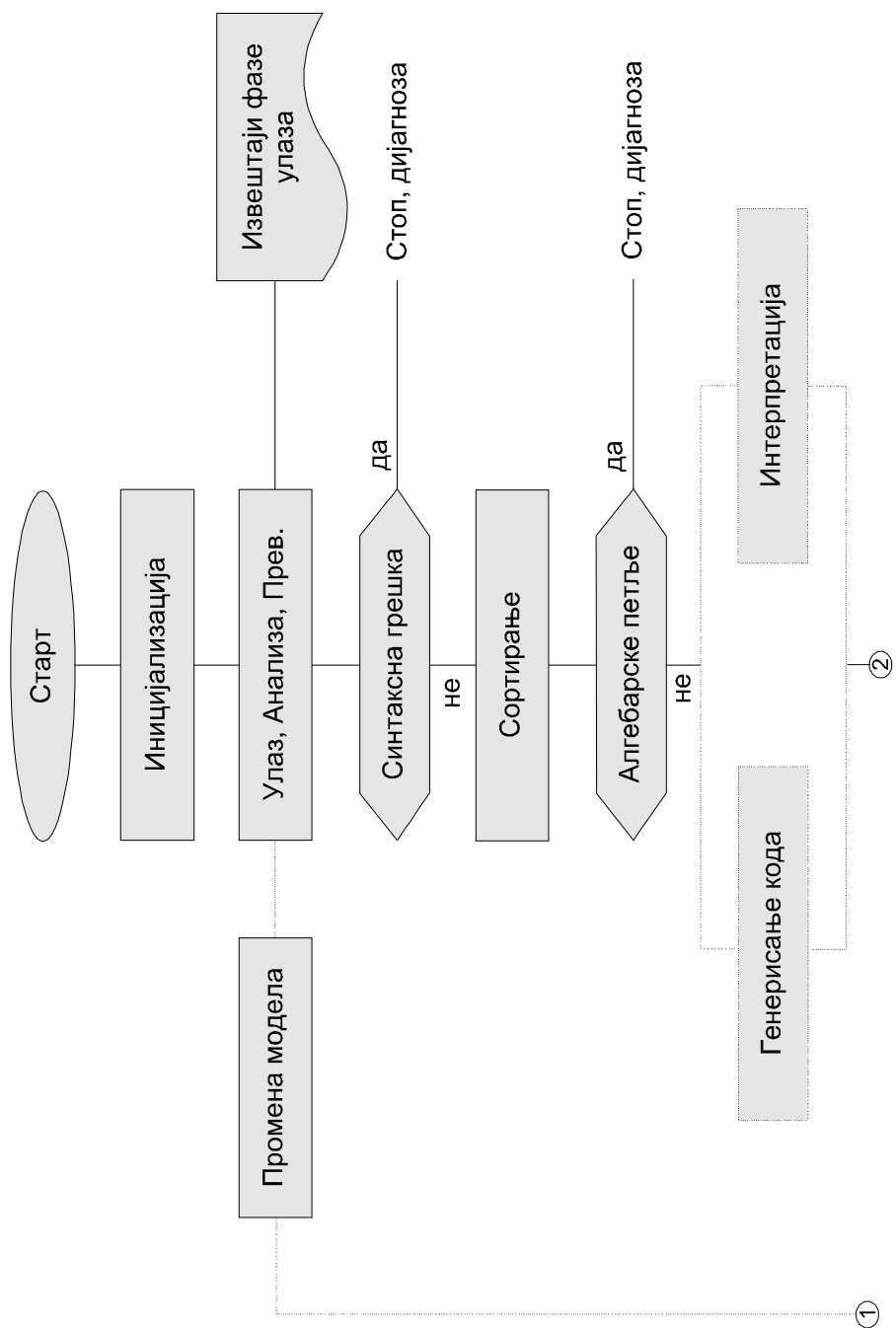
Други модул процесора служи за извршавање симулације. Он врши иницијализацију, израчунава вредности свих алгебарских функција, врши интеграцију свих променљивих, води рачуна о броју понављања и завршава симулацију у складу са датим условима. Тај

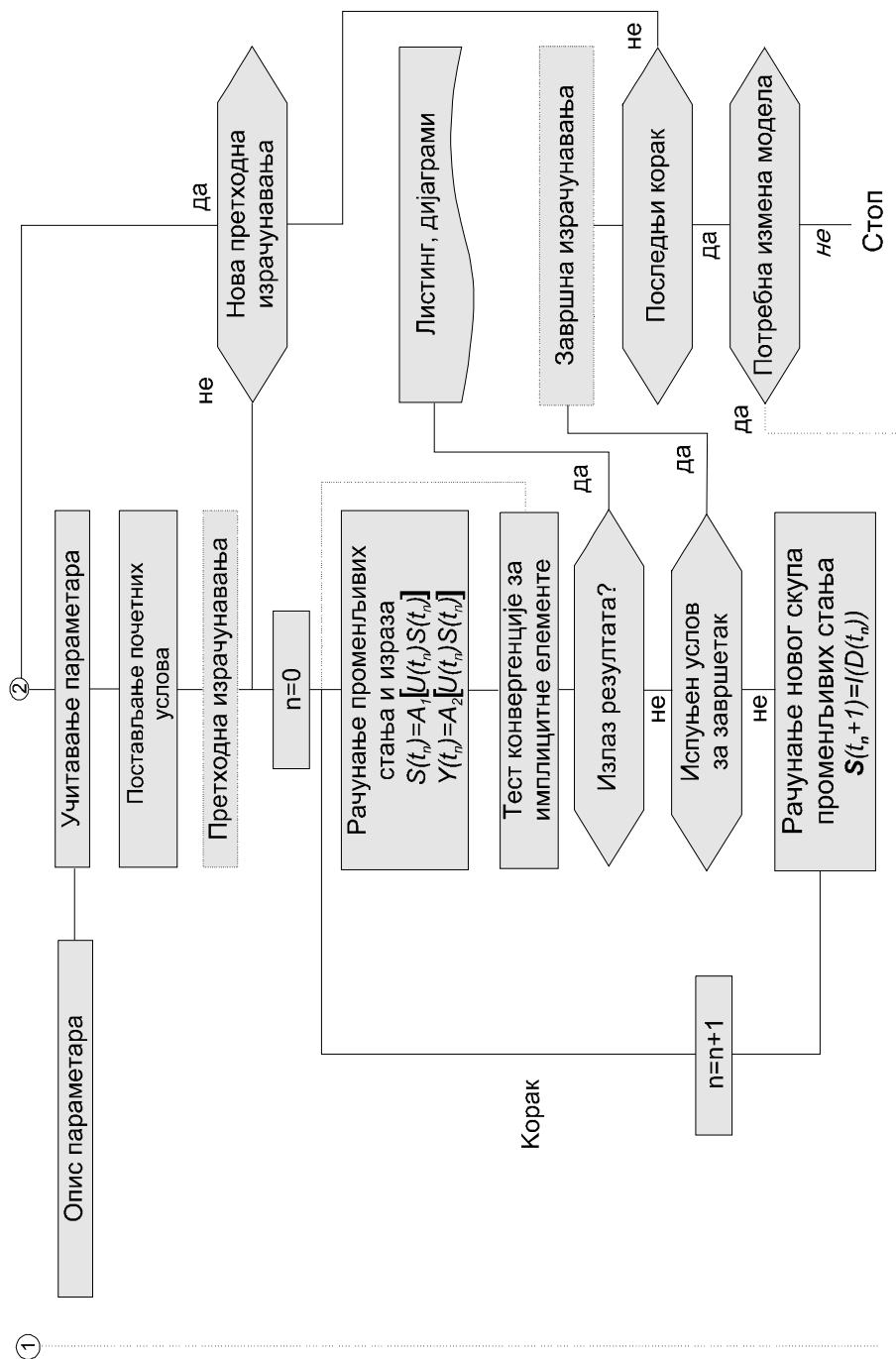
модул се назива *симулатор*. Преводацац и симулатор чине главни и најважнији део процесора.

Контролу тока симулације врши модул који се назива *монитор*. Он само чита контролне наредбе и позива друге модуле који извршавају неку од наведених фаза рада симулатора.

*Интерпретер* и монитор су модули који се не разликују много од сличних модула у другим програмским системима, што значи да нису специфични за један симулациони систем. Много специфичнији је симулатор који обезбеђује извршење симулације и у себи садржи низ алгоритама нумеричке математике за одређивање вредности функција, израчунавање имплицитних алгебарских операција, нумеричко интегрисање и временско кашњење. Симулатор такође поседује програмске модуле за извештавање о току и резултатима симулације.

На слици 6.8 дат је блок дијаграм тока типичног процесора за симулацију континуалних система.





Слика 6.8. Блок дијаграм тока типичног процеса за симулацију континуалних система

## СИМУЛАЦИЈА ДИСКРЕТНИХ ДОГАЂАЈА

Симулација дискретних догађаја је метода симулационог моделирања система код којих се дискретне промене стања у систему или његовом окружењу догађају дисконтинуално у времену. Углавном се користи за анализу динамичких система са стохастичким карактеристикама.

С обзиром да се под догађајем подразумева дискретна промена стања ентитета система (тј. променљиве која описује то стање), *догађај* наступа у одређеном тренутку времена. Типичан пример система са дискретним догађајима је банка, где се променљиве стања, као што је број клијената у банци као и стања на њиховим рачунима, мењају искључиво у одређеним тренуцима времена, када клијент уђе у банку, изврши неку трансакцију и изађе.

Самопослуга, телефонска централа, болница, сервис за оправку аутомобила, али и други системи масовног опслуживања, такође су примери система са дискретним догађајима. Посматрајмо пример сервиса за оправку аутомобила. Можемо уочити неколико дискретних догађаја као што су: долазак аутомобила у сервис, радник прихвата аутомобил и почиње поправку, оправљени аутомобил напушта сервис, итд. Овакви догађаји представљају промене променљивих система које се дешавају у дискретним тренуцима времена. Долазак возача у сервис је дискретан догађај који увећава за један број аутомобила у сервису, док одлазак возача након поправке представља дискретан догађај који тај број умањује за један. Почетак поправке аутомобила је дискретан догађај који умањује за један број аутомобила који чекају на оправку, итд.

Како су величине у оваквим системима најчешће случајне природе, говоримо о дискретним *стохастичким* системима. Време потребно за оправку аутомобила зависи од низа фактора: од сложености квара, од тога да ли сервис располаже делом који треба заменити, од тога да ли је за оправку потребна асистенција другог сервисера,



итд. За моделара система, овакве варијације не могу се предвидети, оне су случајне величине. За њихово описивање могу се користити различити статистички модели, односно расподеле вероватноћа које одговарају датој појави.

## 7.1 Формални опис система са дискретним догађајима

Формални опис система дискретних догађаја може се представити уређеном шесторком  $M_d$ :

$$M_d = \langle U, S, Y, \delta, \lambda, \tau \rangle$$

где су:

$U$  - скуп екстерних догађаја,

$S$  - скуп секвенцијалних стања дискретних догађаја,

$Y$  - скуп излаза,

$\delta$  - квази-преносна функција;

задаје се следећим функцијама:

$\delta^\Phi$  - пресликава  $S \rightarrow S$ ;

показује у које ће стање прећи систем из стања  $s$ , уколико не наступи ни један екстерни догађај,

$\delta^{ex}$  - пресликава  $U \times S \times T \rightarrow S$ ;

показује у које ће стање прећи систем из стања  $s$ , када наступи догађај  $u$  у временском тренутку  $t$ ,

$\lambda$  - излазна функција; пресликава  $S \rightarrow Y$

$\tau$  - функција наступања времена; пресликава  $S \rightarrow R^+_{0,\infty}$ ;

показује колико ће дуго систем остати у стању  $s$ , пре него што наступи наредна промена стања, под претпоставком да неће наступити ни један екстерни догађај.

## 7.2 Догађај, активност и процес

Код модела система са дискретним догађајима, поред концепата који описују структуру, као што су објекти, релације између њих и њихови атрибути, уведени су и концепти за опис динамике. То су:

- ♦ догађај,
- ♦ активност и
- ♦ процес.

Догађај представља дискретну промену стања ентитета у систему или његовом окружењу. Између два узастопна догађаја стање система се не мења. Он може наступити због:

- ♦ уласка/изласка привременог ентитета у односно из система (долазак новог клијента, одлазак клијента), или
- ♦ промене атрибута појединих објеката система (опслужилац у систему постаје слободан или заузет).

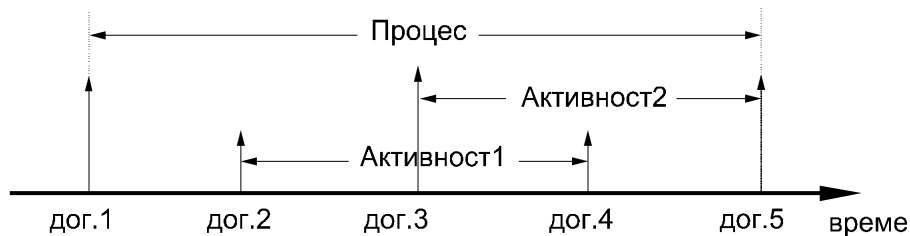
Условни догађаји могу да наступе тек када је испуњен услов њиховог наступања. Обично су повезани са заузимањем неког ресурса, односно почетком активности.

Безусловни догађаји су они чији је једини услов да текуће време симулације буде једнако времену његовог наступања. То је, по правилу, планирано време завршетка неке активности. Обично су повезани са ослобађањем ресурса који су били активирани у току трајања активности.

Активност је скуп догађаја који мењају стање једног или више ентитета. Трајање активности се може унапред дефинисати, али може да зависи и од испуњења неких услова у моделу, тако да је време завршетка такве активности непознато. За активност са непознатим временом завршетка у литератури се може наћи и израз "задржавање" а углавном се односи на задржавање у редовима чекања.

Процес је низ узастопних, логички повезаних догађаја кроз које пролази неки привремени објекат. Другим речима, процес је хронолошки уређена секвенца догађаја која описује једну појаву од настајања до терминирања. У симулацији, процес може да обухвата део или целокупан "живот" привременог ентитета.

Однос догађаја, активности и процеса може се графички представити као на слици 7.1.



Слика 7.1 Однос догађаја, активности и процеса

За илустрацију ових појмова, послужимо се примером сервиса за оправку аутомобила. Догађаји би били: долазак возача (аутомобила) у сервис, заузимање канала и отпочињање поправке, завршетак оправке аутомобила, почетак прања аутомобила након поправке, завршетак прања, одлазак возача (аутомобила) из сервиса након обављене поправке и прања. Запазимо такође следеће две активности: сервисирање (оправка) аутомобила и прање аутомобила (ове се активности не преклапају као што је то случај са активностима на слици 7.1). Процес би обухватио све догађаје од доласка возача у сервис до његовог одласка након завршене поправке и прања.

Користећи се претходно дефинисаним појмовима, можемо описати симулацију дискретних догађаја на следећи начин (*Shannon, 1975*):

#### Ентитети

који се описују **атрибутима**  
и узајамно делују учествујући у **активностима**  
под одређеним **условима**  
стварају **догађаје**  
који мењају **стања система**.

### 7.3 Развој симулације дискретних догађаја

У оквиру овог поглавља разматрају се кључни елементи развоја симулационих програма у симулацији система са дискретним догађајима. То су: механизам помака времена, приступи генерисању догађаја и основне стратегије симулације.

### 7.3.1 Механизам помака времена

У симулацији система са дискретним догађајима користе се два основна механизма помака времена: помак времена за константни прираст и помак времена на наредни догађај (*Law i Kelton, 1982*).

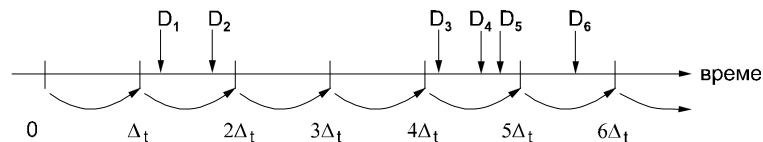
#### 7.3.1.1 Помак времена за константни прираштај

Код ове стратегије, време у симулационом моделу се мења тако да се увек додаје константни прираштај. Након сваког помака времена, односно ажурирања вредности симулационог сата, испитује се да ли је у претходном интервалу времена требало да дође до наступања неких догађаја. Уколико јесте, тада се ти догађаји планирају за крај интервала.

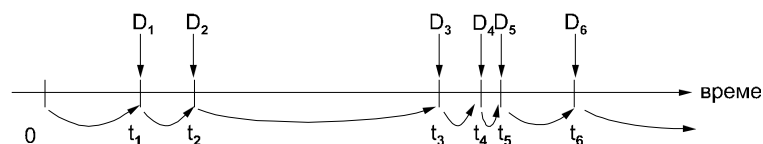
Приступ је приказан на Слици 7.2\_а. Видимо да у првом интервалу нема догађаја, док су у другом интервалу два догађаја  $D_1$  и  $D_2$ , а оба се изводе у тренутку  $2\Delta t$ . Стога је потребно одредити и начин избора редоследа извођења догађаја  $D_1$  и  $D_2$ .

Овај приступ је најједноставнији, али има одређене недостатке. Померањем догађаја на крај временског интервала у којем би они требало да наступе, уводи се грешка у извођење симулације. Осим тога, догађаји који у стварности нису истовремени у овом се приступу приказују као истовремени, а потом се одређује редослед њиховог извођења (који се може разликовати од редоследа извођења у стварности). Смањењем временског прираста те се грешке смањују, али се зато повећава време које се троши на извођење симулације. При томе је све већи број временских интервала у којима нема догађаја, па се они узалуд прелазе у симулацији.

Овај механизам помака времена може се успешно применити у ситуацијама када се догађаји заиста нижу један по један у константним временским интервалима, нпр. у економским системима где се стања мењају у константним временским периодима.



(а) Помак времена за константни прираст



(б) Помак времена на следећи догађај

Слика 7.2 Механизми померања времена у симулацији дискретних догађаја

### 7.3.1.2 Помак времена на наредни догађај

Код овог приступа, симулациони сат се помера на време у којем ће наступити први наредни догађај (или више њих). У том тренутку се догађај изведе и направи се одговарајућа промена стања система; затим се поново испитује који ће догађај следећи наступити, итд. Симулација се завршава када нема више догађаја или када је задовољен неки унапред дефинисани услов завршетка симулације (нпр. тачно одређено трајање симулације). Оваквим приступом се избегава грешка у времену извођења догађаја, а уједно се прескачу и интервали у којим нема догађаја.

На слици 7.2\_(б) дат је пример оваквог приступа. Запажамо да симулациони сат редом "скаче" са тренутка наступања догађаја  $D_1$  на тренутак наступања догађаја  $D_2$  итд. Иако доста компликованији за реализацију, овај приступ је знатно ефикаснији од претходног и не уноси грешке у времену извођења догађаја. Због тога се и користи у свим кључним симулационим језицима и програмским пакетима за симулацију система са дискретним догађајима.

### 7.3.2 Генерисање догађаја

У рачунарској имплементацији симулационог процеса, догађај се описује са више атрибута, који формирају *слог* догађаја. С обзиром на променљив број догађаја у времену, слогови догађаја се меморишу у *листама* догађаја. Слог и листа догађаја приказани су на слици 7.3.

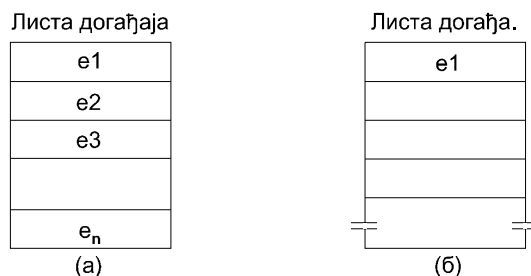


Слика 7.3. Слог и листа догађаја

Постоје два приступа генерисању догађаја и то:

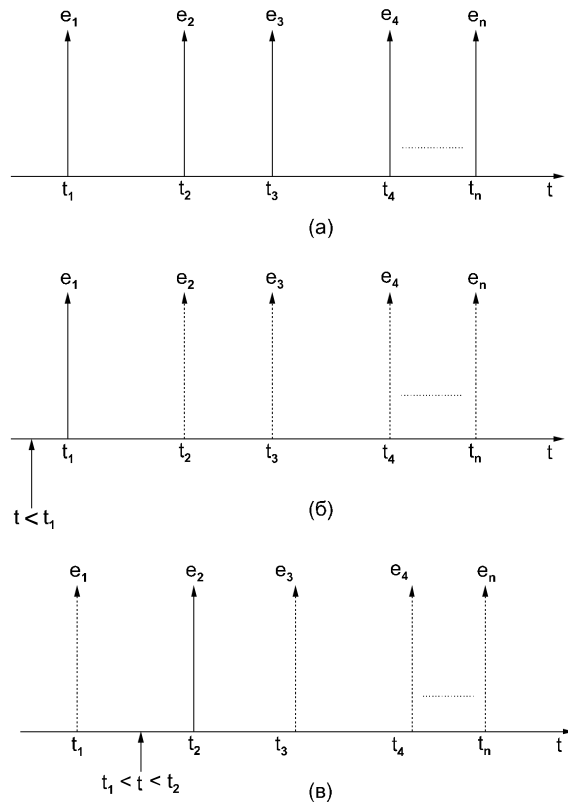
1. *Дефинисање догађаја унапред (fixed event strategy)*. Код овог приступа сви догађаји су унапред познати и дефинисани, а листа догађаја садржи слогове свих догађаја.
2. *Приступ заснован на наредном догађају (next event strategy)*. Код овог приступа, познат је једино први наредни догађај, а листа догађаја садржи само један слог, слог познатог догађаја. При извршавању догађаја, планира се и убацује у листу његов наследник.

На слици 7.4 приказан је изглед листи догађаја за случај (а) када се користи приступ унапред дефинисаних догађаја и (б) када се користи приступ заснован на наредном догађају.



Слика 7.4. Листе догађаја

Слика 7.5. даје графички приказ поменутих приступа за генерисање догађаја: (а) унапред дефинисани догађаји, (б) приступ заснован на наредном догађају за тренутак  $t < t_1$ , (в) приступ заснован на наредном догађају за тренутак  $t_1 < t < t_2$ .

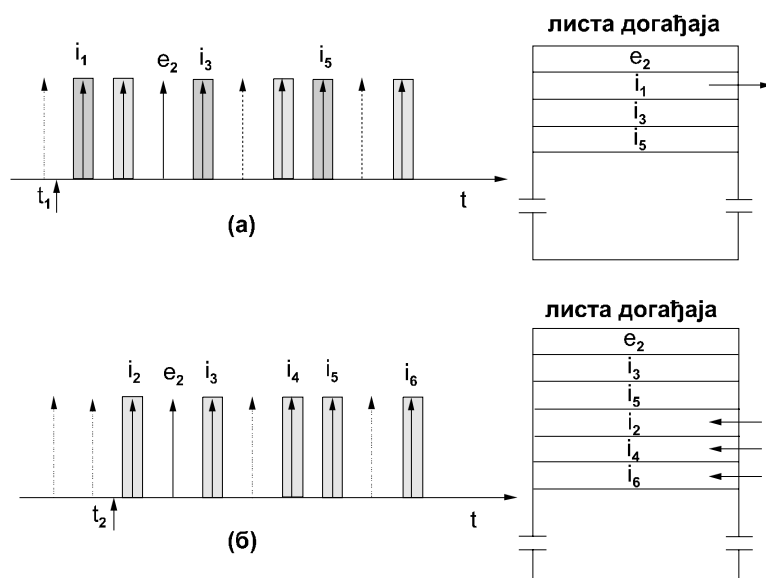


Слика 7.5. Различити приступи генерисања догађаја

Догађаје можемо сврстати у две основне категорије у односу на место настанка (генерисања):

1. *Екстерни догађаји* су они догађаји који не зависе од модела. Они представљају утицај околине на систем. У ту врсту догађаја убрајамо долазак возача у ауто-сервис, појаву телефонског позива, долазак купца у самопослугу, итд.
2. *Интерни догађаји* зависе од модела и у њему се генеришу. Примери интерних догађаја су долазак купца у ред за касу, прекидање везе по завршетку разговора, завршетак операције болесника итд.

На слици 7.6 приказана је листа догађаја у тренутку  $t = t_1$ , пре генерисања интерних догађаја  $i_2, i_4, i_6$  - (а) и листа догађаја у тренутку  $t = t_2$ , након генерисања интерних догађаја у моделу за случај упаред дефинисаних догађаја (б).



Слика 7.6 Екстерни и интерни догађаји и одговарајуће листе

## 7.4 Стратегије извођења симулације

Концепти као што су догађај, активност и процес чине полазну основу за дефинисање стратегија извођења симулације система са дискретним догађајима. Те стратегије су (*Fishman, 1978*):

1. Распоређивање догађаја (*Event scheduling approach*)
2. Сканирање активности (*Activity scanning approach*)
3. Интеракција процеса (*Process interaction approach*)

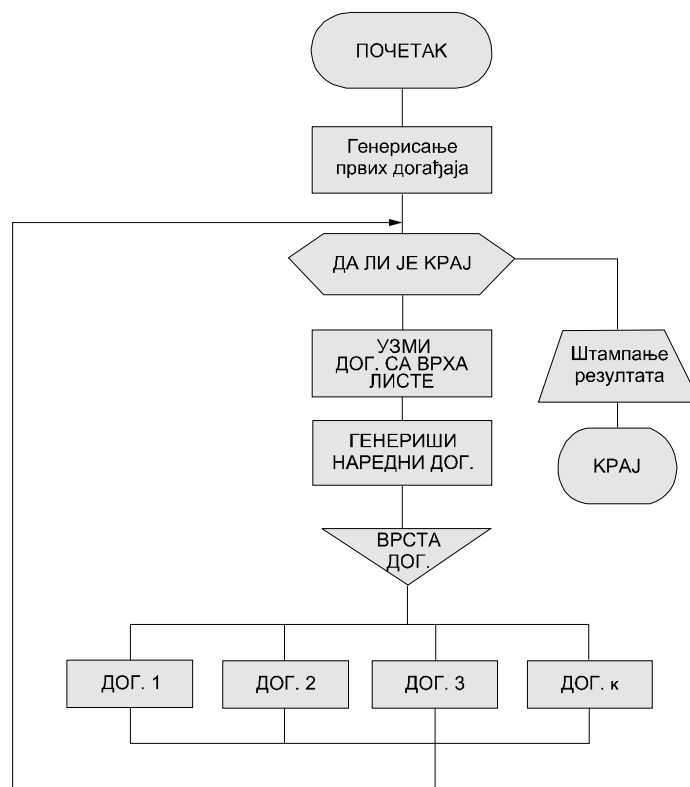
### 7.4.1 Распоређивање догађаја

Механизам распоређивања догађаја подразумева да се догађаји планирају унапред и држе у листи будућих догађаја, најчешће сортирани по времену наступања и приоритету.



Процедура планирања догађаја је следећа: генерише се слог догађаја. Затим се доделе вредности његових атрибута. Атрибути могу бити: идентификатор типа догађаја, време његовог наступања, приоритет као и други, зависно од типа догађаја. Затим се догађај ставља у листу будућих догађаја. Листа будућих догађаја је уређена по времену наступања догађаја и њиховом приоритету.

Функционисање симулатора се одвија на следећи начин : са листе будућих догађаја узима се први догађај. Тада се ажурира симулациони сат на време његовог наступања. На основу типа изабраног догађаја, позива се одговарајућа процедура која извршава сва ажурирања у моделу и симулатору, повезана са наступањем изабраног догађаја. Када се изврше сви догађаји који имају исто време наступања, симулациони сат се ажурира на време следећег догађаја, из листе будућих догађаја. Алгоритам је приказан на слици 7.7.



Слика 7.7 Дијаграм тока распоређивања догађаја

### 7.4.2 Сканирање активности

Активност смо дефинисали као скуп операција које трансформишу стања ентитета. При томе, једну активност може сачињавати више догађаја. Да би се извршила симулација система са дискретним догађајима, потребно је реализовати рачунарски програм који сканира и распоређује активности модела. Сканирање активности подразумева да се догађаји имплицитно распоређују тако да се промена стања извршава преко функција које се називају активности. Свака активност има *услов* и *акцију* (слика 7.8).

А К Т И В Н О С Т И	УСЛОВИ		АКЦИЈЕ	
	A1	УСЛОВИ 1	АКЦИЈЕ 1	
	A2	УСЛОВИ 2	АКЦИЈЕ 2	
	A3	УСЛОВИ 3	АКЦИЈЕ 3	

Слика 7.8 Табеле активности

За сваки временски корак, активности се сканирају и тражи се прва активност која има задовољен услов. Тада се извршава одговарајући програмски сегмент који специфицира акцију за задату активност. Процес сканирања наставља се све дотле док све активности не буду блокиране. Онда и само онда се симулациони сат ажурира за следећи временски корак.

За извршење активности у реалном систему потребно је извесно време, док при симулацији није потребно пратити кретање активности од почетка до завршетка. Само се прелази са једне активности на другу дефинишу експлицитно. Иронија такве ситуације је да активности које у реалном свету трају извесно време, при симулацији не троше рачунарско време јер се не дефинишу експлицитно, док прелази са активности на активност који се у реалном свету одвијају тренутно, троше извесно рачунарско време јер се описују експлицитно одређеним секцијама програма.

Сканирање активности има предности над распоређивањем догађаја у случајевима када је број активности у моделу мали, а број догађаја у оквиру активности велики. У том случају се применом сканирања активности штеди на рачунарским операцијама које се односе на стављање догађаја на листу и његово скидање са листе догађаја.

### 7.4.3 Интеракција процеса

Интеракција процеса представља технику симулације која је настала комбинацијом распоређивања догађаја и сканирања активности.

Процес можемо посматрати као скуп узајамно искључивих активности, повезаних тако да терминирање једне активности дозвољава иницијализацију неке друге активности из скупа активности процеса. С обзиром на то да модел система представља скуп паралелних процеса од којих неки могу бити узајамно искључиви, главни проблем је синхронизација процеса. Овај прилаз моделирању система са дискретним догађајима, могући конфликт који проистиче из преклапања процеса, решава се увођењем две наредбе које се употребљавају при спецификацији догађаја. То су:

- ♦ WAIT и
- ♦ DELAY

и то у оба контекста, условном и безусловном.

WAIT наредба у условном контексту има облик:

WAIT UNTIL <услов>.

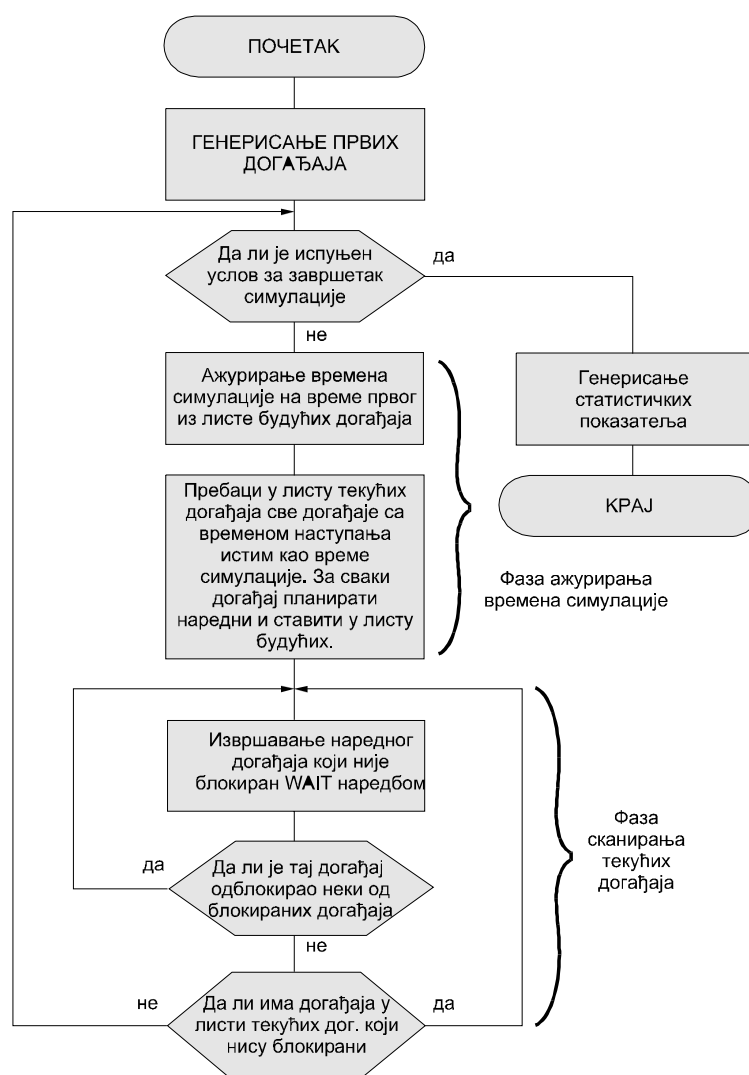
Уколико је услов задовољен, процес који чека на тај услов, наставиће свој ток. У супротном, биће блокиран. У том случају испитује се који од осталих процеса може да се активира и он се извршава. На тај начин се уграђују добре особине методологије сканирања активности.

- ♦ Наредбом DELAY која се у безусловном контексту у литератури означава и са

ADVANCE <број временских јединица>

наставак процеса одлаже се за специфициран број временских јединица. У том случају, испитује се који од осталих процеса може да оствари свој ток. На тај начин се у ову методологију преносе добре особине методологије распоређивања догађаја.

Одговарајући алгоритам рада процесора дат је на слици 7.9.



Слика 7.9 Алгоритам рада процесора при интеракцији процеса

У оквиру процесора, одржавају се две листе догађаја:

1. *Листа текућих догађаја*: садржи условне догађаје; сортирана је по приоритету.
2. *Листа будућих догађаја*: садржи распоређене догађаје који тек треба да наступе; сортирана је по времену наступања.

Планирање догађаја процесор врши по стратегији наредног догађаја. При сваком премештању догађаја, из листе будућих догађаја у листу текућих догађаја, планира се његов наследник који се ставља у листу будућих догађаја, да тамо чека своје време наступања.

Пре почетка симулације, за сваки процес генерише се први догађај и ставља се у листу будућих догађаја. Затим се време симулације ажурира на време догађаја са врха листе будућих догађаја.

Рад процесора одвија се у две фазе:

1. Фаза ажурирања времена симулације,
2. Фаза сканирања листе текућих догађаја.

У фази ажурирања времена симулације, симулациони сат се поставља на вредност времена наступања првог догађаја из листе будућих догађаја, а затим се пребаце сви догађаји са тим временом наступања у листу текућих догађаја. При томе се за сваки премештен догађај планира наследник, и ставља се у листу будућих догађаја.

У фази сканирања листе текућих догађаја сканира се листа текућих догађаја и уколико има неблокираних догађаја, они се извршавају. Када нема догађаја у листи текућих догађаја, или су сви догађаји блокирани (WAIT наредбом), тада се преноси први догађај из листе будућих догађаја и ажурира се сат симулације на његово време.

Симулација се одвија све док се не задовољи критеријум за завршетак симулације. Тада се врши завршна обрада резултата симулације и процесор завршава рад.

Интеракцију процеса користе познати симулациони језици GPSS и SIMULA. Због комбиновања добрих особина распоређивања догађаја и сканирања активности, интеракција процеса представља логичан избор за пројектовање модерних симулационих система.

## ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И СИМУЛАЦИЈА

Област вештачке интелигенције (*artificial intelligence*), као мултидисциплинарна област рачунарских наука последњих година доживљава убрзани развој. Иако су прва теоријска истраживања, започета педесетих година овога века, остала без значајнијих практичних резултата, данас се може рећи да је низ усвојених теорија, техника и алата проистеклих у овој области, пронашло своју практичну и комерцијалну примену. С једне стране, то се може приписати развоју рачунарске технике и паду цена хардвера, а с друге стране, промењеном тежишту истраживања у самој области вештачке интелигенције, са нагласком на примењена истраживања. Један од драгоцених и неспорних доприноса вештачке интелигенције "лежи" у томе да она може да помогне у спречавању губитка стручног знања које представља најдрагоценији капитал сваког друштва.

Еволуциони путеви вештачке интелигенције и симулације почели су значајније да конвергирају тек у последњој декади двадесетог века, иако су истраживања у обе области започета знатно раније. Вештачка интелигенција проистекла је из истраживања у области когнитивне психологије и математичке логике, чије је основно тежиште било на објашњењу рада људског мозга и изради алгоритама опште намене за решавање различитих проблема. С друге стране, симулација је настала из потребе да се схвати и проучи понашање сложених, динамичких система из реалног окружења. Иако се, на први поглед, основни предмет изучавања разликује у ове две дисциплине, постоје преплитања идеја и сличности које су мотивисале истраживаче из обе области на узајамну комуникацију. За илустрацију поменимо чињеницу да су секције за вештачку интелигенцију постале саставни део сваке веће конференције из области симулације од 1985. године, а да је број радова у таквим секцијама између 1985. и 1986. године забележио раст од 150%. Прва самостална конференција у Европи "Вештачка интелигенција и симулација" одржана је такође 1985. године

(*Kerchoffs, Vansteenkiste and Zeigler, 1986*). У области вештачке интелигенције (ВИ), Национална конференција за вештачку интелигенцију одржана 1986. године укључила је први *Workshop* на тему "Вештачка интелигенција и симулација". Разматрајући могући интерфејс и мотивацију за повезивање ВИ и симулације, поћи ћемо од појма "вештачка интелигенција". Вештачка интелигенција је научна дисциплина посвећена "опремању" рачунара интелектуалним способностима као што су опажање, резонување и учење. Пионири у овој области, *Barr* и *Feigenbaum* дефинишу вештачку интелигенцију као "део рачунарских наука који се бави дизајнирањем и изградњом интелигентних рачунарских система, тј. система који показују карактеристике које повезујемо са интелигенцијом у људском понашању - разумевање језика, учење, схватање или резонување, решавање проблема, итд." (*Barr i Feigenbaum, 1981*).

На први поглед, могло би се рећи да наведена дефиниција не успоставља никакав однос између симулације и вештачке интелигенције. Међутим, истраживачи и практичари у обе научне дисциплине сусрећу се са сличним проблемима приликом моделирања сложених система, нарочито у случајевима делимичног познавања и разумевања система. Решења до којих се долазило независно у обе дисциплине, често су се преклапала концептуално иако не и терминолошки. На пример, често коришћена шема за представљање знања у базама знања, *If-Then* правило, наликује методи која се у симулацији користи за тестирање промена стања. Свакако, основни разлог међусобног интересовања је тај што свака од ових дисциплина има своје специфичне предности које могу бити комплементарне, па се решења, алати, технике и методе из једне дисциплине могу са успехом применити у другој. Све ово је утицало да последњих година нагло порасте интересовање за повезивање симулације и вештачке интелигенције иако се мотиви повезивања међусобно разликују.

С једне стране, истраживачи ВИ препознали су важност симулације. На пример, *Campbell* у свом раду коментарише програмирање на вишем нивоу у ВИ на следећи начин: "Они који се баве објектно-оријентисаним програмирањем у ВИ откривају да су одређене технике и лакоћа са којом их је могуће користити, на пример, наслеђивање особина, већ развијене и имплементирани од стране дизајнера симулационих језика". С друге стране, уочено је да велики програмски пројекти у ВИ "поседују неке компоненте приступа математичког моделирања, као у симулацији, али и друге,

богатије структуре, које немају аналогију у том моделирању" (Campbell, 1986). Свакако, могуће је уочити и разлике. Једна од њих је у томе што модели ВИ користе хеуристички приступ као суштински, док то у симулацији није уобичајено. Друга значајна разлика се огледа у чињеници да је у ВИ предмет интересовања интелигентно понашање. У том погледу, ВИ апликације најчешће захтевају да се усмери експлицитна пажња на узроке и последице, док су стандардни математички модели пасивни у том погледу (Paul, 1989).

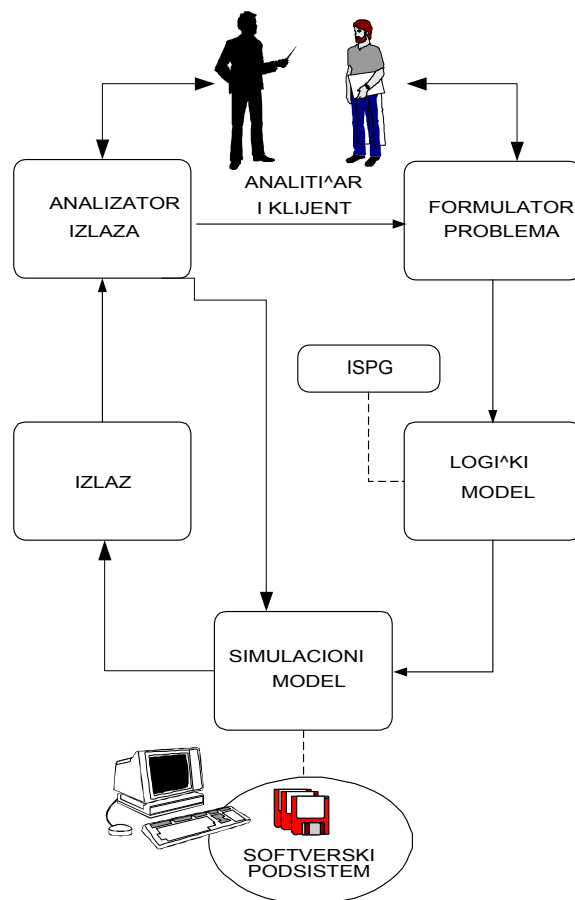
С друге стране, пораст интересовања за вештачку интелигенцију и експертне системе подстакао је и одређени број истраживача у области симулације да истакну сличности између техника ВИ и симулације и да сугеришу важност коришћења комплементарних техника ВИ у симулацији. Ören је био један од првих присталица коришћења техника ВИ код рачунарски подржаних система за изградњу модела (Ören, 1979) који би укључивали спецификацију компоненти модела, интерфејсе између компоненти и вредности за параметре модела. Поједини истраживачи из области симулације посматрали су симулацију и експертне системе као комплементарне технике, које се могу комбиновати како би се обезбедиле моћне функције за доношење одлука (Gaines and Shaw, 1985; Garzia, Garzia and Zeigler, 1986; Moser, 1986; Shaw and Gaines, 1986) или које као такве могу представљати интелигентне платформе за помоћ при извођењу експеримената или код анализе резултата симулације (Haddock, 1987; O'Keefe, 1986).

Као пример могућег коришћења ВИ у симулацији, размотримо пројекат групе истраживача из области симулације, назван *Computer-Aided Simulation Modeling - CASM*. Пројектни тим CASM радио је на изради флексибилног плана за развој рачунарске подршке симулационом моделирању. У наставку следи преглед овога плана у најкраћим цртама (Paul, 1989).

Основни процес развоја симулационог модела према CASM приказан је на слици 8.1. Аналитичар формулише проблем на структуриран начин, на пример преко дијаграма тока или преко дијаграма циклуса активности (Clementson, 1982). Тако дефинисан логички модел смешта се у интерактивни генератор симулационих програма - *ISPG (Interactive Simulation Program Generator)*. Постоји више таквих генератора: *CASP (Clementson, 1982)*; *DRAFT (Mathewson, 1985)*; *CASM.AUTOSIM (Paul and Chew, 1987)* идр. Већина ISPG интерактивно комуницира са корисником о проблему укључујући и квантитативна питања која се



тичу узимања узорак, аритметичких и почетних услова и сл. Генератор програма тада аутоматски пише симулациони модел користећи се при томе расположивим рачунарским ресурсима и софтверским подсистемима. Под контролом аналитичара, модел се стартује и добија се излаз. Добијени излаз се може потом користити за исправке на логичком моделу и рачунарском програму (уколико *ISPG* није довољно поуздан). За приказивање динамике понашања симулационог модела често се користи рачунарска графика.



Слика 8.1 Процес изградње симулационог модела (према CASM)

Анализирајући претходно описани поступак, намеће се питање да ли се могу уочити елементи вештачке интелигенције и да ли она може бити од помоћи у описаном процесу симулације?

Под претпоставком да наведени процес функционише на задовољавајући начин, може се запазити да су преостале само две активности које и даље захтевају интензиван људски рад, а то су: формулисање проблема и анализа резултата симулације. Ове активности представљају "интелигентне" доприносе аналитичара и он тежи да их у што већој мери побољша сопственим искуством. Управо у том делу симулационог процеса могуће је увести елементе вештачке интелигенције.

Формулисање проблема са слике 8.1 врши се помоћу софтвера за ВИ који помаже аналитичару да уз помоћ клијента адекватно формулише проблем. Стручност аналитичара може се такође "чувати" и у анализатору излаза, као помоћ при одлучивању о томе на који начин тумачити резултате симулације. Осим тога, могуће је користити и базу података о логичком моделу ради боље документованости симулационог модела.

Корист од повезивања ВИ и симулације уочили су и многи други аутори те се у стручној литератури јавља све више радова који повезују ове две дисциплине. Истраживачи у области симулације дошли су до закључка да симулациони софтвер "пати" од одређених слабости: корисници су захтевали знатно реалистичније моделе, тражили су се бољи интерфејси за спрегу са корисницима, а посебно боља подршка за кориснике са мало или без искуства, као и већа помоћ код "извлачења" информација из великих количина расположивих података (*Widman i Loparo, 1989*). У области ВИ, истраживачи су дошли до закључака да многи њихови програми "падају у воду" када се суоче са проблемима из реалног света. Они су такође имали проблема при резонувању о феноменима који се мењају у времену, као и код предвиђања понашања сложених стохастичких система. Све ово је утицало да у последње време развојни путеви ове две научне дисциплине све брже конвергирају.

Изградња и коришћење симулационих модела је сложен посао који захтева експертизу из домена разних теоријских подручја, укључујући статистику, системску анализу, рачунарство, нумеричку математику као и подручје коме припада систем који се моделира. У том послу искуство, предвиђање и сналажљивост могу да играју значајну улогу, чиме се у сваком кораку симулационог процеса ствара могућност за асистенцију експертних система или других апликација из домена вештачке интелигенције.

## 8.1 Историјски развој вештачке интелигенције

Вештачка интелигенција као научна дисциплина настаје захваљујући споју рачунарске технике са покушајима научника да докуче човечју интелигенцију путем њене формализације и емуляције. Тако је створена ова мултидисциплинарна област која још еволуира, а окупља знања из различитих грана, као што су: рачунарство, математика, логика, лингвистика, психологија, филозофија, кибернетика и друге.

Корени вештаче интелигенције датирају још из првих дана развоја рачунарских машина. Године 1843, *Ada Augusta Byron*, ћерка песника *Lord Byron*-а и покровитељ *Charles Babbage*-а поставила је питање "може ли *Babbage*-ова аналитичка машина, прва програмибилна рачунарска машина на свету, да мисли?" Њој у част, Министарство одбране САД, назвало је свој стандардни програмски језик *ADA*. Не дуго затим први истраживачи покушали су да се баве проблемима типа *puzzles*, играњем шаха и превођењем једног језика на други (*Barr, Feigenbaum and Cohen, 1981, 1982*) иако се тада такви покушаји нису називали "вештачка интелигенција". Током Другог светског рата, *Norbert Winer* и *John Neumann* повезали су принципе кибернетике са реализацијом сложених одлука и контролним функцијама на машинама.

Прекретница у развоју вештачке интелигенције настаје 1956. године када је одржана чувена *Dartmouth*-ска конференција и ова дисциплина постала засебно поље истраживања рачунарских наука са нагласком на неалгоритамском карактеру интелигентне људске активности (*Charniak and McDermott, 1985*).

Основу досадашњих истраживања у области ВИ представља систем физичких симбола који су поставили *Newell* и *Simon*. Како је непознавање можданих процеса човека постало примарни ограничавајући фактор у развоју ВИ, решење је пронађено у идеји да се човеков интелект, односно процес размишљања, симулира на нижем нивоу сложености структуре, дакле са мање детаља. *Newell* и *Simon* су увели систем физичких симбола којима се симулира човеков мисаони апарат, а сам мисаони процес представљен је као манипулација физичким симболима. Досадашња искуства говоре у прилог овој хипотези, мада је могуће да ће се временом показати да је она само делимично тачна.

Првобитно сматрана за "науку о методама оспособљавања рачунара за вршење задатака и поступака који се обично сматрају интелигентним када су извођени од стране људи", нагласак је испрва био на изналажењу општих начела интелигенције и њиховом уграђивању у универзалне програме за решавање свих врста проблема. У почетку, вршени су покушаји да се пронађу алгоритми који би опонашали људску интелигенцију, решавајући проблеме без веће присутности а приори информација. У овом раздобљу се појављују класични ВИ пројекти *HPS (Newell and Simon, 1972)*, *EPAM (Feigenbaum, 1963)*, популарно је аутоматско доказивање теорема, аутоматско превођење природних језика и слично. Показало се, међутим, да је овакав прилаз превише амбициозан и нереалан. На малом скупу поједностављених проблема, ови су програми давали добре резултате, али ширењем тога скупа и додавањем реалнијих ситуација, њихова ефикасност је нагло падала, будући да је сложеност проблема експоненцијално расла.

У даљој фази, почетком седамдесетих година, напушта се полако идеја обавезног симулирања човечијих мисаоних (когнитивних) процеса и иде се на рачунарски погодну формализацију, односно стандардизацију прилаза разним проблемима, са нагласком на идејама представљања и претраживања. У том раздобљу се појављују системи са мањим претензијама од ранијих, али чији импресивни резултати опадају при напуштању вештачки строго омеђених домена разматрања. Низ познатих пројеката карактерише овај период. Виноградов *SHRDLU (Winograd, 1972)* је, на пример, истакао неке основне принципе у разумевању природних језика и демонстрирао неке од предности тзв. процедуралног представљања знања. *Waltz*-ов програм за разумевање цртежа (*Waltz, 1975*) даје, за конструкцију одређеног скупа рогљастих геометријских тела представљену видљивим обрисима и ивицама, одговарајући опис присутних елемената. Међутим, предвиђена генерализација свих ових система на проблеме шире од првобитно задатих је увек наилазила на непремостиве тешкоће.

Коначно, крајем 70-тих година је преовладало мишљење да за практичне резултате, сами по себи сложени механизми управљања и претраживања и формализације представљања нису довољни и да је за неку задату проблематику, ВИ програме потребно снабдети експлицитним и екстензивним знањем из дате области. Истовремено се створио интерес и за практичну примену производа истраживања ВИ и самим тим започео период интензивније комерцијалне примене.

## 8.2 Основни концепти вештачке интелигенције

У досадашњем развоју ове дисциплине, појам вештачка интелигенција изазивао је многе расправе међу научницима, истраживачима, рачунарским стручњацима али и међу футуристима који су се бавили њеним проучавањем. Неки посматрачи тврдили су чак да ствар као што је ВИ и не постоји. Уколико је нешто вештачко, говорили би, тада није интелигентно. Други су оспоравали оваква становишта и покушавали су да дају ближа одређења појма ВИ.

У *Artificial Intelligence and Expert Systems Sourcebook* (Hunt, 1986) ВИ се третира као "област која се бави изналажењем рачунарских програма који треба да учине рачунаре паметнијим". Стога се "истраживање у вештачкој интелигенцији фокусира на развој рачунарских приступа интелигентном понашању", при чему су наглашена два циља: "да се машине учине кориснијим и да се схвати интелигенција."

Оваква полемичка размишљања довела су до ситуације да је прецизно разграничавање овог појма веома тешко. Чак ни познати тротомни приручник из ВИ (Barr, Feigenbaum and Cohen, 1981, 1982) не издваја и не обрађује посебно овај појам. Стога, морамо се задовољити емпиријским дефинисањем, односно проналажењем заједничког у системима (програмира) који спадају у домен ВИ (Widman i Loparo, 1989):

### 1. Симболичко уместо нумеричког израчунавања.

Основна карактеристика по којој се прави разлика између метода ВИ и нумеричких метода јесте да је базична јединица израчунавања у ВИ симбол, а не број. Наравно, сама ова чињеница није довољна да би се разликовали програми ВИ. Друге класе програма као што су преводиоци или системи за претраживања база података, такође обрађују симболе, али се третирају као ВИ програми. С друге стране, неуронске мреже, за које се најчешће сматра да су део ВИ, зависе од стриктних нумеричких израчунавања при прикупљању и коришћењу знања.

### 2. Неалгоритамски приступ решавању проблема.

Друга карактеристика ВИ програма је та да се структура програма не изражава експлицитно алгоритмом - секвенцом корака које програм извршава при решавању одређеног проблема. Класични

програми уобичајено следе добро дефинисане алгоритме који тачно специфицирају како се на основу улазних варијабли могу добити излазне величине (процедурално програмирање). Код програма ВИ, низ корака које програм прати и извршава зависи од конкретног проблема који се решава; програм одређује како одабрати секвенцу корака који воде ка решењу проблема (декларативно програмирање). На најнижем нивоу посматрања, наравно, и ВИ програми се могу третирати као процедурални пошто се безусловно преводе и извршавају као бинарни код. Из ове перспективе, може се полемисати о томе да декларативна природа програма из области ВИ лежи у нивоу који корисник одабере за посматрање приликом имплементације програма. Међутим, овде ћемо сматрати да су ВИ програми они који се имплементирају коришћењем техника за решавање проблема, за које се генерално сматра да су декларативне.

### 3. Закључивање засновано на знању.

Трећа карактеристика ВИ програма је та да они укључују чињенице и релације о делу реалног света или "домена знања" у коме функционишу. За разлику од класичних програма за посебне намене, као што је на пример програм за књиговодство, ВИ програми могу да праве разлику између резонувања (механизма закључивања) и постојећег знања (база знања). Како је база знања експлицитна и издвојена од механизма закључивања, програм може да "размишља" о свом сопственом знању као о улазним подацима.

### 4. Применљивост код лоше структурираних проблема и података.

Четврта карактеристика ВИ програма односи се на њихову ефикасност у раду са лоше структурираним проблемима. Код таквих проблема, класични програми су углавном неприменљиви. Проблем се третира као лоше структуриран уколико се алгоритам за његово решавање не може изразити експлицитно или уколико су неопходни подаци некомплетни, односно непрецизно специфицирани.

У табели 8.1 сумиране су основне разлике између приступа вештачке интелигенције и конвенционалног програмирања (*Hunt, 1986*).

Табела 8.1 Разлике ВИ и конвенционалног програмирања

Вештачка интелигенција	Конвенционално програмирање
Претежно симболичка обрада	Претежно нумеричка обрада
Хеуристичко претраживање	Алгоритми
Управљачке структуре најчешће издвојене од знања	Информације и управљање интегрисани заједно
Једноставно модификовање, ажурирање и проширивање	Тешко модификовање
Толеришу се погрешни одговори	Неопходни су тачни одговори

### 8.3 Основни елементи вештачке интелигенције

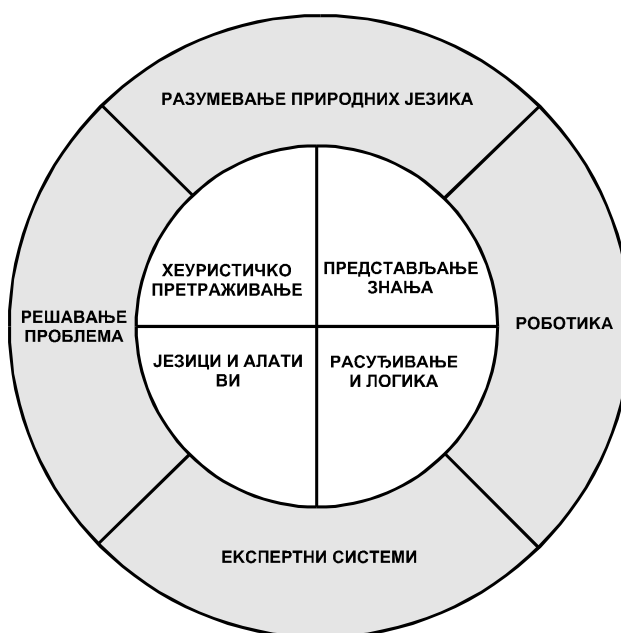
Елементи ВИ могу се графички представити као интегрисани круг на слици 8.2. У средини се налазе основни елементи из којих су компоноване апликације приказане на спољном прстену (*Hunt, 1986*).

#### 8.3.1 Хеуристичко претраживање

Посматрајући на слици 8.2 квадрант означен као хеуристичко претраживање, можемо третирати решавање проблема у ВИ као избор између алтернативних решења. Према томе, могуће је представити простор резултујућих стања као хијерархијску структуру коју називамо пирамида или дрво. Резултујући пут води од почетног стања (чвор-корен), преко различитих грана дрвета и завршава се у неком чвору који се назива "циљни чвор".

Код сложених и обимних проблема, очигледно је да би цртање оваквог дрвета које би обухватило све могуће путеве и изналажење најбољег решења на тај начин, представљало тежак или узалудан посао. Стога, дрво је најчешће имплицитно, односно рачунар генерише гране и чворове приликом тражења решења.

Код крајње једноставних проблема, исправан али временски дугачак приступ обухватао би изналажење одговарајуће шеме претраживања која би се потом следила све до коначног решења. Конвенционално програмирање примењује такав приступ где су познати сви кораци и елементи у проблему. Наравно, ово ограничава домен конвенционалног израчунавања на проблеме који се могу до детаља анализирати.



Слика 8.2 Елементи вештачке интелигенције

У проблемима којима се бави ВИ, а они су најчешће обимни и сложеније су природе, често се користи хеуристика (систем правила-пречица) која омогућава да се полазни проблем ограничи на "разумну" величину. Према томе, хеуристичке методе имају за циљ да ограниче простор стања решења, користећи информације о природи и структури посматраног проблема. Хеуристичко претраживање један је од основних доприноса вештачке интелигенције решавању проблема. По својој природи, хеуристика може да доведе до грешака. Она не гарантује увек да ће одговор бити исправан, али се њеном применом повећава вероватноћа изналажења употребљивог одговора.



### 8.3.2 Представљање знања

По својој природи, знање је сложеније и вредније од информације. Термин знање најчешће се односи на скуп информација о специфичном подручју посматрања, које су структуриране на начин да као такве могу да буду корисне. За разлику од података, који су пасивни, за знање кажемо да је активно. Када за некога кажемо да поседује много знања о нечему, то не значи само да он "чува" мноштво чињеница, већ и да може да користи те информације за анализу проблема и доношење одлука.

Сврха представљања знања је да се оно организује у такву форму да ВИ програм може директно да га користи у процесу одлучивања, планирању, препознавању објеката и ситуација, извођењу закључака и подржавању осталих когнитивних функција. Стога је представљање знања кључно питање код експертних система, препознавања ликова и препознавања природних језика.

Шеме за представљање знања делимо на декларативне и процедуралне. Декларативне се односе на представљање чињеница и тврдњи, док се процедуралне односе на акције које треба предузети. Декларативне шеме обухватају релационе (семантичке мреже) шеме и логичке шеме. О појединим најзначајнијим начинима представљања знања биће речи у делу текста о експертним системима.

### 8.3.3 Логичко закључивање

Логичко закључивање - доношење закључака на основу логике, уобичајено се спроводи "доказивањем теорема". Најпопуларнији метод за аутоматско доказивање теорема је процедура резолуције (разлагања). То је општи аутоматски метод за одређивање да ли теорема (хипотеза-закључак) произилази из постављеног скупа премиса (аксиома). Прво се, коришћењем стандардних идентитета оригиналне премисе, у форму реченице стављају закључци који се проверавају (потврђују). Потом се негира закључак који се проверава. Следи аутоматско извођење нових реченица коришћењем резолуције и других процедура. Уколико се дође до контрадикције, теорема је доказана. Метода резолуције није погодна код сложених проблема пошто простор претраживања генерисан на овај начин расте експоненцијално са бројем формула које се користе за опис проблема.

Постоје и други приступи доказивању теорема и решавању проблема (*Hunt, 1986*) који превазилазе оквире овога излагања те неће бити разматрани.

#### 8.3.4 Језици и алати вештачке интелигенције

Вештачка интелигенција је настала као експериментална наука са циљем да развије рачунарске програме који би опонашали интелигентно (људско) понашање. Ово се показало као веома тежак посао који је захтевао најбоље програмске алате.

Један од проблема који је утицао на карактеристике језика ВИ био је неопходност динамичке алокације меморије, за разлику од статичке алокације коју су примењивали дотадашњи језици за конвенционално програмирање. Други, можда значајнији аспект који је утицао на профилисање посебних ВИ алата и језика, огледао се у непредвидивости структура и форми које би добијали подаци током извршења неког програма.

На карактеристике ВИ језика утицала је и чињеница да су истраживачи у области ВИ утврдили да рекурзивно извршавање програма значајно поједностављује писање самог програма. Стога су језици и алати ВИ морали да пруже подршку рекурзивној обради. На крају, програми ВИ превасходно се баве манипулацијом симболима, а мање самим нумеричким израчунавањем. Ова је чињеница такође битно утицала на специфичности језика који су развијени за ВИ.

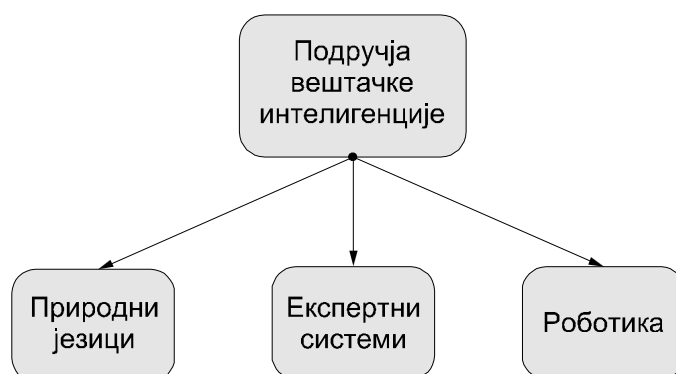
Најпознатији програмски језици, који су прилагођени апликацијама које се решавају техникама вештачке интелигенције су *LISP*, *PROLOG* и *SMALTALK*.

#### 8.4 Основна подручја вештачке интелигенције

Системи вештачке интелигенције могу се класификовати у три основна подручја (*Rauch-Hindin, 1986*), (слика 8.3):

- ◆ експертни системи,
- ◆ природни језици,
- ◆ роботика (препознавање ликова, говора и додира, ...)

Експертни системи су програми који користе процес закључивања (резоновања) налик људском код решавања проблема из разноврсних домена. Ово закључивање заснива се на експертском људском знању, које је кодирано у програму у виду посебне структуре која се назива база знања. Посебан механизам користи расположиво знање у процесу закључивања, са циљем да дође до решења проблема. На тај начин решавају се проблеми који су изван домаћаја конвенционалних рачунарских програма.



Слика 8.3 Основна подручја ВИ

Друго подручје, названо системи природних језика, обухвата програме који препознају природни језик корисника, на пример енглески. Ширење подручја примене рачунара праћено је чињеницом да се све већи број људи користи рачунаром у свакодневном послу. То је наметнуло питање интеракције рачунара и људи који нису рачунарски образовани, односно људи којима непознавање програмских језика, оперативних система и принципа функционисања рачунара представља ограничење у комуникацији са рачунаром. Како је број тих људи сваки даном све већи, јавила се потреба за развојем система који би дозвољавали коришћење природних језика у комуникацији корисника са рачунаром. У том смислу, један број истраживача у области ВИ бави се пословима препознавања и тумачења људског говора од стране рачунара. Програми који су с тим циљем развијени, омогућавају лакшу комуникацију са људима, пошто елиминишу потребу да се уче стилизовани, формални рачунарски језици.

Трећи тип програма ВИ обухвата једноставне системе за перцепцију вида, говора и додира. Рачунарски визуелни системи, на

пример, могу да интерпретирају визуелне сцене или да доносе закључке о квалитету или физичкој орјентацији одређених објеката који су у фокусу телевизијске камере. Њихове могућности су ограничене, а услови под којима се препознавање врши, строго су одређени. Ово се подручје ВИ у ширем смислу третира као роботика.

## 8.5 Експертни системи

Истраживања у области вештачке интелигенције у последње две деценије дала су многе важне резултате. Међу најзначајније свакако спада развијање и примена програма који су познати под називом *експертни системи*. Ови системи данас су предмет проучавања читаве једне гране вештачке интелигенције и рачунарске технике уопште, која се назива инжењерство знања (*knowledge engineering*).

Експертни системи (ЕС) су "интелигентни" програми у које је на погодан начин уграђена велика количина висококвалитетног знања из неког домена људске активности, а који могу да процесирају то знање у циљу успешног решавања одређеног проблема на начин који би се сматрао интелигентним када би те исте проблеме решавао човек.

У којој ће мери један ЕС у свом раду испољавати способност интелигентног решавања проблема који му је задат, зависи пре свега од знања које је у њега уграђено (*Waterman, 1985*). Сматра се да највеће и најквалитетније знање из неке области имају људи који су у тој области експерти. Зато се настоји да знање које су уграђује у ЕС током његовог развоја, по свом квалитету и количини буде у што већој мери налик знању експерата у тој области.

Рад већине данашњих експертних система заснива се на симболичком представљању и процесирању уграђеног знања. Знање се представља преко формалних симбола и погодних структура података исказаних у неком програмском језику, а проблеми се решавају извођењем индуктивних и дедуктивних закључака путем манипулације тим симболима и структурама.

Један од основних разлога због којих се имплементирају експертни системи је жеља да експертско знање у разним областима постане доступно кроз примену рачунара. Експерти нису увек на

располагању када треба решити неки проблем, а када су доступни, њихове услуге су по правилу скупе. Осим тога, људско знање је подложно "деградацији" односно временом се губи под налетом нових информација. Зато је потребно стално примењивати знање да би се задржао његов квалитет. Насупрот томе, знање које је уграђено у ЕС је постојано и увек доступно, без обзира колико често се систем користи.

Карактеристично је и то да поједини стручњаци имају тешкоће у артикулацији, документовању и преношењу свог знања другим људима, док су те активности код ЕС аутоматизоване и стога веома једноставне.

Важно је напоменути и то да при развоју ЕС долази до експлицитног формализовања знања од стране експерата, што омогућава даље продубљивање сазнања у доменима у којима се ови програми примењују.

## 8.6 Дефинисање експертних система

У литератури појам "експертни систем" дефинише се на различите начине, што је и разумљиво када се узме у обзир чињеница да се ради о разноврсним програмима који могу бити имплементирани у различитим језицима и за различите оперативне системе.

Експертни системи се могу дефинисати уже или шире посматрано. Уже посматрано, дефиниције се односе на технике експертних система које олакшавају процес програмирања рачунара и чине га ефикаснијим. Шире посматрано, ове технике представљају само први корак процеса који ће трансформисати начин рачунарске обраде података, превodeћи технологију програмирања са поља нумеричке обраде на поље логичке обраде и симболичког програмирања.

У литератури је често сврставање дефиниција ЕС-а у две основне групе (*Pedersen, 1989*). У прву групу спадају оне дефиниције ЕС-а које примарно објашњавају "како су ЕС имплементирани". На пример: "Експертни системи су рачунарски програми који користе технике закључивања."

У другу групу дефиниција спадају оне које потенцирају аспект "људског знања", на пример: "Експертни системи су рачунарски

програми који користе људско знање за решавање проблема који уобичајено захтевају људску интелигенцију". Иако су појмови људског знања и људске интелигенције још увек недовољно прецизно одређени, ова дефиниција је прихватљивија у односу на претходну.

Дефинишући ЕС-е, бројни аутори усмеравају пажњу на поједине аспекте овог појма. Тако на пример, *Basden* ставља нагласак на *методологију* и означава експертне системе као "рачунарске системе који могу чувати људско знање било које врсте и који могу то знање обрађивати на начин који је сличнији људима, него што то чине конвенционални рачунарски системи" (*Basden, 1983*).

С друге стране, *Bramer* ставља нагласак на *особине* и дефинише експертни систем као "рачунарски систем који сачињава организовано знање које се тиче неког специфичног подручја људске вештине, довољно да се он понаша као вешт и ефикасан консултант. Дакле, то је систем специјалне намене креиран да усвоји вештину неког експерта као што је доктор медицине, хемичар или машински инжењер" (*Bramer, 1982*).

Интересантна је и дефиниција коју је дао *Stefic* са групом аутора, која успоставља везу између експертних система и вештачке интелигенције: "Експертни системи су програми за решавање суштинских проблема за које се генерално признаје да су тешки и да захтевају експертизу. Каже се да су засновани на знању пошто њихове особине зависе од коришћења чињеница и хеуристика које користе експерти. ЕС су коришћени као средства за истраживања у области ВИ уз образложење да представљају снажан алат у истраживањима везаним за решавање проблема" (*Stefic i dr., 1982*).

Анализирајући наведене дефиниције, можемо резимирати да су експертни системи рачунарски програми развијени са циљем да послуже као консултанти у решавању сложенијих проблема, који захтевају људску експертизу.

## 8.7 Експерти и експертни системи

Оно што неког стручњака у одређеном домену чини експертом, састоји се од одређеног знања из тог домена, способности разумевања проблема и задатака из тог домена, као и вештине и искуства које он примењује при решавању тих проблема. Захваљујући својој обучености и искуству, стручњак је у стању да обавља послове које већина других људи не може да обавља. Стручњаци своје знање користе на ефикасан начин, у стању су да брзо дођу до решења, да при томе користе разне досетке, да објасне и образложе оно што раде, да одреде степен важности неког податка за решење проблема и да елиминишу ирелевантне податке из разматрања.

Експерти су у стању да конкретан проблем који решавају препознају као пример неког типског задатка са којим су добро упознати. Другим речима, поред познавања основних чињеница, дефиниција и теорија из књига и других референци у некој области, експерт поседује и неке личне способности, сналажљивост и "осећај" за проблеме, дакле све оно што чини његово "приватно знање". Управо та врста знања омогућује му да путем тзв. "здраворазумског резонувања" примењује оно опште знање из књига, које је доступно и другима. Ова врста знања обично се назива хеуристичким знањем. На основу хеуристичког знања, експерт може између осталог да:

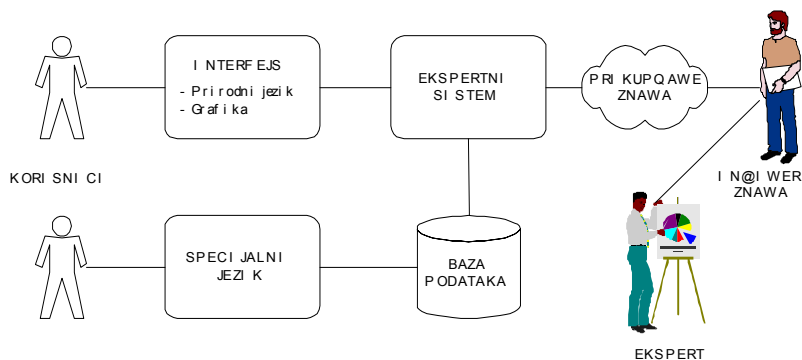
- ◆ препозна на који ће начин најбрже доћи до решења;
- ◆ осети када је приступ решавању неког проблема исправан, када погрешан, а када само вероватно добар;
- ◆ се сналази у ситуацијама када су подаци којима располаже некомплетни или недовољно тачни.

Расветљавање, тумачење, презентација и репродуковање оваквог знања на рачунару представља централни задатак при развијању ЕС. Акценат код ЕС, дакле, није на прецизно дефинисаним и увек важећим алгоритмима и процедурама у неком домену људских активности, већ на декларативном и хеуристичком знању.

Постоји више разлога због којих се код ЕС главна пажња обраћа на чињенично и хеуристичко знање, а не на алгоритамска решења проблема из појединих домена. Најважнији су следећи (*Девеџић и Божовић, 1994*):

- ♦ За многе проблеме и не постоје јединствена и прихватљива алгоритамска решења, будући да се такви проблеми јављају у сувише сложеним физичким и/или социјалним условима, који се не могу прецизно описати и анализирати.
- ♦ Чест је случај да постојећи математички и други алгоритми не омогућају представљање знања потребног за извођење одређених закључака у вези са посматраним проблемом или за коришћење различитих извора знања.
- ♦ Знање стручњака је нешто што има одговарајућу цену и вредност. Прикупљање тог знања од стручњака и формално представљање таквог знања на рачунару може значајно да смањи цену репродуковања и коришћења знања специјалисте.

На слици 8.4 дата је блок шема која приказује релације између ЕС и окружења у коме он ради (Meyers, 1986). Знање које се уграђује у ЕС, као најважнији део самог система, потиче од особе која је експерт у неком домену. По правилу, експерт није стручњак и за област рачунарства па је потребан посредник који ће помоћи да се од стручњака добије и пренесе у рачунар знање које овај поседује.



Слика 8.4 Релације између ЕС и његовог окружења

Инжењер знања (*knowledge engineer*), односно стручњак за ЕС, настоји да у консултацијама са експертом или на неки други начин дође до чињеничног и хеуристичког знања из дате области, да то знање кодира на одговарајући начин и унесе га у ЕС. Овај процес је од изузетне важности за развијање, тестирање и дограђивање ЕС и обично се означава као процес прикупљања знања (*knowledge*



*aequisition*). Инжењер знања често има на располагању посебан програмски пакет који олакшава тај посао. С друге стране, корисник најчешће комуницира са ЕС преко терминала, користећи посебне програмске пакете који олакшавају комуникацију и уносе већи степен интелигенције у читав систем. То могу да буду разне врсте графичких интерфејса, системи за комуникацију неким формалним или природним језиком, као и системи за препознавање говора. Најзад, иако се ЕС развијају за поједине узане домене у некој области људске делатности, данас све више постоји тенденција везивања тих система за базе знања шире намене, које садрже општије знање од оног које се непосредно уграђује у ЕС. Претраживање ових база података у новије време је олакшано коришћењем посебних језика који служе као интерфејс према кориснику.

## 8.8 Експерни системи и конвенционални програми

Конвенционални програми углавном се употребљавају за обраду великих количина података који су најчешће нумеричког типа. Ова обрада врши се према јасним и тачно дефинисаним алгоритмима, који корак по корак воде систем (програм) ка решењу проблема. Уколико програм по својој семантици одговара постављеном проблему и уколико су улазни подаци тачни, конвенционални програм ће резултирати тачним решењем постављеног проблема. Конвенционални програми раде на начин који је најчешће само програмерима разумљив.

Експертни системи се понашају другачије. Углавном манипулишу симболичким подацима и не раде по унапред задатим алгоритмима, или бар не по алгоритмима у класичном значењу те речи. Неалгоритамски приступ једно је од основних обележја ЕС-а. ЕС су у великој мери интерактивни и у сваком тренутку се могу зауставити. Тада корисник експертног система може да прегледа закључке до којих се дошло до тада, да врати процес закључивања од почетка, да унесе нове чињенице, и сл. Експертни системи дају закључке који не морају да буду ни тачни ни погрешни већ су у већој или мањој мери вероватни и поуздани.

Проблеми који се решавају коришћењем ЕС-а најчешће су слабо структурирани, те не подлежу математичком моделирању и формализацији. То значи да је стандардни алгоритамски приступ практично неупотребљив у решавању оваквих проблема.

Класични принципи програмирања дефинишу алгоритме и податке. Алгоритми изражавају начин долажења до решења. Они су јасни и експлицитни, чак и кад се користе сложене програмске структуре (петље, гранања, рекурзије). Човеково знање није погодно за такве моделе јер је структурирано на другачији начин. Експерти не раде строго алгоритамски. Они се користе не само знањем, већ и искуством. Често на основу искуства и расуђивања одлучују како ће решавати проблем.

Пошто стандардни алгоритамски приступ није од велике користи, код експертних система се приступа употреби *хеуристика*. Хеуристика се дефинише као скуп емпиријских и сврсисходних потеза, (правила, поступака) који у својој укупности, опортунистички примењени, обично воде решењу (*Pidd, 1989*). Примена хеуристика доприноси да се скрати средњи број покушаја у току решавања неког проблема. Хеуристика као метода захтева извођење следећих корака:

- ◆ разбијање проблема на мање проблеме или подпроблеме са одређеном организацијом циљева и подциљева понашања;
- ◆ дефинисање оцена карактеристика везаних за дату област примене експертног система, ради рангирања и избора алтернатива;
- ◆ за решавање подциљева користе се рекурзивне методе;
- ◆ хеуристички кораци у решавању проблема за прелазак у наредни корак користе стање из претходног корака.

Осим хеуристике, за градњу експертних система су потребни и неки други елементи: рачун предиката, симболичко програмирање, и др. на којима се на овом месту нећемо посебно задржавати.

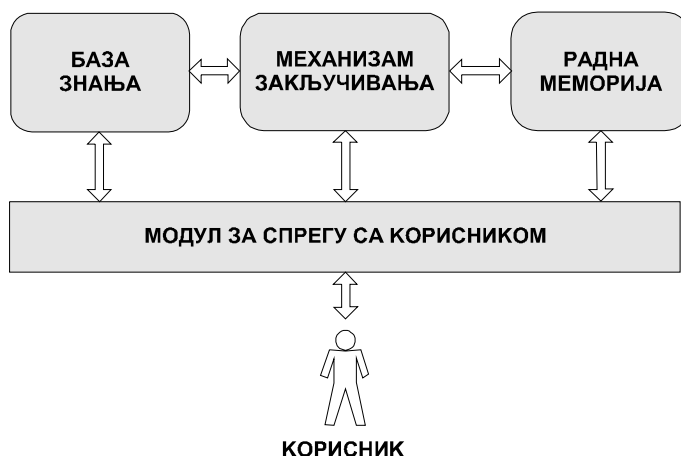
У Табели 8.2 обједињене су основне разлике између конвенционалног програмирања и експертних система.

Табела 8.2 Поређење конвенционалне обраде и ЕС

КОНВЕНЦИОНАЛНИ ПРОГРАМИ	ЕКСПЕРТНИ СИСТЕМИ
Алгоритми	Хеуристике
Представљање и коришћење података	Представљање и коришћење знања
Циклички процеси	Процеси закључивања
Знање и методе знања су помешани	Одвојен модел решавања (база знања) од дела који управља базом знања (механизам закључивања)
Знање организовано у два нивоа: - подаци - програми	Знање организовано у три нивоа: - подаци (база података) - база знања - управљачка структура
Ново знање захтева репрограмирање	Ново знање се додаје без репрограмирања

## 8.9 Структура експертних система

Основни саставни делови сваког ЕС, без обзира на величину, сложеност, начин коришћења и подручје примене су: база знања (*knowledge base*), механизам закључивања (*inference engine*) и радна меморија (*working memory*) (Hayes-Roth, Waterman and Lenat, 1983). У већини случајева, ЕС имају и интерфејс према кориснику (*user interface*). Код ЕС консултативног типа, он је обавезан саставни део, као и претходна три. Овако дефинисана структура ЕС приказана је на слици 8.5.

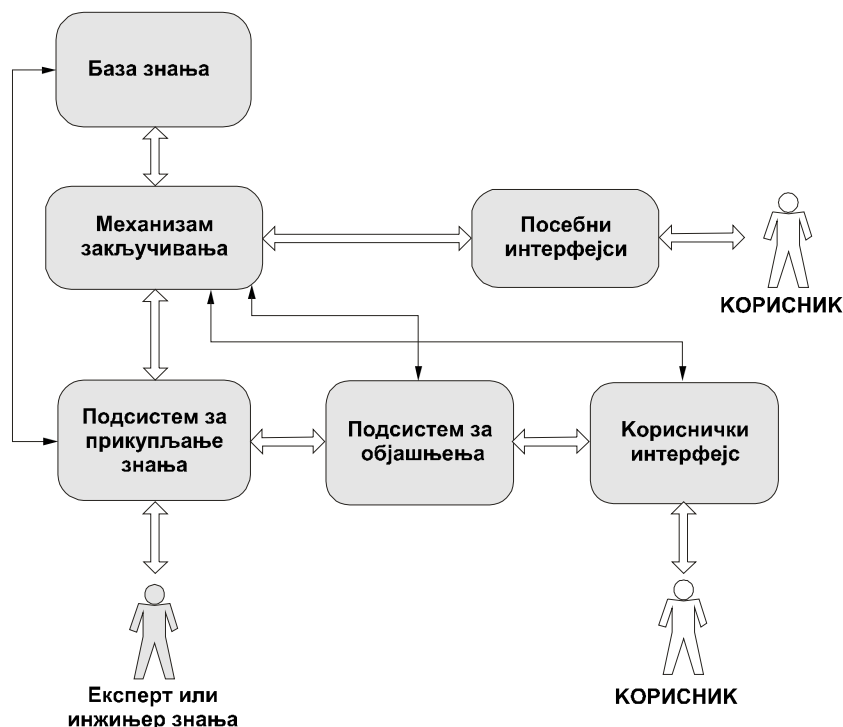


Слика 8.5 Основна структура експертног система

Често, међутим, поред набројаних основних елемената, архитектуру експертног система сачињавају и помоћни модули (*Harmon i Sawyer, 1990; Harmon, Maus i Morrissey, 1988*), као што су подсистем за прикупљање знања (*knowledge acquisition subsystem*), посебни интерфејси (*special interfaces*) и систем за објашњења (*explanation subsystem*). На слици 8.6 приказана је овако дефинисана, проширена структура ЕС-а.

База знања ЕС је специјализована, јединствена за конкретни ЕС и садржи знање експерта из одређене области. Оно је унето у ЕС кроз систем прикупљања знања и не мења се током рада система. Радна меморија садржи тренутне податке о проблему који ЕС решава. Ти подаци су променљиви и својим вредностима одражавају тренутно стање у процесу решавања проблема. Механизам закључивања је програм који на основу тих променљивих података и фиксног знања уграђеног у базу знања решава проблем, односно обавља задатак који се поставља пред ЕС. Преко интерфејса према кориснику одвија се комуникација између система и корисника и презентација резултата.

У наставку текста детаљно су описани различити начини представљања експертског знања, манипулисање подацима у радној меморији, као и начин функционисања ЕС, тј. начин рада његових активних делова - механизма закључивања и интерфејса према кориснику.



Слика 8.6 Проширена структура експертног система

## 8.10 Представљање знања

Знање из домена примене ЕС уноси се у базу знања у виду тврдњи, чињеница, принципа и правила која важе у том домену. То знање се може представити у бази знања коришћењем разних техника. Често се користе и комбинације две или више техника, зависно од домена примене, процеса прикупљања знања и расположиве програмске подршке за синтезу базе знања. Технике представљања знања груписане су око два основна приступа проблему представљања знања у ЕС, декларативног и процедуралног.

Декларативни приступ подразумева представљање знања у облику независних модуларних целина, исказа, чињеница и структура података карактеристичних за домен примене ЕС. Ти елементи знања су пасивни, што значи да не представљају програмске

целине које у себи садрже експлицитан низ програмских команди и експлицитан редослед извршавања тих команди. Једини програм који користи те елементе при решавању проблема је механизам закључивања. Основна предност декларативног приступа у представљању знања је модуларност знања. Измене у бази знања и њена надградња су захваљујући томе олакшане. Нажалост, интерпретација тако представљеног знања може да смањи брзину рада ЕС.

Процедурални приступ се односи на директно уношење знања у програмске процедуре, у виду експлицитног програмског кода. Механизам закључивања при раду позива те процедуре и тако користи знање уграђено у њих. Процедуре се брже извршавају, али је тежа њихова измена и уношење новог знања у систем.

У досадашњем развоју области ЕС највише су се примењивале следеће три технике за представљање знања:

- ◆ продукциона правила (*production rules*)
- ◆ семантичке мреже (*semantic nets*)
- ◆ оквири (*frames*)

Све три технике су у највећој мери декларативне природе, иако могу да имају и неке имплицитне или експлицитне процедуралне елементе. Оне задовољавају три класична критеријума за представљање знања у ЕС (*Barr, Feigenbaum i Cohen, 1981*):

- ◆ Могућност проширивања - програми и структуре података требало би да буду флексибилни како би се проширивање базе знања вршило без великих промена програма.
- ◆ Једноставност - представљање знања треба да буде једноставно чак и за онога ко се не бави информатиком. Једноставност концепта може да се оствари ако се знање изражава на што је могуће хомогенији начин или ако се развију посебне функције за приступ нехомогено представљеним знањима.
- ◆ Експлицитност - знање треба да буде представљено експлицитно. Ово је важно у откривању грешака и давању објашњења о раду ЕС.

### 8.10.1 Продукциона правила

Експертни ситеми код којих је знање представљено у облику правила често се називају продукциони системи (*rule-based systems*). Правила се могу схватити као елементи знања, односно елементарне количине знања из одређеног домена (*Michaelsen, Michie and Boulanger, 1985*).

Правило представља логичку релацију између елемената проблемског подручја и може се изразити на следећи начин:

"ако" X "тада" Y

што значи : "ако важи претпоставка X тада се може закључити Y" или "ако је ситуација X тада се предузима акција Y".

Другим речима, правило је логички израз типа АКО-ТАДА (*If-Then*) са значењем: ако важи нека премиса (скуп премиса), тада се може извести закључак (скуп закључака) или се може предузети нека акција (више акција). Овакав начин изражавања знања је врло природан и одговара тежњи да знање буде модуларно. Такође је задовољен захтев за лако модификацијом базе знања јер се нова правила додају прилично независно од осталих, а систем је транспарентан, тј. лако се објашњава начин на који се долази до закључака. Због свог облика, продукциона правила су погодан начин за представљање оне врсте знања која је заснована на логици.

Закључци се изводе поређењем групе правила са скупом чињеница или знања о тренутној ситуацији. Уколико је "ако" део правила задовољен, извршава се акција одређена са "тада". Када се ово догоди, каже се да је правило *испаљено*, *извршено* или *окинуто*. На пример:

Премисе	If	Број слободних канала > 0	(and)
		Број слободних линија > 0	(and)
		Позиваоц ≠ Позвани	(and)
		Прекид > Почетак	(and)
Закључак	THEN	Разговор је у току	

Описано правило садржи четири премисе (IF-клаузуле) и један закључак (THEN-клаузулу). У општем случају, правило може да садржи већи број IF и THEN-клаузула. IF-клаузуле и THEN-клаузуле се најчешће могу формулисати у виду простих логичких исказа. Значење тих исказа обично је испитивање вредности одређених параметара у IF-клаузулама, односно додељивање, брисање или измена вредности параметара у THEN-клаузулама. THEN-клаузуле могу такође да представљају и акције које треба извршити када су задовољене премисе правила. Интересантно је приметити да акција која следи по окидању правила може да се односи и на модификацију базе знања, односно додавање новог правила у базу или промену већ постојећег правила.

Основна идеја код продукционих ЕС је у томе да се на проблем, описан подацима у радној меморији, итеративно примењују правила из базе знања. Уколико се опис проблема "подудара" са IF-клаузулама једног или више правила, решење проблема се може наћи у THEN-клаузули тог/тих правила.

Важно је нагласити да се продукциона правила ЕС не налазе имплементирана у програмском коду, већ су смештена у бази знања као подаци, са произвољним редоследом. Механизам закључивања садржи у себи посебан програм, тзв. интерпретатор правила (*rule interpreter*) који је задужен за процесирање и интерпретацију ових правила у току рада система.

У горњем примеру подразумева се да премисе важе са апсолутном сигурношћу да би закључак правила могао да се изведе, а само извођење закључка такође се врши са апсолутном сигурношћу. Чест је случај, међутим, да експерти при решавању проблема изводе закључке који важе са сигурношћу мањом од 100%, као и да се закључци доносе на основу података који нису у потпуности поуздани. Продукциона правила омогућују да се прикаже и неизвесност у доношењу одлука (*Pedresen, 1989; Harmon and Sawyer, 1990*). Један од најчешћих начина за представљање неизвесности је коришћење фактора извесности (поузданости, вероватноће) у клаузулама правила. Фактори извесности су бројеви који представљају степен извесности са којим се одређене чињенице могу сматрати важећим, односно вероватноћу да важе одређени закључци у случају да су присутни само неопходни фактори који их подржавају.

У радној меморији, током рада ЕС налазе се чињенице, тврђења, закључци и други релевантни подаци везани за опис проблема који



се тренутно решава. Ови подаци су променљиви, генеришу се, мењају или губе важност током рада система. У системима заснованим на правилима, подаци у радној меморији организовани су у облику тврдњи, по облику врло сличних клаузула правила. На основу иницијалних података у радној меморији и правила у бази знања, ЕС изводи закључке у вези са постављеним проблемом.

### 8.10.2 Семантичке мреже

Мреже су најопштија и најстарија презентациона шема у вештачкој интелигенцији. Семантичке мреже су групе објеката, односно чворова (*nodes*), који су повезани оријентисаним луковима (*arcs*), односно гранама (*links*), које представљају бинарне релације између објеката, односно стављају објекте у одређени однос (*Waterman, 1985*). Чворови и гране су означени именима објеката, односно релација које повезују објекте, чиме је одређена семантика мреже. Реч "објекат" овде се користи у апстрактном значењу. У суштини, чворови могу да представљају физичке објекте, појмове, особе, догађаје, итд. Гране могу да означавају различите врсте односа између објеката, а типични односи су односи припадања врсти, однос поседовања, однос атрибута, исл. Знање представљено семантичком мрежом може се схватити и као скуп бинарних исказа који описују те односе.

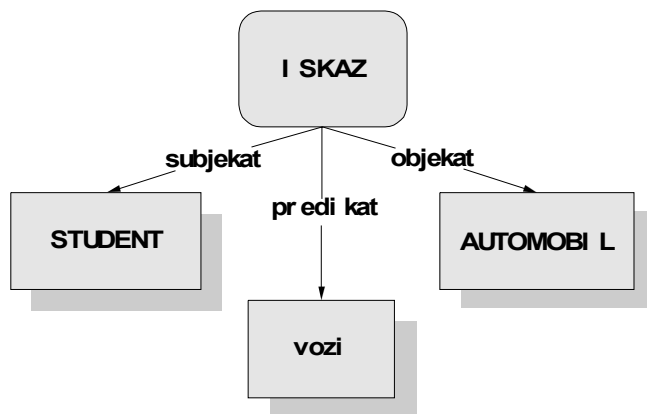
Пар чворова повезан релацијом, може се посматрати као једноставан исказ, на пример:

СТУДЕНТ  $\xrightarrow{\text{ВОЗИ}}$  АУТОМОБИЛ

Запазимо да је лук усмерен од субјекта (студент) ка објекту (аутомобил). Исти исказ могуће је представити и у облику приказаном на слици 8.7 који је погоднији пошто омогућава приказ сложенијих исказа. Наиме сваки чвор може бити повезан са неограниченим бројем других чворова, чинећи тако мрежу исказа.

Тројке облика објекат-атрибут-вредност (О-А-В тројке) су специјалан случај семантичких мрежа са само три врсте чворова (објекти, атрибути и вредности) и две врсте грана (јесте и има). Гране "објекат-атрибут" су типа "има", а гране "атрибут-вредност" су типа "јесте". Ако се у неком ЕС посматра само један објекат, О-А-В, тројке се редукују у парове облика Атрибут-Вредност (А-В парови).

A-B парови су слични тројкама O-A-B, с тим да код A-B парова не постоји могућност представљања већег броја објеката који су у неком међусобном односу.

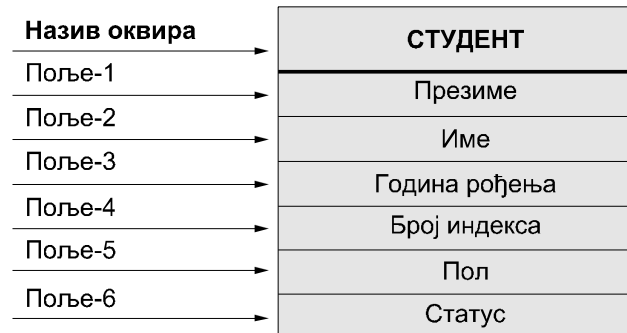


Слика 8.7 Један начин приказивања исказа у семантичким мрежама

Тренутни подаци у радној меморији са којима ради неки ЕС чија је база знања представљена преко семантичке мреже су конкретни објекти и њихове просте релације са објектима дефинисаним у мрежи. Од ЕС се захтева да изведе закључак о неким сложенијим односима и особинама датих објеката.

### 8.10.3 Оквири

Оквир је комплексна структура помоћу које се може представити неки појам или објекат (*Minsky, 1985*). Сваки оквир припада једном објекту и садржи произвољан број означених поља (*slots*) у које се смештају чињенице од значаја за тај објекат, односно атрибути који представљају карактеристичне особине помоћу којих се описује посматрани објекат (слика 8.8). Сваки атрибут има низ својстава (*facts*), као што су на пример стварна или подразумевана (*default*) вредност атрибута. Својство атрибута не мора да буде само неки пасивни епитет, већ може да буде и произвољна процедура, која се извршава аутоматски под одређеним околностима (на пример при промени вредности атрибута). Поља могу бити празна или могу указивати на неки други оквир. На тај начин се скуп оквира повезује у мрежу оквира.



Слика 8.8 Пример једног оквира

У општем случају, слотове можемо попунити на следећи начин:

- ♦ декларативним представљањем - једноставним навођењем тврдњи које се сматрају тачним;
- ♦ процедуралним представљањем - скупом инструкција које по извршењу дају резултат конзистентан са почетном чињеницом;
- ♦ процедуралним придруживањем - у овом случају слот садржи низ инструкција за одређивање улаза у слот.

Оквири такође могу да садрже подоквири (*subframes*), који поред свих атрибута оквира у којима се налазе и које од њих наслеђују, могу да имају и неке специфичне атрибуте. Подоквири, опет могу да имају своје подоквири, итд. Подоквири су повезани са надређеним оквирима преко атрибута "је (сте)" или "врста".

Оквири су погодни за кодирање оне врсте знања која се односи на стереотипне карактеристике појединих класа и подкласа појмова, односно објеката. Из овог кратког описа јасно је да оквири омогућују мешовито представљање знања, комбинујући декларативне и процедуралне елементе у опису неког појма или објекта.

## 8.11 Процес закључивања

У процесу закључивања, механизам закључивања, на основу иницијалних података о проблему у радној меморији и знања које се налази у бази знања, покушава да пронађе решење проблема.

Уобичајено је да се од корисника захтева да унесе иницијалне податке који се смештају у радну меморију ЕС. Механизам закључивања затим примењује на те податке знање из базе знања, генеришући нове податке у радној меморији. Тиме се скуп података у радној меморији проширује. Ново стање у радној меморији може да буде довољно за решавање постављеног проблема и у том случају се процес закључивања завршава. У противном, проширени скуп података се поново обрађује коришћењем знања из базе знања, што доводи до нове промене стања у радној меморији. Процес се итеративно наставља док се не дође до стања у радној меморији које је довољно за решавање постављеног проблема или док се не покаже да такво решење није могуће добити. У класичном случају консултативних ЕС, механизам закључивања током тражења решења проблема може да захтева од корисника да унесе додатне податке уколико је то потребно (*Pedersen, 1989*).

У системима заснованим на правилима, интерпретатор правила механизма закључивања функционише на следећи начин. Иницијални подаци у радној меморији се третирају као појединачни узорци општијих тврђења дефинисаних у премисама и закључцима правила из базе знања. Интерпретатор правила претражује базу знања да би пронашао она правила чијим клаузулама одговарају подаци у радној меморији. Ово претраживање се назива препознавање облика (*pattern-matching*), и може се вршити на разне начине.

Тривијалан начин био би да се свако правило у бази знања упореди са свим подацима у радној меморији, али би то трајало дуго, осим у случају када је број правила у бази знања мали (неколико десетина). Зато се не ради на тај начин, већ се полази од чињенице да је основна ствар у интелигентном решавању проблема селективно и ефикасно долажење до решења на основу скупа могућих алтернатива. Интерпретатор обично користи неку стратегију за ефикасније претраживање базе знања, са циљем да се испитују само релевантна правила, она за која постоји јасна евиденција да барем делимично одговарају ситуацији у радној меморији. Коришћење таквих стратегија омогућује да се у раним фазама решавања проблема из скупа података у радној меморији издвоје они најкориснији и да се избегне испитивање оних алтернатива које не могу да доведу до правог решења. Две најпознатије стратегије тог типа су уланчавање унапред (*forward chaining*) и уланчавање уназад (*backward chaining*) и свака од њих се може реализовати применом више различитих алгоритама.

### 8.11.1 Уланчавање унапред

Уланчавање унапред полази од премиса правила. У сваком циклусу, интерпретатор испитује вредности премиса релевантних правила, поредећи их са подацима у радној меморији и одређује која су правила задовољена. Правило се сматра задовољеним када свака његова премиса одговара неком податку у радној меморији. Задовољено правило се може применити (каже се још и активирати, извршити или окинути), што значи да његове *THEN*-клаузуле представљају или истинита тврђења о проблему који се решава и као такве се могу додати у радну меморију, или акције које треба извршити (обично резултује променом стања у радној меморији).

Ако се при претраживању релевантних правила покаже да ни једно није задовољено, систем закључује да нема довољно података да би могао да реши проблем. У том случају, ЕС може да обустави решавање проблема или да захтева од корисника да унесе додатне податке. Ако се покаже да је само једно правило задовољено, оно се примењује. Уколико дође до конфликтног случаја када је задовољено више правила, стратегија претраживања унапред налаже да се одабере једно од њих које ће се применити. За доношење такве одлуке најчешће се примењује неки хеуристички поступак за резолуцију (разрешавање) конфликта (*conflict resolution*) (McDermott and Forgy, 1978). Применом одабраног правила долази до измене стања у радној меморији, па ЕС испитује да ли је том променом проблем решен. Уколико јесте, обавештава корисника о томе, а у супротном отпочиње нови циклус закључивања, препознавањем узорака у новонасталој ситуацији у радној меморији. Правила из скупа задовољених конфликтних правила која нису одабрана у претходном циклусу остају у том скупу, док на њих не дође ред у наредним циклусима, или док се променом стања у радној меморији не створе услови по којима неко од конфликтних правила више није задовољено. Интерпретатор уклања таква правила из скупа конфликтних правила.

Треба напоменути да једно правило може да буде задовољено са више различитих података из радне меморије. У скупу конфликтних правила се, у суштини, налазе конкретни узорци задовољених правила (*rule instantiations*). Један конкретан узорак задовољеног правила у складу је са тачно једним подскупом скупа података у радној меморији који задовољавају премисе правила. Псеудокод стратегије уланчавања унапред дат је на слици 8.9.

1. Формирати празан скуп конфликтних правила (КП);
2. Формирати скуп релевантних правила (РП);
3. Додати у КП свако правило из РП чији је *If*-део задовољен подацима тренутно присутним у радној меморији;
4. Ако КП није празан скуп:
  - а) Одабрати неко правило (П) из КП;
  - б) Избрисати П из КП;
  - с) Извршити *Then*-део правила П;
  - д) Ако је проблем решен, приказати решење и тиме завршити процес закључивања; у противном, вратити се на корак 2;
4. Захтевати уношење додатних података или обуставити процес закључивања и обавестити корисника да решење проблема није пронађено.

Слика 8.9 Стратегија уланчавања унапред (псеудокод)

### 8.11.2 Уланчавање уназад

Код уланчавања уназад, механизам закључивања претпоставља да важи неко решење проблема (циљ) из скупа могућих решења и проналази правило чије *Then*-клаузуле означавају то решење. У општем случају, таквих правила има више, али ради једноставности објашњења полазимо од претпоставке да постоји само једно такво правило.

Претпостављено решење, односно *Then*-део правила које означава то решење, назива се текућа (тренутна) хипотеза или текући (тренутни) циљ. За пронађено правило интерпретатор настоји да провери да ли је задовољено, што би значило да је претпостављено решење проблема тачно и процес закључивања би тиме био завршен. У том циљу интерпретатор покушава да провери да ли су задовољене премисе, односно *If*-клаузуле посматраног правила. Провера задовољености премиса се обавља на тај начин што се свака премиса правила проглашава за нову текућу хипотезу и тиме се поступак рекурзивно понавља за сваку од њих. Тиме се, заправо врши декомпозиција текућег циља на подциљеве (*subgoals*). Сваку текућу хипотезу интерпретатор третира као међукорак ка финалном решењу. Свака текућа хипотеза може се потврдити уколико је у складу са тренутним подацима у радној меморији или даљом рекурзијом.

У случају да се не нађе ни једно правило које потврђује претпостављено решење, систем поставља нову текућу хипотезу из скупа могућих решења. Уколико није потврђено ни једно од могућих решења, механизам закључивања обично захтева од корисника додатне информације. Због чињенице да код претраживања уназад механизам закључивања увек полази од неког претпостављеног решења (циља), механизми закључивања који користе ову стратегију често се називају "механизми вођени циљевима" (*goal-driven*) за разлику од механизма са претраживањем унапред који су "вођени подацима" (*data-driven*) (Winston, 1984) Псеудокод стратегије уланчавања уназад дат је на слици 8.10.

1. Формирати празан скуп меморисаних хипотеза (MX);
2. Формирати скуп могућих решења (MP) проблема;
3. Ако је скуп MP празан, захтевати уношење додатних података и вратити се на корак 2, или обуставити процес закључивања и обавестити корисника да решење проблема није пронађено.
4. Претпоставити да важи неко решење (P) из скупа MP;
5. Изабрати P из скупа MP;
6. Пронаћи правило чији *Then*-део задовољава претпостављено решење P и прогласити га за текућу хипотезу (TX);
7. Ако су све *If*-клаузуле правила TX задовољене:
  - а) Ако је скуп MX празан, P представља решење проблема које треба приказати кориснику и тиме завршити процес закључивања; у противном, прогласити хипотезу која је последња унета у скуп MX за текућу хипотезу, избрисати је из скупа MX и вратити се на корак 7;
8. Ако је било која *If*-клаузула правила TX у супротности са подацима у радној меморији:
  - а) Избрисати све евентуално преостале хипотезе из скупа MX;
  - б) Вратити се на корак 3;
9. Ако се за неку *If*-клаузулу правила TX не може установити на основу података тренутно присутних у радној меморији да ли је задовољена или не:
  - а) Унети правило TX у скуп MX;
  - б) Пронаћи правило чији *Then*-део задовољава ту *If*-клаузулу и прогласити га за текућу хипотезу;
  - с) Вратити се на корак 7.

Слика 8.10 Стратегија уланчавања уназад (псеудокод)

У процесу закључивања корисник може да захтева од ЕС да пружи објашњења за доношење одређених одлука или за постављање захтева кориснику за уношење додатних података.

Типичан случај је да корисник тражи од ЕС објашњење зашто или како се дошло до неког објашњења, одлуке, или захтева. ЕС на то може да одговори путем приказивања примењених правила и претходних текућих циљева на терминалу, уз одговарајуће коментаре. Тиме се омогућује кориснику да сагледа кораке закључивања које је механизам закључивања до тог тренутка предузео и логику коју је користио.

Поред наведених, и други елементи управљања могу да буду уграђени у механизам закључивања. Један од њих су приоритети података у радној меморији, којима се омогућује да механизам закључивања најпре испитује оне податке који се сматрају најважнијим за процес закључивања. Механизам закључивања такође може да води статистику о коришћењу појединих правила и да у појединим ситуацијама бира за извршавање оно правило које се најчешће или најређе користи. Заједничка особина свих тих елемената управљања је да инжењер знања при синтези ЕС не може да преко њих утиче на управљање процесом закључивања јер су они уграђени у механизам закључивања.

С друге стране, управљање процесом закључивања може да буде у извесној мери под контролом инжењера знања преко одређених елемената базе знања. Већ у самом начину на који је знање преведено у скуп правила, налазе се имплицитно елементи управљања. Такође се кроз правила могу дати и разне експлицитне информације о управљању. У ту сврху најчешће се користе тзв. метаправила. Она су истог облика као и сва друга правила, али не представљају знање из домена примене ЕС, већ контролно управљачко знање о начинима примењивања осталих правила у процесу закључивања ("знање о знању"). У њима се може тачно навести, на пример, редослед по коме треба приступити подциљевима у претраживању уназад или коју групу правила треба претражити да би се дошло до циља без разматрања осталих правила.



## 8.12. Симулација заснована на знању

Као метода за моделирање и анализу сложених реалних система, симулација се у науци и пракси користи већ годинама. Нарастајући захтеви, условљени повећањем сложености система који се моделирају, условили су појаву бољих, ефикаснијих и софистициранијих симулационих алата, нарочито последњих година. С једне стране, то је допринело да се значајно прошири круг корисника симулације, док је с друге стране, за истраживаче и аналитичаре то значило значајно поједностављење развоја нових модела. Међутим, и поред тога, поједине проблеме било је тешко или немогуће решавати традиционалним приступом у симулацији.

У традиционалном приступу (слика 8.11), аналитичар би креирао модел система, користећи при том неки програмски језик опште намене или специјализовани симулациони језик. Након валидације и верификације симулационог модела, приступило би се експериментисању са моделом. Добијени резултати касније би се анализирали како би се донела одређена одлука, односно испунио циљ симулационе студије.



Слика 8.11 Традиционални приступ у симулационом моделирању

Овако дефинисан, традиционални приступ у симулацији који подразумева и одговарајуће софтверске алате који корисницима и аналитичарима стоје на располагању, показао се као ефикасан само код оних проблема који су стриктно везани за конвенционално моделирање и симулацију. Овако дефинисано подручје примене у литератури је познато и као "симулација у малом" (*simulation in small*) (Spriet i Vansteenkiste, 1982).

С друге стране, појам "симулација у великом" (*simulation in large*), подразумева укључивање и свих оних активности које се на први поглед нису могле доводити у значајнију везу са симулацијом, али које се у крајњем случају нису могле избећи приликом извођења експеримената. Ове операције укључују претходну обраду и пост-

обраду информација, евалуацију и тумачење информација добијених симулацијом, и слично.

Како је циљ симулације, између осталог, и то да се преузме (аутоматизује) део послова од истраживача, функције које су се развијале у оквиру овог ширег приступа, као на пример, разни помоћни програми и програмски пакети, водиле су ка испуњењу тако постављеног циља.

Нова и напреднија решења у том правцу наглашена су у истраживањима која су спроведена у последње две деценије и која су резултирала практичном применом апликација и метода вештачке интелигенције, посебно експертних система. Апликације везане за базе знања и одлучивање на основу закључака до којих се долази претраживањем таквих база знања, омогућиле су значајан корак напред ка потпуном аутоматизовању процеса анализе система на основу симулације.

Комплексност проблема, присутна у реалним апликацијама, омогућава коришћење ВИ у симулацији на два начина (*Paul, 1989*). Прво, могуће је креативност симулационог моделирања побољшати додавањем софтверских апликација вештачке интелигенције у виду разних алата за анализу и друго, могуће је делове животног циклуса симулације моделирати помоћу ВИ апликација. Проблеми везани за прикупљање потребног знања за формирање симулационог модела и извођење симулације налажу потребу да се из тог знања извуче максимум користи. То заправо значи да процес симулације, уместо да само даје одговоре на питања типа "шта-ако", треба да омогући и широки опсег метода закључивања, одлучивања и претраживања, које су иначе доступне и добро развијене у ВИ. Овако дефинисан "широки" приступ симулацији најчешће се назива симулација заснована на знању (*knowledge-based simulation*) (*Rothenberg, 1989*).

### 8.13. Аспекти интеграције ЕС и симулационих модела

Приликом разматрања могуће интеграције експертних система са симулационим моделима, могу се уочити неке њихове карактеристичне разлике и сличности (*Lehmann, 1986*). Прва сличност се огледа у томе што се оба ова приступа могу користити за чување и примену знања о структури, организацији и апликацији динамичких система (дискретних или континуалних). Разлика лежи у

томе како се ово знање представља и како им се приступа у процесу резонувања. Док симулација користи технике нумеричког, процедуралног израчунавања, знање је код експертних система декларативно, а обрада симболичка. Поред тога, и симулациони модели и системи засновани на знању представљају знање и експертизу о посматраном систему и његовом понашању у модуларној форми. На пример, у симулационим моделима дискретних догађаја, ово се знање у процедуралном облику имплементира преко концепата као што су догађај, активност или процес. У експертним системима, за представљање знања о систему користе се правила, оквири или семантичке мреже, дакле декларативне форме. За разлику од већине других програма, где је управљање током извршавања програма одређено нумерички израчунатим величинама, код симулације, као и код ЕС, контрола тока програма зависи од логичких одлука. Но, и ту постоје одређене разлике. На пример, у симулацији дискретних догађаја за управљање током извршавања може се користити стратегија наредног догађаја, за разлику од механизма закључивања, као што су *foreward* или *backward chaining* код ЕС-а. Следећа разлика између симулације и ЕС тиче се представљања и обраде непотпуног, неизвесног или хеуристичког знања. У стохастичкој симулацији, неизвесно знање се изражава преко расподела вероватноће, док се код ЕС оно представља вероватноћама, *fuzzy*-логиком или *fuzzy*-скуповима.

Ово поређење указује на то да симулација и експертни системи нуде различите методе, технике и алате за прикупљање и представљање знања, као и различите технике процесирања тог знања и закључивања. Приближна подручја примене симулационих модела и експертних система, с једне стране, и различите технике реализације, с друге стране, нуде бројне могућности за корисно и ефикасно повезивање и интеграцију система заснованих на знању са симулационим моделима и обратно.

## 8.14. Симулациони процес и експертни системи

У наставку је укратко дат преглед могућих примена експертних система у појединим фазама симулационог процеса, са нагласком на практичним резултатима уколико такви постоје (*Widman i Loparo, 1989*).

### 8.14.1 Изградња модела

Постоје три основна начина како ЕС могу да побољшају перформансе симулације у овој фази. То су: експертска помоћ новим корисницима, побољшање форми представљања реалних система и смањење обима израчунавања у симулацији.

Експертни систем може да помогне неискусним корисницима код спецификације одређеног модела и код избора најбољег модела за одређени циљ симулације. Постоји више система који помажу новим корисницима приликом спецификације модела. Они углавном користе графички интерфејс преко кога корисници размештају "сличице" које представљају компоненте модела. Систем користи технологију ЕС како би проверио да ли су све неопходне компоненте присутне и да ли су добро повезане. Модел се потом преводи у код неког стандардног симулационог језика. Примери таквих система су *SES (Adelsberger idr, 1986)* и *ECO (Muetzelfeldt idr., 1985)*.

Када је на располагању више унапред дефинисаних модела, корисник се сусреће са проблемом избора оног модела који је најпогоднији за проучавање система који се посматра и који највише одговара постављеним експерименталним условима. Ово је фаза, такође, погодна за примену ЕС (*Kerchoffs i Vansteenkiste, 1986*).

Побољшање форми представљања реалног система може бити од веће важности него експертска помоћ новим корисницима, али може захтевати знатно веће време за реализацију. Проблем који се овде јавља односи се на чињеницу да постојећи симулациони модели не могу да представе на адекватан начин (1) сложено одлучивање какво је на пример неопходно код моделирања организационих циљева или војних стратегија и тактика, или (2) неизвесне или некомплетне моделе или податке, који су иначе веома чести у доменима инжењерства и управљања. Постојећи симулациони програми користе у те сврхе логику стабла одлучивања или табела одлучивања, што повлачи одређене недостатке: неизвесност се представља расподелом вероватноће чак иако формалне вероватноће не могу пружити најбољи опис у свим случајевима.

Уграђивањем посебног модула у виду ЕС у симулациони модел, могуће је прецизније моделирати ове сложене "објекте", чиме се повећава поузданост симулационог модела. Интересантан пример

оваквог концепта је коришћење ЕС у моделу за представљање понашања заступника у преговорима између радника и руководства (*Ahamed i Roman, 1985*).

Када је циљ серије симулационих експеримената верификација скупа предложених корака, најчешће није неопходно прецизно израчунавати све нумеричке вредности свих променљивих у моделу. Код таквих симулационих експеримената, екстензивна израчунавања се понекад могу заменити квалитативном симулацијом која ограничава вредности у одређени интервал. Квалитативна симулација, осим што смањује рачунарске трошкове, обезбеђује и експлицитни траг о зависностима између променљивих. Пример коришћења ЕС у овом домену је систем за квалитативну симулацију производње полупроводника (*Mohammed i Simmons, 1986*).

#### 8.14.2 Процена параметара

Помоћ статистичара је драгоцен новим корисницима који се сусрећу са пословима процене параметара модела и тестирањем улазних података. Пример ЕС који се користи за те сврхе је *EDA*, статистички програм за савете који комбинује симболичко и нумеричко израчунавање и саветује нове кориснике великих статистичких пакета, како и које статистичке процедуре да користе за проблеме које разматрају.

#### 8.14.3 Валидација и верификација модела

Валидација и верификација модела су експертски послови за које се могу формулисати прагматична правила. Експертни системи, осим што могу да пружају помоћ код отклањања грешака у моделима, могу да дају и савете, на пример, аутоматским тестирањем модела са подацима који су посебно припремљени да "уздрмају" модел у кључним тачкама. Такође, ЕС могу да анализирају погрешне резултате тест симулација и да укажу на најчешће изворе грешака. Поред тога, ЕС могу да пруже савете о правилном избору улазних података који се користе за валидацију и верификацију модела.

#### 8.14.4 Планирање симулационих експеримената

Постоје три основна начина како ЕС могу да помогну кориснику код планирања симулационих експеримената: савети засновани на општим статистичким и симулационим принципима, савети засновани на знању из посматраног домена и савети засновани на претходном искуству са одабраним моделом.

Планирање симулационих експеримената зависе од њихове сврхе ради које се исти обављају. То може бити: евалуација перформанси, поређење, предвиђање, анализа осетљивости, оптимизација, успостављање функционалних веза, проучавање понашања система, и сл. Како је планирање симулационих експеримената релативно лак посао за експерте, а релативно тежак посао за нове кориснике и како исти има велику економску вредност, он представља идеалан посао за подршку ЕС.

Спецификација симулационог модела најчешће захтева детаљно знање о ограничењима и релацијама које проистичу из одређеног домена или система у оквиру неког домена. Грешке које настају приликом изградње модела могу се избећи правовременим давањем одређених савета и критика. Један такав систем развијен је у области медицине (*Miller, 1986*).

Када ЕС буду у стању да уче на основу искуства, они ће моћи да саветују кориснике о томе када намеравана симулација неће дати очекиване резултате, шта треба изменити и које информације треба додати. Такви савети могу у значајној мери да уштеде време и трошкове, посебно тада када више појединаца или група користи исте моделе за експериментисање.

#### 8.14.5 Анализа резултата

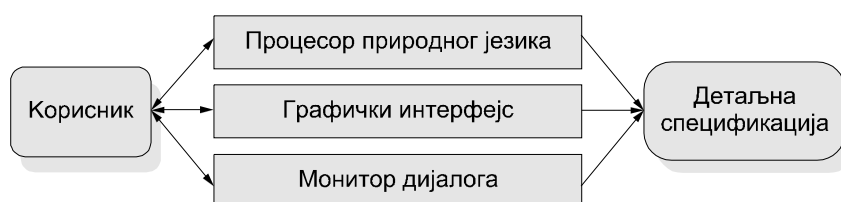
Један од кључних захтева код симулационих окружења заснованих на примени ЕС тиче се њихове могућности да помажу корисницима приликом измена симулационих модела са циљем да се постигну постављени циљеви симулације. На пример, уколико се користи модел производње како би се донела одлука о томе како смањити фреквенцију застоја који су проузроковани недостацима материјала на складишту, систем може да одреди редослед којим ће се параметри модела мењати са циљем да се што пре достигну постављени циљеви и да потом настави по том плану.

Од свих корака у процесу симулације, овај је са становишта потребног знања најинтензивнији пошто се од система захтева да схвати начин функционисања модела, али и да поседује знање о општим симулационим методама. Овако сложен посао реализован је на Carnegie Mellon Универзитету у виду познатог *KBS* система (Fox и др., 1989).

## 8.15 Спецификација симулационог модела преко ЕС

Основни циљ спецификације софтвера је да обезбеди опис шта рачунарски програм треба да ради, независно од тога да ли је тај програм нека апликација везана за рачуноводствени систем, програм за управљање производњом или симулациони модел неког реалног система који се може представити формализмом дискретних догађаја. Начини на које се до спецификације долази могу бити бројни и разноврсни.

Спецификација симулационог модела укључује различите активности где за аналитичара од велике помоћи могу бити неке од метода и техника ВИ, посебно експертни системи и разумевање природних језика (Radenkovic, Markovic i Vukmirovic, 1993). За разлику од традиционалног приступа који је приказан на слици 8.11, на слици 8.12 приказана су три могућа начина аутоматског прикупљања података ради спецификације модела (Murray i Sheppard, 1988). Сваки од приказаних приступа има за циљ да обезбеди детаљну спецификацију симулационог модела.



Слика 8.12 Аутоматизовани приступи прикупљања података

У првом случају, корисник задаје опис система на свом матерњем језику, нпр. енглеском, који се преко процесора природног језика (*natural language processor*) преводи у детаљну спецификацију модела. Интерфејс заснован на процесору природног језика има ту

предност што је релативно једноставан за корисника и што пружа велику флексибилност код спецификације. Међутим, он захтева знање из домена система који се моделира, а такође значајна пажња се мора посветити и лингвистичким аспектима препознавања језика. Ово подразумева уграђивање лексичког знања о речима и њиховим деловима и синтаксног знања о груписању речи у смислене фразе, као и разрешавање проблема везаних за могућност двојаког или вишеструког тумачења одређених речи, односно израза.

Други начин подразумева коришћење графичког интерфејса (*graphical interface*), који је посебно користан и једноставан за описивање структуре модела, односно компоненти модела и њихових међусобних веза. С друге стране, овај начин спецификације модела захтева од корисника да буде упознат са скупом расположивих "сличица", како би одабрао одговарајуће и преко њих представио различите аспекте система који се моделира.

Како су за изградњу модела дискретног система потребни и неки додатни детаљи као, на пример, расподеле случајних променљивих или дисциплине везане за редове чекања, то се најчешће у комбинацији са графичким интерфејсом користи процесор природног језика или монитор дијалога (*dialog monitor*), како би се комплетирао спецификација.

Када су прикупљене све потребне информације, односно извршена спецификација модела, могуће је аутоматски конструисати модел, било коришћењем генератора програма било посебног система за аутоматизовање програмирања.

У симулацији базираној на знању (или правилима), овако дефинисани приступ аутоматском програмирању обезбеђује да се аутоматизује не само процес спецификације модела, већ и процес изградње (кодирања) модела. Како постоје извесна преклапања у знању које је приликом прикупљања података неопходно за спецификацију модела и знању које се користи у процесу програмирања код традиционалног приступа са слике 8.11, ово се знање може чувати као део базе правила која би омогућила аутоматизацију и једне и друге активности. На тај начин, корисник који добро познаје систем који се симулира, али нема претходног искуства у имплементационом симулационом језику, може да креира извршне симулационе моделе.



У наставку текста разматрају се могућности спецификације симулационог модела дискретног система коришћењем експертног система, основни аспекти развоја таквог система, његова структура и методологија за пројектовање.

## 8.16. Развој експертног система за спецификацију симулационог модела

Код развоја експертног система за спецификацију симулационог модела дискретно-стохастичког система потребно је, пре свега, за конкретан симулациони модел утврдити догађаје и атрибуте догађаја којима се ови описују, а потом и активности које преводе систем из једног у друго стање.

Савремене методе развоја система најчешће инсистирају на успостављању базе за дефинисање система. Основна филозофија ових метода се базира на концепту одвојених (самосталних) функционалних захтева модела који репрезентују домен проблема. Модел се дефинише догађајима (акцијама), њиховим атрибутима и скупом процеса који описују редослед тих догађаја. Модел служи као база у којој су сажете све функционалне информације о стању околине. Ова метода не обезбеђује експлицитно представљање пословне политике, тако да се она уграђује у програмску логику.

### 8.16.1 Уграђивање правила

Уколико би се систем могао представити на исти начин како га види корисник и уколико би било могуће експлицитно утврдити његову репрезентацију у рачунару, тада би се свака спецификација система изазвана променама захтева могла лако идентификовати и успешно исправити. Нека истраживања су показала да правила играју главну улогу у описивању система, пошто дефинисање неког динамичког система обично укључује описивање трансакција, управљања, закључака и правила потребних за рад система.

Правило се обично састоји из два дела: премисе и акције. Премиса је спој исказа, а акција производи један или више закључака који се могу извести ако су премисе задовољене. Представљање знања у облику правила има своје предности које су засноване на следећим разлозима (*Poo and Layzdel, 1990*):

- ♦ правила обезбеђују природни механизам за описивање проблема и одлучивање,
- ♦ правило се може посматрати као модуларни сегмент знања унутар домена проблема,
- ♦ правила могу бити употребљена за дефинисање знања или информација о другим правилима, чиме се обезбеђује контрола над спецификацијом система.

Спецификација модела има три главне области у којима се правила могу применити. То су:

- ♦ област акција ентитета
- ♦ област атрибута ентитета
- ♦ област извођења веза.

### 8.16.2 Област акција ентитета

Ентитет модела симулира ентитет у реалности. Како се ентитет реалног система мења под утицајем неке акције, тако ентитет модела симулира те акције, односно долази до промене стања ентитета након сваког завршетка акције.

#### 8.16.2.1 Правила интегритета ентитета

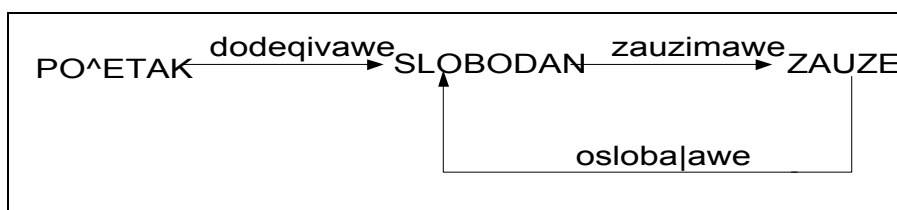
Свака акција укључује два стања: почетно стање и завршно (одредишно) стање. Почетно стање је стање у коме се ентитет тренутно налази, док је одредишно стање оно у које ће ентитет доћи када се посматрана акција заврши. Валидност промене стања дефинисана је структуром (историјом, животним циклусом) ентитета. Промена стања неког ентитета као резултат завршетка неке акције може се приказати у графичком облику на следећи начин:

ПОЧЕТНО СТАЊЕ —→ акција —→ ОДРЕДИШНО СТАЊЕ
--

Промена стања може се такође представити и на следећи начин:

ОД <почетно стање> НАКОН <акција> ДО <одредишно стање>

На пример, ако модел система електронске телефонске централе (ЕТС) има ентитет КАНАЛ, историја животног циклуса овог ентитета може се приказати на следећи начин:



Табела 8.3 приказује могућа прелазна стања ентитета КАНАЛ и ентитета ЛИНИЈА у моделу система ЕТС:

Табела 8.3 Прелазна стања ентитета канал и линија

	Почетно стање	Акција	Одредишно стање
Прелазна стања	почетак	додељивање	слободан
	слободан	заузимање	заузет
	заузет	ослобађање	слободан

Употреба израза заснованих на правилима у овој области, у вези је са дозвољеним акцијама над текућим стањима ентитета. Правила дефинишу скуп дозвољених акција за одређено стање ентитета. Овај тип правила познат је као "Правила интегритета ентитета" (*EIR - Entity Integrity Rule*) и она дефинишу могуће прелазе из једног стања ентитета у друго. Промена стања је валидна под условом да задовољава структуру ентитета.

Према томе, било који покушај промене стања који није садржан у скупу *EIR*, сматра се као нарушавање ограничења ентитета,

сигнализирајући или неважећу акцију или акцију изван контекста. *EIR* има бројне предности над конвенционалним представљањем система у много чему као што је, на пример, прављење експлицитних веза између стања и акција у дијаграму промене стања. Коришћењем правила, могуће је генерисати скуп промена стања за сваки ентитет, а саме промене стања се могу боље представити у облику графа него конвенционалном хијерархијском структуром.

#### 8.16.2.2 Пре-акциона условна правила

Постоје ситуације када морају да буду испуњени додатни услови пре него што може да започне извршење неке акције. Да би се једна акција извршила, претходно треба испитати неки услов (прост или сложен) и, уколико је он испуњен, тек тада се може прећи на извршење акције. Провера таквих услова једноставно се може представити у облику правила. Овај тип правила је познат као пре-акциона условна правила (*PACR - Pre-Action Conditional Rule*). *PACR* везује претходни услов за акцију и може се у општем облику приказати као:

ПРЕ <акција> УСЛОВ <услов>

На пример, пре заузимања КАНАЛА (акција), треба проверити да ли је испуњен услов да је број слободних КАНАЛА већи од 0.

ПРЕ заузни канал УСЛОВ број слободних канала > 0

#### 8.16.2.3 Пост-акциона окидачка правила

У неким системима, чест је случај да је потребно извршити одређене функције или послове након завршетка неких других акција. Овакав приступ се може експлицитно реализовати преко посебне врсте правила која називамо пост-акционим окидачким правилима (*PATR - Post-Action Trigger Rule*). Ова правила изражавају везу између акције и функције (у конвенционалном смислу) када су одређени услови испуњени. Овај тип правила се

може представити у општем облику као:

НАКОН <акција> AND <услов> CALL <функција(аргумент)>

На пример, када се изврши заузимање канала, а уколико је број слободних линија већи од 1 (услов) треба позвати (окинути) процедуру за генерисање разговора.

ПОСЛЕ заузми канал AND број слободних линија > 1  
CALL генериши-разговор(...)

Уведена синтакса правила није стриктна и може се реализовати на различите начине. У експертним системима код којих се користи форма продукционих правила за представљање знања у бази знања, горепоменута правила углавном имају облик типа *IF* <услов(и)> *THEN* <акција>.

### 8.16.3 Област атрибута ентитета

Атрибут, као и сваки податак, треба заштитити од неконзистентних вредности. Две врсте неконзистентних вредности често се везују за атрибуте: неконзистентан тип и опсег вредности. Неконзистентност типа се односи на додељивање недозвољеног типа вредности атрибуту. На пример, атрибут <заузетост> ентитета <канал> дефинисан логичким типом <*True, False*> не може прихватити алфабетску или нумеричку вредност. Придруживање опсега вредности подразумева додељивање вредности из специфицираног опсега одређеном атрибуту. Када се додели вредност која је изван валидног опсега, она не може бити прихваћена и мора постојати механизам који ће такву вредност одбацити.

Додавање конзистентних ограничења, као што су тип и опсег вредности, и представљање тих ограничења у облику правила, омогућава чвршћу и лакшу контролу вредности атрибута. Обезбеђивање типа и опсега вредности атрибута није нова идеја; већина конвенционалних програмских језика има те могућности. Нажалост, представљање ових ограничења је често уграђено

унутар програмских рутина, чиме се отежава одржавање система. Предност представљања типа и опсега вредности атрибута помоћу правила лежи у томе што се опис атрибута диже на ниво погодан за разумевање од стране корисника, а самим тим и за његову валидацију. Могу се посматрати три типа правила која су погодна за дефинисање ограничења атрибута, која су објашњена у наставку текста.

#### 8.16.3.1 Правило о типу атрибута

Правило о типу атрибута (*Attribute Definition Rule - ADnR*) дефинише тип атрибута. На пример, атрибут дефинисан типом целобројан, може узети само целобројну вредност и било која друга вредност је недозвољена. Дефиниција типа атрибута обезбеђује да се атрибуту могу доделити само вредности прописаног типа.

#### 8.16.3.2 Правило о опсегу вредности атрибута

Овим правилом (*Attribute Constraint Rule - ACR*) се дефинише опсег дозвољених вредности атрибута. Опсег може бити дефинисан константом (константама) или вредношћу (вредностима) неког другог атрибута. На пример:

број канала > 2 и број канала < 50 или  
број линија > број канала

#### 8.16.3.3 Правило о зависности атрибута

Слично је са *ACR* по томе што ограничава опсег дозвољених вредности атрибута, али се опсег вредности у *ADyR* (*Attribute Dependency Rule*) одређује на основу услова који га ставља у зависност од вредности другог атрибута. На пример:

АКО веза = 1 ТАДА трајање разговора > 0  
АКО веза = 2 ТАДА трајање разговора = 0

Другим речима, атрибут *трајање разговора* има вредност која зависи од вредности атрибута веза.

#### 8.16.4 Област извођења веза

У овој области се дефинишу правила условних релација и врше разна израчунавања у процесу симулације. Могу се посматрати два типа правила, чији опис следи.

##### 8.16.4.1 Правило закључивања

Ако се посматра условна релација  $\langle y1 \rangle$  која везује два атрибута  $\langle a1, a2 \rangle$  за ентитет  $\langle E1 \rangle$  то се може представити на следећи начин:

$\langle y1 \rangle \text{ АКО } \langle (a1, a2) = 0 \rangle$

Другим речима,  $\langle y1 \rangle$  удружује два атрибута,  $a1$  и  $a2$  унутар једног правила и каже се да је то условно правило (*Inference Rule - IR*) засновано на два атрибута. Дефиниција овог правила се обично представља на следећи начин:

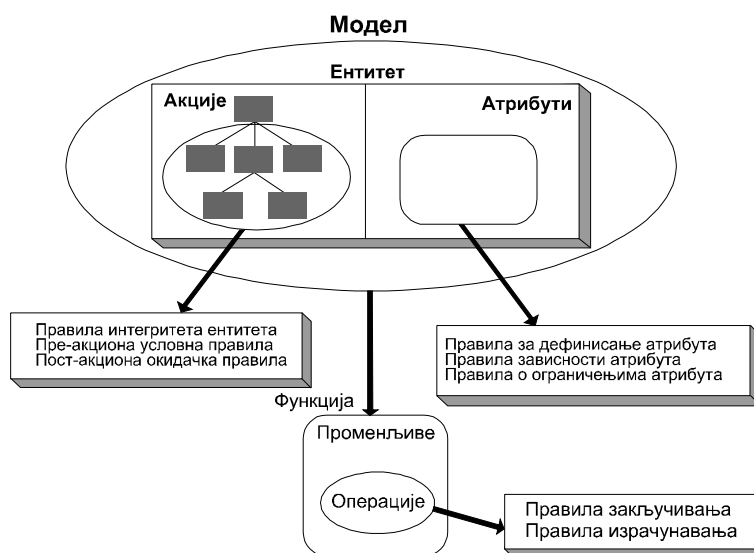
IF  $\langle \text{услов} \rangle$  THEN  $\langle \text{процес} \rangle$

Бројна понављања ове дефиниције могу бити се наћи у различитим деловима система. Ако је дефиниција правила апстрактно представљена у спецификацији, тада су измене правила усмерене само на дефиницију правила, а не и на место појављивања правила. У многим системима у којима се правила јављају у експлицитном облику, нпр. IF  $\langle a=b \rangle$  THEN ..., захтев за измену услова правила имаће за последицу тражење свих места где се то правило јавља у систему; тек након тога измена ће бити могућа. Међутим, ако се правило представи у апстрактном облику IF  $\langle c \rangle$  THEN ..., где је дефиниција правила  $\langle c := "a=b" \rangle$ , тада ће за измену бити довољно променити само дефиницију, што је свакако лакши начин. Ова форма представљања нарочито је погодна у симулацији система.

#### 8.16.4.2 Правило израчунавања

Овај тип правила (*Computation Rule*) се може комбиновати са другим, напред наведеним типовима. Тако је назван јер се атрибути повезују преко аритметичких формула. Један атрибут који се налази на више места у моделу, мења своју вредност изменом функције која му додељује нову вредност. На тај начин, није потребно тражење свих места појављивања одређеног атрибута већ је довољно извршити модификацију у функцији тог атрибута.

На слици 8.13 приказана су места дефинисања правила у спецификацији модела.



Слика 8.13 Места дефинисања правила у спецификацији модела

### 8.17 Стратегија трофазне симулације као продукциони систем

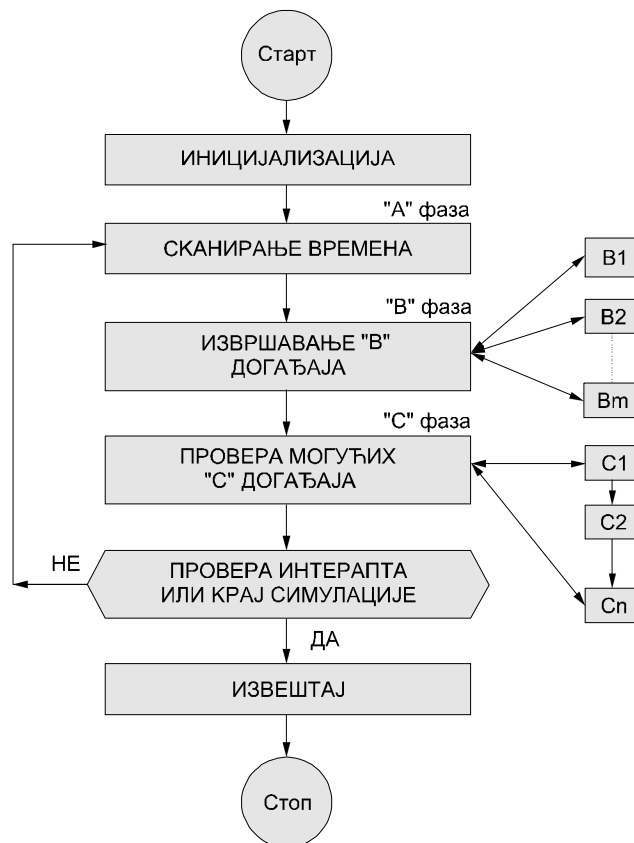
Разматрајући питања симулације дискретних система у ранијим излагањима, напоменули смо да је интеракција процеса специфична методологија симулације, настала комбинацијом распоређивања догађаја и сканирања активности. Показаћемо да је могуће једну такву структуру реализовати у виду продукционог



експертног система на основу постојећих сличности (*Doukidis, 1987*).

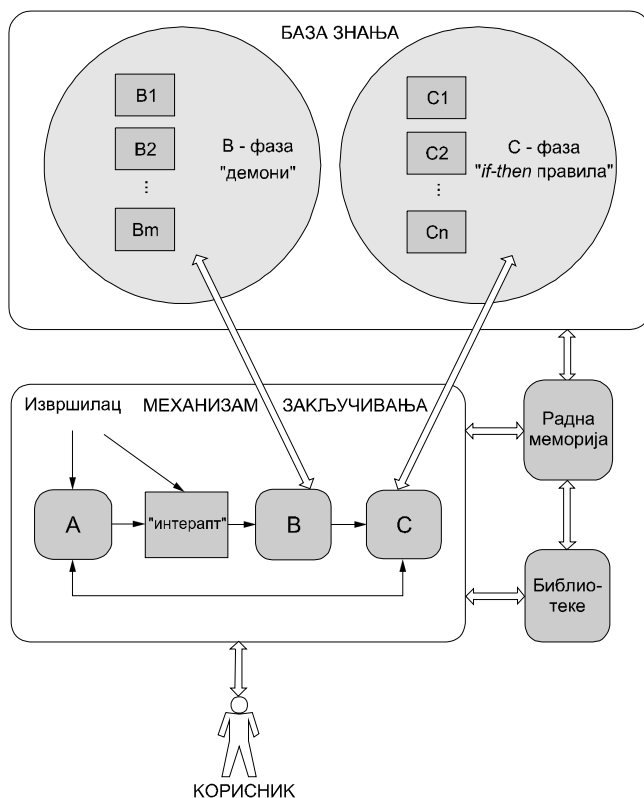
Претпоставимо да се методологија интеракције процеса може упростити тако што се извршавање условних догађаја и догађаја који се безусловно извршавају у оквиру фазе сканирања листе текућих догађаја, раздваја у две секвенцијалне фазе. Под тим претпоставкама, интеракција процеса може се упрошћено представити у форми такозване трофазне симулације са следећим фазама (слика 8.14):

1. фаза А - ажурирање времена,
2. фаза В - извршавање безусловних догађаја,
3. фаза С - извршавање условних догађаја.



Слика 8.14 Стратегија трофазне симулације

Приказана структура може се представити у виду специфичног продукционог система који је шематски приказан на слици 8.15.



Слика 8.15 Интеракција процеса као продукциони систем

База знања садржи скуп правила која описују опште знање о посматраном проблему. Правила се састоје из два дела. Први део чине безусловни догађаји (В догађаји) који представљају завршетке активности, односно наступају тада када симулациони сат означи завршетак неке активности. У терминологији ВИ, оваква правила (догађаји) означавају се као "демони", пошто немају условни део, већ само део који обухвата акцију или серију акција. Правило ове врсте бива "испаљено" када наступи одговарајући В догађај. Други део чине С догађаји или условни догађаји који представљају типична "Ако-Тада" (*If-Then*) продукциона правила. Код ових правила, акција која следи *Then*-део правила извршава се увек када је задовољен услов из *If*-дела правила, односно у терминологији симулације, када се испуни потребан услов за почетак неке

активности, тада се предузима једна или више одговарајућих акција.

Радна меморија продукционог система са слике 8.15 чува текуће знање о систему. Структурама података из радне меморије управља механизам закључивања, преко позива библиотеке симулационих модула. Библиотека симулационих модула креира структуре података и дозвољава њихово брисање и/или ажурирање. Основни део радне меморије сачињавају циљеви и чињенице, као и у сваком другом моделу продукционог система. Чињенице, које описују симулациони модел у сваком тренутку времена, представљају стања ентитета и редова. Њих користе правила у процесу закључивања. Циљеви су елементи механизма који управља временом и који представља план наступања В догађаја. Као и код других продукционих система, циљеви усмеравају и управљају процесирање система указујући на стања која треба достићи. У трофазној стратегији симулације, особине симулационих циљева су време (када нешто треба да се догоди) и приоритет (релативни приоритет активности).

Механизам закључивања са слике 8.15 контролише време, терминирање, позиве В догађаја и врши тестирање свих продукционих правила везаних за С догађаје. По начину закључивања, симулациони систем није типичан продукциони систем. Наиме, док продукциони систем прво пронађе сва правила која могу бити задовољена текућим садржајем података у радној меморији, а потом употреби стратегију селекције да одреди које ће од њих да примени, у симулацији се једноставан скуп "метаправила" извршава по реду (*Paul, 1989*). Метаправила А фазе ажурирају време симулације увек када се одабере нови догађај за извршење, односно постављају симулациони сат на време када треба да наступи тај догађај. Метаправила прекида (*interrupt*), уобичајена за све продукционе системе, проверавају да ли је испуњен услов за прекид симулације. Метаправила В фазе извршавају све В догађаје идентификоване метаправилима А фазе. На крају, метаправила С фазе тестирају продукциона правила за сваки С догађај по реду и "испаљују" она правила код којих се услови део правила слаже са текућим подацима у радној меморији. Четири наведена метаправила примењују се уз стално понављање, све док метаправило интерпта не укаже на прекид.

Механизам закључивања користи стратегију уланчавања унапред приликом испитивања и извршавања продукционих правила С догађаја. Ово је логичан избор за стратегију управљања процесом

закључивања када постоји више једнако прихватљивих циљних стања и само једно почетно стање. Такође, не постоји предефинисано завршно стање према коме би стратегија уланчавања уназад могла да тежи, пошто је такво стање заправо оно што сам модел (симулација) треба да открије.

Пошто управљање временом није "јача" страна програма вештачке интелигенције, а с друге стране је одлично разрађено у различитим симулационим техникама, наведени пример се може сматрати потенцијалним доприносом симулације истраживањима у области вештачке интелигенције.

## 8.18 Језици вештачке интелигенције у симулацији

Од језика вештачке интелигенције, за симулацију су посебно значајна два: PROLOG и LISP. Како разматрање појединих карактеристика наведених језика превазилази оквире ове књиге, задржаћемо се само на кратком прегледу значајнијих радова који је сачинио *Ray J. Paul*, а који се односе на примену језика ВИ у симулацији (*Paul, 1989*).

Ето разматра систем писан у TS-PROLOG – у који комбинује симулацију и решавање проблема. У том систему, модел је представљен као хијерархија независних компоненти за које је могуће дефинисати циљеве. Систем обезбеђује веома флексибилан механизам комуникације, омогућава да понашање било које компоненте буде континуално или дискретно. Такође, обезбеђена је могућност "хода унатраг" ради испитивања већег броја алтернатива у случају да се дође на "слепи" колосек при претраживању циљева.

*Cleary* је приказао прелиминарну верзију конкурентног логичког програмског језика за симулацију система са дискретним догађајима, писану у језику CONCURRENT PROLOG.

*Flitman* и *Hurion*, као и *O'Keefe* и *Roach* развијају нешто традиционалнији симулациони систем на PROLOG-у који користи методологију трофазне симулације као основу за продукциони систем, сличан оном који је приказан на слици 8.15.

*O'Keef* и *Roach* су израдили PROLOG систем који представља имплементацију GPSS језика. У њега је уграђена и колекција

PASCAL рутина за генерисање случајних променљивих и за статистику.

*Birtwhistle* и *Cendal* расправљају предности LISP-а као имплементационог језика за израду окружења за симулацију система са дискретним догађајима.

Треба напоменути и значајан допринос бројних аутора (*Bauman i Turano; Middleton i Zanconato; Klahr; Wilkinson; Ruiz-Mier i Talavage; Barton; McRoberts i dr.*) који су развили своје ВИ језике специјалне намене, применљиве у симулацији. Њихов број је свакако већи; у овом кратком прегледу споменута су само нека имена.

Између осталог, издвојили бисмо и рад *Paul-a i Doukidis-a* везан за систем разумевања људског говора, који се користи као помоћ при анализи и доношењу одлука у формулисању симулационих модела под називом NULS.

## **ЈЕЗИЦИ ЗА СИМУЛАЦИЈУ КОНТИНУАЛНИХ СИСТЕМА**

Први корак у изучавању система јесте стварање модела. Модел је опис реалног система који садржи само оне компоненте које су релевантне за његово изучавање. Симулацију, ипак, изводимо на моделима система, а не на самим системима. Модел је дакле основна претпоставка симулације. Дobar логички модел представља неопходан, али не и довољан услов за извршавање симулације. Први корак у симулацији је формирање инструкција којима се описује модел система и њихово уношење у рачунар. Те инструкције морају да буду разумљиве од стране рачунара. Језици који омогућавају опис модела на начин разумљив од стране рачунара су уопштено названи програмски језици.

Писање симулационих програма, у суштини, није тежак посао, али је тачност у овој фази симулације барем од истог значаја као и тачност при изради логичког модела, пошто директно утиче на квалитет добијених резултата.

За израду симулационих програма, аналитичару симулације је на располагању:

1. неки од бројних симулационих језика специјалне намене,
2. неки од општих језика вишег нивоа, или, евентуално,
3. неки генерички модел вођен подацима.

Постоји већи број специјалних симулационих језика, прилагођених различитим рачунарима и оперативним системима. Треба нагласити да је већина њих намењена за програмирање једне класе модела. Обично се врши подела на језике за симулацију континуалних, дискретних и хибридних модела.

## 9.1 Језици за симулацију континуалних система

У новије време симулација континуалних система врши се на дигиталним рачунарима захваљујући постојању специјалних језика и пакета за њихову симулацију. Ти језици се састоје од симулационог језика, процесора и скупа блокова (функција).

Модели са континуалним променама стања су генерално сачињени од низа диференцијалних и диференцијалних једначина које се обично решавају нумерички. Назив континуална симулација се користи зато што је независно променљива у једначинама које би биле решаване, обично време: желимо да израчунамо стање система у неком тренутку у будућности полазећи од познате почетне конфигурације. Ово се може, такође, назвати временски континуална симулација. Ипак, језици континуалне симулације се користе за решавање детерминистичких диференцијалних једначина. Решавање стохастичких диференцијалних једначина је далеко теже.

Планирање континуалне симулације је прилично различито од планирања симулације дискретних догађаја. Генерално, диференцијалне једначине у моделу могу да садрже случајне променљиве за представљање шума, на пример, или друге случајне поремећаје. Ове случајне променљиве су генерисане коришћењем истих техника као код симулације дискретних догађаја. Код континуалне симулације аналитичар треба да изабере одговарајући алгоритам и дужину корака симулације. Уколико је модел део реалног система тада је од важности и брзина израчунавања. Генерално, континуална симулација обухвата и проблеме нумеричке анализе.

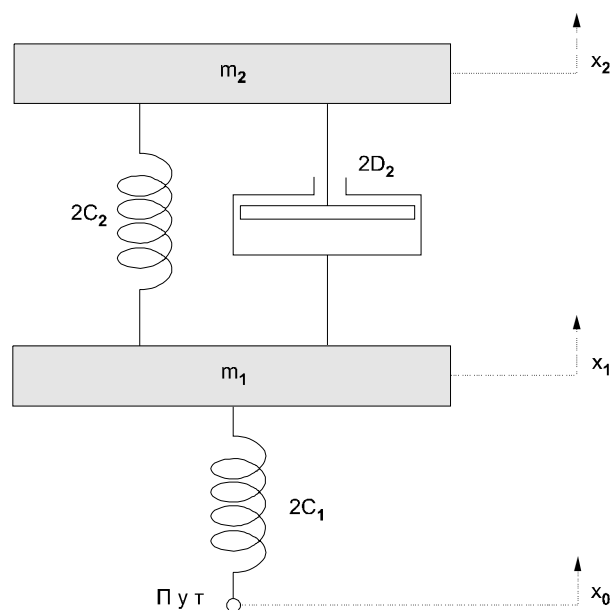
Постоје различите варијанте симулационих језика за континуалну симулацију базиране на следећим приступима:

- ♦ Дефинисање система врши се помоћу подпрограма писаних у стандардним програмским језицима као што су FORTRAN или C. Тим подпрограмима описује се десна страна једначина модела у простору стања којима је систем дефинисан.
- ♦ Уношење једначина модела у простору стања врши се помоћу специјално пројектованог командног језика.
- ♦ Визуелно описивање модела система помоћу блок дијаграма.

За програмирање континуалних модела најчешће се користе CSMP (IBM), DINAMO (M.I.T.), ACSL i CSSL. У новије време појавио се програмски језик вишег нивоа намењен симулацији динамичких континуалних и дискретних система, TUTSIM. Такође, у оквиру Matlab-a, као посебан Toolboks, постоји програмски пакет SIMULINK.

Први приступ користи се у стандардним језицима за симулацију као што су, на пример, TUTSIM, CC, ACSL, CSSL. Други приступ се среће код SIMON-a. Трећи приступ је све распрострањенији и среће се код CSMP-a, MATRIX-a, VISSIM-a, SIMULINK-a итд.

У наставку текста даћемо кратак опис неких симулационих језика и пакета као што су TUTSIM и SIMULINK, а детаљније ћемо приказати CSMP. За илустрацију, анализирајмо модел система "вешања" на аутомобилу који се може поједностављено приказати као на слици 9.1 (W. K. Giloi, 1975).



Слика 9.1 Упрошћена шема "вешања" на аутомобилу

Уведимо следеће ознаке:

$m_1$  - маса осовине и два точка

$m_2$  - маса шасије аутомобила



- $C_1$  - коефицијент опруге која представља гуму  
 $C_2$  - коефицијент опруге која представља гигањ  
 $D_2$  - коефицијент трења код амортизера  
 $x_0$  - дејство (функција) пута  
 $x_1$  - померај осовине  
 $x_2$  - померај шасије

Пошто на свакој осовини постоје два гигања, две гуме и два амортизера, параметре  $C_1$ ,  $C_2$  и  $D_2$  дуплирамо.

Да бисмо извршили симулацију овог модела, неходно је прво дефинисати математички модел система. У складу са d'Ambert-овим законом, разматрани систем се може описати следећим диференцијалним једначинама другог реда:

$$m_1 x_1'' + 2D_2(x_1' - x_2') + 2C_2(x_1 - x_2) + 2C_1(x_1 - x_0) = 0$$

$$m_2 x_2'' + 2D_2(x_2' - x_1') + 2C_2(x_2 - x_1) = 0$$

Дате једначине треба решити по изводу највишег реда, након чега се добија математички модел система у облику који је погодан за формирање симулационог блок-дијаграма.

$$x_1'' = -\frac{2D_2}{m_1}(x_1' - x_2') - \frac{2C_2}{m_1}(x_1 - x_2) - \frac{2C_1}{m_1}(x_1 - x_0)$$

$$x_2'' = -\frac{2D_2}{m_2}(x_2' - x_1') - \frac{2C_2}{m_2}(x_2 - x_1)$$

Уведимо следеће смене:

$$\frac{2D_2}{m_1} = k_1; \quad \frac{2C_2}{m_1} = k_2; \quad \frac{2C_1}{m_1} = k_3; \quad \frac{2D_2}{m_2} = k_4; \quad \frac{2C_2}{m_2} = k_5$$

Крајњи облик математичког модела посматраног система, након уведених смена, постаје:

$$x_1'' = -k_1(x_1' - x_2') - k_2(x_1 - x_2) - k_3(x_1 - x_0)$$

$$x_2'' = -k_4(x_2' - x_1') - k_5(x_2 - x_1)$$

Претпоставимо да уведене константе имају следеће реалне вредности:

$$k_1 = 26.7 \text{ s}^{-1}; k_2 = 1950 \text{ s}^{-2}; k_3 = 9350 \text{ s}^{-2}; \\ k_4 = 2.19 \text{ s}^{-1}; k_5 = 160 \text{ s}^{-2},$$

које ће се користити при реализацији датог модела у симулационим језицима и пакетима чији опис следи.

## 9.2 TUTSIM

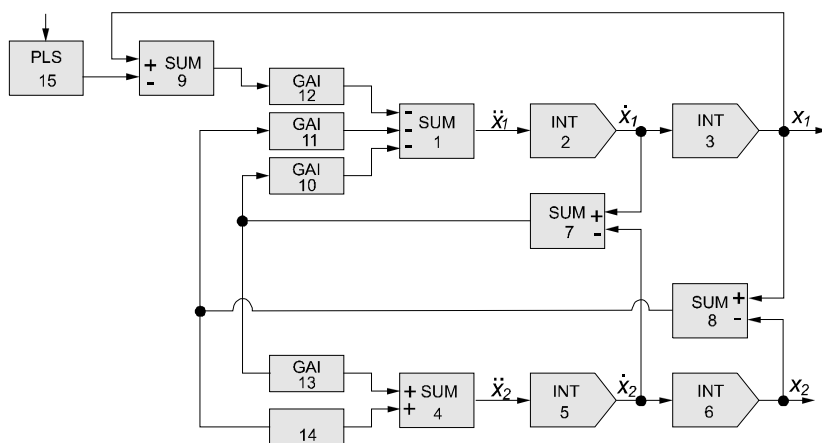
TUTSIM је програмски језик високог нивоа чија је намена симулација динамичких континуалних и дискретних система, и то како линеарних тако и нелинеарних. Може да се користи и за симулацију логичких кола. TUTSIM, такође, има и могућност адаптације на програмске језике вишег нивоа као што је FORTRAN. То омогућава обављање евентуалних додатних захтева (*TUTSIM - User's Manual*, 1989; Д. Антић, Г. Голо, 1996;)

Понашање реалних система се обично описује линеарним или нелинеарним диференцијалним једначинама. Да би се могла извршити симулација помоћу TUTSIM-а, неопходно је познавање модела система. Блокови које користи TUTSIM обављају одређене математичке операције, те због тога корисник не мора да разматра алгоритам за решавање постављеног проблема, већ на основу једначина система саставља симулациону шему реализовану помоћу TUTSIM блокова на основу које се врши симулација. Резултати симулације су обично временске функције и могу бити приказани графички на монитору рачунара или у виду нумеричких података.

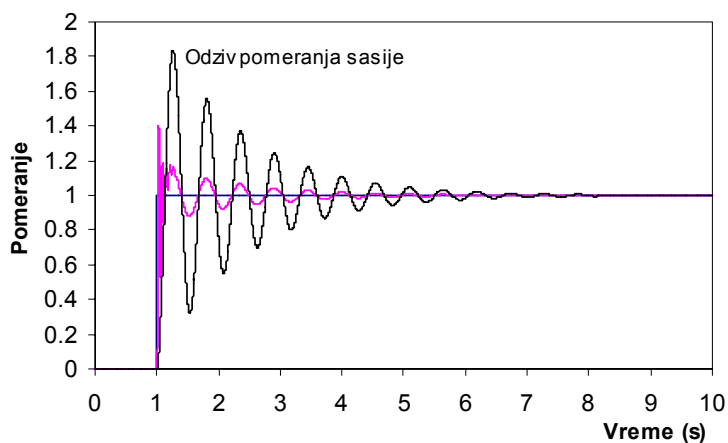
TUTSIM поседује нелинеарне и специјалне математичке блокове који омогућавају решавање низа проблема симулације. Како је овај програмски пакет написан као асемблерски језик, то омогућава директно извршавање по уношењу у рачунар.

### 9.2.1 Решење система “вешања” аутомобила у TUTSIM-у

Следи решење примера система “вешања” на аутомобилу у симулационом језику TUTSIM. Најпре је дата симулациона блок шема на којој су дати блокови са функцијама које извршавају, као и бројеви блокова, на основу којих је и написан програм (слика 9.2). Резултати симулације, у овом случају одзиви померања осовине и шасије, приказани су на слици 9.3. Код програма је приказан на слици 9.4.



Слика 9.2 Блок шема посматраног модела у TUTSIM –у



Слика 9.3 Одзиви померања осовине и шасије

```

Model File: vvoz.sim
Date:                2 / 23 / 1999
Time:                16. 24
Timing:              0.0010000 ,DELTA ;10.0000, RANGE
Plot Blocks and Scales:
Format:
  Block No, Plot-MINimum, Plot-MAXimum; Comment
Horz:   0 ,    0.0000 ,    10.0000 ;Time
Y1:    15 ,    0.0000 ,    2.0000 ;Dejstvo puta
Y2:     3 ,    0.0000 ,    2.0000 ;Polozaj osovine
Y3:     6 ,    0.0000 ,    2.0000 ;Polozaj sasiје
Y4:     ,    ,    ,    ;

MODEL:
      1 SUM -10 -11 -12 ;Ubrzanje osovine
0.0000      2 INT  1      ;Brzina osovine
0.0000      3 INT      2      ;Polozaj osovine
      4 SUM 13 14      ;Ubrzanje sasiје
0.0000      5 INT  4      ;Brzina sasiје
0.0000      6 INT  5      ;Polozaj sasiје
      7 SUM  2 -5      ;Razlika brzina
      8 SUM  3 -6      ;Razlika polozaја
      9 SUM  3 -15     ;Razlika polozaја
26.7000     10 GAI  7      ;Konstanta k1
1.950E+03    11 GAI  8      ;Konstanta k2
9.350E+03    12 GAI  9      ;Konstanta k3
2.1900       13 GAI  7      ;Konstanta k4
160.0000     14 GAI  8      ;Konstanta k5
1.0000       15 PLS      ;Dejstvo puta
100.0000
1.0000
    
```

Слика 9.4 Симулациони модел у TUTSIM-и

## 9.3 SIMULINK

За описивање модела SIMULINK нуди три могућности, које се могу комбиновати (Д. Антић, Г. Голо, 1996):

- ♦ опис модела коришћењем виших програмских језика (Fortran, C)
- ♦ опис модела помоћу MATLAB функција и
- ♦ графичко претстављање модела помоћу блокова.

Трећа могућност (коришћење блокова) је најпопуларнији, најпогоднији и најбржи начин за едитовање модела до постизања жељеног резултата. Блокови су разврстани у библиотеке које се могу допунити блоковима које формира корисник, а такође се могу формирати и нове библиотеке.

Углавном, библиотеке садрже све блокове који су потребни за симулацију различитих система. Постоји више врста генератора сигнала чији се параметри могу мењати, а за посматрање резултата симулације на располагању је више блокова. Такође, за графичко приказивање могу се користити и 2D и 3D функције MATLAB-а.

Код сложенијих и гломазних система постоји могућност обједињавања групе блокова који се у шеми представљају као један блок чиме се добија на прегледности модела. Такође, код сложених модела где симулација може да траје веома дуго, постоји могућност одвијања симулације у позадини.

За добро извођење симулације, најважнији фактор, ипак, остаје човек, јер резултати симулације су у функцији квалитетног моделирања система.

За анализу модела могу се користити три начина, који нису строго раздвојени, већ се могу користити комбиновано зависно од нивоа, односно степена развоја модела.

Први начин је интерактивни рад применом команди из менија SIMULINK-а. Овакав начин је једноставан за коришћење. Обично се користи у фази формирања модела система и фази исправљања грешака.

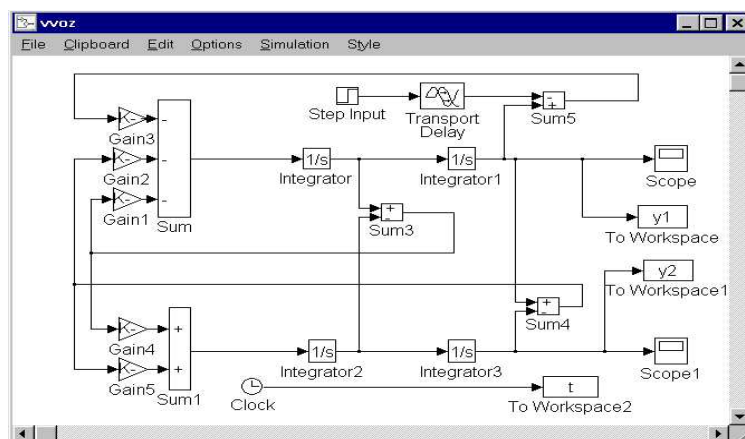
Други начин је коришћење уграђених функција за симулацију и анализу командне линије. Овај начин обезбеђује већу флексибилност у раду. Резултат симулације можемо пребацити у радно окружење MATLAB-а, а затим користити уграђене алате за анализу и визуелизацију коју поседује MATLAB.

Трећи начин је сложенији, али и најефикаснији. Сваки модел пројектован у SIMULINK-у је доступан MATLAB-у, као S функција. Свака S функција има име које је идентично имену модела у SIMULINK-у. Позивање S функција може се извршити на различите начине, зависно од тога која својства модела су од интереса. Сви

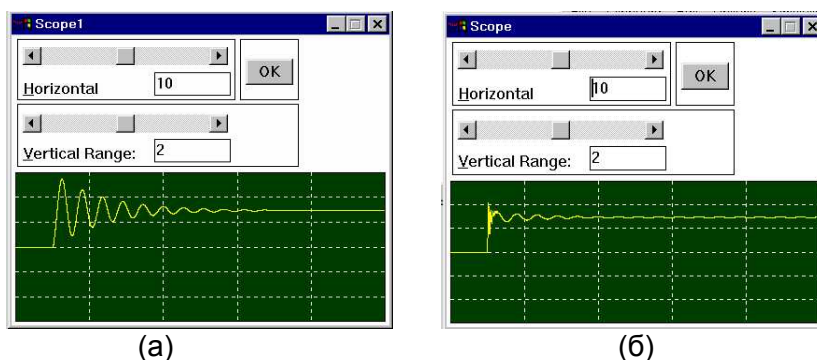
методи за анализу који су уграђени у SIMULINK-у приступају моделу преко S функција.

### 9.3.1 Решење система “вешања” аутомобила у SIMULINK-у

У наставку је дато решење примера система “вешања” на аутомобилу реализовано у програмском пакету SIMULINK. На слици 9.5. приказана је блок шема система “вешања” на аутомобилу. На сликама 9.6 дати су одзиви померања осовине (а) и шасије (б). Константе из примера  $k_1, k_2, k_3, k_4, k_5$  овде су дате у виду појачања:  $Gain1, Gain2, \dots, Gain5$ .



Слика 9.5 Симулациона шема система “вешања” на аутомобилу

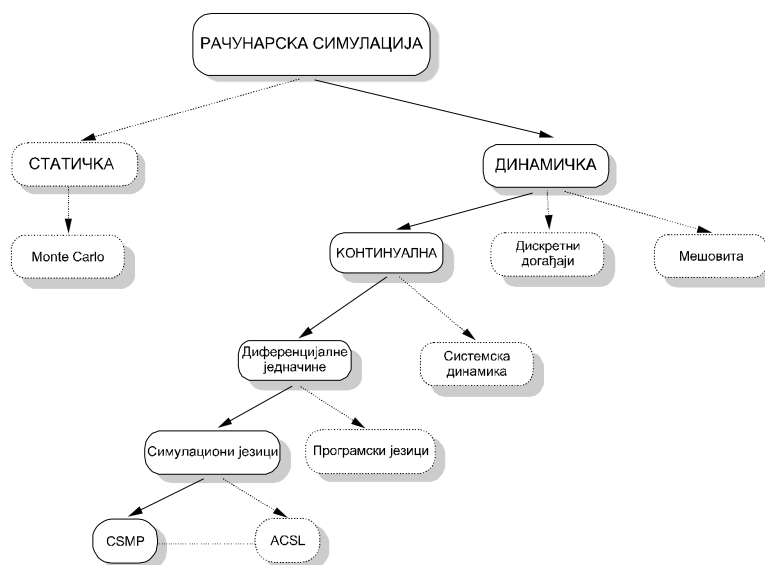


Слика 9.6 Одзив померања осовине (а) и шасије (б)

## 9.4 Симулациони језик CSMP

Симулациони језик CSMP (*Continuous System Modeling Program*) спада у класу специјализованих, блок оријентисаних језика за симулацију континуалних система који се могу описати формализмом обичних диференцијалних једначина (слика 9.7).

Развијен је у IBM-у раних 60-тих година и представљао је прави аналогни симулатор. Након прве верзије, развијене су и верзије CSMP II и CSMP III (IBM 1130). Ови програми се нису могли користити у интерактивном режиму, нити је било могуће истовремено коришћење са различитих терминала. Поменути недостаци елиминисани су у интерактивној верзији симулационог језика CSMP 11/70, развијеној на Факултету организационих наука у Београду, за рачунар PDP 11/70. У односу на изворни IBM програм, извршене су модификације ради бржег извођења имплицитних операција, уведен је командни језик за управљање програмом, а имплементирано је и неколико нових елемената (Бингулац и Стојић, 1971; Бингулац 1983).



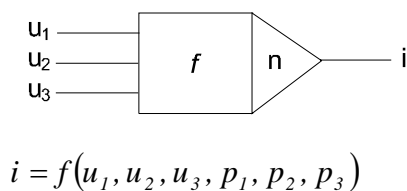
Слика 9.7. Место симулационог језика CSMP у класификацији симулационих језика

У наредној фази развоја језика на ФОН-у, прво је реализована верзија на програмском језику BASIC (CSMP/MICRO) која је била погодна за примену на већини кућних микро рачунара (Раденковић, 1984). Потом се дошло на идеју да се CSMP имплементира у програмском језику који је истовремено брз, једноставан, структуриран и расположив на већини PC рачунара који су преовладали на тржишту. Одабран је програмски језик PASCAL, а за имплементацију је коришћена верзија 5.0 фирме Borland за PC рачунаре. У овој верзији CSMP језика још више је унапређен кориснички интерфејс, као и целокупна ефикасност програма (Марковић, 1989).

Садашњи примат оперативног система Windows '95, '95/NT наметнуо је потребу за новом верзијом језика CSMP, а уједно омогућио развој квалитативно бољег корисничког интерфејса. Обзиром да је наследник програмског језика Pascal у Windows '95/NT окружењу Delphi, он је и одабран за имплементацију нове верзије језика CSMP која је развијена у Лабораторији за симулацију Факултета организационих наука под називом CSMP-W95/NT и о којој ће бити више речи у наставку текста (Милићевић, Марковић и Раденковић, 1998; 1999).

#### 9.4.1 Начин рада CSMP-а

Симулациони језик CSMP може да се сврста у класу програма за решавање система диференцијалних једначина код којег се систем једначина специфицира преко блок оријентисаног језика. Сваки од елемената који се специфицира у конфигурацији има свој идентификациони и графички симбол. У општем облику елемент се графички може приказати као на слици 9.8.



Слика 9.8 Графички приказ елемента у општем облику



Сваки од расположивих елемената специфицира релацију од највише три улазне променљиве  $u_1, u_2, u_3$  и три параметра  $p_1, p_2, p_3$ . Излаз  $i$  је скалар чија вредност зависи од конкретне релације  $f$  за дати елемент.

Рад програма се заснива на алгоритму за сортирање редних бројева блокова. Он даје редослед којим је потребно рачунати вредности излаза блокова тако да при одређивању вредности излаза било ког блока унутар конфигурације вредности излаза блокова на који су везани улази тог блока морају бити претходно одређене. По завршетку сваког интервала интеграције, вредности константи и меморијских елемената су познати. На основу њихових вредности, коришћењем резултата сортирања, могуће је одредити вредности излаза осталих блокова у моделу. Ако је конфигурација конзистентна, сви излази блокова у конфигурацији могу да се одреде на основу резултата сортирања.

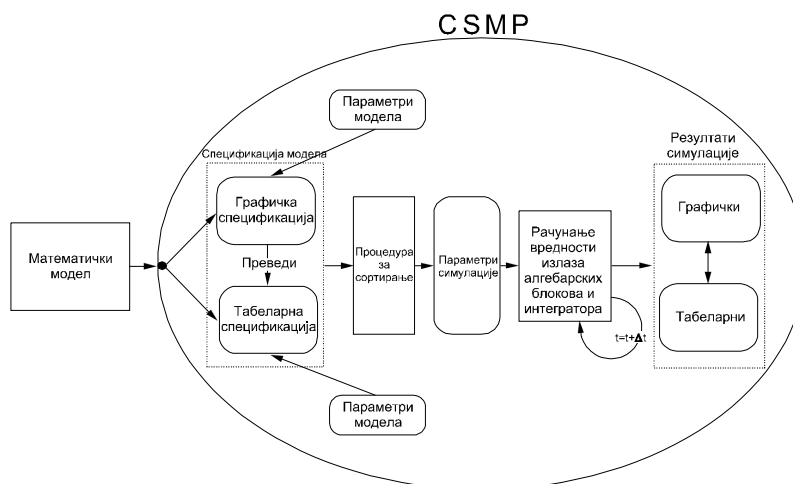
Уколико модул за сортирање примети нерегуларну спецификацију конфигурације, тада се штампа одговарајућа порука. Већина грешака које се јављају су резултат алгебарских петљи, што се догађа при имплицитним операцијама са немеморијским елементима. За симулацију која захтева имплицитне операције са немеморијским елементима, постоје посебни елементи (блокови) који ове недостатке отклањају.

Програм користи нумеричку интеграцију за одређивање вредности излаза интегратора. Систем диференцијалних једначина се обрађује као векторска једначина. При свакој половини интервала интеграције, програм помоћу одговарајућих подпрограма одређује вредности излаза модела. Добијене вредности се користе за одређивање вредности излаза интегратора у наредној половини интервала интеграције. Интеграција се врши методом *Runge-Kutta II* реда, која при погодном одабраном интервалу интеграције даје задовољавајуће резултате (Пејовић, 1983).

#### 9.4.2 Симулациони језик CSMP-W95/NT

Симулациони језик CSMP-W95/NT поседује *user – friendly* интерфејс, графичко и табеларно приказивање резултата и посебан модул за графичку спецификацију конфигурације модела и њено превођење у табеларни облик који користи симулатор (слика 9.9).

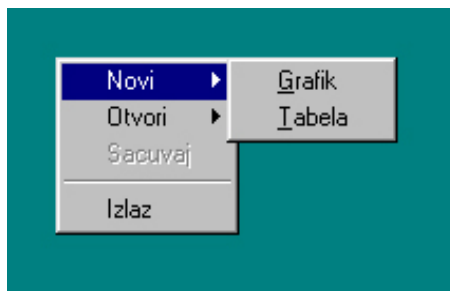
Могућности програма и начин његовог коришћења описани су и пропраћени на раније уведеном примеру (слика 9.1).



Слика 9.9 Структура језика CSMP-W95/NT

#### 9.4.2.1 Дефинисање конфигурације

Симулациони језик CSMP-W95/NT омогућава два начина уноса модела: табеларни и графички. Ове опције учљиве су на почетку стартовања апликације (слика 9.10). Могуће је учитати постојећи (меморисани) модел или креирати нови.



Слика 9.10 Опције за задавање конфигурације

Избором опције за унос новог модела преко табеле, корисник се одриче могућности да за дати модел добије графичку блок шему; у супротном, након уноса графичког модела, табеларни модел се мора генерисати.

Након избора опције *Novi* → *Tabela* отвара се дијалог који од корисника захтева да унесе број блокова модела (програм не дозвољава да се наруши интегритет уносом негативног броја или знака који није цифра тј. позитиван цео број). У нашем примеру тај број је 14.



Слика 9.11 Дијалог прозор за унос броја блокова

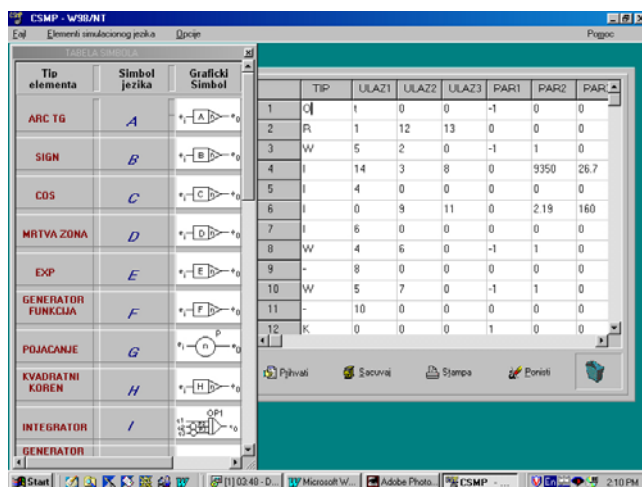
Унесена вредност потврђује се притиском на дугме *prihvati*, након чега се на екрану отвара табела са оноликим бројем редова за унос блокова колико је претходно дефинисано.

Блокови се задају преко тастатуре или методом *poovuci i pusti (drag & drop)* из помоћног панела који се отвара у менију *Елементи симулационог језика* → *otvori tabelu simbola* (слика 9.12)

Уношење блокова у табелу (симболи, улази и параметри) подлеже провери интегритета од стране програма. У случају нарушавања интегритета, програм сигнализира грешку.

Сваком блоку дефинисаном у конфигурацији могу се придружити највише три параметра. Неки елементи користе сва три параметра, а неки ниједан. Код блокова који немају сва три параметра, одговарајућа поља у табели су неприступачна, тако да корисник не може да наруши интегритет модела приликом уношења конфигурације. Код елемената који имају меморијска својства (интегратор, јединично кашњење, и сл.), први параметар се користи да зада почетни услов, односно вредност излаза блока за  $t = 0$ .

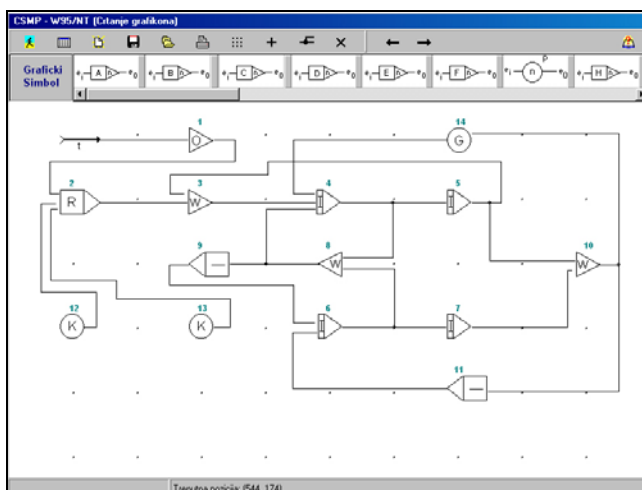
Током уношења табеларног модела, кориснику су на располагању опције за штампање и снимање модела. У случају погрешног уноса, поред стандардног брисања садржаја ћелије табеле, на располагању је и брисање целог реда методом "*drag and drop*" (са било које ћелије у том реду до канте за отпатке у доњем десном углу панела).



Слика 9.12 Табела за унос конфигурације модела

По завршеном уносу модела, притиском на дугме *prihvati*, програм врши контролу унесених података (позива процедуру за сортирање блокова) и информисе корисника о регуларности унесене конфигурације.

Други начин за задавање нове конфигурације је коришћење алата за графичку спецификацију *Novi → Grafik*.

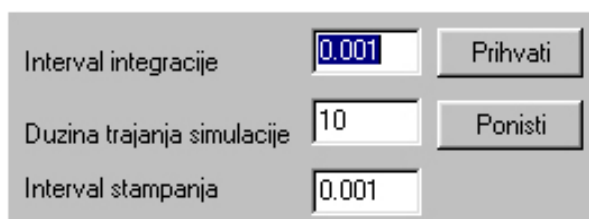


Слика 9.13 Задавање конфигурације графичким алатом

На врху радне површине налазе се иконе за отварање постојећег графика из фајла, прављење новог графика и снимање графика. Поред се налазе иконе за цртање веза између блокова, штампање графикона и брисање блокова. Такође, постоји и икона за превођење графичког модела у табеларни модел. У следећем реду налазе се графички симболи расположивих блокова, поређани на *scroll bar*-и. Задржавањем над сваким симболом блока приказује се његов назив. Методом "*drag and drop*" , блокови се пребацују на радну површину, а избором везе повезују. Пуштањем блока на радну површину, отвара се дијалог који омогућава унос његових параметара, док се улази у сваки блок аутоматски генеришу његовим повезивањем.

#### 9.4.2.2 Дефинисање вредности од значаја за симулацију

Након уношења конфигурације и провере њене исправности, а пре самог извођења симулације, неопходно је задати вредности за *интервал интеграције*, *дужину трајања симулације* и *интервал штампања резултата* (слика 9.14).



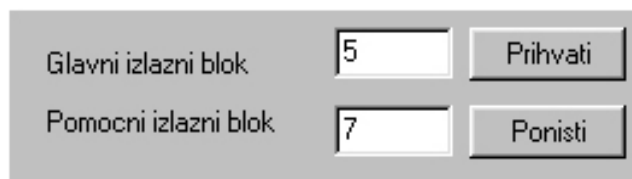
Слика 9.14 Дијалог прозор за унос параметара симулатора

Процес симулације на дигиталном рачунару је по својој природи дискретан процес. Због тога је потребно пре почетка симулације дефинисати интервале времена за које ће се вршити израчунавање вредности излаза блокова у моделу. Ови интервали времена се називају *интервали интеграције*. За одређивање интервала интеграције не постоји формално правило, већ се он одређује експериментом. При одабиру ове вредности треба бити обазрив, јер се при сувише великом интервалу интеграције губи увид у понашање система за време тог интервала, а могућа је и појава нумеричке нестабилности. С друге стране, код сувише малих вредности интервала интеграције, може доћи до нагомилавања

грешака услед великог броја рачунских операција, а и сам процес симулације дуже траје. Препоручује се да при првом експерименту интервал интеграције буде бар 10 пута мањи од најбрже временске константе у систему (Раденковић, 1984).

Резултати симулације се не морају штампати за сваки интервал интеграције. *Интервал штампања* резултата симулације треба да буде цео умножак интервала интеграције и задаје се у горе приказаном дијалогу.

У наредном кораку неопходно је дефинисати редне бројеве блокова (највише два) чији излази представљају резултате симулације (слика 9.15) који се прате.



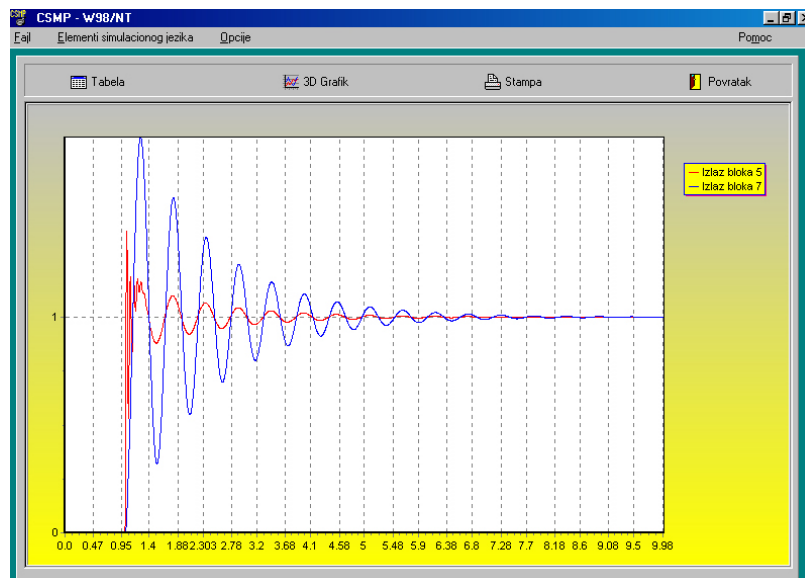
Дијалог прозор са два реда поља за унос и дугмад за потврду и повраћање.

Glavni izlazni blok	<input type="text" value="5"/>	<input type="button" value="Prihvati"/>
Pomocni izlazni blok	<input type="text" value="7"/>	<input type="button" value="Ponisti"/>

Слика 9.15 Дијалог прозор за унос излазних блокова

Започињање симулације врши се притиском на тастер *Симулација* (који до тада није био расположив). Након обављене симулације, на располагању су графички и табеларни резултати блокова који се прате (слике 9.16 и 9.17).

Графички приказ резултата симулације могућ је у 2D и 3D варијанти и бира се тастером на врху графика. Постоји могућност увеличавања графика одабиром одређеног сегмента, лак прелаз са графика на табеларне резултате, као и штампање резултата у обе варијанте.



Слика 9.16 Графички приказ резултата симулације

	Vreme	Izlaz bloka 5	Izlaz bloka 7
1	0.0	0	0
2	0.002	0	0
3	0.005	0	0
4	0.008	0	0
5	0.011	0	0
6	0.014	0	0
7	0.017	0	0
8	0.02	0	0
9	0.023	0	0
10	0.026	0	0
11	0.029	0	0
12	0.032	0	0
13	0.035	0	0
14	0.038	0	0
15	0.041	0	0
16	0.044	0	0
17	0.047	0	0
18	0.05	0	0

Слика 9.17 Табеларни приказ резултата симулације

## ЈЕЗИЦИ ЗА СИМУЛАЦИЈУ ДИСКРЕТНИХ ДОГАЂАЈА

Моделирање система или објеката у дискретним тренуцима времена помоћу рачунара назива се дигиталном симулацијом. Дигитална симулација дискретних догађаја бави се моделирањем система на дигиталним рачунарима, где се променљиве стања могу представити скупом дискретних догађаја. Код ове класе система, промене стања се дешавају када се деси догађај. Техника моделирања зависи од природе интервала одабирања. Временски интервали могу да буду случајни или детерминистички. У принципу, разликујемо две врсте симулације. Асинхрона симулација је она у којој се догађаји дешавају у произвољним тренуцима времена. Код синхроне симулације време напредује за константни интервал одабирања (временски корак).

Постоји више различитих стратегија управљања током симулације. Најопштији алгоритми су прилази засновани на догађајима, активностима (трофазни прилаз) и процесима. Наравно, свака од ових структура може се пресликати у било коју другу.

Међу три најпознатија језика за симулацију дискретних система спадају: GPSS (*Gordon, 1961*), SIMULA (*Dahl-Nygaard, 1962*) и SIMSCRIPT (*Markowitz, 1962*). У тексту који следи, највише простора биће посвећено језику GPSS.

### 10.1 Прилаз заснован на наредном догађају

Процедура се састоји у томе да се одреди наредни догађај и да се изврше све промене зависне од тог догађаја. Програм заснован на догађају мора да укључи блокове кода који одговарају на догађаје система и активности које би могле да следе те догађаје. Овде програм захтева експлицитну идентификацију подскупа активности које користе ресурсе ослобођене од стране догађаја. На оваквом



прилазу засновани су симулациони језици као што су SIMSCRIPT, GASP и др.

### 10.1.1 SIMSCRIPT

Језик SIMSCRIPT развијен је од стране Markovitz-а у RAND корпорацији, најпре као транслятор који преводи изворни SIMSCRIPT програм у еквивалентне FORTRAN наредбе. Затим је направљен компајлер за SIMSCRIPT који представља стандард и назван је SIMSCRIPT II.5. Подељен је у пет нивоа, који омогућавају лако програмирање. Нивои 1 до 3 садрже особину која се налази код већине језика вишег нивоа. Нивои 4 и 5 уводе концепте моделирања који су потребни у симулацији. На пример, класе перманентних и привремених ентитета са атрибутима. Ниво 5 служи за управљање симулационим временом, а укључује и могућност прикупљања статистике. За све дефинисане променљиве, које се посматрају, систем ће прикупити све захтеване статистике, при чему корисник не треба да задаје посебан код. Најважнији блокови су ентитети, атрибути и скупови. Ентитети означавају објекте система, атрибути карактеристике ентитета, а скупови - груписање ентитета.

## 10.2 Трофазни прилаз

Трофазни прилаз, као што и сам назив указује, карактеришу три фазе извршавања, а то су:

Померање симулационог сата на време наредног распоређеног догађаја.

Извршавање свих планираних (распореджених) догађаја чије је време наступања једнако садашњем тренутку.

Тестирање свих условних догађаја и извршавање оних чији су услови задовољени.

Код овог прилаза, планирани догађаји (В – догађаји) и услови догађаји (С – догађаји) програмирају се као посебне процедуре. Детаљнији опис овог прилаза дат је у поглављу 8.17.

Трофазна стратегија највише одговара за моделирање система код којих постоје сложенији услови за додељивање ресурса, односно сложеније интеракције између ентитета и ресурса система. Код ове стратегије сва правила одлучивања су сконцентрисана у условном догађају. Веома је погодна за примену симулације са визуелним излазом зато што се екран може ажурирати више пута, на пример после извршавања распоређених догађаја, после извршавања условних догађаја или након ажурирања симулационог сата.

Главни недостатак трофазне стратегије јесте њена релативна неефикасност, тј. са повећањем величине модела повећава се и број условних догађаја као и број бескорисних позивања условних догађаја.

На трофазном прилазу заснивају се језици као што је ECSL, SIMON, и др.

### 10.2.1 ECSL

Најпре се појавио CSL у току 1960/61. године као транслатор FORTRAN-а. Након више верзија, коначно CSL постаје ECSL језик, написан од стране Alan-а Clemantsou-а са Универзитета у Birmingham-у. ECSL је написан у FORTRAN-у и може се користити на сваком рачунару за који постоји компајлер за FORTRAN. Наравно, то никако не подразумева да корисник мора да познаје FORTRAN. Касније су уграђене могућности визуелног интерактивног моделирања.

## 10.3 Прилаз заснован на процесима

Ова стратегија се може посматрати као комбинација стратегије распоређивања догађаја и сканирања активности. Овај прилаз је, вероватно, најбољи за системе масовног опслуживања где ентитети чекају на опслугу од стране ресурса. Овде се нагласак ставља на моделирање динамичких ентитета у симулацији (праћење њиховог напредовања кроз модел).

Овај прилаз је основа структуре процеса код симулационих језика SIMULA и GPSS. Овде се даје кратак осврт развоја симулационог језика SIMULA, док ће опис језика GPSS бити дат у поглављу 10.6.

SIMULA је процесно орјентисан симулациони језик, развијен раних 60-тих година. Стандардни пакет је SIMULA Begin (*G. M. Birtwistle at all, 1973*). SIMULA заснива свој почетак на ALGOL-у. Основна структура језика су класе, а дефинсање класа система обезбеђује распоређивање догађаја и манипулисање редовима.

Основне идеје су даље развијене у DEMOS-у (Discrete Event Modelling on Simula). Ово је нова класа која има уграђене концепте за синхронизацију активности. Сама структура је врло различита од стандардне структуре SIMULA-е.

Компајлери језика SIMULA били су расположиви за широку класу рачунара као што су: IBM, DEC 10, UNIVAC 1100 Series, ICL 2900, PERO, VAX (VMS i UNIX), Honeywell Bull DPS8, Cyber, Spery, и др.

Поред тога што је SIMULA изведен језик из ALGOL-а, постоје реализације симулационих алата заснованих на процесима у PASCAL-у. У ту сврху развијене су верзије MICRO PASSIM (*Barnett, 1980*), Pascal-SIM (*O'Keefe, 1986*), SIMPAS (*Bryant, 1986*), SIMTOOLS (*Seila, 1986*).

## 10.4 Објектно орјентисани симулациони језици

Објектно орјентисани симулациони језици усредсређени су на индивидуалне објекте који се повезују са другим објектима комуникацијом преко порука и који имају могућност да наслеђују особине других објеката - веома слично као код SIMULA-е. Предност објектно орјентисаног програмирања у симулацији заснива се на могућности вишеструке употребе кода.

SMALLTALK омогућава кориснику да развије моделе у слободном програмском окружењу. Улаз симулације је заснован на графичком приказу модела и представљању одређеног броја параметара од стране корисника. SMALLTALK је интерпретерски језик, те је његово извршавање нешто спорије.

## 10.5 Језици за хибридную симулацију

Ако модел садржи континуалне и дискретне променљиве (комбиновани модел), тада корисник који жели да напише програм има на располагању једну од четири алтернативе:

1. да употреби неки од језика опште намене,
2. да побољша неки од симулационих језика за континуалне моделе, тако да се могу прихватити и дискретни модели,
3. да побољша неки од симулационих језика за дискретне моделе тако да се могу прихватити и континуални модели,
4. да употреби неки од симулационих језика за комбиноване моделе (GASP, SLAM, APLS, CDSP, CLASS, и др.)

У пракси, највероватнији је избор прве или четврте варијанте, јер друга и трећа захтевају доста времена за развој - побољшање језика.

## 10.6 Симулација на језику GPSS

GPSS (General Purpose Simulation System) представља интерпретерски језик за симулацију дискретних - стохастичких система. Прву верзију овог језика (GPSS-I) развио је Geoffrey Gordon 1961. године, за тадашње рачунаре IBM-704 и IBM-709. Од тада, овај језик је прошао кроз уобичајену серију промена и побољшања, тако да је сада у употреби последња верзија овог језика GPSS-V (*T. J. Schriber, 1974*), коју испоручује IBM. Такође постоје верзије других произвођача софтвера, као што су GPSS/66 (Honeywell Series 60 level 66) и GPSS/UCC (University Computing Corporation's). Поједини произвођачи софтвера, да би избегли строге ауторске прописе, дају својим верзијама GPSS језика друга имена, али се у суштини ради о истом језику који је у опште узев добро стандардизован.

Такође, постоје верзије за персоналне рачунаре. Такве верзије су на пример: GPSS/PC развијена од стране Minueman software, САД, која у себи садржи интерактивну графику и анимацију. Затим, верзија GPSS/H развијена од стране Wolverine Software Corporation, SAD (*GPSS/H Reference Manual, 1989; J. Banks et al, 1989; T. Schriber, 1991*).

Због потреба образовног процеса на Факултету Организационих наука у Београду развијена је верзија GPSS језика, названа GPSS/FON, реализована на језику PASCAL ( *Б. Раденковић,*

1989). Основа за имплементацију овог језика је одређени скуп инструкција из верзије GPSS -V. При имплементацији су учињена извесна побољшања у односу на стандардни GPSS у погледу формата програма и дужине идентификатора.

GPSS је симулациони систем у којем се на једноставан начин помоћу наредби уграђеног језика дефинише структура модела и врши симулација. По завршеној симулацији, на располагању су статистички показатељи о понашању модела у току симулације.

GPSS је језик оријентисан на процесе. Модел у GPSS језику представља се у облику дијаграма тока. За конструисање дијаграма тока, на располагању је преко педесет различитих блокова, од којих сваки има своје име, специјални симбол и намену. Да би конструисао модел, програмер бира изврстан број блокова и повезује их на одређени начин, а затим добијену структуру преводи у скуп GPSS наредби. Иако модел у GPSS-у на први поглед изгледа исто као и програм у било којем општенаменском програмском језику вишег нивоа, разлике између GPSS блок наредби и наредби стандардних језика су значајне. Сваком блоку одговара једна линија у GPSS програму, а један подпрограму у Асемблеру представља један блок. Блокови су статички објекти. То су редови, уређаји (facility), складишта (storage), логички прекидачи (logical switch), табеле и др.

Ентитете (објекте модела) можемо поделити у четири класе:

1. динамички ентитети,
2. статички ентитети,
3. статистички ентитети и
4. ентитети операција.

### 10.6.1 Динамички ентитети

*Трансакције* су једини динамички објекти GPSS језика које се “крећу” кроз блокове. Трансакције имају низ карактеристика које називамо параметрима. Тако свака трансакција, осим што заузима извесни блок, може са собом да “понесе” и скуп параметара. Трансакције су активни објекти. У сваком тренутку времена може постојати више трансакција на различитим местима у блок дијаграму. Трансакције представљају саобраћајне јединице као што

су: клијенти на опслуживању, телефонски позиви, возила у сервису, авиони, бродови и др. Свака трансакција може према својој потреби да резервише одређене ресурсе у моделу, који су увек ограничени (број шалтерских места у пошти, број канала у ауто-сервису, број канала у телефонској централи и слично).

### 10.6.2 Статички ентитети

Ентитети друге класе представљају елементе опреме система (ресурсе) на које се дејствује трансакцијама. То су пасивни објекти. Ту спадају: уређаји, складишта, логички, прекидачи. Уређај може да услужи само једну трансакцију у неком тренутку времена. На пример, службеник на шалтеру, док у луци, полетно-слетна стаза, продавац и др. Складиште може да услужи више трансакција истовремено (меморија рачунара, аутобус, воз, авион, паркинг и др.). Логички прекидачи могу да буду постављени помоћу трансакција у једно од три стања: укључено, искључено и инвертовано, а затим да буду тестирани од стране других трансакција на основу чега се управља током трансакције.

### 10.6.3 Статистички ентитети

У ову групу ентитета спадају редови и табеле. Ако је уређај заузет, или ако је складиште пуно, тада трансакције формирају ред. Мерење таквих редова је један од важнијих функција симулационих програма. Симулатор аутоматски подржава неке статистике које се односе на уређаје, складишта и редове и у том циљу формира одговарајуће табеле. Корисник може да добије и друге статистике уколико жели. Тако, на крају симулације, корисник може да добије табелу транзитних времена чије се табелирање врши фреквенцијски унутар сваког опсега. Табела садржи у основи фреквенцијске класе у којима су представљене нумеричке вредности неког атрибута.

### 10.6.4 Ентитети операција

Ентитети операција, који се називају “блоковима“, чине четврту класу ентитета. Слично блок дијаграму, они пружају логику система, дају инструкције трансакцијама где треба да иду и шта даље да

раде. Блок дијаграм представља операције које се врше унутар датог система. Сваки блок показује тип операције коју обавља, а линије између блокова указују на ток саобраћаја. Циљ је да се створи блок дијаграм који јасно показује све тачке где се доносе одлуке у систему и који може да се користи за верификацију свих могућих услова који настају за време рада таквог система. Такав блок дијаграм је уствари модел система.

Унутар блок дијаграма могу да настану четири основна типа догађаја:

1. стварање или уништавање трансакција,
2. промена нумеричких атрибута ентитета,
3. кашњење трансакције за одређени износ времена,
4. промена тока трансакције кроз блок дијаграм.

#### 10.6.5 Основни концепти GPSS језика

Трансакцију генерише блок GENERATE. Она се креће кроз модел све док не наиђе на блок који нема услова да је прими или не наиђе на блок TERMINATE који уклања трансакцију из модела. Такође, постоје блокови који задржавају трансакцију за одређени период симулационог времена. Уколико неки блок нема услова да прими трансакцију, тада трансакција чека да се испуни услов даљег кретања кроз модел. Пролаз трансакције кроз одређене блокове изазива промене које утичу на стања модела и његово окружење.

Генерално, трансакције у GPSS-у се чувају у два листама: листи текућих догађаја (LTD) и листи будућих догађаја (LBD) (current events chain - CEC, future events chain - FEC). Трансакције из LTD настоје да се крећу кроз блок дијаграм. Поједини блокови могу да одбију пријем трансакције и на тај начин зауставе њено кретање. Трансакције из LBD нису спремне за кретање. Тек касније, када се ажурира симулациони сат (онда када се испуни услов за кретање), ове трансакције се пребацују у LTD, како би наставиле кретање кроз модел.

У свакој тачки симулационог времена (симулациони сат задат нумеричким атрибутом C1 узима само целобројне вредности), GPSS сканира све трансакције у LTD. Свака трансакција која има испуњен услов даљег кретања то и чини. Кретање трансакција се

прекида због једног од три разлога: трансакција је уништена, блок одбија да прими трансакцију или је трансакција ушла у блок који је задржава извршан период симулационог времена. Уколико је у питању трећи разлог, трансакција се пребацује из LTD у LBD. Уколико је кретање трансакције изазвало промену стања модела, GPSS поново обавља сканирање, али од почетка LTD; у супротном, наредна трансакција из LTD се сканира. Пре или касније, наступа ситуација да ниједна трансакција из LTD не може да се креће.

Када наступи тај тренутак, GPSS испитује LBD. Листа будућих догађаја је уређена тако да се на њеном почетку налазе трансакције које услов за кретање кроз модел стичу раније, док су на крају оне трансакције које морају дуже да чекају. Када ниједна трансакција из LTD не може да настави кретање, GPSS ажурира симулациони сат на време прве трансакције из LBD. Све трансакције из те листе, које су сада стекле услов за кретање, копирају се из LBD у LTD и сканирање LTD се понавља.

Свака трансакција има свој приоритет, који се може мењати са протоком симулационог времена и тренутним положајем трансакције у блок дијаграму. Листа текућих догађаја је сортирана по опадајућем редоследу приоритета, тако да се трансакције са већим приоритетом крећу пре оних чији је приоритет мањи у сваком тренутку симулационог времена.

### 10.6.6 Врсте наредби у GPSS-у

У GPSS -и разликујемо следеће врсте наредби:

1. Декларационе наредбе ентитета,
2. Блок наредбе,
3. Контролне наредбе.

Декларационим наредбама ентитета дефинишу се атрибути појединих перманентних ентитета у програму. Ту спадају наредбе за декларацију уређаја, складишта, табела (хистограма), функција и др. У пољу ЛОКАЦИЈЕ специфицира се посебно дефинисано складиште, табела и слично. У пољу ВАРИЈАБЛЕ могу да буду уписани различити аргументи, чије значење зависи од типа наредбе која се користи. Декларационе наредбе ентитета се могу писати



било где унутар програма, али је уобичајено да се пишу на почетку програма.

Блок наредбе чине основу модела система који се симулира, а служе за његову спецификацију. Модел се састоји од низа блок наредби повезаних у облику блок дијаграма. Блок наредба се извршава када трансакција приликом кретања кроз модел наиђе на блок. Ефекат извршења блок наредбе зависи од природе специфициране наребе. Свака блок наредба може да има идентификациони број, који се пише у пољу локације. У већини случајева то није потребно, јер ће сам програм аутоматски исписати секвенцијалне бројеве блок наредби. Међутим, ако корисник жели да се позове на неки посебан блок, он може да унесе у поље ЛОКАЦИЈЕ одређени мнемонички симбол. У пољу ОПЕРАЦИЈЕ пише се тип блок наредбе, а у пољу ВАРИЈАБЛЕ пишу се аргументи специфицираног блока.

Важније блок наредбе GPSS језика биће детаљно обрађене у наредним поглављима.

Контролне наредбе служе за контролу извршења симулације, а могу имати утицај на статистику о понашању ентитета у току симулације. Симулатор захтева неке информације управљања, такве као дужина симулације која се жели, када је крај наредби програма и др. У пољу ОПЕРАЦИЈА пише се тип наредбе, а у поље ВАРИЈАБЛЕ уносе се аргументи специфициране наредбе. Поље ЛОКАЦИЈЕ се не користи од стране контролних наредби. То су наребе SIMULATE, START, RESET и CLEAR чија је семантика описана у наредним поглављима.

Трајање симулације може бити ограничено бројем трансакција које су прошле кроз модел, што се контролише погодном употребом наредби START и TERMINATE или временски (фиксно), што је одређено тзв. тајмером који чини пар наредби GENERATE и TERMINATE.

#### 10.6.7 Основни скуп наредби GPSS језика

У наредном поглављу биће описан основни скуп наредби језика GPSS /ФОН. При дефинисању синтаксе наредби, операнди који су наведени у угластим заградама, [ ], могу да се изоставе. У том случају они узимају подразумеване вредности. Операнди који нису

означени угластим заградама морају да се специфицирају. Мора се водити рачуна да опсези вредности операнада морају да буду унутар одређених вредности, у супротном, приликом превођења пријављује се грешка.

#### 10.6.7.1 Генерисање трансакција

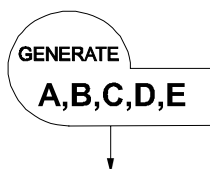
Трансакција је једини динамички ентитет у GPSS програму. Трансакцију генерише блок GENERATE и, уколико је наредни блок у стању да је прими, трансакција почиње кретање кроз модел. У супротном, трансакција остаје у блоку GENERATE све док се не оствари услов њеног даљег кретања. У том случају, средњи временски интервал између стварања трансакција биће већи од средње вредности која је задата у пољу А, што може да доведе до грешке. То се може избећи постављањем блока који не одбија улазак трансакцијама одмах после блока GENERATE (на пример, то може да буде блок ADVANCE 0).

При дефинисању модела, концепт трансакције се користи за представљање кретања саобраћајних јединица кроз систем опслуживања, као што су: клијенти у поштама, купци који долазе у самопослугу, возила која долазе у сервис или појава телефонског позива при симулацији телефонске централе.

Сваки GPSS програм мора да садржи бар један блок GENERATE. Уколико је то потребно, модел може садржати више GENERATE блокова, од којих сваки ради независно у односу на све друге. Блок GENERATE је једини блок у који трансакције не улазе.

Синтакса, семантика наредбе као и графички симбол блока GENERATE могу се претставити на следећи начин:

GENERATE A,[B],[C],[D],[E]



где су:

A - средње време између генерисања  $\tau$  (трансакција)

B - модификатор чија се вредност користи за модификацију

- вредности из поља А.
- С - време генерисања прве  $\tau$
- D - укупан број  $\tau$  које генерише овај блок
- E - приоритет трансакције.

Поље В је модификатор чија се вредност користи за модификацију вредности из поља А. То може да буде модификатор у виду интервала или у виду функције. Модификатор у виду интервала се користи када је време између долазака равномерно расподељено у неком задатом опсегу. На пример, ако су времена између долазака трансакција униформно расподељена у интервалу  $10 \pm 5$  временских јединица, тада је средња вредност 10, а вредност полуинтервала униформног одступања је 5. Као резултат добијамо униформно расподељене целе бројеве од 5 до 15, тј. {5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}. Само једна од могућих целобројних вредности се одабира за сваку појединачну трансакцију. Свака од могућих целобројних вредност се одабира са једнаком вероватноћом, укључујући и крајње тачке.

*Пример:*

GENERATE 10,5

Модификатор у виду функције се користи када су времена између долазака трансакција расподељена у виду неке сложеније расподеле. У том случају, време између долазака се израчунава множењем средње вредности дефинисане у пољу А са добијеном вредношћу функције у пољу В.

*Пример:*

GENERATE 80,FN\$EXPO

#### 10.6.7.2 Временско задржавање трансакција

Приликом симулације система опслуживања, потребно је на неки начин симулирати време задржавања на месту опслуживања. Једини блок у GPSS-у који може да задржава трансакције за одређени временски период је блок ADVANCE. Када трансакција наиђе на овај блок, њено кретање кроз модел се прекида за специфицирани број временских јединица. По истеку

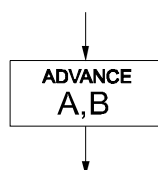
специфицираног броја временских јединица, трансакција наставља кретање кроз модел.

Ова наредба користи се за симулацију времена проведеног на опслуживању на шалтерима у пошти, при куповини карата, на каси самопослуге, времена које је потребно за обраду производа у једној фази производње, време које возило проведе на поправци, или времена разговора при симулацији телефонске централе итд.

Када год трансакција наиђе на блок ADVANCE, она се пребацује из листе текућих догађаја (LTD) у листу будућих догађаја (LBD). Тек након што се симулациони сат ажурира за специфицирани број временских јединица (из наредбе ADVANCE), трансакција се враћа у LTD и наставља се њено напредовање кроз модел.

Синтакса, семантика наредбе, као и графички симбол блока ADVANCE могу се представити на следећи начин:

ADVANCE A[,B]



где су:

A - средње време задржавања  $\tau$

B - модификатор средње вредности задржавања  $\tau$ .

Модификатор може да буде у виду интервала или у виду функције, као у случају блока GENERATE.

Уколико је полуинтервал униформног одступања већи од средњег времена задржавања трансакције, GPSS процесор пријављује грешку у току извршења симулације.

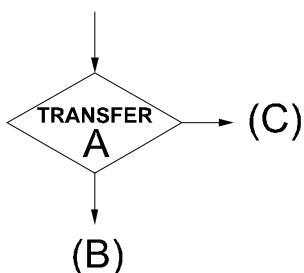
### 10.6.7.3 Статистичко гранање трансакције

При симулацији система опслуживања, некад је потребно извршити избор извесног процента трансакција и преусмерити их на други део модела. Тако, на пример, при одређивању шкарта у симулацији

производног процеса, при одређивању броја путника који силазе из аутобуса у симулацији градског превоза, итд. Наведени случајеви могу се симулирати коришћењем статистичке TRANSFER наредбе.

Синтакса, семантика наредбе, као и графички симбол блока су:

TRANSFER .A,[B],C



где су:

A - вероватноћа скока на блок чије је обележје дато у пољу C,

B, C - обележја блокова у које одлази  $\tau$  после гранања.

Ако је у пољу A блока TRANSFER написан децимални разломак, тада се врши случајни избор између блокова који су дефинисани у пољима B и C. Вероватноћа избора блока специфицираног у пољу C задаје се децималним разломком у пољу A. Вероватноћа се пише у форми .d или .dd или .ddd где  $d \in [0,1,2,\dots,9]$ .

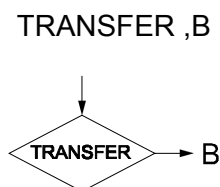
*Напомене:*

- ♦ исходи скокова су статистички независни.
- ♦  $\tau$  остаје у блоку TRANSFER, уколико блок у који она треба да пређе не може да је прими, све док се не испуни услов пријема.

#### 10.6.7.4 Безусловни скок

У случајевима када је потребно безусловно усмерити трансакцију на неки други део модела, користи се наредба безусловног трансфера.

Синтакса, семантика наребе, као и графички симбол блока су:



где је:

B - обележје блока на који  $\tau$  одлази безусловно.

*Напомена:*

Уколико блок чије је обележје дефинисано у пољу B не може да прихвати  $\tau$ , она ће бити задржана у блоку TRANSFER. Када се услов за пријем оствари,  $\tau$  аутоматски одлази на назначени блок.

#### 10.6.7.5 Уклањање трансакције из система

Када трансакција изврши предвиђени ток кроз модел, она постаје неактивна и нема више никаквог утицаја на модел. С обзиром на то да трансакција заузима изваншар меморијски простор који се користи за запис њених атрибута, држање у меморији неактивних трансакција, нарочито код већих модела, може потпуно да онемогући симулацију. Због тога у GPSS-у постоји наредба за уклањање трансакције из модела.

Блок TERMINATE уклања трансакције из модела. Он се користи за представљање завршетка пута трансакције у систему. Број блокова TERMINATE је произвољан, али бар један блок мора да има дефинисано поље A, чија је вредност једнака јединици или већа, да би се симулација завршила.

Синтакса, семантика наредбе, као и графички симбол блока могу се представити на следећи начин:

TERMINATE [A]



где је:

A - вредност за коју се умањује терминациони бројач када  $\tau$  дође на овај блок.

*Напомена:*

Терминациони бројач (ТБ) је јединствена локација на нивоу GPSS програма, која се умањује кад год нека трансакција дође на неки од блокова типа TERMINATE који у пољу A имају неку вредност. Када вредност ТБ падне на нулу, симулација се аутоматски завршава. Почетна вредност ТБ дефинише се помоћу контролне наредбе START.

#### 10.6.7.6 Стартовање симулатора

Након завршеног дефинисања модела, да би се извршила симулација, потребно је стартовати симулатор. Стартовање симулатора врши се наредбом START.

Синтакса и семантика ове наредбе је:

START A [,B] (контролна наредба)

где су:

A - почетна вредност терминационог бројача,  
B - користи се за укидање штампања резултата по завршетку симулације. Да би се то постигло, потребно је у поље B унети ознаку NP (No Print).

*Напомена:*

Наредбом START започиње се процес симулације (извршења GPSS програма). Она се налази на крају пакета блок наредби и декларационих наредби.

#### 10.6.7.7 Дефинисање почетка и краја GPSS програма

Контролна наредба SIMULATE је прва наредба GPSS програма. Ако се она изостави, програм ће се превести, али се неће извршити (наредба START тада нема ефекта).

Контролна наредба END је задња наредба GPSS програма. Она означава крај пакета за GPSS преводилац и уједно је извршна наредба после које се управљање предаје оперативном систему.

Синтакса ових наредби је једноставна:

SIMULATE	(контролна наредба)
END	(контролна наредба)

Симулација престаје када терминални бројач падне на нулу. Уколико је потребно да се симулација врши на задатом интервалу времена, потребно је увести TAJMEPE.

За илустрацију наредби GPSS језика које смо упознали до сада, послужимо се једним једноставним примером.

##### *Пример 1.*

Машина производи делове сваких 5 минута. Делови иду на контролу квалитета. Контролу врши група од неограниченог броја инспектора. Време прегледа је  $4 \pm 3$  минута. Зна се да машина производи 10% шкарта.

Симулирати процес прегледа 1000 делова. За временску јединицу узети 1 минут.

##### *Решење:*

При дефинисању овог модела, машина се представља блоком GENERATE који генерише трансакције сваких 5 временских јединица. Трансакција представља део који се израђује на машини. Контрола квалитета се представља блоком ADVANCE који задржава трансакцију за време контроле од  $4 \pm 3$  мин. С обзиром на то да је при контроли 10% производа шкарт, распоред прихваћених и одбачених делова врши се статистичком TRANSFER наредбом, која обезбеђује гранање трансакција са вероватноћом од 0.1. Када делови које је произвела машина (трансакције) буду прихваћени

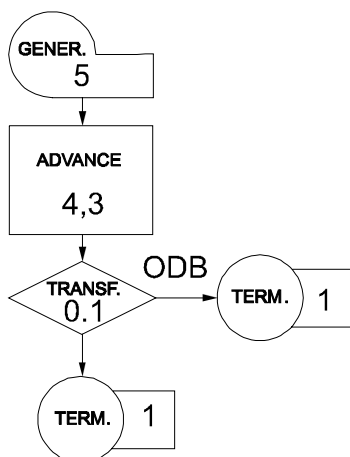


или одбачени, они нису више од значаја за наш симулациони модел, па се због тога уклањају из модела наредбом TERMINATE. Вредност декремента терминационог бројача је један, па је време симулације ограничено бројем терминираних трансакција, који је задат START наредбом. Задата вредност терминационог бројача је 1000.

Одговарајући GPSS програм може се написати на следећи начин:

	SIMULATE	Vrši se simulacija
	GENERATE 5	Mašina proizvodi delove svakih 5 vj.
	ADVANCE 4,3	Kontrola kvaliteta 4±3 min.
	TRANSFER .1,,ODB	Kontrolori kvaliteta vrše kontrolu
	TERMINATE 1	Ispravni delovi se uklanjaju iz modela
ODB	TERMINATE 1	Odbačeni delovi se uklanjaju iz modela
	START 1000	Startovanje simulatora TB = 1000
	END	Fizički kraj simulacionog programa

Одговарајући блок дијаграм за овај програм је:



Резултати симулације за овај пример, коју је извршио GPSS процесор су:

GPSS/FON Ver.1.2, Simulating results  
Relative clock 5004 Absolute clock 5004

Block counts	Block Current	Total
1	0	1000
2	0	1000
3	0	1000
4	0	906
5	0	94

Резултате тумачимо на следећи начин:

Симулација је трајала 5004 в.ј. односно 5004 минута, што је разумљиво, јер производња сваког комада траје 5 минута, односно 1000 комада 5000 минута. Време прегледа варира од 1-7 минута, а у нашем случају генератор случајних бројева (ГСБ) је одредио да преглед последњег комада траје 4 минута, па укупно време симулације износи 5004 (5000 + 4) минута. За 4 минута није произведен ниједан нови производ, што значи да је кроз блокове 1, 2 и 3 прошло по 1000 производа (трансакција), од чега 906 исправних, а 94 неисправна. Такав однос исправних и неисправних производа добро се слаже са почетном претпоставком о 10% шкарта производње (проценат добијен симулацијом је 9,4%).

### 10.6.8 Основни перманентни ентитети

Перманентни ентитети (објекти) постоје у GPSS симулатору од момента увођења до завршетка симулације. Сваки од перманентних ентитета има своје атрибуте који се ажурирају увек када трансакција покуша (успешно или неуспешно) да приступи ентитету. Атрибути перманентних ентитета штампају се аутоматски по завршетку симулације, док су у току симулације на располагању у облику стандардних нумеричких атрибута (CHA).

Перманентни ентитети се дефинишу на два начина:

1. Декларационим наредбама (STORAGE, TABLE...) и
2. Наредбама које означавају приступ ентитету (ENTER, QUEUE, LINK...)

Уколико се перманентни ентитети специфицирају декларационим наредбама, тада они постоје од самог почетка симулације. Ако је перманентни ентитет специфициран блоком који означава приступ ентитету, тада се ентитет уводи у модел тек када трансакција први пут наиђе на тај блок. У моделу, перманентни ентитет остаје до завршетка симулације.

#### 10.6.8.1 Уређаји

Уређај (FACILITY) је перманентни ентитет који представља једног опслуживоца. Уређај може истовремено да прихвати само једну трансакцију. Ако  $\tau$  покуша да пређе у запоседнут уређај, она се

аутоматски задржава у претходном блоку, све док се уређај не ослободи.

Концепт уређаја користи се код система опслуживања где постоји само један опслужилац који је у стању да опслужује само једног клијента (један шалтер у пошти, једна каса у самопослузи, један канал у ауто сервису, једна телефонска линија итд.).

Посебно важан концепт код уређаја је концепт пријемптирања помоћу којег је могуће симулирати опслужиоца који опслужује клијенте са различитим приоритетом.

Концепт пријемптирања се огледа у следећем:

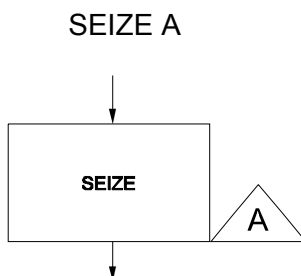
Уколико је уређај запоседнут трансакцијом са приоритетом  $(n)$ , а уређај покуша да запоседне трансакција са приоритетом  $(n1) > (n)$ , тада се трансакција која је запосела уређај уклања из уређаја и ставља на стек пријемптирања који постоји на сваком уређају. Њена обрада се прекида и уређај запоседа трансакција са вишим приоритетом. Када трансакција са вишим приоритетом напусти уређај, тада се скида трансакција са врха стека пријемптирања и наставља обраду од тачке где је била пријемптирана.

Могуће је пријемптирање у више нивоа, зависно од приоритета трансакција које долазе на уређај.

Запоседање и ослобађање уређаја врше наредбе SEIZE и RELEASE.

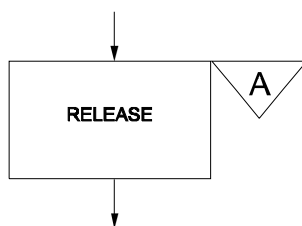
Синтакса, семантика наредби, као и графички симболи ових блокова су:

Запоседање уређаја:



Ослобађање уређаја:

RELEASE A



где A представља број, односно име уређаја.

### Пример 2.

У пошти постоји један шалтер за пријем пакета. Доласци клијената на шалтер за пакете дефинисани су према униформној расподели  $250 \pm 50$  секунди. Време опслуге на шалтеру за пакете је, такође, дефинисано униформном расподелом у опсегу  $200 \pm 100$  секунди. Потребно је испитати искоришћеност опслуживоца.

Симулирати временски период од 12 часова. За временску јединицу узети 1 секунду.

### Решење:

При дефинисању овог модела, доласци клијената се представљају блоком GENERATE, који генерише трансакције према униформној расподели  $250 \pm 50$  временских јединица. Трансакција представља клијента који захтева услугу. Опслуга клијената на шалтеру представља се блоком ADVANCE који задржава трансакцију за време које је дефинисано униформном расподелом  $200 \pm 100$  секунди. Након завршетка опслуге, клијент напушта пошту. Трансакција која представља датог клијента треба да буде уклоњена из модела. У ту сврху користи се блок TERMINATE. За дефинисање временског трајања симулације користи се такозвани TAJMER који генерише трансакцију у тренутку када треба да се заврши симулација, која се одмах уништава. Терминациони бројач задаје се наредбом START.

Одговарајући GPSS програм може се написати на следећи начин:

### Датотека изворног програма у GPSS-u:

```
* SIMULACIJA RADA JEDNOG ŠALTERA ZA PRIJEM PAKETA U POŠTI
*
SIMULATE
GENERATE 250,50          Dolazak klijenata
```

SEIZE SLUZH	Zauzmi službenika
ADVANCE 200,100	Vreme obsluživanja
RELEASE SLUZH	Oslobodi službenika
TERMINATE	Klijent napušta sistem
*	
* SEGMENT TAJMERA	
*	
GENERATE 43200	Generisanje Tajmera
TERMINATE 1	Umanji TB za 1
START 1	Početak simulacije
END	Kraj simulacije

### Листинг датотека истог програма:

GPSS/FON - Assembler Ver. 2.01, 1995

```

-----
1|* SIMULACIJA RADA JEDNOG ŠALTERA ZA PRIJEM PAKETA U POŠTI
2|*
3|    SIMULATE
4| 1    GENERATE 250,50    Dolazak klijenata
5| 2    SEIZE SLUZH        Zauzmi službenika
6| 3    ADVANCE 200,100    Vreme opsluge
7| 4    RELEASE SLUZH      Oslobodi službenika
8| 5    TERMINATE          Klijent napušta sistem
9|*
10|* SEGMENT TAJMERA
11|*
12| 6    GENERATE 43200    Generisanje Tajmera
13| 7    TERMINATE 1        Umanji TB za 1
14|    START 1              Početak simulacije
15|    END                  Kraj simulacije

```

Facility symbols and corresponding numbers

1: SLUZH

-----  
No errors detected

### Резултати симулације су:

GPSS/FON Ver. 2.0, Simulating results

Relative clock      43200   Absolute clock      43200

#### Block counts

Block	Current	Total
1	0	172
2	0	172
3	1	172
4	0	171
5	0	171
6	0	1
7	0	1

Facility	Average utilisation	Number entries	Average time/tran	Seizing transact.	Preempting transaction
1	0.806	172	203.491	1	0

Тумачење резултата:

Симулација је трајала 43200 временских јединица. Број клијената који су приступили шалтеру био је 172. Средња искоришћеност радника на шалтеру је 80.60%. Средње време проведено на опслужу је 203.491 временских јединица.

#### 10.6.8.2 Складишта

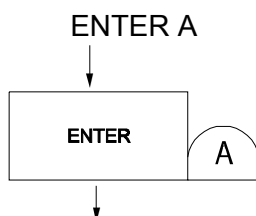
Складиште (STORAGE) је перманентни ентитет који представља групу симултаних опслужилаца. Складиште може истовремено да прихвати више трансакција, зависно од задатог капацитета. Ако  $\tau$  покуша да уђе у пуно складиште, она се аутоматски задржава у претходном блоку, све док нека од трансакција које су у складишту не напусти складиште и тиме ослободи довољно јединица складишта.

Као пример коришћења концепта складишта у симулацији организационих система може се навести самопослуга, која се представља једним складиштем капацитета максималног броја муштерија. Унутар самопослуге, више каса може бити представљено посебним складиштем чији је капацитет једнак броју каса. Такође и корпе у самопослузи представљају ограничени ресурс и могу се посматрати као посебно складиште, чији је капацитет једнак броју корпи.

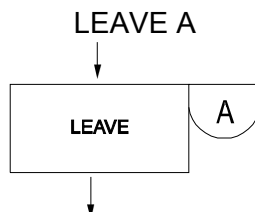
За улазак и излазак трансакције из складишта користе се наредбе ENTER и LEAVE, док се декларација складишта врши декларационом наредбом STORAGE.

Синтакса, семантика ових наредби, као и графички симболи ових блокова, могу се представити на следећи начин:

Улазак у складиште:



Напуштање складишта:



Декларација складишта:

A STORAGE [к] (декларациона наредба)

где су:

к - максимални (почетни) капацитет складишта  
 А - број /име складишта.

### Пример 3.

У пошти постоје два шалтера за писмоносне пошиљке. Доласци клијената на шалтере за пријем писмоносних пошиљака су према униформној расподели од 30 до 60 секунди. Времена опслуживања су униформно расподељена са средњом вредношћу 60 с и полуинтервалом од 20 с. Потребно је утврдити искоришћеност радника на шалтерима. Шалтере треба посматрати као складиште. Симулирати временски прериод од 12 часова. За временску јединицу узети 1 с.

Одговарајући GPSS програм за овај пример био би:

GPSS/FON - Assembler Ver. 2.01, 1995

```

1|* SIMULACIJA RADA PISMONOSNIH SALTERA U POSTI
2|*
3|  SLUZH STORAGE 2      Definisanje skaldišta
4|    SIMULATE
5|  1    GENERATE 45,15   Generisanje transakcija
6|  2    ADVANCE 0        Blok koji uvek prima τ
7|  3    ENTER SLUZH      Službenik počinje da radi
8|  4    ADVANCE 60,20    Vreme opsluzivanja
9|  5    LEAVE SLUZH      Službenik postaje slobodan
10| 6    TERMINATE        Klijent napušta sistem
11|*
12|* SEGMENT  TAJMERA
13|*
14| 7    GENERATE 43200    Generisanje Tajmera
  
```

```

15| 8      TERMINATE 1      Umanji TB za 1
16|        START 1        Početak simulacije
17|        END            Kraj simulacije
    
```

Storage symbols and corresponding numbers

1: SLUZH

-----  
No errors detected

**Резултати симулације су:**

GPSS/FON Ver. 2.0, Simulating results

Relative clock      43200   Absolute clock      43200

Block counts

Block	Current	Total
1	0	973
2	0	973
3	0	973
4	1	973
5	0	972
6	0	972
7	0	1
8	0	1

Storage	Capacity	Average Contents	Average Utilisation	Entries	Average Time/tran	Maximum Contents	Current Contents
1	2	1.353	0.677	973	60.082	1	2

На основу резултата, видимо да је искоришћеност оба радника 67.70%. Број клијената који су били опслужени је 973. Средње време проведено на опслузи је 60.082 временских јединица.

### 10.6.8.3 Логички прекидачи

Логички прекидач (LOGIC SWITCH) је перманентни ентитет који има два стања: укључен и искључен. Представља еквивалент за логичку променљиву која такође може имати два стања: тачно и нетачно. У симулационим моделима користи се за означавање једног од два стања неког ентитета у моделу.

Конкретан пример примене прекидача у симулационом моделу телефонске централе је означавање стања телефонске линије која може бити слободна или заузета. Сваком стању телефонске линије придружује се једно стање прекидача (укључен или искључен).

Операције над прекидачем извршавају се кад трансакција наиђе на блок LOGIC. Те операције су:

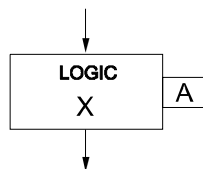


- ♦ укључивање прекидача,
- ♦ искључивање прекидача,
- ♦ инвертовање прекидача.

Инвертовање прекидача представља пребацивање прекидача у супротно стање од оног у којем се налази.

Синтакса, семантика наредбе, као и графички симбол блока представљају се на следећи начин:

LOGIC { X } A



где су:

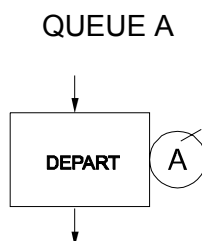
- |     |                                    |
|-----|------------------------------------|
| A   | - број / име прекидача             |
| {X} | - постфикс наредбе:                |
|     | X = R - искључи прекидач (Reset)   |
|     | X = S - укључи прекидач (Set)      |
|     | X = I - инвертуј прекидач (Invert) |

#### 10.6.8.4 Редови

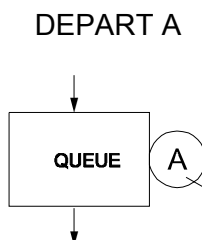
Блок QUEUE (ред) је перманентни ентитет који се уводи у модел ради прикупљања статистике о чекању трансакција на неком од блокова. У самом моделу, редови чекања егзистирају независно од увођења ових перманентних ентитета. Ред као ентитет не утиче на понашање модела, већ само служи за прикупљање статистике о моделу.

У модел може да се уведе већи број редова. Једна  $\tau$  може да буде члан већег броја редова (највише 5).

Пријава трансакције у ред:



Одјава трансакције из реда:



где је:

A - број /име реда

Блок QUEUE безусловно прихвата трансакцију. Уколико наредни блок има услов пријема, одмах је пропушта. Ако наредни блок нема услова за пријем, трансакција остаје у блоку QUEUE све док се не оствари услов пријема. Ако у блоку QUEUE чека више трансакција, одлази прво она која је најдуже чекала.

*Напомене:*

1. Ред чекања може постојати у моделу без обзира на постојање блокова QUEUE /DEPART. Ови се блокови умећу у програм само ради прикупљања статистичких података о чекању трансакција на одређени ресурс у моделу.
2. Ако је трансакција ушла у QUEUE блок, то не значи да она у њему остаје све време, мада остаје у одговарајућем реду.
3. Редови, тј. блокови QUEUE /DEPART ни на који начин не утичу на модел.

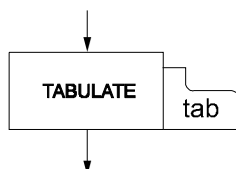
### 10.6.8.5 Табеле (хистограми)

Блок TABLE (табела) је перманентни ентитет који се уводи уколико се жели снимање хистограма за неку величину (Стандардни Нумерички Атрибут - СНА). Убацивање података (ажурирање табеле) у хистограм врши се доласком трансакције на блок TABULATE.

Табела се декларише декларационом наредбом TABLE. Синтакса, семантика ових наредби, као и графички симболи блока TABULATE су:

Ажурирање табеле:

TABULATE tab



Декларација табеле:

tab TABLE A,B,C,D (декларациона наредба)

где су:

- tab - број/име табеле
- A - дефинише величину (СНА) која се снима у табели
- B - горња граница првог интервала хистограма
- C - ширина интервала
- D - број фреквенцијских интервала (класа).

*Напомене:*

1. Табела се ажурира кад год нека  $\tau$  дође на блок TABULATE.
2. Блок TABULATE никада не задржава  $\tau$  и он ни на који начин не утиче на модел.

Уколико желимо да снимимо хистограм времена путовања  $\tau$  кроз модел, од тренутка њиховог генерисања до неког блока, тада се испред тог блока убацује наредба: TABULATE tab, а табела tab се декларише наредбом:

tab TABLE M1,B,C,D

где је M1 CHA који означава време путовања трансакције.

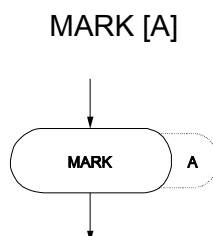
#### 10.6.8.6 Блок MARK

Када програм почне да ради, врши се откуцавање временских јединица, јер у себи има часовник. То је време апсолутног сата (CLOCK TIME). Ово време важи за све трансакције истовремено од тренутка уласка трансакција у модел (систем). Поред овог времена, постоји такозвано MARK TIME које служи да у датом моделу одредимо “транзитно време” за сваку трансакцију. Транзитно време је време путовања трансакције у појединим сегментима модела. MARK TIME се може рачунати за сваку трансакцију. Програм на почетку маркира сваку трансакцију са апсолутним временом (CLOCK TIME) са којим она првобитно улази у модел. У том тренутку “МАРК” ВРЕМЕ једнако је апсолутном времену симулације (CLOCK TIME), док је транзитно време једнако је нули, јер се оно рачуна на следећи начин:

$$\text{ТРАНЗИТНО ВРЕМЕ} = \text{АПСОЛУТНО ВРЕМЕ} - \text{“МАРК” ВРЕМЕ}$$

Произилази да нема потребе за прогресивним повећањем транзитног времена за све трансакције унутар модела са променом (порастом) времена апсолутног сата, јер је то време истовремено и транзитно време трансакције кроз модел (систем).

Ако нас интересује време путовања трансакције кроз одређени сегмент модела, онда је потребно вратити транзитно време трансакције на нулу. У ту сврху користи се блок MARK.



Операнд А дефинише број параметра трансакције у који се уписује текуће време апсолутног сата.

Кад год трансакција уђе у блок MARK, "MARK" ВРЕМЕ те трансакције се изједначује са апсолутним временом (MARK TIME = CLOCK TIME). Операнд А није обавезан и, у зависности од тога, овај блок ради на два начина:

1. Ако се операнд А изостави или је једнак нули, тада се транзитно време трансакције анулира, то јест врши се ресетовање времена путовања трансакције, јер се "MARK" ВРЕМЕ изједначује са CLOCK TIME. (TRANZIT TIME= CLOCK TIME – MARK TIME =0).
2. Ако се користи операнд А, тада он дефинише број параметра трансакције коме ће бити приписано време апсолутног сата. На тај начин можемо да меримо транзитно време кроз одређени сегмент модела. При томе MARK време трансакције се не мења.

*Примери:*

1. MARK      Постави транзитно време на нулу.
2. MARK 5    Вредност апсолутног сата припиши параметру 5
3. MARK 6    Вредност апсолутног сата припиши параметру 6.

Уколико желимо да табелирамо време опслуживања клијаната на шалтеру, онда користимо блок MARK на одговарајућим местима, а то се време може израчунати помоћу наредбе VARIABLE, на следећи начин:

VREME      VARIABLE P6-P5

Такође, можемо регистровати тренутке приступања трансакције на одређено место у блок дијаграму.

Уколико нас интересује време путовања трансакције од неког блока б1 до другог блока б2, тада је потребно одмах иза блока б1 ресетовати (анулирати) време путовања трансакције. То се постиже помоћу блока MARK

### 10.6.9 Генерисање случајних променљивих

Већина система који се симулирају помоћу GPSS-а садрже једну или више случајних активности. Углавном, ове активности се описују у виду статистичких расподела. На пример, многи системи масовног опслуживања претпостављају процес Пуасонових долазака. Да бисмо симулирали такве активности, модел мора да генерише случајне узорке из одговарајућих статистичких расподела.

Времена између долазака и времена опслуге могу да буду детерминистичка, униформно расподељена и могу да буду моделирана са неуниформним расподелама вероватноћа. Већина расподела треба да буде дефинисана у табеларном облику. Функције које за задају у GPSS-у имају само један аргумент и једини начин њиховог задавања је табеларни.

#### 10.6.9.1 Узимање узорака из популације

Генерисање случајних узорака се генерално извршава у два корака. Најпре се вредност извлачи из генератора случајних бројева који дају бројеве униформно расподељене у отвореном интервалу (0,1). Затим се, униформно расподељени случајни узорак пресликава у еквивалентну вредност из циљне расподеле.

Континуалне случајне променљиве могу да узимају било коју вредност у неком интервалу. Времена између долазака и времена опслуге обично су континуалне случајне променљиве.

Функција густине расподеле за популацију униформне и континуалне расподеле на интервалу од 0 до 1 има вредност 1, а ван тога има вредност нулу (у специјалном случају 0-1 униформна расподела функције густине формира квадрат).

Као подршку двокорачном процесу за узимање узорака потребно је да имамо извор случајних бројева униформно расподељених на интервалу 0-1. Такође, неопходно је да дефинишемо начин како да пресликамо случајну вредност из изворне униформне расподеле 0-1 у еквивалентну вредност из циљне популације.

### 10.6.9.2 Генератори униформних случајних бројева

У GPSS-у постоји осам идентичних токова генератора случајних бројева (RN1, RN2, ..., RN8). Они враћају вредности у опсегу од 0 до 999, осим када се користе као аргумент функције и тада дају вредности из популације униформно расподељене на отвореном интервалу 0-1 (отворени интервал је онај који не укључује крајње тачке).

Сваки од ових генератора има неидентичну подразумевану почетну тачку (семе). У GPSS-у се користи један алгоритам (процедура) за добијање униформних случајних бројева на интервалу 0-1 (најчешће Lehmer-ов мултипликативни конгурентни алгоритам). Математички речено, такав генератор је дефинисан као рекурентна релација. Алгоритам је детерминистички, тако да случајни бројеви које он ствара нису стварно случајни, већ су псеудослучајни, што значи да су поновљиви. Питање које се често поставља од стране нових корисника симулације је: зашто не генеришемо стварне случајне бројеве, коришћењем хардверског уређаја пре него технику детерминистичког генерисања псеудослучајних бројева ? Разлог за коришћење технике генерисања псеудослучајних бројева је то што она ствара поновљиве секвенце "случајних" узорака. Поновљивост случајних бројева може се сматрати као предност при планирању статистичких експеримената. Интерно представљање случајних бројева зависи од рачунара на којем се ради.

### 10.6.9.3 Транспарентно узимање узорака у блоковима GENERATE, ADVANCE и TRANSFER

Најједноставнији облик генерисања случајних величина јесте коришћење модификатора у виду интервала, тј. "проширења".

Операнди А и В код блокова GENERATE и ADVANCE могу се користити за специфицирање модификатора проширења у виду полуинтервала униформно расподељених времена између долазака (GENERATE) и времена опслуге (ADVANCE). Код оваквог коришћења блокова GENERATE и ADVANCE процес узимања узорака је транспарентан.

У општем облику блок GENERATE се може користити као

GENERATE A,B

У том случају, времена између долазака трансакција су из расподеле  $A \pm B$ . GPSS узима узорак транспарентно из  $A \pm B$  популације коришћењем генератора RN1 као изворне случајне бројеве, пресликавајући вредност RN1 у еквивалентно време између долазака  $t_d$  према следећој формули (T. J. Schriber, 1991):

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

У том случају вредност  $t_d$  се креће у границама  $(A - B) < t_d < (A + B)$ , што је конзистентно са чињеницом да су времена између долазака расподељена на отвореном интервалу  $A \pm B$ . Ако се ова формула дефинише као аритметички израз, тада се може користити било који од генератора случајних бројева.

Видимо да у случају транспарентног узимања узорака у блоковима GENERATE и ADVANCE, у свим таквим блоковима у моделу користи се генератор RN1. Када се врши транспарентно узимање узорака у два или више таква блока, сваки блок користи само подскуп вредности генерисаних од стране RN1 у посматраном временском опсегу.

Наредба TRANSFER, такође, користи ток случајних бројева RN1.

Формат статистичког блока TRANSFER је:

TRANSFER .A,B,C

Операнди A и B су обележја блокова у моделу. Операнд A указује на вероватноћу са којом се трансакције усмеравају ка блоку чије је обележје дефинисано у операнду C.

У циљу извршавања овог процеса, GPSS извлачи узорак из RN1 и онда користи узимање вредности узорка у циљу одређивања следећег блока у који ће бити упућена трансакција. На пример, претпоставимо да трансакција улази у следећи блок TRANSFER:

TRANSFER .250,BLOK1, BLOK 2

У том случају, позива се генератор RN1 и добијена вредност (то је троцифрени цео број у затвореном интервалу од 0 до 999) пореди се са 250. Ако је извучени број мањи од 250, трансакција се упућује



у блок чије је обележје BLOK 2, у супротном биће упућена у блок чије је обележје BLOK 1.

Као што видимо, у статистичким блоковима TRANSFER постоји транспарентна употреба генератора RN1, као и за транспарентно узимање узорака у блоковима GENERATE и ADVANCE.

#### Пример 4.

Возила долазе на техничку контролу. Последња провера односи се на рад мотора. Уколико је мотор неисправан, шаље се у одељење за поправку. После поправке возило се враћа поново на контролу. Након успешне провере возило иде на паркирање.

На контролном месту, на којем се врши контрола рада мотора, возила пристижу сваких  $10 \pm 5$  мин. Контролу возила врше два радника. Време провере возила је  $20 \pm 5$  мин. Од свих прегледаних возила њих 70% је исправно и одлазе на паркирање. Осталих 30% возила иду у одељење за поправку, где на поправци ради један радник. Време потребно за поправку возила износи  $45 \pm 15$  мин.

Потребно је написати програм на GPSS-у за анализу функционисања овог дела процеса.

Потребно је добити статистике редова возила на месту контроле као и на месту поправке. У том циљу потребно је табелирати:

- ♦ задржавања возила на контролном месту, при чему је горња граница првог интервала 5 мин, ширина интервала је 5 мин, а број класа је 5.
- ♦ задржавања возила на поправци, при чему је горња граница првог интервала 5 мин, ширина интервала је 10 мин, а број класа је 6.

Симулирати временски период од 8 часова, а за временску јединицу узети 0.1 мин.

GPSS/FON - Assembler Ver. 2.01, 1995

```
-----  
1|*  
2|* PROGRAM ZA TEHNIČKU KONTROLU VOZILA  
3|*  
4|  RAD    STORAGE 2          Definisanje kapac. sklad.
```

5	TAB1	TABLE M1,100,50,5	Definisanje tabele
6	TAB2	TABLE M1,300,100,5	Definisanje tabele
7		SIMULATE	Početak simulacije
8	1	GENERATE 100,50	Vozila dolaze na 100±50 v.j.
9	2 KONT	QUEUE RED1	Vozilo staje u red
10	3	ENTER RAD	Radnik vrši kontrolu vozila
11	4	DEPART RED1	Vozilo napušta red
12	5	MARK	Resetovanje tranzitnog vrem.
13	6	ADVANCE 200,50	Vreme kontrole 200±50 v.j.
14	7	LEAVE RAD	Radnik postaje slobodan
15	8	TABULATE TAB1	Tabeliranje vremena kont.
16	9	TRANSFER .300,,OPR	Preusmeravanje vozila
17	10	TERMINATE	Vozilo napušta sistem
18	11 OPR	QUEUE RED2	Red za popravku
19	12	SEIZE PERA	Početak popravke
20	13	DEPART RED2	Vozilo napušta red
21	14	MARK	Resetovanje tranzitnog vrem.
22	15	ADVANCE 450,150	Popravka vozila 450±150 v.j.
23	16	RELEASE PERA	Radnik postaje slobodan
24	17	TABULATE TAB2	Tabeliranje vremena popr.
25	18	TRANSFER ,KONT	Uputi vozilo na ponovnu kont.
26	*		
27	*	SEGMENT TAJMERA	
28	*		
29	19	GENERATE 4800	Generisanje Tajmera
30	20	TERMINATE 1	Umanji TB za 1
31		START 1	Početak simulacije
32		END	Kraj simulacije

Facility symbols and corresponding numbers

1: PERA

Storage symbols and corresponding numbers

1: RAD

Queue symbols and corresponding numbers

1: RED1

2: RED2

Table symbols and corresponding numbers

1: TAB1

2: TAB2

-----  
No errors detected

**Добијени су следећи резултати симулације:**

GPSS/FON Ver. 2.0, Simulating results

Relative clock	4800	Absolute clock	4800
Block counts			
Block Current	Total		
1	0	47	
2	8	56	
3	0	48	
4	0	48	
5	0	48	
6	2	48	
7	0	46	
8	0	46	

9	0	46
10	0	32
11	4	14
12	0	10
13	0	10
14	0	10
15	1	10
16	0	9
17	0	9
18	0	9
19	0	1
20	0	1

Storage	Capacity	Average Contents	Average Utilisation	Entries	Average Time/tran	Current Contents	Maximum Contents
1	2	1.916	0.958	48	194.489	2	2

Table 1

Entries in table	Mean argument	Standard deviation	Sum of arguments
46	197.457	31.883	9083.000

Upper limit	Observed frequency	Percent of total	Cumulative percentage	Cumulative remainder	Multiple of mean	Deviation from mean
100	0	0.000	0.000	100.000	0.506	-3.057
150	2	4.348	4.348	95.652	0.760	-1.488
200	22	47.826	52.174	47.826	1.013	0.080
250	22	47.826	100.000	0.000	1.266	1.648
300	0	0.000	100.000	0.000	1.519	3.216

Table 2

Entries in table	Mean argument	Standard deviation	Sum of arguments
9	469.000	85.910	4221.000

Upper limit	Observed frequency	Percent of total	Cumulative percentage	Cumulative remainder	Multiple of mean	Deviation from mean
300	0	0.000	0.000	100.000	0.640	-1.967
400	1	11.111	11.111	88.889	0.853	-0.803
500	4	44.444	55.556	44.444	1.066	0.361
600	4	44.444	100.000	0.000	1.279	1.525
700	0	0.000	100.000	0.000	1.493	2.689

Queue	Maximum contents	Average contents	Total entries	Zero entries	Percent zeros	Average time/trans	Current contents
1	10	4.868	56	5	8.929	414.768	8
2	4	2.442	14	1	7.143	836.077	4

Facility	Average utilisation	Number entries	Average time/tran	Seizing transact.	Preempting transaction
1	0.955	10	469.000	1	0

У овом примеру постоји транспарентно коришћење тока случајних бројева у наредбама чији су бројеви 1, 6, 9 и 15.

#### 10.6.9.4 Експоненцијална расподела

Експоненцијална случајна променљива, која је континуална, узима све вредности веће од нуле. Експоненцијална случајна променљива се често користи за моделирање времена између долазака када не постоји контрола над процесом долазака. Примери таквог процеса долазака су доласци клијената у пошту, бродова у луку, захтеви од интерактивних терминала корисника у рачунару и сл.

Експоненцијална расподела је потпуно дефинисана са њеном очекиваном вредношћу. Експоненцијална расподела се понекад назива негативна експоненцијална расподела. Узорак из униформног извора 0-1 може се пресликати у узорак из експоненцијалне расподеле са средњом вредношћу  $\mu$  на основу једначине:

$$t_e = \mu * [-\ln * (1 - RN_j)]$$

односно, како је  $(1-RN_j)$  такође случајни број у интервалу 0-1, то се горња једначина може користити у следећем облику:

$$t_e = \mu * [-\ln * (RN_j)]$$

где је  $\ln$  функција природног логаритма, а  $RN_j$  указује на ток генератора случајних бројева,  $j=1,2,3,\dots$

Експоненцијална расподела има високи степен варијабилности. Ово је конзистентно са релативно неконтролабилним процесом, као на пример, долазак клијената у пошту. Уствари, њена варијанса једнака је квадрату њене срење вредности. На пример, експоненцијална случајна променљива са средњом вредношћу 20 има варијансу од 400. Експоненцијална расподела има једну од највећих варијанси (у односу на њену средњу вредност) од општијих типова расподела.

Ако су опсервације времена између долазака расположиве, лако је урадити непосредну проверу да би се видело да ли могу да буду експоненцијално расподељене. Провера обухвата листинг времена између долазака у растућем редоследу и онда цртање ових времена између долазака у полулогаритамској размери према њиховом рангирању на линеарној скали. Ако су времена долазака

експоненцијално расподељена, тачке ће бити близу праве линије (Gordon, 1975).

Поред времена међудолазака, и времена опслуге су понекад експоненцијално расподељена. На пример, трајање неких врста телефонских разговора су експоненцијално расподељена. Такође, време функционисања појединих електронских компоненти (полупроводници, сијалице, и др.) експоненцијално је расподељено.

#### 10.6.9.5 Релације између експоненцијалних и Пуасонових случајних променљивих

Када дискутујемо о процесу долазака, могуће је да говоримо о временима између долазака, са једне стране, или износима долазака, са друге стране. На пример, ако процес долазака има средњу вредност времена између долазака 20 мин, тада је то повезано са средњим износом долазака од 3 на сат. У симулацији, наше интересовање је често усредсређено на времена између долазака, зато што је то расподела из које се узимају узорци за распоређивање будућих долазка.

Време између долазака случајне променљиве је обично континуално. У супротном, износ долазака случајне променљиве је дискретан. Може се показати да, када су времена између долазка експоненцијално расподељена, тада износ долазака повезан са њим следи Пуасонову расподелу.

Да би процес долазака био пуасоновски расподељен, морају да буду задовољене следеће претпоставке:

- ♦ Вероватноћа да се долазак дешава за време малог временског интервала пропорционална је величини интервала. На пример, вероватноћа доласка који се дешава за време следећих 5 с у пуасоновском процесу долазака је двапут веће од вероватноће доласка који се дешава за време следећих 25с.
- ♦ Вероватноћа два или више долазака који се дешавају за време довољно малог временског интервала је незнатно мала (последица ове претпоставке је да се симулациони доласци не дешавају по Пуасоновом процесу доласка. Другим речима, време између долазака никад није једнако нули).

- ♦ Времена између долазака су независна од свих других.

Пуасонови доласци чије је средње време између долазака 100 в.ј. могу се моделирати као:

```
EXPO FUNCTION RN1,C24          Експоненцијална raspodela
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/
.95,2.99/.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/0.995,5.3/
0.998,6.2/0.999,7/0.9998,8
-
-
GENERATE 100,FN$EXPO  Generisanje transakcija
```

Функција EXPO узима ток случајних бројева RN3 као аргумент. C24 указује да је функција континуална (C) и да је дефинисана у 24 тачке. Табелиране тачке дефинишу инверзну кумулативну експоненцијалну расподелу са средњом вредношћу 1. Овде је модификатор у виду функције (FN\$EXPO), па се времена између долазака трансакција добијају множењем дате средње вредности (100) са вредношћу функције (FN\$EXPO).

#### 10.6.9.6 Узимање узорак из Ерлангове расподеле

Узорак из Ерлангове расподеле се може добити сабирањем два или више узорак из експоненцијалне расподеле. Ерлангова расподела се може практично користити за моделирање времена опслуге. Ерлангова расподела је окарактерисана са два параметра: редом (параметром облика) и средњим временом опслуживања. Експоненцијална расподела је Ерлангова расподела првог реда. Искуство је показало да се у пракси најчешће срећу Ерлангове расподеле другог реда. Један од начина да узмемо узорак из Ерлангове расподеле  $k$ -тог реда са очекиваном вредношћу  $\mu$  јесте да формирамо суму од  $k$  узорак из експоненцијалне расподеле са очекиваном вредношћу  $\mu/k$ .

На пример, претпоставимо да је време опслуге окарактерисано Ерланговом расподелом другог реда ( $k = 2$ ) са очекиваном вредношћу 150 в.ј. Тада, овај временски период можемо симулирати са два блока ADVANCE:

```
ADVANCE 75, FN$EXPO
ADVANCE 75, FN$EXPO
```

Овде је очекивана вредност експоненцијалне случајне променљиве 75 в.ј., што је половина од очекиване вредности Ерлангове расподеле (150 в.ј.). Ова два блока ADVANCE се могу заменити једним блоком ADVANCE и једним аритметичким изразом:

```

ERLANG    VARIABLE 75*(FN$EXPO+FN$EXPO)
-
-
-
ADVANCE V$ERLANG

```

Варијанса Ерлангове случајне променљиве  $k$ -тог реда са очекиваном вредношћу  $\mu$  је  $(1/k)$ -ти део варијансе за експоненцијалну случајну променљиву са истом очекиваном вредношћу. На пример, ако време опслуживања има дату очекивану вредност и следи Ерлангову расподелу другог реда, њена варијанса биће једнака половини варијансе која би се добила у случају да је време опслуживања експоненцијално расподељено са истом очекиваном вредношћу.

#### 10.6.9.7 Нормална расподела

Нормална случајна променљива, која је континуална и симетрична, са освртом на њену очекивану вредност, узима вредности у опсегу од минус бесконачно до плус бесконачно. Нормална расподела је потпуно окарактерисана са очекиваном вредношћу  $\mu$  њеном стандардном девијацијом  $\sigma$ .

Нормална случајна променљива са очекиваном вредношћу 0.0 и стандардном девијацијом (или варијансом) 1.0 назива се стандардизована нормална случајна променљива. Ретко је да посматрамо нормалне вредности иза три стандардне девијације од средње вредности, зато што је скоро читава популација (99.7%) укључена унутар тог опсега. Вредност из било које нормалне расподеле може да буде стандардизована. На пример, ако је  $x$  вредност извучена из нормалне расподеле са средњом вредношћу  $\mu$  и стандардном девијацијом  $\sigma$ , вредност из стандардизоване нормалне расподеле добија се на основу израза:

$$z = \frac{x - \mu}{\sigma}.$$

На основу централне граничне теореме, расподела суме (средња вредност) низа независних идентично расподељених случајних променљивих из било које расподеле прилази нормалној за велики број чланова у суми. У већини случајева, централна гранична теорема је задовољена када је обухваћена сума од 30 опсервација (или мање).

Нормалној расподели подвргавају се времена опслуге, као што је време потребно за израду неког производа. Слично, време за пренос поруке у комуникационом систему може да буде нормално расподељено.

Метод који се често користи за узимање узорака из нестандардне нормалне расподеле је да конвертујемо узорак из униформне расподеле 0-1 у вредност из стандардне расподеле, а онда да конвертујемо вредност из стандардне расподеле у еквивалентну вредност из нестандардне нормалне популације на основу израза:

$$x = \sigma * z + \mu$$

Времена између долазака, као и времена опслуге не могу да буду негативна, те се узимање узорака из нормалне расподеле мора користити са опрезношћу у том погледу. На пример, ако се време опслуге покова нормалној расподели са средњом вредношћу 150 в.ј. и стандардном девијацијом 25 в.ј, то се може моделирати на следећи начин:

```
SNORM FUNCTION RN2,C25          Standardizovana normalna rasp.
0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2/.06681,-1.5/
.11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6/.34458,-.4/
.42074,-.2/.5,0/.57926,.2/.65542,.4/.72575,.6/.78814,.8/.84134,1/
.88493,1.2/.93319,1.5/.97725,2/.99379,2.5/.99865,3/.9997,4/1,5

NRASP VARIABLE 150+25*FN$SNORM
-
-
-
ADVANCE V$NRASP
```

Функција SNORM узима ток случајних бројева RN2 као аргумент. C25 указује да је функција континуална (C) и да је дефинисана у 25 тачака. Табелиране тачке дефинишу инверзну кумулативну стандардизовану расподелу са средњом вредношћу 0 и стандардном девијацијом 1. Аритметички израз (FN\$NRASP)



трансформише узорак стандардизоване нормалне расподеле у узорак циљне нормалне расподеле са  $\mu = 150$  и  $\sigma = 25$ .

#### 10.6.9.8 Статистичка независност у симулацији

Уколико промене које се дешавају у једном делу модела не утичу на понашање других делова модела, тада извори случајности у моделу морају да буду статистички независни. Посматрајмо систем једноканалног опслуживања, на пример, шалтер за пријем пакета у пошти (пример 2). Питање статистичке независности је од важности чак и у овом простом моделу, када су у питању униформне расподеле. Утицај статистичке независности је знатно уочљивији када су у питању други облици расподела (на пример експоненцијална).

##### Пример 5.

Посматраћемо једну јавну телефонску говорницу. Доласци клијената су пуасоновски са средњим временом између долазака 120 s. Време опслуге (трајање разговора) је експоненцијално расподељено са средњом вредношћу од 100 с. Када клијент дође, уколико је говорница заузета, он стаје у ред. Потребно је добити статистике везане за ред, као и за телефонску говорницу. Симулирати временски период од 18 часова, а за временску јединицу узети 1 с.

Најпре ћемо написати програм коришћењем истог тока псеудослучајних бројева RN3.

\* SIMULACIJA RADA JEDNE JAVNE TELEFONSKE GOVORNICE

\*

SIMULATE

\*

EXPO FUNCTION RN3,C24                      Експоненцијална raspodela  
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/  
.75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/  
.95,2.99/.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/0.995,5.3/  
0.998,6.2/0.999,7/0.9998,8

\*

GENERATE 120,FN\$EXPO	Dolazak klijenata
QUEUE 1	Klijent pristupa redu
SEIZE 1	Klijent ulazi u govornicu
DEPART 1	Klijet napušta red
ADVANCE 100,FN\$EXPO	Trajanje razgovora
RELEASE 1	Klijent oslobađa govornicu
TERMINATE	Klijent napušta sistem

```

*
* SEGMENT TAJMERA
*
    GENERATE 64800      Generisdanje Tajmera
    TERMINATE 1        Umanji TB za 1
    START 1            Početak simulacije
    END                Kraj simulacije
    
```

Добијени су следећи резултати симулације:

GPSS/FON Ver. 2.0, Simulating results

Relative clock      64800   Absolute clock      64800

Block counts

Block Current Total

1	0	517
2	0	517
3	0	517
4	0	517
5	0	517
6	0	517
7	0	517
8	0	1
9	0	1

Queue	Maximum contents	Average contents	Total entries	Zero entries	Percent zeros	Average time/trans	Current contents
1	9	1.470	517	157	30.368	183.888	0

Facility	Average utilisation	Number entries	Average time/tran	Seizing transact.	Preempting transaction
1	0.734	517	91.890	0	0

Овде су доласци клијената и трајање разговора случајне променљиве. Експериментисање са моделом највероватније ће укључити испитивање ефеката променљивог износа долазака, времена опслуге, или оба. Ако су обе случајне променљиве генерисане коришћењем истог тока псеудослучајних бројева, тада ће промена било које од њих унети промену код обе случајне променљиве. Због тога што се исти ток случајних бројева користи за узимање два типа узорака, промена код износа долазака или код времена опслуге утицаће на процес узимања узорака за обе случајне променљиве. У датом примеру, употреба два тока независних псеудослучајних бројева (на пример, RN3 и RN7) даће веродостојније резултате. У том случају, износ долазака и време опслуге могу се мењати независно једно од другог. Тиме се

омогућава експериментисање са системом који се моделира на адекватнији начин. У овом случају програм би изгледао овако:

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

*Добијени су следећи резултати симулације:*

GPSS/FON Ver. 2.0, Simulating results

Relative clock      64800   Absolute clock      64800

Block counts

Block Current Total

1	0	540
2	4	540
3	0	536
4	0	536
5	1	536
6	0	535
7	0	535
8	0	1
9	0	1

Queue	Maximum contents	Average contents	Total entries	Zero entries	Percent zeros	Average time/trans	Current contents
-------	------------------	------------------	---------------	--------------	---------------	--------------------	------------------

1	15	3.031	540	104	19.259	364.284	4
---	----	-------	-----	-----	--------	---------	---

Facility	Average utilisation	Number entries	Average time/tran	Seizing transact.	Preempting transaction
1	0.825	536	99.845	1	0

Упоредивањем резултата у оба случаја могу се уочити знатне разлике, што указује на значајну статистичку зависност код узимања узорака у првом случају.

Такође, и код транспарентног узимања узорака настаје статистичка зависност. Ако посматрамо пример 4, уочићемо да се наредбе са бројевима 1, 6, 9 и 15 експлицитно позивају на исти ток псеудослучајних бројева RN1. Ако је потребна статистичка независност, употреба ове особине у више од једне тачке у моделу мора да се избегне, јер се у датом случају све наредбе имплицитно позивају на ток псеудослучајних бројева RN1.

Да бисмо избегли статистичку зависност у датом примеру, потребно је да користимо следећи програм, у којем се добијају веродостојнији симулациони резултати.

\* PROGRAM ZA TEHNIČKU KONTROLU VOZILA

\*

	SIMULATE	
DOLAZ	VARIABLE (100-50)+(RN7*(2*50))/1000	Dolasci
USLUG1	VARIABLE (200-50)+(RN4*(2*50))/1000	Kontrola
USLUG2	VARIABLE (450-150)+(RN3*(2*150))/1000	Popravka
RAD	STORAGE 2	Definisanje kapaciteta sklad.
TAB1	TABLE M1,100,50,5	Definisanje tabele
TAB2	TABLE M1,300,100,5	Definisanje tabele
*		
KONT	GENERATE V\$DOLAZ	Vozila pristižu svakih 100(50 v.j.
	QUEUE RED1	Vozilo staje u red
	ENTER RAD	Radnik vrši kontrolu vozila
	DEPART RED1	Vozilo napušta red
	MARK	Resetovanje tranzitnog vremena
	ADVANCE V\$USLUG1	Vreme kontrole 200(50 v.j.
	LEAVE RAD	Radnik postaje slobodan
	TABULATE TAB1	Tabeliranje vremena kontrole
	TRANSFER .300,,OPR	Preusmeravanje vozila
	TERMINATE	Vozilo napušta sistem
OPR	QUEUE RED2	Vozilo staje u red za popravku
	SEIZE PERA	Radnik počinje sa popravkom
	DEPART RED2	Vozilo napušta red
	MARK	Resetovanje tranzitnog vremena
	ADVANCE V\$USLUG2	Vreme popravke 450(150 v.j.
	RELEASE PERA	Radnik postaje slobodan
	TABULATE TAB2	Tabeliranje vremena popravke
	TRANSFER ,KONT	Uputi vozilo na popravku

```
*  
*      SEGMENT TAJMERA  
*  
      GENERATE 4800      Generisanje Tajmer transakcije  
      TERMINATE 1        Umanji TB za 1  
      START 1            Početak simulacije  
      END                Kraj simulacije
```

#### 10.6.9.9 Остале наредбе GPSS језика

У претходним поглављима дат је краћи приказ основних наредби GPSS језика. Већи број других наредби стоји на располагању свима који програмирају у GPSS-у, али синтакса и семантика тих наредби по свом обиму превазилази оквире ове књиге, те неће бити изложена. Напоменимо укратко да GPSS између осталог омогућава и рад са меморијским локацијама, аритметичким изразима, корисничким редовима, као и дефинисање различитих функција, индиректно адресирање, задавање параметара трансакцијама, и др. У циљу нумеричке обраде коју захтевају сложенији модели, GPSS процесор омогућује приступ разним интерним варијаблама симулатора, атрибутима трансакција и перманентних ентитета, које заједнички називамо стандардним нумеричким атрибутима (СНА) (Б. Раденковић, А. Марковић, 1992, 1995; T. J. Schriber, 1974; T.M. O' Donovan, 1979).

#### 10.6.9.10 Симулација телефонске централе на GPSS -у

Поступак симулације дискретних система на GPSS-у приказаћемо на примеру симулације телефонске централе. Поред наредби GPSS језика које су детаљно обрађене у претходним поглављима, у овом примеру су коришћене и друге GPSS наредбе, чија синтакса није дата у овој књизи.

Претпоставимо да је потребно извршити симулацију рада једне аутоматске телефонске централе. На централу је везан већи, али ограничен број претплатника (линија). Ради сажетости, уводимо следећу терминологију:

- ♦ позивалац је претплатник који покушава да успостави везу, тј. позива неки број;
- ♦ позвани је претплатник са којим позивалац покушава да успостави везу.

Веза између две линије (претплатника) у централи се остварује преко канала, чији је број такође ограничен, али знатно мањи од броја претплатника. Постојање слободног канала је услов да би се остварила веза између две линије.

За дати пример уводимо следеће полазне претпоставке:

- ◆ позивалац не може позвати сам себе,
- ◆ ако је позвани број заузет, не долази до успостављања везе и сматра се да је позивалац спустио слушалицу, односно да је његов број слободан,
- ◆ свака успостављена веза заузима један канал,
- ◆ ако нема слободних канала, не долази до успостављања везе и сматра се да су и позивалац и позвани број слободни,
- ◆ ако је позвани број слободан и има слободних канала, долази до заузимања канала и успостављања везе,
- ◆ кад се успостави веза, подразумева се да су и позивалац и позвани број заузети,
- ◆ сматра се да трајање телефонског разговора има експоненцијалну расподелу,
- ◆ по завршетку разговора долази до ослобађања линије позиваоца и позваног, као и канала који се користио за ту везу.

Улазни подаци за модел који могу да се мењају су:

- ◆ број телефонских претплатника у систему телефонске централе,
- ◆ број канала који су на располагању за успостављање веза,
- ◆ просечно време трајања разговора,
- ◆ просечно време између два позива у систему и
- ◆ број покушаних позива које треба симулирати.

Као резултат, симулација треба да обезбеди одговарајући скуп податка.

Ток симулације:

- ◆ порука о сваком покушаном позиву,
- ◆ порука о свакој успостављеној вези,
- ◆ порука о неуспостављеним везама због заузетости канала,
- ◆ порука о неуспостављеним везама због заузетости позиваног,

- ♦ порука о сваком завршетку разговора.

Преглед одабраних параметара модела:

- ♦ број линија, тј. телефонских претплатника,
- ♦ број расположивих канала у централи,
- ♦ број позива за које се врши симулација,
- ♦ просечно време трајања разговора,
- ♦ просечно време између два позива у систему.

Статистика симулације:

- ♦ укупан број позива који су симулирани,
- ♦ број неуспелих позива због заузетости канала,
- ♦ број неуспелих позива због заузетости линија,
- ♦ број позива који су у току,
- ♦ време за које су обављени сви позиви, тј. време симулације.

Параметри модела који се симулира су следећи:

- ♦ максимални број линија (MAXL) је 100,
- ♦ максимални број канала (MAXK) је 20,
- ♦ средње време између два позива (TPOZ) је 15 s,
- ♦ средње време трајања разговора (TRAZ) је 150 s.

Метод изградње GPSS програма је следећи:

- ♦ свакој телефонској линији се додељује један прекидач (SWITCH), при чему положај прекидача "укључен" означава да је линија заузета;
- ♦ каналима се додељује једно складиште капацитета двадесет;
  - ♦ уводи се један тип трансакција са два параметра:
    - ♦ p1 - број позиваоца,
    - ♦ p2 - позвани број;
    - ♦ временска јединица је 1 секунда;
    - ♦ симулација траје 1000 позива искључујући прелазно стање 50 позива.

SIMULATE

\*

EXPO1 FUNCTION RN2,C24

Експоненцијална raspodela

0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/  
 .75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/  
 .95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/  
 .999,7/.9998,8

EXPO2 FUNCTION RN3,C24 Eksponecijalna raspodela

0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/  
 .75,1.38/.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/  
 .95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/  
 .999,7/.9998,8

*	GENERATE X\$TPOZ, FN\$EXPO1	GENERISANJE POZIVA
	TEST G V\$LIN, 2, ODB	POZIV SE ODBACUJE AKO JE BROJ
*		SLOBODNIH LINIJA MANJI OD 2
BROJ1	ASSIGN 1, V\$UNIF	ODREDJIVANJE BROJA POZIVAOCA
	GATE LR *1, BROJ1	DA LI JE POZIVALAC ZAUZET
	LOGIC S *1	POZIVALAC POSTAJE ZAUZET
BROJ2	ASSIGN 2, V\$UNIF	ODREDJIVANJE POZIVNOG BROJA
	TEST NE P1, P2, BROJ2	TESTIRANJE DA LI JE P1 <> P2
	GATE SNF KANAL, ZKAN	AKO IMA SLOB. KANALA ZAUZMI KAN.
UZMIK	ENTER KANAL	ZAUZIMANJE KANALA
	GATE LR *2, ZLIN	DA LI JE POZIVNI BROJ SLOBODAN
	LOGIC S *2	ZAUZMI POZIVNI BROJ
	ADVANCE X\$TRAZ, FN\$EXPO2	TRAJANJE RAZGOVORA
	LOGIC R *1	POZIVAO POSTAJE SLOBODAN
	LOGIC R *2	POZVANI POSTAJE SLOBODAN
	LEAVE KANAL	OSLOBADJA SE JEDAN KANAL
	SAVEVALUE USPOZ+, 1	REGISTR. BROJA USPEŠNIH POZ.
	TERMINATE 1	UKLANJANJE TRANSAKCIJE
*		
ODB	SAVEVALUE ODPOZ+, 1	REGISTR. BROJA ODBAČENIH POZIVA
	TERMINATE	UKLANJANJE TRANSAKCIJE
*		
ZKAN	LOGIC R *1	AKO NEMA SLOB. KANALA POZIVALAC
*		OSTAVLJA SLUŠALICU
	SAVEVALUE NUPOZ+, 1	REGISTR. BROJA NEUSPEŠNIH POZIVA
*		ZBOG ZAUZETOSTI KANALA
	TERMINATE 1	UKLANJANJE TRANSAKCIJE
ZLIN	LOGIC R *1	POZIVALAC OSTAVLJA SLUSALICU JER
*		JE POZVANI BROJ BIO ZAUZET
	LEAVE KANAL	OSLOB. KANALA; POZVANI BROJ JE
*		BIO
ZAUZET		
	SAVEVALUE NPOZB+, 1	REGISTR. BROJA NEUSPESNIH POZ.
*		ZBOG ZAUZETOSTI LINIJE POZVANOG
	TERMINATE 1	UKLONI TRANSAKCIJE (NEUSP. POZ.)
*		
KANAL	STORAGE 20	BROJ KANALA
*		
LIN	VARIABLE X\$MAXL-S\$KANAL*2-2	BROJ SLOBODNIH LINIJA
UNIF	VARIABLE X\$MAXL*RN1/1000+1	IZABIRANJE LINIJE
*		
	INITIAL X\$MAXL, 100	CENTRALA IMA 100 LINIJA
	INITIAL X\$TRAZ, 150	SREDNJE VREME TRAJANJA RAZ.
	INITIAL X\$TPOZ, 10	SREDNJE VREME IZMEDJU POZ.
*		
	START 50, NP	50 POZIVA PRELAZNI REŽIM
	RESET	RESETOVANJE SIMULATORA



```

INITIAL X$USPOZ,0
INITIAL X$ODPOZ,0
INITIAL X$NUPOZ,0
INITIAL X$NPOZB,0
*
START 1000                                SIMULIRATI 1000 POZIVA
END

```

*Добијени су следећи резултати симулације:*

GPSS/FON Ver. 2.0, Simulating results

Relative clock      9623   Absolute clock      10203

Block counts

Block	Current	Total
1	0	1002
2	0	1002
3	0	1336
4	0	1336
5	0	1002
6	0	1013
7	0	1013
8	0	1002
9	0	999
10	0	999
11	0	765
12	13	776
13	0	763
14	0	763
15	0	763
16	0	763
17	0	763
18	0	0
19	0	0
20	0	3
21	0	3
22	0	3
23	0	234
24	0	234
25	0	234
26	0	234

Storage	Capacity	Average Contents	Average Utilisation	Entries	Average Time/tran	Current Contents	Maximum Contents
1	20	11.896	0.595	1010	113.341	13	20

SaveValues

X\$1 = 100

X\$2 = 150

X\$3 = 10

X\$4 = 763

X\$6 = 3

X\$7 = 234

Симулацијом су добијени следећи резултати: од 1000 позива, број успешних позива био је 763, број неуспешних позива због заузетости канала био је 3, а неуспешних позива због заузетости линије било је 234.

## ДИНАМИКА СИСТЕМА И ПРОГРАМСКИ ЈЕЗИК SDS

Динамика система (*system dynamics*) је методологија истраживања, моделирања и симулације сложених динамичких система. Овом методологијом најчешће се моделирају економски, друштвени и биолошки системи, али се успешно може применити и у другим областима. Ниво посматрања система који се моделира, а потом симулира, може се значајно разликовати од случаја до случаја. На пример, од модела нивоа неког сегмента предузећа, затим модела целог предузећа, преко модела националне економије, све до модела светске популације или екологије. По правилу, овакви системи нису континуалног карактера, већ мењају своја стања кроз већи број појединачних дискретних догађаја. У динамици система, дискретни догађаји се посматрају у агрегираном облику, тако да се могу описати преко континуалних токова, а промене у систему се апроксимирају диференцијалним једначинама.

### 11.1 Историјски преглед динамике система

У наставку је укратко дат историјат настанка динамике система као научне дисциплине (*Munitić, 1989*).

Динамика система се заснива на системском прилазу проблемима управљања сложеним динамичким системима. Један од пионира таквог приступа је Ludwig von Bertalanffy (1930), који је применио принципе структуре система у проучавању биолошких организама. У исто време у САД је почео развој системског инжењерства, првенствено за потребе тадашње војне индустрије. Значај ових радова

огледа се, пре свега, у уочавању аналогних начина понашања појединих биолошких и техничких система. Године 1948. Norbert Wiener објављује свој рад "Cybernetics" у коме објашњава сличности и разлике регулације и управљања код биолошких, техничких, друштвених и економских система.

Као научна дисциплина, динамика система је почела да се развија на Massachusetts Institute of Technology, 1961. године, под руководством проф. Jay Forrester-a, који је први применио широке принципе кибернетике у истраживању функционисања индустријских система. Циљ ових истраживања било је предузеће као динамички систем, због чега је ова дисциплина првобитно и названа индустријска динамика – дисциплина која изучава силе које делују и кретања која оне изазивају у предузећу (Forrester, 1961). Тек касније, због проширења подручја примене, ова дисциплина добија свој данашњи назив динамика система.

## 11.2 Системи са и системи без повратног дејства

Основне класе система који се моделирају и симулирају у динамици система су системи са повратним дејством. Уколико у неком систему постоје релације између елемената система такве да један елемент посредно, преко других елемената утиче сам на себе, кажемо да у таквом систему постоји повратно дејство (*feedback loop*). У супротном, реч је о систему без повратног дејства (Рајков, 1988). Другим речима повратно дејство је затворени круг узрока и последица који утиче на то да неки почетни узрок има индиректан утицај на самога себе. Постојање једног или више кола повратног дејства у систему изазива његово сложено понашање у времену.

Систем без повратног дејства карактерише се тиме што његов излаз зависи од улаза, али улаз не зависи од излаза. У системима ове класе, прошли догађаји не утичу на будуће и сам систем не контролише своје понашање, односно они функционишу без сазнања о резултатима које остварују. Тачност функционисања ових система одређена је само квалитетом међусобне зависности елемената који их сачињавају. Код ових система не јављају се проблеми везани за нестабилност који се

јављају код система са повратним дејством. Већина техничких система припада овој класи.

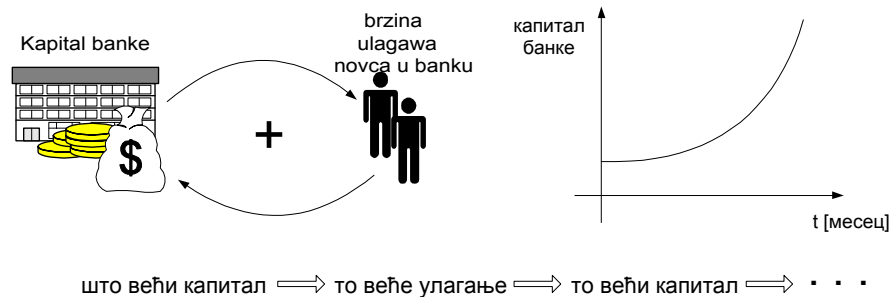
Систем са повратним дејством је онај чији улаз зависи или је контролисан ранијим догађајима у њему, односно зависи, на неки начин, од излаза или неког елемента система који се контролише. Систем са повратним дејством има структуру у којој постоји барем једно коло повратног дејства кроз које прошле активности контролишу будуће (*Рајков, 1988*).

### 11.3 Поларитет кола повратног дејства

Повратно дејство може бити позитивно и негативно. Повратно дејство које повећава утицај поремећаја на улаз назива се позитивно. Позитивно повратно дејство значи да неки узрок, преко ланца последица доводи до промена увек у истом смеру. Тако се сталним повратним деловањем постиже стални раст или стално смањење тог узрока (*Черић, 1993*). Појава кола позитивног повратног дејства у систему је последица природе међусобне зависности елемената система. Акције унутар кола позитивног повратног дејства стално повећавају разлику између стварног и жељеног стања система. Ова кола у систему имају улогу у остваривању циљева који се односе на раст и развој.

Једноставан пример позитивног кола повратног дејства је оно у коме се прати утицај брзине улагања новца на величину капитала банке. Претпоставимо да у почетном тренутку посматрања банка располаже одређеним капиталом. Висина тог капитала одређује положај банке на тржишту новца. Већи капитал значи предност за банку јер привлачи улагаче, на пример преко величине каматне стопе коју банка одређује или преко гаранција коју банка пружа у погледу сигурности улога. Уз одређену стопу улагања, могуће је за дати (почетни) капитал банке одредити количину новца која ће бити уложена у банку у посматраној јединици времена (нпр. месец), која ће повећати почетни капитал банке. Због повећаног капитала банке, на почетку наредног месеца, количина уложеног новца у том месецу биће још већа, што ће даље повећати капитал банке, итд.

На слици 11.1 приказано је описано повратно дејство и одговарајући дијаграм раста капитала банке у времену.



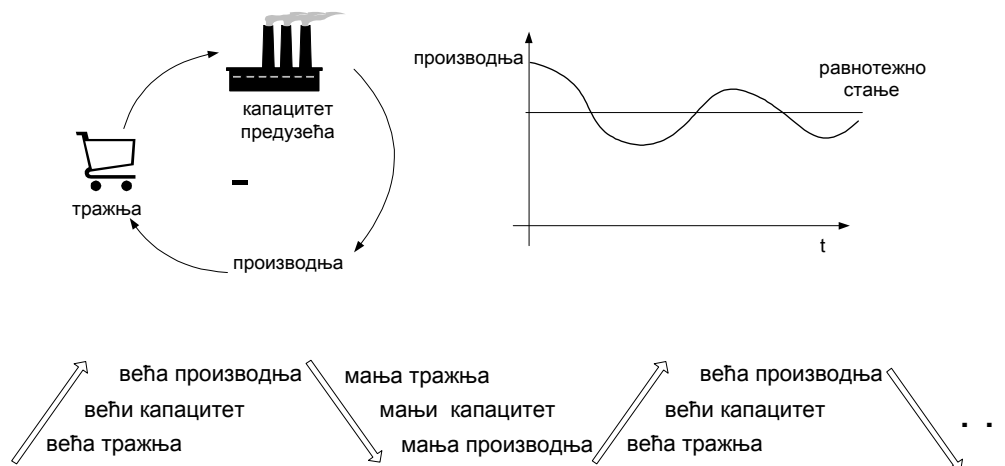
Слика 11.1 Позитивно коло повратног дејства

Негативно повратно дејство је оно које смањује утицај поремећаја на улаз. Оно омогућава успостављање равнотежног стања у систему, када је оно нарушено неким поремећајем. Негативно повратно дејство заправо значи да неки узрок, преко ланца последица доводи до промене смера сопственог деловања. Уколико посматрани узрок порасте изнад неког равнотежног стања, повратно дејство ће смањити тај узрок и обратно, када узрок постане мањи од равнотежног стања, тада ће повратно дејство проузроковати повећање тог узрока. На тај начин позитивно коло повратног дејства "стабилизује" узрок око неког равнотежног стања (Черић, 1993).

Функционисање негативног повратног дејства објаснићемо на једноставном примеру у коме се посматра узајамни утицај тражње за производима неког предузећа, капацитета предузећа и производње.

Повећана тражња на тржишту за производима предузећа утиче на повећање улагања у проширење капацитета предузећа. Већи капацитети омогућавају већу производњу тражених производа, а већа производња брже задовољава тражњу, која се даље смањује. Смањење тражње у наредном циклусу посматрања, условиће мање ангажовање расположивих капацитета, односно мању производњу, што ће поново довести до веће тражње, итд.

Описано повратно дејство и одговарајуће понашање система у времену (величина производње) приказани су на слици 11.2.



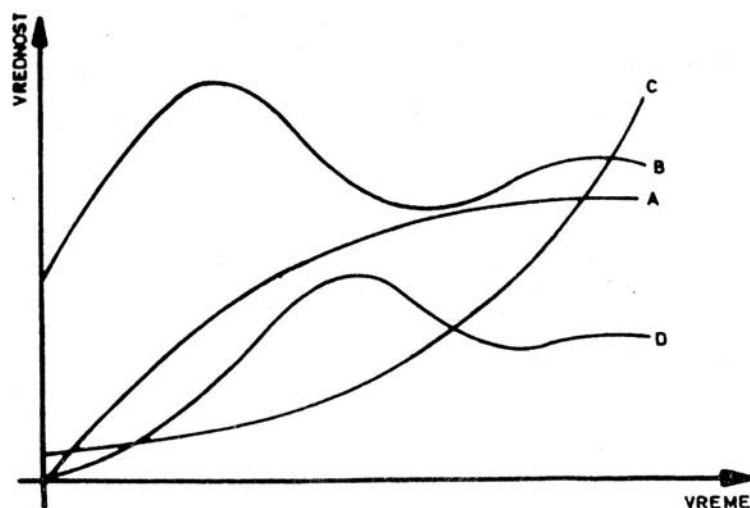
Слика 11.2 Негативно коло повратног дејства

Описана повратна дејства по правилу се не остварују у кратким ланцима узрока и последица, какви су приказани у претходним примерима. У реалном систему, она заправо настају у знатно дужем и сложенијем низу узрока и последица. Такође, реални системи најчешће садрже мешавину позитивних и негативних кола повратног дејства која одређују њихово понашање у времену. Њихова анализа подразумева одређивање оних фактора који доминантно утичу на понашање система.

На слици 11.3 приказани су неки карактеристични облици понашања система са повратним дејством (Рајков, 1988).

Крива **A** је типична крива за једноставнији систем са повратним дејством код кога променљива расте са опадањем величине прираштаја (однос величине излаза и улаза). После извесног времена прираштај тежи нули и крива се асимптотски приближава некој вредности. Пример оваквог понашања је раст у запошљавању. У систему доминира негативно коло повратног дејства.

Крива **B** описује сложеније достизање коначне вредности (осциловање око коначне вредности). Може настати због претераног кашњења у колу повратног дејства или због нагле корекције разлике између стварног и жељеног стања. Примери оваквог понашања су: раст или опадање индустријске производње унутар економских циклуса и промена цене условљена законом понуде и потражње. Системом доминира негативно коло повратног дејства.



Слика 11.3 Неки типични облици понашања система са повратним дејством

Крива **C** приказује експоненцијални раст. Примери оваквог понашања су: размножавање ћелија и раст продаје који зависи само од броја продаваца. Доминантан утицај има позитивно коло повратног дејства.

Крива **D** приказује почетни експоненцијални раст, а потом флукуације у достизању неке вредности. Примери оваквог понашања су: понашање броја животиња и раст новог производа. У почетку у систему доминира позитивно, а потом негативно коло повратног дејства.



## 11.4 Основни појмови система са повратним дејством

За описивање система са повратним дејством, у динамици система употребљавају се следећи основни појмови:

- ◆ стања система
- ◆ промене стања система
- ◆ кашњења

*Стања система (level - ниво)* представљају стања ресурса, односно различите акумулације ресурса. Стања су мерљиве величине, а њихове димензије изражавају се у јединицама ресурса. Неки примери стања система су: број становника, количина воде у резервоару, количина робе на залихама, новац на рачуну итд. Стањима система се може управљати и на тај начин утицати на брзину њихове промене и на њихово равнотежно стање. Своју вредност, ове величине имају и када систем не функционише.

*Промене стања система (rate – брзина)* или *функције одлучивања* су акције или токови који доводе до повећања или смањења стања система, у јединици времена. То су променљиве које показују брзину претварања ресурса из једног у друго стање. У динамици система, ове се величине посматрају као просечне брзине промене у неком периоду времена, а изражавају се у јединицама ресурса у јединици времена.

*Кашњења* представљају консеквенце промена које се не дешавају истовремено када су покренуте. Наиме, у техничким, економским, друштвеним али и у већини других система, промене се не дешавају истовремено када су покренуте, већ је неопходно да протекне одређено време (кашњење) да би се промена изазвана на једном елементу у систему пренела на други елемент, који је са њим повезан. Један део кашњења се односи на материјална кашњења, а други на кашњења информација. На пример, време које протекне од момента поручивања робе од добављача до момента када та роба стигне до складишта поручиоца представља класичан пример кашњења. На нека кашњења се може утицати (нпр. на време обраде поруџбине), а на нека не (нпр. на време које је потребно да пошта достави поруџбину). Кашњења у систему могу имати значајне последице на понашање система, и она по правилу изазивају осцилације у понашању система.

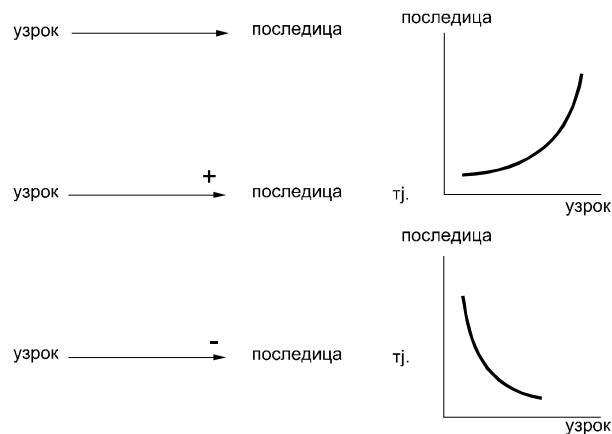
У већини случајева када је могуће смањити кашњења, смањиће се и осцилације и систем ће радити уједначеније.

## 11.5 Концептуални модели динамике система

Концептуални модели омогућавају експлицитни приказ елемената система и релације које између њих постоје у графичком облику. На тај начин они олакшавају развој симулационих (рачунарских) програма и повећавају њихову поузданост. У динамици система користе се две класе концептуалних модела: диаграми узрочних петљи и дијаграми токова (Черић, 1993).

### 11.5.1 Дијаграми узрочних петљи

У анализи система са повратним дејством уобичајено је да се најпре одреде везе између елемената система и да се идентификују врсте тих веза, а потом идентификују повратна дејстава у систему и одреди њихов поларитет. Најједноставније је да се то изрази у графичком облику. За приказивање узрочно последичних веза између елемената система са повратном дејством у динамици система користе се *дијаграми узрочних петљи*. Графички симболи ових дијаграма приказани су на слици 11.3.



Слика 11.3 Приказивање релација у дијаграмима узрочних веза

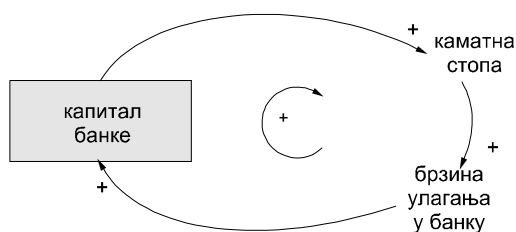
Линија са стрелицом означава смер везе између узрока и последице. Ако је на крају стрелице знак плус, то значи да се узрок и последица мењају у истом смеру, а ако је на крају стрелице знак минус, тада се узрок и последица мењају у супротном смеру.

Полукружна стрелица у средини петље са знаком плус означава позитивно повратно дејство, а са знаком минус негативно повратно дејство.

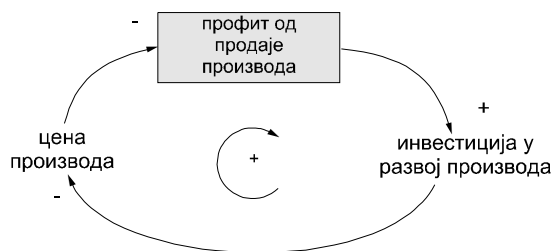
Код сложенијих кола, тип повратног дејства се најлакше може одредити на следећи начин:

Само позитивне везе:	позитивно повратно дејство
Паран број негативних веза:	позитивно повратно дејство
Непаран број негативних веза:	негативно повратно дејство

На слици 11.4 приказано је неколико једноставнијих примера дијаграма узрочних петљи код система са позитивним повратним дејством.



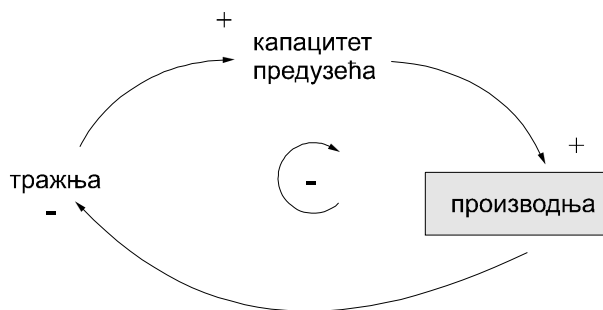
(а) Пораст капитала банке везан за пораст каматне стопе



(б) Предузеће које инвестира у развој производа

Слика 11.4 Неки примери позитивног повратног дејства

На слици 11.5 приказан је пример дијаграма узрочних петљи код система са негативним повратним дејством.



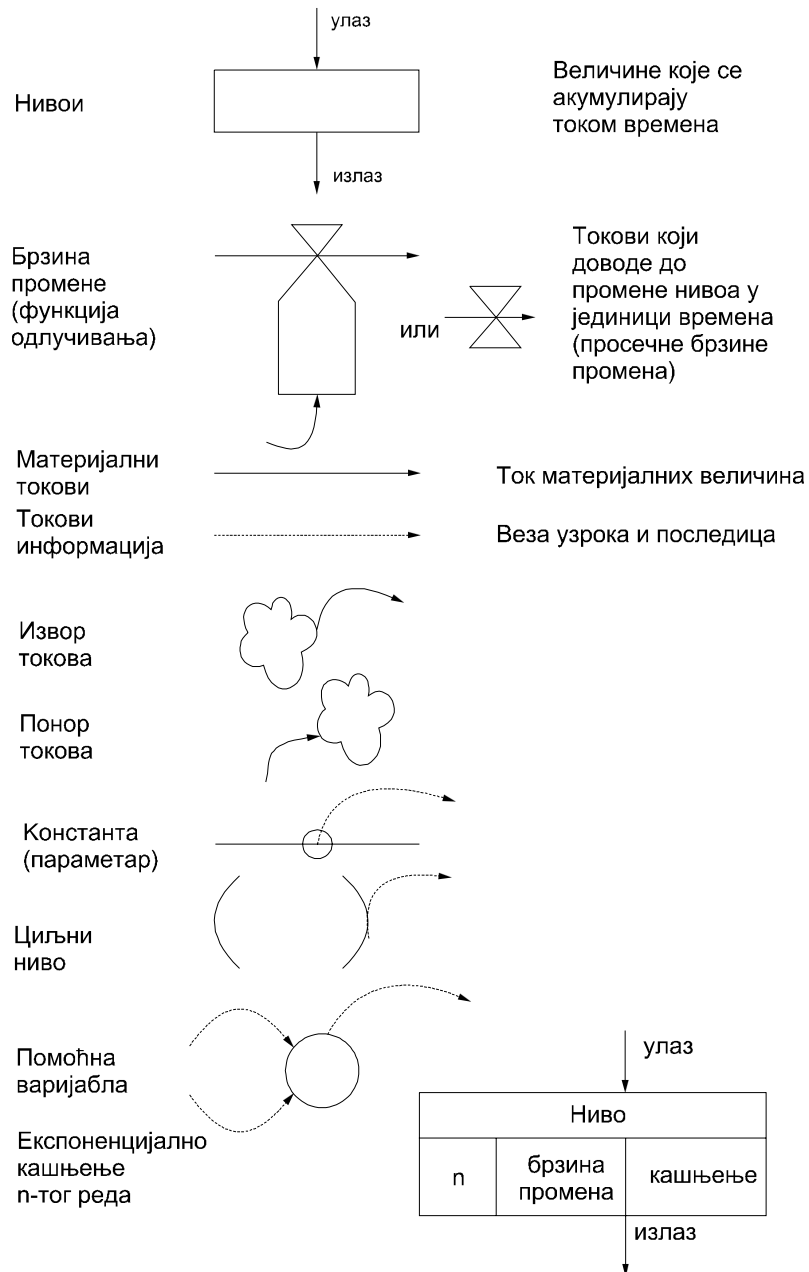
Тражња и капацитет производње

Слика 11.5 Пример негативне повратног дејства везе

У пракси, ретко се срећу системи који се састоје само од једне петље повратног дејства, како је то приказано у претходним примерима, већ је уобичајено да се системи описују већим бројем петљи које су повезане преко заједничких елемената који у њима учествују.

### 11.5.2 Дијаграми тока

Концептуални модели којима се идентификују стања система, промене стања система и кашњења у систему називају се *дијаграми тока* и служе за прецизније графичко представљање посматраног система. Ови модели чине основу за писање рачунарских модела у динамици система. Основни симболи који се користе за конструисање дијаграма тока дати су на слици 11.6.



Слика 11.6 Елементи дијаграма тока

Линијама се означавају токови у систему који представљају протоке материјала, енергије и информација. Њима се дефинише начин на који ресурси у систему прелазе из једног стања у друго. Материјални токови означавају се пуном, а информациони испрекиданом линијом. Сваки ток има одговарајући симбол за извор (почетак тока изван система) и понор (одредиште тока изван система).

Стања система су величине које се акумулирају током времена. На дијаграму тока представљају се правоугаоникима унутар којих се уписује њихов назив. Промене стања одређују протоке на токовима. То су просечне брзине (промена стања у јединици времена) које се често називају и функције одлучивања пошто се преко њих управља стањима система. Промене стања које се налазе на улазном току стања система, називају се улазне промене стања, док су излазне промене стања оне које се налазе на излазном току тог стања. Константе су величине у моделу које се не мењају током времена. Помоћне променљиве се налазе на токовима информација и служе за опис елемената промене стања. Јављају се као издвојени елементи јер имају своје независно значење.

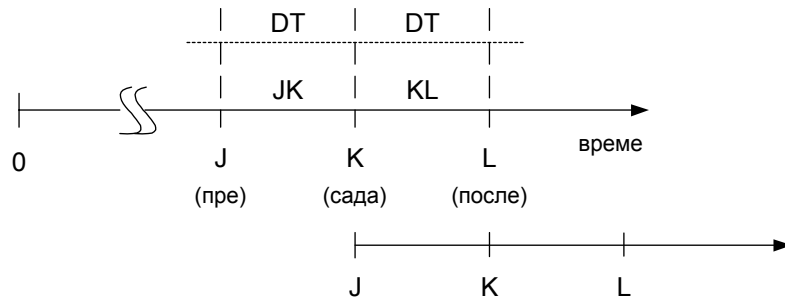
Поред набројаних, постоје и елементи за опис експоненцијалних кашњења на материјалним и информационим токовима који се овде неће посебно разматрати.

## 11.6 Рачунарски модели у динамици система

На основу израђеног дијаграма тока који представља детаљан концептуални модел система, ствара се рачунарски модел система. Рачунарски модели динамике система приказују квантитативне везе између елемената система и омогућују извођење планираних симулационих експеримената са моделом система. Они се приказују у облику система диференцијалних једначина. Једначине модела приказују промене вредности променљивих модела између узастопних, временских тренутака, удаљених за изабрани, константни, временски корак. Да би се извела симулација, односно добило понашање система у времену, потребно је решити систем диференцијалних једначина рачунарског модела, за шта се користе нумеричке методе интеграције.

### 11.6.1 Описивање промене времена у једначинама модела

Стања система у динамици система израчунавају се у дискретним тренуцима времена, рецимо  $t=0, \dots, T$ , међусобно удаљеним за константан временски интервал  $DT$ . Величину  $DT$  бира моделар и од ње зависи тачност резултата и време извођења симулације. Математички гледано, овај приступ подразумева коришћење диференцијалних једначина. Временска оса у динамици система садржи три узастопна, еквидистантна тренутка времена ( $t-1, t, t+1$ ), који се означавају са J, K и L (слика 11.7).



Слика 11.7 Временска оса у динамици система

Временски тренутак J представља претходно, тренутак K садашње, а тренутак L будуће време. Основни временски интервал је време између два узастопна временска тренутка и означава се са  $DT$ . Лако је уочити да је:

$$\begin{aligned} K &= J + DT \\ L &= K + DT \end{aligned}$$

Интервал времена између тренутака J и K обележава се са JK, а интервал између K и L са KL. Основи интервал времена одређује се на основу расположивих података и може представљати дан, недељу, месец, квартал, годину итд.

Вредности стања система обележавају се на следећи начин:

стање.J ; стање.K ; стање.L

Вредности промена стања представљају просечне брзине промена стања и између два узастопна временска тренутка. Третирају се као константе, а означавају се на следећи начин:

промена стања.JK ; промена стања.KL

Сада размотримо следећу једначину (*Kleijnen & Groenendaal, 1992*)

$$y_t = y_{t-1} + \Delta y$$

Уколико је брзина промене између  $t-1$  и  $t$  константна, тада следи:

$$\frac{\Delta y}{\Delta t} = c(t-1, t)$$

Комбиновањем горња два израза добија се:

$$y_t = y_{t-1} + \Delta y \cdot c(t-1, t)$$

Превођењем горњег израза у симболичку нотацију која се користи у динамици система, добија се:

$$\text{стање.K} = \text{стање.J} + \text{DT}(\text{промена.JK})$$

што представља општу једначину за одређивање вредности елемента стања система. Уколико посматрано стање система има и улаз и излаз, тада ће, по дефиницији, брзина промене у времену бити једнака разлици улазних и излазних токова, односно

$$\text{промена.JK} = \text{улаз.JK} - \text{излаз.JK}$$

Решавање једначина стања и једначина промене стања у времену, врши се у следећим корацима (*Черић, 1993*):

1. Време се помера на тренутак K.



2. Израчунавају се вредности стања система у тренутку K. За прорачун се користе једначине стања система.
3. Израчунавају се вредности свих промена стања за наредни интервал  $KL$  ) на основу добијених вредности стања система у тренутку K).
4. Преименују се стања система и промене стања система:  $K \rightarrow J$  ;  $L \rightarrow K$ , да би се омогућило коришћење истих једначина.
5. Време се помера за  $DT$  и нови садашњи тренутак поново називамо K.

Цео описани циклус се понавља док се симулација не заврши (не постигне задато време трајања симулације. Да би прорачун могао да отпочне, неопходне су почетне вредности за сва стања система, као и вредности за све константе и параметре система. На тај начин добија се узастопни низ вредности променљивих модела у временским тачкама, међусобно удаљеним за  $DT$ . Тај дисконтинуални низ може представљати апроксимацију континуалног временског понашања променљивих модела.

## 11.7 Програмски пакет SDS

SDS је програмски пакет за симулацију динамике система. Првенствено је намењен за симулационе моделе који се могу описати елементима стања, елементима промене стања и дискретном променом времена, при чему се може користити Euler-ова метода или метода Kutta-Merson-a, која омогућава контролу тачности рачунања за нумеричко решавање система диференцијалних једначина. Такође, са успехом се може користити за решавање система диференцијалних једначина типа:

$$\left( \frac{d}{dt} \right)_i = f_i(x, y_1, y_2, \dots, y_n), \quad i = 1, 2, \dots, n$$

Програмски пакет SDS омогућује:

- ♦ симулацију понашања система и
- ♦ анализу понашања система.

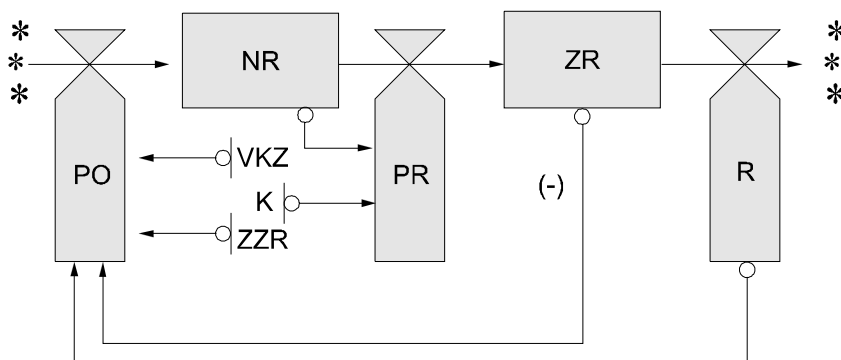
Детаљнији приказ језика SDS може се наћи у раду (Рајков и Раденковић, 1994)

### 11.7.1 Пример програма у SDS –у.

Вербални модел:

Предузеће се бави производњом и продајом одређене робе. Сектор продаје врши испоруке робе купцима са залиха робе. Попуњавање залиха се врши поручивањем од производње. Пријем наручене робе се обавља са извесним закашњењем. Поручивање робе се врши увек када су залихе робе мање од неке жељене количине. Испорука робе је константна у прва три месеца и износи 100 комада, а затим се повећава за 25% и више се не мења. Капацитет магацина је 500 комада, а у тренутку праћења залихе су износиле 300 комада.

Структурни модел (дијаграм тока) система управљања залихама дат је на слици 11.8.



Слика 11.8 Дијаграма тока модела управљања залихама

## Симулациони модел у SDS језику је:

## USVOJENE OZNAKE

E1.KL	IR	i	KOM/MES	ISPORUČENA ROBA
E2.K	ZR	z	KOM	ZALIHE ROBE
E3.K	NR	n	KOM	NEISPORUČENA ROBA OD DOBAV.
E4.KL	PR	p	KOM/MES	PRIMLJENA ROBA
E5.KL	PO	o	KOM/MES	PORUČENA ROBA
P1	VKZ	-	MES	VREM. KORIGOVANJA ZALIHA
P2	ZZR	-	KOM	ŽELJENE ZALIHE ROBE
P3	K	-	MES	KAŠNJENJE U PRIJEMU ROBE
P4	PSM	-	-	PROMENA STRUKTURE MODELA
P5	SKOK	-	KOM	PROMENA TRAŽNJE
P6	STAR	-	MES	VREME STARTA PROMENE TRAŽNJE

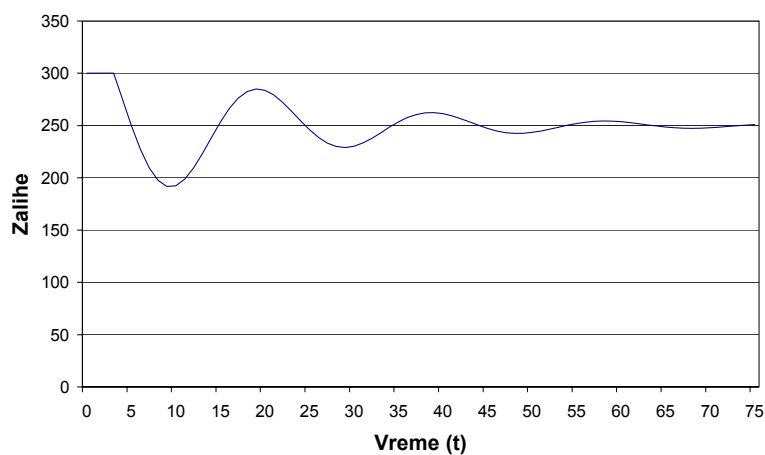
## VELIČINA KORAKA SIMULACIJE

DT=1.  
 BROJ KORAKA SIMULACIJE  
 BS=75.  
 BROJ ELEMENATA MODELA  
 BE=5.  
 BROJ PARAMETARA MODELA  
 BP=6.

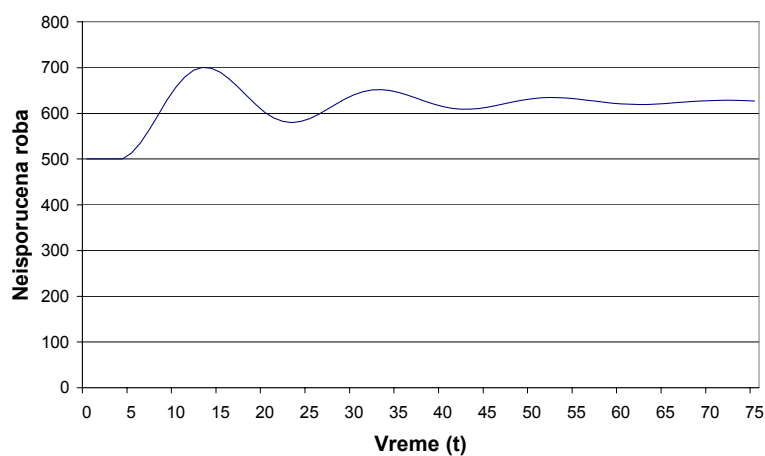
## MODEL

E1.KL=100.+STEP(P5,P6)  
 P5=25.  
 P6=3.  
 $E2.K = E2.J + DT * (E4.JK - E1.JK)$   
 $E2.0 = 300.$   
 $E3.K = E3.J + DT * (E5.JK - E4.JK)$   
 $E3.0 = 500.$   
 $E4.KL = E3.K / P3$   
 P3=5.  
 GO TO (10,20),INT(P4)  
 $E5.KL = E1.JK + (P2 - E2.K) / P1$   
 GO TO 30  
 $E5.KL = (P2 - E2.K) / P1$   
 CONTINUE  
 P1=2.  
 P2=500.  
 P4=2.  
 CALL SDS

Резултати симулације:



Слика 11.9 Промена стања залиха у функцији времена



Слика 11.10 Промена стања неиспоручене робе у функцији времена

# П Р И Л О Г

## ВЕРОВАТНОЋА И СТАТИСТИКА У СИМУЛАЦИЈИ

У стохастичким симулационим моделима, за опис система и његовог окружења, користе се случајне променљиве. За анализу и синтезу таквих модела, као и за анализу резултата симулационог експеримента, нужно је познавање основних концепата вероватноће и статистике. Због тога, у овом прилогу дајемо кратак преглед основних концепата вероватноће и статистике без детаљног удубљивања у доказе који се могу наћи у литератури (Вуковић, 1998; Вукадиновић и Поповић, 1996). Ознаке у овом поглављу су преузете из наведених референци.

Све догађаје везане за једну појаву или процес делимо у три класе:

- ♦ немогући догађаји,
- ♦ сигурни догађаји и
- ♦ случајни догађаји.

### П.1 Случајни догађаји

У природи постоје појаве које се могу потпуно описати помоћу сигурних и немогућих догађаја. То су појаве детерминистичког карактера. Међутим, у природи, а нарочито у друштву, много је већи број појава које се не могу потпуно описати помоћу сигурних и немогућих догађаја.

Узмимо пример бацања коцке. Знамо да ће при бацању пасти један од бројева  $\{1,2,3,4,5,6\}$ , али не знамо који ће пасти. Овакву врсту догађаја који се у датој појави могу остварити а исто тако и не остварити, називамо случајни догађаји, док појаве које изучавамо преко случајних догађаја називамо експерименти.

Скуп могућих исхода експеримента зовемо скуп елементарних догађаја, и било који резултат експеримента потпуно се описује једним и само једним елементом из скупа могућих исхода. Скуп елементарних догађаја може бити: коначан, бесконачан и пребројив, и бесконачан и небројив.

Пре него што пређемо на разматрање случајних променљивих, неопходно је истаћи неке карактеристике случајних догађаја:

1. Случајни догађај је подскуп скупа елементарних догађаја.
2. Немогућ догађај је подскуп скупа елементарних догађаја који нема ниједан елемент, и означава се са  $\emptyset$ .
3. Нека је дат случајан догађај  $A$ . Супротан догађај је догађај  $\bar{A}$  који се оствари онда и само онда када се  $A$  не оствари.
4. Супротан догађај немогућег догађаја је сигуран догађај.
5. Дата су два случајна догађаја  $A$  и  $B$ . Ако се сваки пут када се оствари догађај  $A$  остварује и  $B$ , кажемо да  $A$  имплицира  $B$  [ $A \Rightarrow B$ ].
6. За два случајна догађаја  $A$  и  $B$  кажемо да су једнаки ако истовремено  $A$  имплицира  $B$  и  $B$  имплицира  $A$ , тј. ако је [ $A \Rightarrow B$  и  $B \Rightarrow A$ ].
7. Нека су дати случајни бројеви  $A$  и  $B$ . Унија догађаја  $A$  и  $B$  је догађај који се оствари онда и само онда када се оствари бар један од догађаја  $A$  и  $B$ , и означава се са  $A \cup B$ .
8. Пресек догађаја  $A$  и  $B$  је случајан догађај који се остварује онда и само онда кад се остваре оба догађаја, тј. када се истовремено оствари догађај  $A$  и догађај  $B$ , и означава се са  $A \bullet B$  или  $AB$  или  $A \cap B$ .
9. За два догађаја  $A$  и  $B$  кажемо да се међусобно искључују ако је њихов пресек немогућ догађај, тј. ако је  $A \cap B = \emptyset$ .
10. Ако се два догађаја  $A$  и  $B$  међусобно искључују, тада је њихова унија збир тих догађаја и означавамо то са  $A \cup B = A + B$ .

11. За операције уније и пресека који су дефинисани на случајним догађајима важе следећа три закона:

♦ Комутативност:

$$A \cup B = B \cup A \text{ и } A \cap B = B \cap A$$

♦ Асоцијативност:

$$(A \cup B) \cup C = A \cup (B \cup C) \text{ и } (A \cap B) \cap C = A \cap (B \cap C)$$

♦ Дистрибутивност:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

## П.2 Вероватноћа

Вероватноћа случајних догађаја је свака функција  $P$  дефинисана на случајним догађајима која пресликава случајне догађаје у реалне бројеве и која има следеће особине:

1. Ненегативност. Сваком случајном догађају  $A$  функција  $P$  придружује ненегативан број  $P(A)$ .
2. Нормираност. Сигурном догађају функција  $P$  придружује број један.
3. Адитивност. Ако се случајни догађаји  $A_1, A_2, \dots$  међусобно искључују, онда функција  $P$  придружује њиховој унији број који је једнак збиру бројева које функција  $P$  придружује догађајима: тј.

$$P(A_1 + A_2 + A_3 + \dots) = P(A_1) + P(A_2) + P(A_3) + \dots$$

Потребно је навести још неке особине вероватноће:

1. Вероватноћа суротног догађаја једнака је:  $P(A) = 1 - P(\bar{A})$ .
2. Вероватноћа немогућег догађаја једнака је нули.
3. Вероватноћа било ког случајног догађаја је број који се налази између  $0 < P(A) < 1$ .



### П.2.1 Условна вероватноћа

Пракса намеће проблем налажења вероватноће догађаја  $A$ , ако се догађај  $B$  већ десио, и таква вероватноћа  $P(A/B)$  назива се условна вероватноћа

$$P(A/B) = \frac{P(A \cdot B)}{P(B)}.$$

Условна вероватноћа задовољава сва три аксиома којима је дефинисана вероватноћа.

### П.3 Случајне променљиве

Случајна променљива  $\tilde{X}$  може се дефинисати као променљива која представља исход експеримента.

Ако са  $X = \{x_1, x_2, x_3, \dots\}$  означимо резултат експеримента, онда кажемо да променљива  $\tilde{X}$  може узети једну вредност из скупа могућих резултата експеримента и то на случајан начин.

Потребно је утврдити колико често, односно са којом вероватноћом променљива  $\tilde{X}$  узима поједине вредности из датог скупа могућих вредности.

У пракси се посматрају две категорије случајних променљивих и то:

- ♦ дискретне и
- ♦ континуалне.

Променљива  $\tilde{X}$  назива се случајна променљива дискретног типа ако вредности које она може узети образују коначан или пребројив низ реалних бројева  $\{x_1, x_2, x_3, \dots\}$  а узимање сваке од ових вредности је случајан догађај са одређеном вероватноћом.

За променљиву  $\tilde{X}$  кажемо да је случајна променљива континуалног типа ако узимање било које вредности из интервала  $(-\infty; +\infty)$

представља случајан догађај и ако постоји функција  $f(x)$  таква да је вероватноћа да ће се  $X$  наћи у произвољно малој околини тачке  $x$  у интервалу

$$\left[ x - \frac{\Delta x}{2}; x + \frac{\Delta x}{2} \right].$$

#### П.4 Функција густине расподеле и функција расподеле

Функција  $f(x)$  назива се функција густине расподеле случајне променљиве  $\tilde{X}$ . Функција густине  $f(x)$  мора да задовољи следећа два услова:

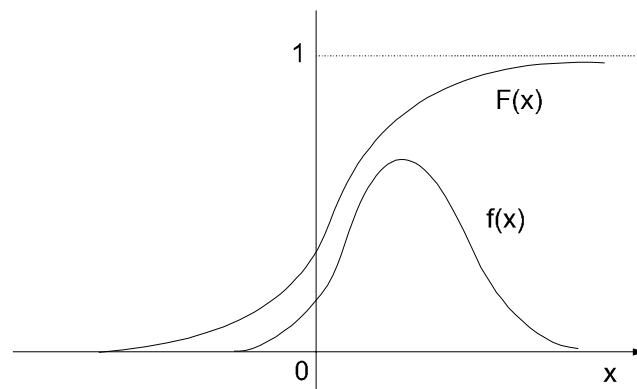
Вероватноће да се случајна променљива нађе у неком интервалу  $(x_1, x_2]$  је :

$$P(x_1 < X < x_2) = \int_{x_1}^{x_2} f(x) dx .$$

У пракси је често потребан податак о вероватноћи да случајна променљива  $\tilde{X}$  узме вредност не мању од  $x$ , при чему је  $x$  унапред одређен број из интервала  $(-\infty; +\infty)$ . Нпр. интересује нас догађај да ће у аутоматској телефонској централі број позива бити мањи од неког броја који представља капацитет централе. Та вероватноћа је:

$$P(X > x) = 1 - P(X \leq x).$$

Функција расподеле случајне променљиве  $\tilde{X}$  је функција  $F(x)$  која за свако  $x$  из  $(-\infty; +\infty)$  представља вероватноћу да случајна променљива  $X$  неће узети вредност већу од  $x$ , тј.  $F(x) = P(X \leq x)$ .



Слика П.1 Графички приказ функције густине расподеле и функције расподеле случајне променљиве

Функције расподеле окарактерисана је следећим особинама:

1. Функција расподеле има вредности између нула и један, тј.  $0 < F(z) < 1$ .
2.  $F(-\infty) = 0$ .
3.  $F(+\infty) = 1$ .
4. Функција  $F(x)$  је монотонно неоппадајућа функција.
5. Код случајне променљиве  $\tilde{X}$  дискретног типа функција расподеле  $F(x)$  је степенаста функција која у тачкама  $x_1, x_2, x_3, \dots$  има скокове једнаке одговарајућим вероватноћама.
6. Ако је  $\tilde{X}$  непрекидна случајна променљива функцијом густине расподеле,  $f(x)$ , тада је за свако

$$x \in (-\infty; +\infty) \Rightarrow F(x) = \int_{-\infty}^x f(x) dx .$$

### П.4.1 Трансформација случајне променљиве

Нека случајна променљива  $\tilde{X}$  има функцију расподеле  $F(x)$ , са функцијом густине расподеле  $f(x)$ .

Ако случајна променљива  $\tilde{Y}$  представља функцију од случајне променљиве  $\tilde{X}$ , тада  $G(y)$  представља функцију расподеле две случајне променљиве, а  $g(y)$  функцију густине расподеле.

Ако је  $h(x)$  туђа функција, тада је

$$G(y) = F(h^{-1}(y)) \text{ па је } F(x) = G(h(x))$$

У случају линеарне трансформације  $Y=aX+b$ , функција расподеле  $G(y)$  се одређује као

$$G(y) = F\left(\frac{y-b}{a}\right)$$

Док се функција густине расподеле налази као извод  $G(y)$ , тј.

$$g(y) = \frac{1}{|a|} \cdot f\left(\frac{y-b}{a}\right)$$

### П.4.2 Нумеричке карактеристике функције расподеле

Прва група параметара случајних променљивих са неком расподелом су параметри који показују централну тенденцију расподеле, и то су:

- ♦ аритметичка средина ( $\bar{X}$ ),
- ♦ математичко очекивање, очекивана вредност ( $E(x)$ )
- ♦ хармонијска средина ( $H$ ),
- ♦ геометријска средина ( $G$ ),
- ♦ модус ( $M_o$ ) и
- ♦ медијана ( $M_e$ ).

Математичко очекивање окарактерисано је следећим особинама:

1. Математичко очекивање од константе  $c$  једнако је  $c$ .
2. Очекивање производа константе  $c$  и функције  $g(X)$  једнако је производу те константе  $c$  и очекивања функције  $g(X)$ , тј:

$$E\{c g(X)\} = c E\{g(X)\}.$$

3. Очекивање збира две функције случајне променљиве једнако је збиру њихових очекивања, тј.

$$E\{g(X)+h(X)\}=E\{g(X)\}+E\{h(X)\}.$$

4. Очекивање линеарне функције случајне променљиве једнако је линеарној функцији очекивања те променљиве:

$$Y=aX+b \quad E(Y)=aE(X)+b.$$

5. Очекивање линеарне комбинације две случајне променљиве  $\tilde{X}$  и  $\tilde{Y}$ , је једнако линеарној комбинацији њиховог очекивања,  $Z=aX+bY$ ;  $E(Z)=aE(X)+bE(Y)$ . Очекивање збира или разлике такве две случајне променљиве једнако је збиру, односно разлици њихових очекивања

$$E(X \pm Y)=E(X) \pm E(Y).$$

Хармонијска средина је дата преко очекивања реципрочне вредности случајне променљиве изразом

$$\frac{1}{H} = E\left(\frac{1}{X}\right).$$

Геометријска средина је дата изразом

$$\log G = E(\log X).$$

Модус случајне променљиве  $X$  је она вредност  $M_o$  за коју расподела односно густина има највећу вредност.

Медијана је она вредност случајне величине  $M_e$  за коју је

$$P(X \leq M_e) = P(X > M_e).$$

Друга група параметара су параметри који представљају мере варијабилности случајне променљиве, то су:

- ◆ средња девијација,
- ◆ варијанса,
- ◆ стандардна девијација и
- ◆ коефицијент варијације.

Средња девијација је математичко очекивање апсолутне вредности разлике  $X$  и његовог очекивања, тј.

$$e_m = E(|X - \mu|).$$

Варијанса случајне променљиве  $\tilde{X}$  представља математичко очекивање квадрата одступања случајне променљиве  $X$  од  $\mu$ :

$$\sigma^2 = E(X - \mu)^2.$$

Особине варијансе:

1. Варијанса случајне променљиве  $\tilde{X}$  је једнака нули онда и само онда када је  $X = \text{const.}$
2. Ако је случајна променљива  $Y$  линеарна функција  $Y = aX + b$ , тада је варијанса  $Y$  једнака

$$\sigma^2(Y) = a^2 \sigma^2(X).$$

3. Математичко очекивање квадрата разлике  $X$  и константе  $c$  је минимално ако је та константа  $c = \mu$ , а његова минимална вредност је једнака варијанси случајне променљиве  $\tilde{X}$ , тј.

$$\min_c E(X - c)^2 = E(X - \mu)^2 = \sigma^2$$

*Стандардна девијација* случајне променљиве  $\tilde{X}$  је позитивна вредност корена варијансе.

*Коефицијент варијације* случајне променљиве  $\tilde{X}$  је процентуално изражен количник

$$V = \frac{\sigma}{\mu} \cdot 100\%$$

## П.5 Стандардизована случајна променљива

На крају можемо увести стандардизовану случајну променљиву која представља однос:

$$Z = \frac{X - \mu}{\sigma}$$

*Математичко очекивање* случајне променљиве  $Z$  једнако је нули, што се може и доказати:

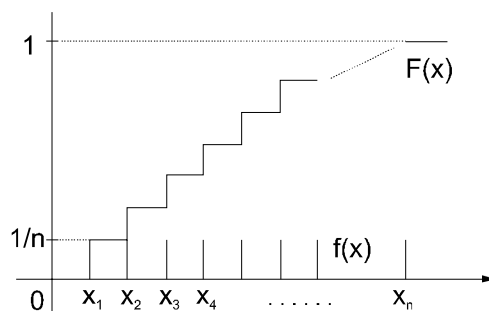
$$E\left(\frac{X - \mu}{\sigma}\right) = \frac{1}{\sigma} [E(X) - E(\mu)] = \frac{1}{\sigma} [\mu - \mu] = 0$$

*Варијанса* случајне променљиве  $Z$  једнака је јединици, тј.  $\sigma^2 = 1$  што се на сличан начин може доказати.

## П.6 Типичне расподеле случајних променљивих

### П.6.1 Дискретна униформна расподела

Дискретна униформна расподела случајне променљиве  $X$  може се графички представити на следећи начин:



Слика П.2 Функције расподеле и густине расподеле за униформну дискретну расподелу

Случајна променљива  $\tilde{X}$  узима вредности из скупа  $\{x_1, x_2, \dots, x_n\}$ .

Вероватноћа да случајна променљива  $\tilde{X}$  узима вредност  $x_i$  је

$$P(X = x_i) = \frac{1}{n} \quad i = 1, 2, \dots, n$$

при чему је математичко очекивање

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

док је варијанса једнака

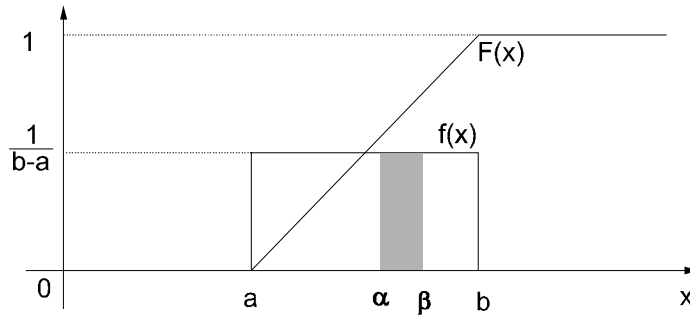
$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

Одговарајућом аналогијом може се дефинисати и континуална униформна расподела.



### П.6.2 Континуална униформна расподела

Континуална униформна расподела може се графички приказати на следећи начин:



Слика П.3 Функције расподеле и густине расподеле за униформну континуалну расподелу

где је  $x \in X = [a, b]$  интервал у  $R$ , и

$$f(x) = \begin{cases} \frac{1}{b-a} & a \leq x < b \\ 0 & \text{другачије} \end{cases}$$

Вероватноћа да ће се случајна променљива  $\tilde{X}$  наћи у интервалу  $\{\alpha, \beta\}$  једнака је

$$P(\alpha \leq x < \beta) = \frac{\beta - \alpha}{b - a}$$

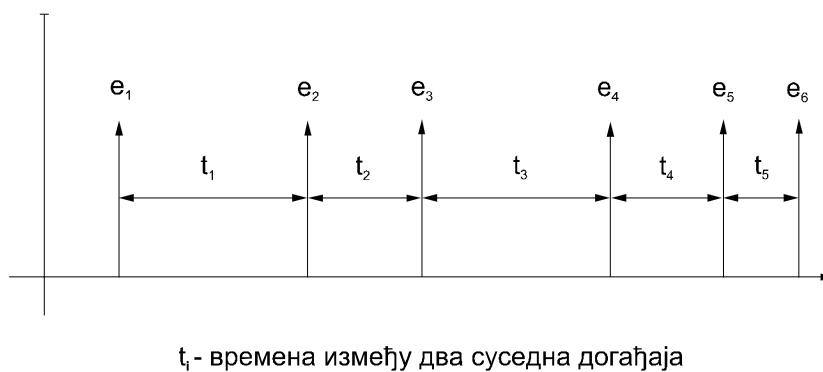
У пракси се често користи случајна променљива на "јединичном интервалу",

$$\tilde{r} \in [0,1) \quad f(x) = \begin{cases} 1 & , x \in [0,1) \\ 0 & , x \notin [0,1) \end{cases}$$

Случајна променљива са униформном расподелом на јединичном интервалу коју означавамо са  $\tilde{r}$ , користи се, осим у основном облику, и за генерисање других типова расподела.

### П.6.3 Експоненцијална расподела

За представљање реалних појава као што су долазак купаца у самопослугу, појава телефонских позива, појава кварова на машини, и других система опслуживања, искуством је утврђено да је најпогоднији облик експоненцијална расподела. Догађаје у оквиру експоненцијалне расподеле можемо представити следећим графиком:



Сл. П.4 Догађаји са експоненцијалном расподелом

За ову расподелу важе полазне претпоставке:

1. време наступања догађаја не зависи од претходног догађаја,
2. вероватноћа да догађај наступи у малом интервалу времена пропорцијална је дужини тог интервала:

$$P\left(\tilde{t} < t + \frac{\Delta t}{\tilde{t}} \geq t\right) = \frac{P(t \leq \tilde{t} < t + \Delta t)}{P(\tilde{t} \geq t)} = \frac{F(t + \Delta t) - F(t)}{1 - F(t)} = \lambda \cdot \Delta t$$

где је  $P(\tilde{t} < t) = F(t)$  тражена расподела за  $\tilde{t}$ .

Ако  $\Delta t \rightarrow 0 \Rightarrow F' = \lambda(I - F(t))$ ,  $t \geq 0$  па је тражена расподела:

$$F(t) = \begin{cases} I - e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

док је функција густине расподеле:

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

Математичко очекивање параметара ове расподеле једнако је:

$$E(\tilde{t}) = \int_0^{\infty} t \lambda e^{-\lambda t} dt = \frac{1}{\lambda}$$

Стандардна девијација је такође једнака  $\frac{1}{\lambda}$ .

#### П.6.4 Poisson-ова расподела

За случајну променљиву  $\tilde{X}$  кажемо да има Poisson-ову расподелу ако може узети вредност  $k$  из низа ненегативних целих бројева  $[0, 1, 2, 3, \dots]$  са вероватноћом

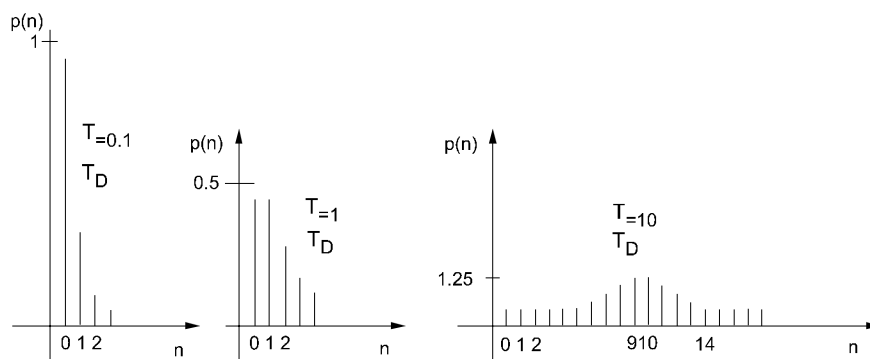
$$P(k) = P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda},$$

при чему је  $\lambda > 0$ , реалан број и представља параметар расподеле.

Ако интервали између два суседна догађаја ( $\tilde{t}$ ) имају експоненцијалну расподелу, тада је вероватноћа да се  $n$  догађаја деси на интервалу  $T$  једнака

$$P(\tilde{n} = n) = \frac{(\lambda T)^n e^{-\lambda T}}{n!} \quad n = 0, 1, 2, \dots$$

где је  $\tilde{n}$  број догађаја у јединици времена.



Слика П.5 Облици Poisson-ових расподела за различите односе

### П.6.5 Нормална (Gauss-ова) расподела

Користи се за представљање стохастичких појава уношењем шума у детерминистичке променљиве. Дефинисана је са 2 параметра

- ♦  $\mu$  - средња вредност
- ♦  $\sigma$  - стандардна девијација

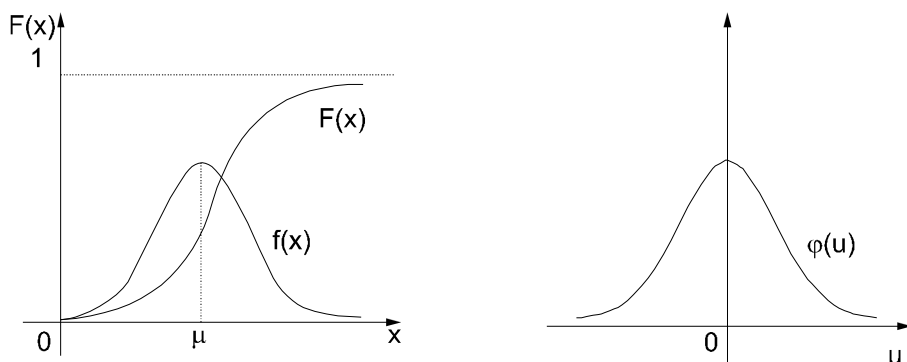
$$P(x < X) = \Phi\left(\frac{x - \mu}{\sigma}\right)$$

$$\Phi(z) = \int_{-\infty}^z (\xi) d\xi$$

Функција стандардизоване нормалне расподеле за коју важи да је  $\mu = 0$   $\sigma = 1$  има облик:

$$\varphi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \quad \left( z = \frac{x - \mu}{\sigma} \right)$$

Графички облик функције расподеле и функције густине расподеле за нормалну расподелу дат је на слици П.6.



Слика П.6 Нормална расподела

На другом графикону је стандардизована нормална расподела са вредностима  $\mu = 0$   $\sigma = 1$ .

## П.7 Генерисање случајних бројева

У току спецификације стохастичких модела за променљиве које описују промену времена, улаза и стања, дефинишу се статистичке расподеле које их описују са најмањом грешком у односу на реални систем. При симулацији тако дефинисаних модела јавља се потреба за рачунарским алгоритмима који ће генерисати случајне променљиве за описивање промена времена и стања у моделу са неком задатом расподелом.

У статистичкој теорији се доказује да је случајну променљиву са задатом расподелом могуће генерисати на основу једне или више независних случајних променљивих са униформном расподелом. На тај начин се проблем своди на генерисање случајне променљиве са униформном расподелом.

Случајну променљиву са униформном расподелом могуће је добити једном од следећих метода:

- ♦ мануелне методе,
- ♦ табеле случајних бројева,

- ♦ методе за рад са аналогним рачунарима и
- ♦ методе за рад са дигиталним рачунарима.

### П.7.1 Мануелне методе

Мануелне методе су малог опсега и због тога су неприменљиве у симулацији. Евентуално се могу користити за обучавање кадрова. У ове методе спадају: извучење бројева из кутије, бацање коцкица итд.

### П.7.2 Табеле случајних бројева

Табеле случајних бројева су серије случајних бројева (СБ) који се производе на погодан начин, помоћу неког физичког извора или неком од метода извлачења бројева.

Највећа до сада публикована табела случајних бројева, садржи 1.000.000 цифара (a million random digitis, Rand Corporation, The Free Press, 1955). Бројеви су добијени на следећи начин. Био је конструисан специјалан рулет који се покретао и заустављао уз помоћ електронике. Покретни диск се нагло заустављао и бирала се цифра коју је показивала стрелица

Овако добијене цифре се скупљају у четвороцифрене или петоцифрене бројеве и уносе у табелу случајних бројева. Иако овакав поступак поседује сва својства случајности, ипак су се после дужег времена почеле да добијају цифре које, према провери статистичким тестом, нису имале равномерну расподелу.

### П.7.3 Методе за генерисање СБ на аналогном рачунару

Ове се методе базирају на коришћењу случајних физичких процеса који се одвијају у аналогном рачунару и дају "праве случајне бројеве". Неподесне су за рад, јер у случају поновљене симулације са истим редоследом случајних бројева, физички процес даје нове случајне бројеве.

### П.7.4 Методе за генерисање СБ на дигиталном рачунару

Методе за генерисање случајних бројева базирају се на следећим поступцима:

1. Коришћење физичких извора + A/D конверзија,
2. Коришћење табела случајних бројева на дисковима рачунара,
3. Коришћење алгоритама за добијање псеудослучајних бројева.

Коришћење физичких извора (бели шум, радиоактивни распад) и конверзија физичких величина у дигиталне је скуп и неприступачан процес за свакодневну примену.

Табеле случајних бројева на масовним меморијама рачунара су добар али спор начин за добијање случајних бројева при симулацијама сасвим просечних система који у току симулације траже неколико милиона случајних бројева.

Алгоритми за генерисање псеудослучајних бројева су брз начин за генерисање случајних бројева. Међутим квалитет таквих генератора зависи од алгорита и рачунара на коме се имплементира. Дужина секвенце бројева без понављања је ограничена а униформност и резолуција може да буде променљива. Због тога је потребно пре употребе оваквих генератора испитати њихову резолуцију, униформност и дужину секвенце случајних бројева без понављања стандардним статистичким методама (нпр.  $\chi^2$  тест) и утврдити да ли добијене карактеристике одговарају захтевима модела.

Међутим, без обзира на наведене недостатке алгоритамских генератора случајних бројева (ГСБ), данас су претежно у употреби овакви генератори.

Од избора алгорита зависи и квалитет генератора. Постоје многи алгоритми за генерисање случајних бројева, али се углавном користе конгруентни генератори. Њих карактерише једноставност и брзина.

### П.7.5 Линеарни конгруентни ГСБ

Да бисмо објаснили механизам рада конгруентних генератора случајних бројева потребно је најпре дефинисати шта су то конгруентни бројеви: Конгруентни бројеви су такви бројеви који су међусобно дељиви без остатка.

Користећи особине конгруентних бројева, можемо дефинисати следеће врсте линеарних конгруентних генератора псеудо-случајних бројева са униформном расподелом:

1. Мултипликативни  $Z_i = (Z_{i-1} \cdot a) \bmod m$
2. Мешовити  $Z_i = (Z_{i-1} \cdot a + c) \bmod m$
3. Адитивни  $Z_i = (Z_{i-1} \cdot a + Z_{i-k} \cdot b) \bmod m$

где су:

$Z_i$  - псеудо случајни број,  
 $Z_0$  - почетна вредност или семе генератора,  
 $a$  - мултипликатор,  
 $b$  - константа,  
 $c$  - константа  
 $m$  - модулус.

Од наведених генератора, највише је у употреби линеарни мултипликативни конгруентни генератор псеудослучајних бројева, па ће њему бити посвећена посебна пажња.

Линеарни мултипликативни конгруентни генератор случајних бројева генерише случајне бројеве по следећој формули:

$$Z_i = (a * Z_{i-1} + b) \bmod m$$

$$m = 2^{b-1}$$

где су :

$Z_i$  - псеудослучајни број,  
 $Z_0$  - почетна вредност (семе)  
секвенце псеудо случајних бројева,  
 $a$  - мултипликатор,  
 $b$  - број битова целобројне променљиве у рачунару.

Доказује се да је максимална могућа дужина низа генерисаних бројева, без понављања  $N_{max} = m - 1$ .

Да би се постигло максимално могуће  $N_{max}$ ,  $a$  и  $Z_0$  треба да испуњавају следеће услове



$$a = 8 \cdot j \pm 3 \quad j = 1, 2, 3, \dots$$

$$Z_0 = 1, 3, 5, 7, \dots$$

тада је

$$\frac{m}{4} \leq N_{\max} \leq m - 1$$

Квалитет генератора случајних бројева првенствено зависи од броја бита целобројне променљиве. Погодно изабраним мултипликатором, за исти број бита целобројне променљиве могуће је побољшати квалитет генератора за 75%. У литератури постоји низ радова који се баве проблематиком избора мултипликатора.

Ради јаснијег схватања дат је пример имплементације линеарног мултипликативног генератора са штампањем првих 10 случајних бројева. У овој имплементацији за генерисање случајних бројева користи се целобројна променљива дужине 32 бита ( $b = 32$ ). Мултипликатор је 16383. За семе генератора узет је број 1. Имплементација је тестирана на TURBO PASCAL-u.

```

PROGRAM RndTest;
TYPE Rand = RECORD
    Seed : LongInt;      {Seme - четворобajтна prom.}
    Mult : LongInt;      {Мултипликатор}
    Val : Real;          {Случајан број у интервалу 0-1}
END;
VAR i : INTEGER;
    Rnd : Rand;
PROCEDURE Randu (VAR Rnd : Rand);
BEGIN
    Rnd.Seed := Rnd.Seed * Rnd.Mult;
    IF Rnd.Seed < 0 THEN
        Rnd.Seed := Rnd.Seed + MaxLongInt + 1;
        Rnd.Val := Rnd.Seed / MaxLongInt
    END;
BEGIN {Главни програм}
    Rnd.Seed := 1;      {Дефинисање семена генер.}
    Rnd.Mult := 16383;  {Дефинисање мултипликатора}
    FOR i := 1 TO 10 DO {Извлачење 10 случајних број.}
        BEGIN
            Randu(Rnd); {Генерисање случ. броја}
            Writeln(Rnd.Val) {Штампање}
        END
    END.

```

По извршењу програма биће одштампано 10 псеудослучајних бројева у интервалу  $(0, 1)$ . Уколико корисник жели да генерише већи број независних токова псеудо-случајних бројева, потребно је да у

сегменту иницијализације програма, дефинише одговарајући број различитих почетних вредности семена генератора, при чему вредност мултипликатора може да остане непромењена за све токове.

## П.8 Тестови за проверу ГСБ

Псеудослучајни бројеви добијени из генератора случајних бројева треба да задовоље особине које се односе на униформност расподеле унутар интервала  $(0,1)$  и независност појављивања. Наиме, иако је генерисање бројева линеарним конгруентним генераторима строго детерминистички алгоритам, генерисани бројеви морају да задовоље одређене статистичке тестове и тада се сматрају случајним.

### П.8.1 $\chi^2$ -тест

Овај тест служи за испитивање да ли неки скуп генерисаних случајних бројева има претпостављени распоред. Неки скуп од  $n$  елемената (случајних бројева) може се груписати према некој особини у  $r$  група. Нека се у  $i$ -тој групи нађе  $f_i$  елемената тако да је

$$f_1 + f_2 + \dots + f_r = n$$

Вероватноћа да ће се елеменат  $x$  наћи у групи  $f_i$  је  $p_i$ .

Познато је да број елемената који према теоријској расподели треба да се нађу у  $i$ -том интервалу износи  $np_i$ .

Да би претпоставка  $(H_0)$  да дати скуп подлеже одређеној расподели била у важност, треба да се нађе вредност израза

$$2 \sum (f_i - np_i) \ln \frac{f_i}{np_i}$$

и да се она упореди са вредношћу  $k$  која се налази у таблицама за  $r-1$  степене слободе и за вероватноћу која одговара задатом нивоу значајности.

Ако је  $k < k_0$  сматра се да скуп подлеже претпостављеној расподели, тј усваја се хипотеза  $H_0$ ; у супротном се хипотеза одбацује.

### П.8.2 Колмогоров-Смирнов тест

Овај тест омогућава поређење да ли испитивани скуп случајних бројева подлеже претпостављеној расподели.

Нека је генерисано  $n$  случајних бројева. Треба прво обавити њихово сређивање по величини, тј.

$$x_1 < x_2 < \dots < x_n.$$

Код униформне расподеле, вероватноћа појављивања сваког од њих износи  $1/n$ . Кумулативна функција расподеле може се конструисати на тај начин што ће код сваког  $x$  доћи до пораста функције за  $1/n$ .

Пошто је кумулативна функција униформне расподеле  $F_n(x)$  позната, то се може израчунати изразом

$$D_n = \max_{-\infty < z < \infty} |F(z) - F_e(z)|.$$

Уколико је  $D_n(x)$  мање од критичне вредности  $d_0$  за праг значајности  $\alpha$ , нулта хипотеза (хипотеза да се ради о униформној расподели) биће прихваћена.

Предност овог теста над  $k$ -тестом је у томе што се могу тестирати и мали скупови бројева. Међутим, потреба да се бројеви среде по редоследу захтева доста меморијског простора, јер је број генерисаних бројева обично врло велики.

Постоји и читав низ других тестова за проверу квалитета генератора случајних бројева као што су:

- ◆ покер тест,
- ◆ тест серије истих цифара,
- ◆ интервални тест и
- ◆ тестови корелације,

али се они ређе користе.

## П.9 Генерисање случајне променљиве са задатом расподелом

Случајне променљиве чије вредности генеришемо у моделу подлежу некој расподели, која може бити теоријска или емпиријска, а по природи континуална или дискретна.

Постоји неколико метода за генерисање вредности случајних променљивих на дигиталним рачунарима, као што су:

- ♦ метода инверзне трансформације
- ♦ метода одбацивања,
- ♦ метода правоугаоне апроксимације и
- ♦ метода сумирања.

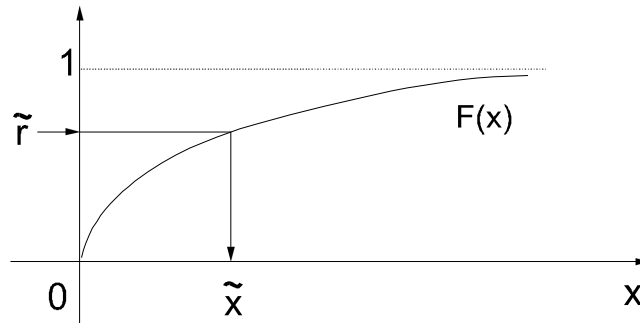
Заједнички задатак за ове методе је да генеришу случајну променљиву са неком другом расподелом, ако располажемо генератором случајних бројева са униформном расподелом. Претпоставља се да је генератор нормиран.

### П.9.1 Метода инверзне трансформације

Уводи се смена  $\tilde{X} = h(\tilde{r})$  таква да случајној променљивој  $\tilde{X}$  одговара жељена функција расподеле  $F(X)$ .

$$F(X) = P(\tilde{X} < X) = P(\tilde{r} < r) = P(\tilde{r} < h^{-1}(X)) = U(h^{-1}(X)) = h^{-1}(X) \\ h(\tilde{r}) = F^{-1}(\tilde{r})$$

Примена ове методе ограничена је на функције расподеле које имају инверзну функцију и монотоне су. Углавном се примењује за генерисање случајних променљивих са експоненцијалном расподелом.



Слика П.7 Графички приказ методе инверзне трансформације

### П.9.2 Метода одбацивања

Услов за примену ове методе је ограничен опсег дефинисаности функције густине расподеле.

$$f(X) \approx 0 \text{ за } X \notin [a, b]$$

Ако се на површини  $(b-a)*c$  генерише низ тачака са униформном расподелом и ако се одбаце све тачке које падају изнад криве  $f(X)$  преостале тачке имаће функцију густине расподеле  $f(X)$ .

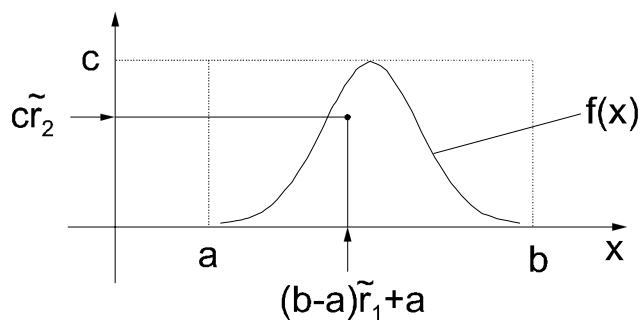
Алгоритам :

```

REPEAT
    генериши  $r_1$  и  $r_2$  ;
     $X_1 := (b - a) * r_1 + a$  ;
UNTIL  $r_2 * c \leq f(X_1)$ 
 $X := X_1$ 

```

С обзиром на то да се приликом сваког промашаја наново генеришу случајни бројеви са униформном расподелом, као и на то се троши извесно рачунарско време, за примену ове методе погодније су расподеле које имају већу површину ограничену апсцисом и кривом функције расподеле.



Слика П.8 Графички приказ методе одбацивања

### П.9.3 Метода правоугаоне апроксимације

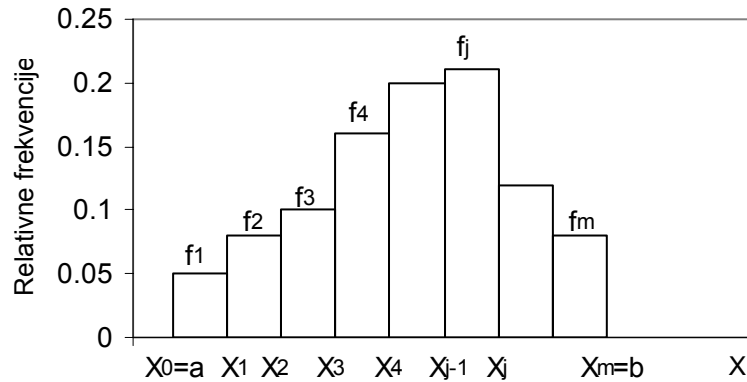
Метода правоугаоне апроксимације користи се за генерисање емпиријских расподела, када је функција расподеле монотона и задата паровима тачака.

Код многих реалних проблема, вероватноћа да ће се десити догађај изражава се у виду емпиријских података. Ти подаци се могу груписати у произвољан број фреквенцијских класа ( $m$ ) које сачињавају хистограм.

Свака класа је представљена правоугаоником чија се доња и горња граница могу означити са  $X_{j-1}$ ,  $X_j$ ,  $j=1,2,\dots,m$ . Укупан број реализација случајне променљиве у оквиру једне класе се назива апсолутна фреквенција и означава се са  $n_j$ ,  $j=1,2,\dots,m$ .

Релативна фреквенција сваке класе,  $f_j = 1/n_j$  представља вероватноћу да ће реализација случајне променљиве  $\tilde{X}$  узети вредност из класе  $j$  (слика П.9), тј:

$$X_{j-1} \leq \tilde{X} \leq X_j$$



Слика П.9 Хистограм расподеле релативних фреквенција

Збир релативних фреквенција је једнак јединици, односно:

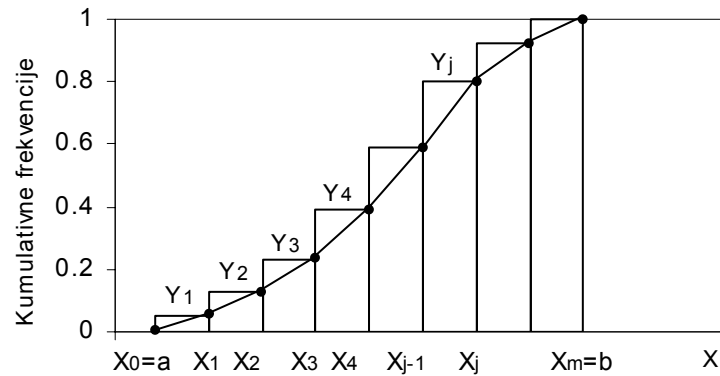
$$\sum_{j=1}^m f_j = 1$$

На основу добијеног хистограма може се лако конструисати функција расподеле, при чему је потребно израчунати кумулативну суму претходних фреквенција (слика П.10), т.ј.

$$\begin{aligned} Y_1 &= f_1 \\ Y_2 &= f_1 + f_2 \\ &\vdots \\ Y_j &= f_1 + f_2 + \dots + f_j \\ &\vdots \\ Y_m &= f_1 + f_2 + \dots + f_m = 1 \end{aligned}$$

Кумулативна сума претходних фреквенција,  $Y_j$ , представља вероватноћу да случајна вредност  $\tilde{X}$  не прелази вредност  $X_j$ .

Да би омогућили примену методе инверзне трансформације, потребно је апроксимативно одредити функцију расподеле, тако што се кроз кумулативни хистограм провлачи континуална крива. То се најлакше може урадити повлачењем сегмената правих линија, као што је приказано на слици П.10.

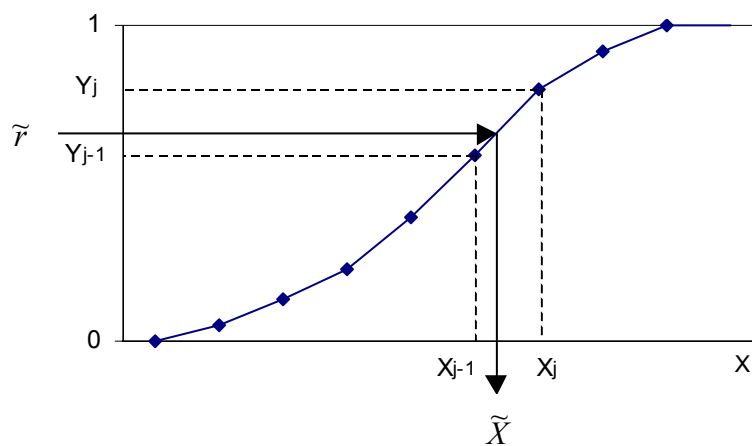


Слика П.10 Кумулативна сума релативних фреквенција

За добијање одговарајуће вредности  $\tilde{X}$  потребно је генерисати униформно расподељен случајни број  $\tilde{r} \in [0, 1)$  и користити једначину за линеарну интерполацију, т.ј.

$$\tilde{X} = X_{j-1} + \left( \frac{\tilde{r} - Y_{j-1}}{Y_j - Y_{j-1}} \right) \cdot (X_j - X_{j-1})$$

што је илустровано на слици П.11.



Слика П.11 Линеарна апроксимација функције расподеле



Једини преостали проблем јесте одређивање класе која одговара случајној величини  $\tilde{r}$ . То се може решити одговарајућим претраживањем, стартујући од крајње леве класе (т.ј.  $j=1$ ) и сукцесивним поређењем вредности  $\tilde{r}$  са вредностима  $Y_j$ . Тражена класа је прва класа за коју је  $\tilde{r} \leq Y_j$ .

При имплементацији ове методе на рачунару морају бити познате следеће вредности: доња и горња граница апцисе функције расподеле ( $a=X_0, b=X_m$ ), као и вредности кумулативних фреквенција  $Y_j$  (слика П.10). За случај када су класе једнаких ширина, њихове границе се рачунају на следећи начин:

$$X_{j-1} = a + \left( \frac{b-a}{m} \right) \cdot (j-1)$$

$$X_j = a + \left( \frac{b-a}{m} \right) \cdot j$$

Уколико класе хистограма нису једнаке ширине, потребно је поред поменутих вредности учитати и границе свих класа  $X_j, j=0, 1, \dots, m$ .

Горње једначине се могу модификовати у следећим случајевима:

1. Уколико је функција расподеле задата аналитички, а потребно је "припремити" исту као улаз у неки од симулационих програма, тада је могуће извршити поделу вредности функције расподеле на једнаке интервале, за које је потребно израчунати вредности  $X_i, i=1, 2, \dots, n$ .
2. Ако је пожељно изједначити вредности фреквенција класа у хистограму, тада је потребно формирати класе различитих ширина. У том случају су вредности функције расподеле, које одговарају границама класа, на једнаким растојањима.
3. Када се захтева ефикаснији алгоритам за претраживање, при чему грешка апроксимације није критична, могуће је извршити поделу вредности функције расподеле ( $Y$ ) на једнаке интервале, за које је потребно израчунати вредности  $X$ .

У наведеним случајевима, могуће је користити следећи алгоритам, који је ефикаснији у рачунском смислу:

$$\begin{aligned} &\text{генериши } \tilde{r} ; \\ &i := INT \left( \frac{\tilde{r}}{\Delta Y} \right) \\ &\tilde{X} := X_i + A_i * (\tilde{r} - i * \Delta Y) \end{aligned}$$

где је:

$$\begin{aligned} A_i &:= \frac{X_{i+1} - X_i}{\Delta Y} \\ \Delta Y &:= \frac{I}{m} \end{aligned}$$

#### П.9.4 Метода сумирања

Ова метода се користи за генерисање Нормалне (Gauss-ове) расподеле. Заснива се на примени централне граничне теореме.

Централна гранична теорема се може формулисати на следећи начин:

Нека је  $\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_n$  низ независних случајних променљивих са једнаким вероватноћама, свака са средњом вредношћу  $\mu_x$  и коначном варијансом  $\sigma_x^2$ . Њихова средња вредност дата је следећим изразом:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n \tilde{X}_i$$

Тада променљива

$$Y = \frac{(\bar{X} - \mu_x)}{\sigma_x / \sqrt{n}}$$

конвергира ка стандардној нормалној расподели са средњом вредношћу  $\mu$  и стандардном девијацијом  $\sigma/\sqrt{n}$ , тј. за довољно велико  $n$  разлика између променљиве  $Y$  и стандардизоване нормалне променљиве може се занемарити из практичних разлога.

Према томе, да би генерисали узорак из стандардне нормалне расподеле, можемо узети  $n$  независних униформно расподељених случајних бројева  $\tilde{r}_i \in [0, 1)$  са средњом вредношћу  $E(\tilde{r}_i) = 1/2$  и варијансом  $E(\tilde{r}_i - \mu)^2 = 1/12$ . Случајна променљива:

$$Z = \frac{\frac{1}{n} \sum_{i=1}^n \tilde{r}_i - \frac{1}{2}}{\sqrt{\frac{1}{12}} / \sqrt{n}} = \frac{\sum_{i=1}^n \tilde{r}_i - \frac{n}{2}}{\sqrt{\frac{n}{12}}}$$

има нормалну расподелу са средњом вредношћу нула и варијансом један ( $E(Z) = 0$ ,  $E(Z - \mu)^2 = 1$ ), када  $n \rightarrow \infty$ . Задовољавајући резултати апроксимације добијају се за  $n=12$  у ком случају горња једначина постаје:

$$Z = \sum_{i=1}^{12} \tilde{r}_i - 6$$

Уколико је потребно генерисати узорке из нестандардизоване нормалне расподеле, потребно је користити следећи израз:

$$X = \mu + \sigma \cdot Z.$$

### П.9.5 Box-Muller-ов метод

Box-Muller-ов метод је егзактан метод који користи две независне псеудо-случајне променљиве  $r_1$  и  $r_2$  које се користе за генерисање стандардизованих нормално расподељених случајних променљивих коришћењем оба или једног од следећих израза:

$$Z_1 = \cos(2\pi \cdot r_1) \sqrt{-2 \ln(r_2)}$$

$$Z_2 = \sin(2\pi \cdot r_1) \sqrt{-2 \ln(r_2)}$$

Уколико је потребно генерисати узорке из нестандардизоване нормалне расподеле, потребно је користити следећи израз:

$$X = \mu + \sigma \cdot Z$$

Код методе сумирања потребно је за свако  $Z$  имати на располагању 12 различитих вредности  $\tilde{r}_i$ , док су код Box-Muller-ове методе потребне само две такве вредности. Међутим, овај метод захтева израчунавање квадратног корена, логаритма, косинусне или синусне функције, што захтева више рачунарског времена него што је потребно за одређивање једноставне суме. Такође, тачност резултата зависи и од уграђених функција (библиотечких подпрограма) за израчунавање квадратног корена, логаритма, као и синусне и косинусне функције.

## ЛИТЕРАТУРА

Aburdene, M. F. (1988), *Computer Simulation of Dynamic Systems*, Wm. C. Brown Publishers, Dubuque, Iowa.

Антић, Д. и Г. Голо (1996) *Програмски пакети за симулацију динамичких система*, Кантакузин, Крагујевац.

Adelsberger, H. H., U. W. Poch, R. E. Shannon and G. N. Williams (1986), *Rule-Based Object-Oriented Simulation Systems, Intelligent Simulation Environments*, Proceedings of the Conference on Intelligent Simulation Environments (eds. P.A. Luker and H.H. Adelsberger), SCS, 23-25. January, San Diego, CA, pp. 107-112.

Ahmed, S. V. and E. G. Roman (1985), *Extension in the Application of Expert System Concepts to Labor Management Negotiations*, Proceedings in the 1985 Summer Simulation Conference, Chicago, Illinois, SCS, San Diego, CA, pp. 702-703.

Altman, D. (1982), *Osnovi teorije diskretnog modelirawa i simulacije*, Marketing Iskra Delta.

Banks, J. & J. Carson II (1984), *Discrete-Event System Simulation*, Prentice Hall, Englewood Cliffs, New Jersey.

Banks, J., J. S. Carson II, J. Ngosy (1989), *Getting Started with GPSS/H*, Wolverine Software Corporation, Annandale.

Barr, E. A. Figenbaum (1981, 1982), *The Handbook of Artificial Intelligence*, Vol I-III, Vitman, London.

Bartley, P, B. L. Fox, L. E. Schrage (1987), *A Guide to Simulation*, Springer-Verlag, N.Y. Berlin, Haidelberg.

Basden, A. (1983), *On the application of expert systems*, International Journal of Man-Machine Studies, No. 19, pp. 461-477.

Bingulac, S., M. Stojić (1971) *An extension of IBM modeling language to the simulation of hybrid computers preprints*, IFIP Congres, F5-2.

Бингулац, С. (1983) *Опис програмског пакета CSMP 11/70*, Факултет организационих наука, Београд.

Birtwistle, G. M., O. J. Dohl, B. Myhrhang, K. Mygrard (1973), *Simula Begin*, Van Nostrand Reinhold, New York.

Bonnet, A. (1985), *Artificial Intelligence: Promise and Performance*, Prentice-Hall International, London.

Божиновић, Р. (1986), *Експертни системи из зборника радова "Управљање системима"*, редакција Р. Петровић, Научна књига, Београд.

Bramer, M. A. (1982), *A survey and critical review of expert systems research*, Introductory Readings in Expert Systems (ed. D. Michie), pp. 3-29, Gordon & Breach, London and New York.

Bratley, P., L. F. Bennett, L. E. Schrage (1983), *A Guide to Simulation*, Springer - Verlag, New York.

Campbell, J. A. (1986), *Principles of Artificial Intelligence, Artificial Intelligence: Principles and Applications*, (ed. M. Yazdani), Chapman and Hall, London.

Čerić, V. (1993), *Simulacijsko modeliranje*, Školska knjiga, Zagreb.

Чичак, М., С. Весковић, М. Станојевић (1997), "Симулациони модел транспортног процеса угља на индустријској железници Обреновац-Колубара", SYM-OP-IS '97, Зборник радова, стр. 603-606, Београд.

Clementson, T. (1982), *Extended Control and Simulation Language*, Clementson Com Ltd, Birmingham.

CSMP application description, IBM publication No H 20-282-0

Дајовић, С. и В. Вујчић (1981), *Математика II*, Привредна штампа, Београд

Daves, R. & R. O'Keefe (1989), *Simulation Modeling with Pascal*, Prentice Hall, London, New York.

Decker, H. D. & J. R. Geissler (1983), *Modeling and simulation net of agencies with BORIS*, In Model Acceptability, Pergamon Press, NY.

Dekker, L. & Zuidervaat J. (1976), *Automation of hybrid computer system and design aspects of future Hybrid processor*, Proc. of VIII ACIA World Congress, Delft.

Девеџић, В., З. Божовић (1994), *Експертни системи*, ИНФО – часопис за рачунарство и информатику, број 2/94, стр 20-29, Београд.

- Doukidis, G. I. (1987), *An antology on the homology of simulation with artificial intelligence*, Journal of the Operational Research Society, 38.
- Džigurski, O., *Hibridni procesor za simulaciju dinamičkih sistema*, doktorska teza, Elektrotehnički fakultet, Beograd.
- Evans, J. B. (1988), *Structures of Discrete Event Simulation: An Intoduction to the Engagement Strategy*, Ellis Horwood, Chichester.
- Feigenbaum, E. (1963), *The Simulation of Verbal Learning Behaviour, Computers and Thought* (eds. E. Feigenbaum, J. Feldman), McGraw-Hill.
- Fisher, M. J. W. (1982), *The application of visual interactive simulation in the management of continuous process chemical plants*, PH\_D Thesis, University of Warwick.
- Fishman, S. G. (1978), *Principles of discrete event simulation*, Wiley Interscience, New York.
- Fox, M. S. N. Husain, M. McRoberts and Y. V. Reddy (1989), *Knowledge-Based Simulation: An Artificial Intelligence Approach to System Modeling and Automating the Simulation Life Cycle, Artificial Intelligence, Simulation, and Modeling* (eds. L. Widman, K.A. Loparo and N.R. Neilsen), John Wiley & Sons, Inc., New York
- Gaines, B. R., M. L. G. Shaw (1985), *Expert Systems and Simulation*, Proceedings of the 1985 Conference on Artificial Intelligence, Graphs and Simulation (ed. G. Birtwistle), SCS San Diego, CA, (Jan.), pp. 95-101.
- Gaines, R. B. & M. L. G. Shaw (1985), *Expert Sistems and Simulation, AI, Graphic and Simulation* (ed. G. Birtwhistle), Society for Computer Simulation, San Diego, USA.
- Garzia, F. R., M. R. Garzia and B. P. Zeigler (1986), *Discrete-event simulation*, IEEE Spectrum 23, no. 12 (Dec.), pp. 32-36.
- Giloi, W. K. (1975), *Principles of Continuous System and Simulation*, B.G. Teubner, Stuttgart.
- Gordon, G. (1978), *Sistem Simulation*, Prentice - Hall, New Jersey.
- GPSS/H Reference Manual (1989), *Wolwerine Software Corporation*, Third Edition, Annandale.
- Granino A. Korn, John V. Wait (1978), *Digital Continuous - System Simulation*, Prentice - Hall, New Jersey.

Haddock, J. (1987), *An expert system framework based on simulation generator*, Simulation 48, no. 2 (Feb.), pp. 45-53.

Harmon, P. & B. Sawyer (1990), *Creating Expert Systems for Business and Industry*, John Wiley & Sons, Inc., New York.

Harmon, P., R. Maus and W. Morrissey (1988), *Expert Systems: Tools and Applications*, John Wiley & Sons, Inc., New York.

Hayes-Roth, F. D. A. Waterman, D. B. Lenat (1983), *Building Expert Systems*, Addison-Wesley, Reading, MA.

Holmes, M. W. (ed.) (1985), *Artificial Intelligence and Simulation*, Society for Computer Simulation, San Diego, USA.

Hunt, D. V. (1986), *Artificial Intelligence and Expert Systems Sourcebook*, Chapman & Hall, New York.

Hurion, R. D. (1978), *Visual interactive simulation, an aid to decision making*, Omega (6).

*IBM 1130 Continuous System Modeling Program. Program Description and Operations Manual*, SH20-0905. IBM Program Information Department, Hawthorne, N.Y.

*IBM 360 Continuous System Modeling Program. Program Description and Operations Manual*, SH20-0240. IBM Program Information Department, Hawthorne, N.Y.

Kerchoffs, E. J. H. and G. C. Vansteenkiste (1986), *The Impact of Advanced Information Processing in Simulation: An Illustrative Review*, Simulation 46, pp. 17-26.

Kerchoffs, E. J. H., G. C. Vansteenkiste, and B. P. Zeigler (1986), *AI Applied to Simulation. Proceedings of the European Conference at the University of Ghent*, February 25-28, 1985, Ghent, Belgium, Society for Computer Simulation, San Diego, CA.

Kleijnen, J. P. C. (1974), *Statistical Techniques in Simulation*, Vol I i II, Marcel Dekker, New York.

Korn, A. G. (1989), *Interactive Dynamic System Simulation*, McGraw-Hill Book Company.

Крчевинац, С., Ј. Петрић и др. (1982), *Методе планирња у сложеним организацијама удруженог рада*, Научна Књига, Београд.

Law, A. M. and J. S. Kelton (1982), *Simulation Modeling and Analysis*, McGraw-Hill, New York.



Ljung, L. (1976), *System Identification: Advanced and Case Studies*, Academic Press, New York, London.

Ljung, L. (1979), *Identification and System Parameter Estimation*, Pergamon Press, Oxford.

Marković, A. (1989), *Realizacija simulaconog jezika CSMP na programskom jeziku PASCAL*, Diplomski rad, FON, Beograd.

Марковић, А. (1994), *Спецификације симулационих модела коришћењем концепата вештачке интелигенције*, Магистарска теза, Факултет организационих наука, Београд.

Марковић, М., М. Станојевић, Н. Вучинић (1997), "Симулациони модел уског грла са приоритетним правцима", SYMOPIS '97, Зборник радова, стр. 639-641, Бечићи.

Марковић, М., М. Станојевић (1995), "Симулациони модел за утврђивање броја заустављених друмских возила испред пружног прелаза и њиховог времена чекања", Симпозијум о рачунарским наукама и информатици, YU INFO, Зборник радова, Књига 3, стр. 41-44, Брезовица.

Марковић, М., М. Станојевић, (1994) "Симулационо-аналитички модел за прогнозирање саобраћајних незгода на путно-пружним прелазима", Симпозијум о операционим истраживањима, Зборник радова "SYM-OP-IS", стр. 625-628, Котор.

Michaelsen, R. H., D. Michie & A. Boulanger (1985), *The Technology of Expert Systems*, Byte, Vol. 10, No. 4.

Милићевић, Л. А. Марковић, Б. Раденковић (1998) *Моделирање и симулација континуалних система у CSMP-W95/NT симулационом језику*, SYMOPIS '98, Херцег Нови, 21-24. септембар 1998, Зборник радова, стр. 627-630.

Милићевић, Л., А. Марковић, Б. Раденковић (1999) *CSMP-W95/NT – графичко окружење за симулацију континуалних система*, YU-INFO '99, Копаоник, 22-26. март 1999, Зборник радова.

Miller, P. L. (1986), *Expert Critiquing Systems: A Practise-Based Medical Consultation by Computer*, Springer Verlag, New York.

Minsky, M. (1985), *A Framework for Representing Knowledge*, The Psychology of Computer Vision (ed. P.H. Winston), McGraw-Hill.

Mohammed, J. & R. Simmons (1986), *Qualitative Simulation of Semiconductor Fabrication*, Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia, Pennsylvania, (ed. M.

Kaufmann), Los Altos, CA, pp. 794-799.

Moser, J. G. (1986), *Integration of artificial intelligence and simulation in a comprehensive decision-support system*, Simulation 47, no. 6 (Dec.), pp. 223-229.

Muetzelfeldt, A., A. Bundy, M. Uschold and D. Robertson (1985), *ECO - An Intelligent Front End for Ecological Modeling, AI Applied to Simulation*, Proceedings of the European Conference at the University of Gent, (eds. E.J.H Kerchoffs, G.C. Vansteenkiste and B.P. Zeigler), February 25-28, Ghent, Belgium, SCS, San Diego, CA, pp. 67-70.

Murray, K.J. & S. V. Sheppard (1988), *Knowledge-based simulation model specification*, Simulation 50:3, pp. 112-119.

Munitić, A. (1989), *Kompjuterska simulacija uz pomoć systemske dinamike*, Brodsplit, Split.

Naylor, B. et all. (1966), *Computer simulation techniques*, John Wiley & Sons, New York.

Newell and H. A. Simon (1972), *Human Problem Solving*, Prentice- Hall, Englewoods Cliffs.

O' Donovan, T. M. (1979), *GPSS Simulation Made Simple*, John Wiley & Sons, New York (chichester).

O'Keefe, R. (1986), *Simulation and Expert Systems - A Taxonomy and Some Examples*, Simulation 46, no. 1 (Jan.), pp. 10-16.

Oren, I. T. (1979), *Concepts for Advanced Computer Assisted Modelling and Simulation*, North Holland P.C. Amsterdam.

Parnas, D. L. (1969), *On the use transition diagrams in the design of a user interface for an interactive computer systems*, Proc.of the ACM National Conference pp. 379-386.

Paul, J. R. & S. T. Chew (1987), *Simulation modelling using an interactive simulation program generator*, Journal of the Operational Society, 38.

Paul, R. J. (1989), *Artificial Intelligence and Simulation Modelling, Computer Modelling for Discrete Simulation* (ed. M. Pidd), John Wiley and Sons, Chichester.

Pedersen, K (1989), *Expert Systems Programing: practical techniques for rule-based systems*, John Wiley & Sons, Inc., N.Y.

Пејовић, П. (1983) *Нумеричка анализа II*, стр. 57-59. Научна књига, Београд.

Пејовић, П. (1983), *Нумеричка Анализа, I (1 и 2)*, Научна Књига, Београд.

Пејовић, Т. П., Н. Парезановић (1972), *Аналоги електронски рачунари и њихова примена*, Математички Институт, Савремена рачунска техника и њена примена, Београд.

Петровић, Б. (1986), *Примена Z трансформације*, интерна скрипта, Факултет организационих наука.

Pidd, M. (1989), *Computer Modelling for Discrete Simulation* (ed. M. Pidd), John Wiley & Sons Ltd.

Poo, C. C. D. and P. J. Layzell (1990), *Enhancing software maintenance through explicit system representation*, Information and software technology, vol 32, no 3, april 1990, Butterworth & Co Ltd.

Poole, G. T. & J. Z. Szymankiewicz (1977), *Using Simulation to Solve Problems*, McGraw-Hill, London.

Pritsker, A. A. B. (1984), *Introduction to Simulation and SLAM II*, 2nd Edition, Halstead Book Press, New York.

QNX language reference manual, (1989).

Radenković, B., A. Marković i D. Vukmirović (1993), *An Expert System for the Specification of Simulation Model of Input/Output Zone of a High-bay Warehouse*, Second Balcan Conference on Operational Research, Thessaloniki, Greace.

Radenkovic, B., A. Markovic i M. Vulovic, (1997), *Simulation study of production process in sawmilling industry*, Yugoslav Journal of Operations Research (YUJOR), No 1.

Radenkovic, B., A. Markovic, Z. Radojicic (1996), *Production Process Simulation Model Monitoring Using the Computer Simulation*, Yugoslav Journal of Operations Research (YUJOR), Volume 6, No. 2, YU ISSN 0354-0243, pp. 257-265, Belgrade.

Раденковић, Б. (1984) *Програм за симулацију континуалних и дискретних система CSMP/MICRO*, Аутоматика 25 стр. 235-238, Београд.

Раденковић, Б. (1989), *Интерактивни симулациони систем за дискретну стохастичку симулацију организационих система и његова реализација на мини и микро рачунарима*, Докторска теза, Универзитет у Београду, ФОН.

Раденковић, Б. (1987), *Симулациони систем за дискретну – стохастичку симулацију индустријских система и његова реализација на мини и микро рачунарима*, Магистарска теза, Универзитет у Београду, ФОН.

Раденковић, Б., А. Марковић (1995) *Рачунарска симулација и симулациони језици*, скрипта, Факултет организационих наука, Београд.

Рајков, М. (1988), *Теорија динамике организационих и економских система*, КИЗ Култура.

Schmidt, J. W. (1984), *Introduction to Simulation*, Proceedings of the 1984 Winter Simulation Conference, Dalas, Texas, Society for Computer Simulation, San Diego, CA.

Schmidt, J. W. and R. E. Taylor (1970), *Simulation and Analysis of Industrial Systems*, Irwin, Homewood, ILL.

Schriber, T. J. (1974), *Simulation Using GPSS*, John Wiley & Sons, New York.

Schriber, T. J. (1991), *An Introduction to Simulation Using GPSS/H*, John Wiley & Sons, New York.

Shaw, M. L. G. & B. R. Gaines (1986), *A Framework for Knowledge-Based Systems Unifying Expert Systems and Simulation*, Proceedings of the Conference on Intelligent Simulation Environments (eds. P.A. Luker and H.H. Adelsberger), SCS Simulation Series 17, no. 1, San Diego, CA.

Spriet, A. J. & G. C. Vansteenkiste (1982), *Computer-aided modeling and simulation*, Academic Press.

Станојевић, М. (1982), "Симулациони модел уливања саобраћајног тока са споредног на главни пут", Симпозијум о операционим истраживањима, Зборник радова "SYM-OP-IS", стр. 477-484, Херцег Нови.

Станојевић, М. (1983), "Значај симулације у анализи понашања саобраћајних токова, надгледању и управљању саобраћајем", Зборник радова III Југословенског саветовања "Технике регулисања саобраћаја", стр. 173-188, Нови Сад.

Станојевић, М. (1983), "Симулациони модел несигналисане изоловане раскрснице", Зборник радова III Југословенског саветовања "Технике регулисања саобраћаја", стр. 201-209, Нови Сад.

Станојевић, М. (1987), *"Анализа ефикасности функционисања изоловане Y раскрснице моделирањем на дигиталном рачунару"*, Техника-Саобраћај, 34, бр 12, стр. 1233-1236, Београд.

Станојевић, М. (1992) *"Место и улога моделирања у области ПТТ саобраћаја"*, X научно стручни семинар за иновацију знања у области нових технологија у ПТТ, Зборник радова, 209-224, Саобраћајни факултет, Београд.

Станојевић, М. (1993) *"Компаративна анализа појединих класа система масовног опслуживања у поштанском саобраћају"*, XI научно стручни симпозијум за иновацију знања у области нових технологија у ПТТ, Зборник радова, стр. 25-30, Саобраћајни факултет, Београд.

Станојевић, М. (1995) *"Модели за утврђивање квалитета услуга у поштанском саобраћају"*, XIII научно стручни симпозијум за иновацију знања у области нових технологија у ПТТ, Зборник радова, стр. 103-112, Саобраћајни факултет, Београд.

Станојевић, М. (1996) *"Симулациона анализа функционисања шалтерске службе у ПТТ"*, XIV научно стручни симпозијум за иновацију знања у области нових технологија у ПТТ, Зборник радова, стр. 121-136, Саобраћајни факултет, Београд.

Станојевић, М. (1998) *"Симулација као подршка одлучивању у поштанском саобраћају"*, XVI симпозијум о новим технологијама у поштанском и телекомуникационом саобраћају, Зборник радова, стр. 159-168, Саобраћајни факултет, Београд.

Станојевић, М., С. Станковић (1981), *"Симулациони модел саобраћајног тока на деоници пута"*, Зборник радова, III Међународни симпозијум "Компјутер на свеучилишту", стр. 351-351, Цавтат.

Станојевић, М., С. Станковић (1995), *"Симулациона анализа основних променљивих саобраћајног тока"*, Симпозијум о рачунарским наукама и информатици, YU INFO, Зборник радова, Књига 3, стр. 24-29, Брезовица.

Станојевић, М., С. Станковић (1998), *"Симулациона анализа понашања возила у воду"*, Симпозијум о рачунарским наукама и информатици, YU-INFO, Зборник радова, стр. 515-520, Копаоник.

Stefic, M., J. Aikins, R. Blazer, J. Benoit, L. Birnbaum, F. Hayse-Roth, and E. Saceroti (1982), *The organization of expert systems: a tutorial*, Artificial Intelligence, No. 18, pp. 135-173.

TUTSIM (1989) *User's Manual*, IBM PC Computer.

Вуковић, Н. (1998), *PC Статистика и вероватноћа*, ФОН ИД, Београд.

Вушковић, М. (1984), *Симулација и симулациони језици*, Интерна скрипта, Факултет организационих наука.

Весковић, С., М. Марковић, М. Станојевић, Н. Вучинић (1998), "Симулациони модел транспортног процеса на затвореној индустријској железници", Симпозијум о рачунарским наукама и информатици, YU-INFO, Зборник радова, стр. 521-525, Копаоник.

Вукадиновић, С., Ј. Поповић (1996), *Математичка статистика*, Универзитет у Београду, Београд.

Вукадиновић, С., Ј. Поповић (1985), *Метода Монте-Карло*, Саобраћајни факултет, Београд.

Weizenbaum, J. (1980), *Моћ рачунара и људски ум*, Rad, Beograd.

Widman, L. E. and K. A. Loparo (1989), *Artificial Intelligence, Simulation and Modeling: A Critical Survey*, In Artificial Intelligence, Simulation, and Modeling (eds. L. Widman, K. Loparo, and N. Nielsen), John Wiley & Sons, New York.

Winograd, T. (1972), *Understanding Natural Language*, Academic Press, London.

Winston, P. H. (1984), *Artificial Intelligence*, Addison-Wesley, Reading, MA.

Zeigler, P. B. (1976), *Theory of modeling and Simulation*, Wiley, NY.

Zeigler, P. B. (1984), *Multifaceted modeling and discrete event simulation*, Academic Press, London.

Zikic, A. and B. Lj. Radenkovic (1997), *The New Approach to Teaching Discrete Event System Simulation*, International Journal of Engineering Education, Volume 12, Number 6.

Zikic, A., B. Radenkovic (1993), *An application of GPSS/FON in teaching simulation*, International journal of applied engineering education, 247-253, Great Britain.

Žikić, M. A. (1989), *Practical Digital Control*, Edison Wesley and John Wiley & Sons.

Zrnic, Dj., N. Cupric, B. Radenkovic (1992), *A study of material flow systems (input/output) in high-bay warehouses*, Journal of Production Research, vol 30, no. 9, 2137-2149.

Зрнић, Ђ. и Д. Петровић (1994), *Стохастички процеси у транспорту*, Машински факултет Универзитета у Београду.

Зрнић, Ђ., Д. Савић (1987), *Симулација процеса унутрашњег транспорта*, Машински факултет, Београд.