1. Моделирање и модели, врсте модела

Моделирање представља један од основних процеса људскога ума. Оно се најчешће посматра као најзначајније концептуално средство које човеку стоји на располагању. 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. 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.

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).

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.

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.

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:

- Некомплетан опис модела (уколико модел не садржи све ситуације које могу да наступе)
- Неконзистентан опис модела (уколико су у опису модела за исту ситуацију предвиђена два или више правила чијом се применом добијају контрадикторне акције)
- Нејасан опис модела (ако у једној ситуацији треба обавити две или више акција, а при томе није дефинисан њихов редослед)

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. Препорука при изради модела*:*

1. Граница система са околином мора бити одабрана тако да систем, односно његов модел, обухвата само феномене од интереса.

2. Модели 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.

3. Модел не сме сувише да поједностави проблем.

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.

6. Потребна је провера логичке и квантитативне исправности модела.

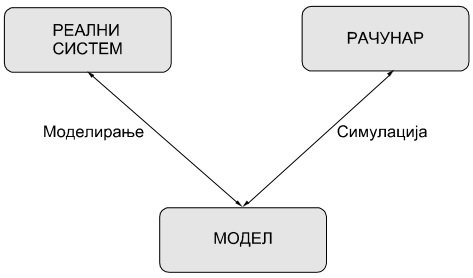
3. Рачунарска симулација

За извесне, углавном случајне улазе, посматрају се одговарајући излази модела. Овај процес назива се симулација. Под симулацијом се подразумева процес изградње апстрактних модела за неке системе или подсистеме реалног света и обављање већег броја експеримената над њима. Када се обавља на рачунару, ради се о рачунарском моделирању и симулацији. У моделирању рачунари се користе у две сврхе: у развоју модела и у извођењу прорачуна на основу створеног модела. Израз моделирање и симулација изражава сложену активност која укључује три елемента: реални систем, модел и рачунар.

Реални систем је уређен, међузависан скуп елемената који формирају јединствену целину и делују заједнички како би остварили задати циљ или функцију. Он 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, то се под моделом може подразумевати скуп инструкција (програм) који служи да се генерише понашање 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.



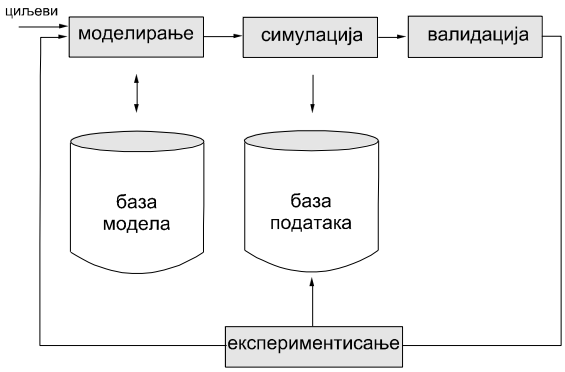
Релације моделирања и симулације

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.

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.



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.

4. Историјски преглед развоја симулације

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.

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

5. 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. 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).

Симулациони модели прикупљају податке о променама стања система и излаза, фокусирајући се на понашање индивидуалних компоненти система

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).

Симулациони модели најчешће су модели динамичких система, тј. система који се мењају у времену. Ови су модели углавном дати у облику концептуалних (структурних) и рачунарских модела. Најчешће, али не и генерално, симулациони модели су модели система који се не могу описати нити решавати математичким средствима.

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.

6. Потреба за симулацијом

Постоји више разлога, али су најважнији следећи*:*

- Експеримент над реалним системом може да буде скуп или чак немогућ
- Аналитички модел нема аналитичко решење (нпр., сложенији модели масовног опслуживања)
- Систем може да буде сувише сложен да би се описао аналитички (нпр. системом диференцијалних ј-на).
- 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
- Понекад треба симулирати услове под којима наступа разарање система
- Време може да буде врло јак разлог да се прибегне симулацији. При симулацији, време се може сажети. У другим случајевима, време се може знатно продужити.
- 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.

7. Могућности примене симулације, предности и недостаци симулације

Навешћемо неколико ситуација када се симулација може успешно применити*:*

- 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.

Као основне предности коришћења симулације наводе се следеће*:*

- 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.
- 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*:*

- 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
- 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.

8. 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.
Основни кораци симулационог процеса:

1. *Дефиниција циља симулационе студије*
   Дефиниција жељеног циља и сврхе студије: проблем који треба решити, 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.

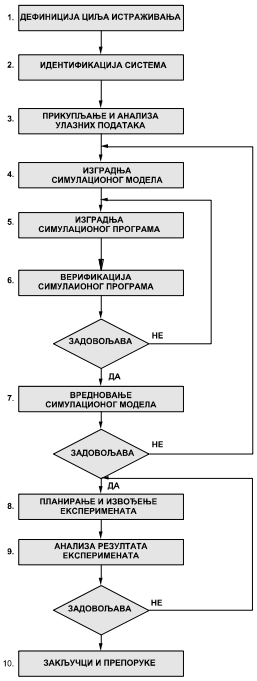
6. *Верификација симулационог програма*
   Тестирање симулационог програма према поставкама симулационог модела (појединачне процедуре, генерисање случајних променљивих, повезивање процедура).

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).

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).

9. *Анализа резултата експеримената*
   Статистичка анализа резултата симулационих експеримената. Током анализе резултата може се показати потреба за извођењем додатних експеримената.

10. *Закључци и препоруке*
    Презентација релевантних резултата на основу којих се могу донети одговарајуће одлуке.



9. 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.

Детерминистички модели су они чије се понашање може предвидети, односно у којима је ново стање система у потпуности одређено претходним стањем. **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. Стање система S_n може променити у једно од стања S'_{n+1}, S''_{n+1} или S'''_{n+1} под утицајем активности *A*.

У **Дискретним моделима** стање система се мења само у појединим тачкама у времену (нема континуалне промене стања). Такве промене се називају догађаји.

У **Континуалним моделима** променљиве стања се мењају континуално у времену. Пример континуалне промене је лет авиона чији се положај и брзина мењају континуално у времену.

10. Врсте симулационих модела

Постоје 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.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 (статистичка симулација), као што јој и име каже, повезана је са случајним феноменима. Она је статички тип симулације код које 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:

1. *Детерминистички проблеми које је тешко или скупо решавати*
   Типичан пример овога типа је рачунање вредности одређених интеграла који се не могу решити аналитички.

2. *Сложени феномени који нису довољно познати*
   За њих је карактеристично да није познат начин узајамног деловања између елемената већ су познате само вероватноће његовог исхода.

3. *Статистички проблеми који немају аналитичка решења*
   Процене критичних вредности или тестирање нових хипотеза.

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. Разликујемо три основна типа континуалних симулационих модела

1. *Модели који се описују обичним диференцијалним једначинама (системи обичних диференцијалних ј-на)*
   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 $\left(y = y(t)\right)$ једне независно променљиве (t), изражавају се математички једначинама у којима се, поред независно променљиве и непознате функције, јављају (обавезно) и изводи те функције $\left(dy/dt\right)$.

2. *Модели који се описују системима парцијалних диференцијалних једначина*
   Парцијалне диференцијалне једначине садрже више од једне независне променљиве $\left(x_j\right)$ по којима се траже изводи зависне променљиве. Типични примери су проблеми аеродинамике, хидродинамике и метеорологије.

3. *Модели динамике система (System Dinamics)*
   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. Модели са повратном везом користе се најчешће за моделирање инжењерских, биолошких, друштвених и економских система.

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.

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.

11. Класификација модела

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. 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.

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 промењиве и сви неаутономни модели називају се отворени модели. Неаутономни модели се такође деле на две групе: са и без излазних променљивих.

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 непребројивог, односно континуалног скупа. У складу са тим разликујемо три класе модела:

- *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.
- *Модели са континуалним стањима* - Код ових модела све три врсте описних променљивих узимају вредности из подскупова реалних бројева.
- *Модели са мешовитим стањима* - Поједине променљиве узимају вредности из разних подскупова чији су елементи дискретне вредности, а остале променљиве из различитих подскупова реалних бројева.

Класификација у односу на природу опсега вредности променљиве "време" - Скуп вредности које се додељују променљивој "време" може бити пребројив или непребројив. Стога разликујемомоделе 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.

Класификација у односу на временску зависност модела - 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.

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.

Класификација у односу на линеарност - Линеарни модели мењају стања и дају излазе поштујући законитости линеарних трансформација.

Класификација према врсти рачунара - Три врсте рачунара се могу користити за симулацију: аналогни, дигитални и хибридни

12. 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 објекат неке класе у оквиру неког формализма, параметрима формализма додељују се 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.

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.

Асоцијација

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

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.

13. 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:

$$\boldsymbol{s}' = f\left(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{p}, t\right)$$
$$\boldsymbol{y} = g\left(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{p}, t\right)$$
$$\boldsymbol{s}(t_0) = \boldsymbol{s}_0$$

Где су:

$\boldsymbol{s}$ - вектор стања димензије n ,

$\boldsymbol{u}$ - вектор улаза димензије m,

$\boldsymbol{y}$ - вектор излаза димензије k,

$\boldsymbol{p}$ - вектор састављен од n_p непознатих параметара,

f и g су одговарајуће векторске функције,

$\boldsymbol{s}_0$ - 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. 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.

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.

Идентификабилност система и параметара зависи од два фактора:

1. *Специфичнсти 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.

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. 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.

14. Статистички приступ процени параметара 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 медијуму, 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.

Оцена средње вредности случајне променљиве

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

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:

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

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

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:

$$\hat{a}_0 = 0$$

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

где је: y_k - текући узорак (*к =1,2,..,N*)

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

15. Оцена непознатог параметра по методи најмањих квадрата

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

$$y_i = x_i a + e_i \qquad i = 1,2,...,N$$

Претпоставке су:

x_i - познате вредности, e_i - грешке мерења

Потребно је проценити непознати параметар a користећи доступне резултате мерења.

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:

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

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.

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

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

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.

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

тада се оцена параметра a може изразити као: $\hat{a}_k = \hat{a}_{k-1} - k_k \left(\hat{a}_{k-1} - y_k \right)$

16. Процена К непознатих параметара

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

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

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

Редни број експеримента	x_1	x_2	.....	x_k	y
1	x_{11}	x_{12}	.....	x_{1k}	y_1
2	x_{21}	x_{21}	.....	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 $x_{j1}, x_{j2}, \ldots, x_{jk}$, тада се као резултат добија y_j. Одређена одступања (грешке) ћемо обележити са e_j. На основу изложеног, за *n* експеримената и *k* променљивих добија се ситем једначина облика:

$$y_1 - \left(a_1 x_{11} + a_2 x_{12} + \ldots + a_k x_{1k}\right) = e_1$$

$$y_j - \left(a_1 x_{j1} + a_2 x_{j2} + \ldots + a_k x_{jk}\right) = e_j$$

$$y_n - \left(a_1 x_{n1} + a_2 x_{n2} + \ldots + a_k x_{nk}\right) = e_n .$$

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

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

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

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

Обележимо са ***y*** *n*-димензиони вектор резултата експеримента, са ***a*** *k*-димензиони вектор непознатих параметара, а са ***X*** матрицу мерења улаза експеримента димензије *(n x k).*

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

Критеријумска функција се тада може представити у облику $J = \left(\boldsymbol{y} - \mathbf{X}\boldsymbol{a}\right)^T \left(\boldsymbol{y} - \mathbf{X}\boldsymbol{a}\right)$

Применом правила диференцирања за матрице и векторе добија се

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

одакле се добија $\mathbf{X}^{\mathbf{T}}\mathbf{X}\boldsymbol{a} = \mathbf{X}^{\mathbf{T}}\boldsymbol{y}$

односно $\boldsymbol{a} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y}$ је релација која даје непознате параметре модела.

17. Валидација и верификација модела

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. Ова два процеса се 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.

Први корак у изградњи модела обухвата посматрање реалног система и интеракција између његових различитих компоненти, као и прикупљање података о понашању система.

Други корак у изградњи модела је формирање концептуалног модела, скупа претпоставки за компоненте модела и структуру система, као и хипотеза за вредности параметара 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.

18. Валидација симулационих модела

Проблем валидације модела настаје пошто се у фази изградње модела уносе многе апроксимације реалног система. 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 фукција треба да буде приближна "оригиналној" функцији у области где ће систем вероватно функционисати.
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:

   a) Временска агрегација
   b) Међу-секторска агрегација
   c) Агрегација помоћних средстава

5. Апроксимација стационарности
   Ствар поједностављује чињеница да се параметри и друге карактеристике система не мењају у времену.

Све претпоставке и апроксимације које уносимо у модел могу, на известан начин, да доведу у питање валидност тог модела. 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.

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. Валидација је 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.. 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.

Практични приступ процесу валидације
Naylor и Finger формулисали су практичан приступ проблему валидације, који се састоји се из три фазе:

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.

За проверу валидности модела, може се користити и анализа осетљивости. То је поступак тестирања осетљивости модела на различите претпоставке и промене улазних величина. 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.

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).

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. 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. У неким случајевима могу постојати одређени подсистеми планираног система, па је могуће спровести само делимичну валидацију модела.

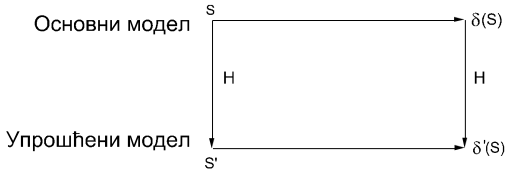
19. Формални критеријум за утврђивање валидности модела

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.

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.

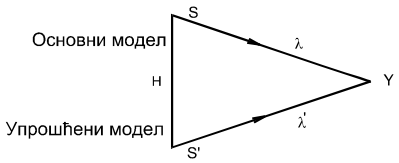
2. Очување функције прелаза стања
   Ако стању s основног модела одговара стање s' упрошћеног модела (тј. применом пресликавања H на стање s добија се стање s'), тада и следећа стања у која ће модел прећи у наредниом тренутку посматрања морају одговарати једно другом. Следеће стање основног модела ће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



   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.

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 (λ) да би се добио упрошћени модел за стање s), даје исти излаз Y као и примена излазне функције упрошћеног модела (λ') на стање s'. 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.



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

20. Верификација симулационих модела

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.

21. Аналогни рачунар као средство за симулацију

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.

Идеја аналогних рачунара састоји се у изналажењу таквих физичких објеката који ће моћи да се користе за анализу математичких модела помоћу којих су ти објекти математички описани. Принципи рада аналогних рачунара засновани су на физичким законима који важе за онај физички објект који се користи у рачунске сврхе.

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. Аналогни рачунари (АР) деле се најчешће у две групе и то:

- 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

Универзални аналогни рачунари граде се као: - Репетитивни АР раде великом брзином и на њима се решење обично понавља 20 до 1000 пута у секунди. Велика учестаност решења омогућава његово визуелно праћење на екрану осцилоскопа. - Спори АР раде у тзв. реалном времену, тј. тако што се решење добија само у једном циклусу рада рачунара.

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.

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. Рачунски елементи којима су опремљени аналогни рачунари деле се у шест група (према функцији):

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.

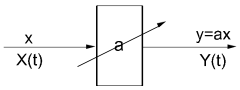
Главни елементи електронског аналогног рачунара су:

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.

22. Основни елементи електронског аналогног рачунара

Потенциометар

Потенциометар је рачунски елемент ЕАР који омогућава да се реализује математичка операција множења промењиве величине са константом, тј. $y(t) = ax(t)$ 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.



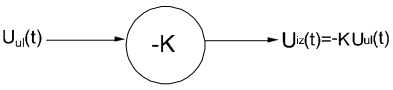
Потенциометри су снабдевени скалама од 0 до 1 и не може никад подесити идеално тачно, јер свако постављање клизача на неку дужину l уноси извесну грешку која је мања, што је дужина отпорника већа.

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.

Појачавач

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.



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.

$$U_{iz} = -kU_{ul}, \quad k >> 1$$

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.

Сабирач

Математичка операција алгебарског сабирања *n* променљивих величина коју желимо да извршимо, дата је изразом: $y(t) = -\sum_{i=1}^{n} a_i x_i(t)$ где су a_i константе којима се множи одговарајућа променљива величина $x_i(t)$.

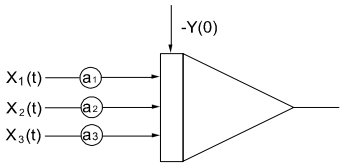
Физичке величине $X_i(t)$, $(i = 1,2,...,n)$ које одговарају променљивим $x_i(t)$ су улазни напони сабирача. Напон $Y(t)$ као излазна величина из сабирача одговара математичкој величини $y(t)$. Сабирач код електронских аналогних рачунара може да обави две математичке операције:

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

Интегратор

Математичка операција која се жели реализовати помоћу интегратора има облик: $y(t) = -a\int_0^t x(t)\,dt$

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



$$Y(t) = -\sum_{i=1}^{3} a_i \int_0^t x_i(t)dt - Y(0)$$

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.

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

Множач

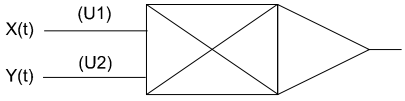
Нека је математичка операција множења две променљиве дата изразом

$$z(t) = x(t)\,y(t)$$

где су $x(t)$ и $y(t)$ променљиве величине са временом. Рачунски елемент на електронском аналогном рачунару који може да обавља математичку операцију 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 множача 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.



$$Z(t) = -\frac{X(t)Y(t)}{U}$$

U – рачунски напон

Сервомножач

Идеја сервомножача потекла је од особине потенциометра да улазни напон множи са неком константом мањом од јединице. За реализацију множача било је потребно направити електромеханички уређај, који може да помера клизач потенциометра у зависности од неке друге променљиве величине, у нашем случају $Y(t)$. Позициони сервомеханизам се састоји од дискриминатора, појачивача, електромотора, механичког редуктора и потенциометра.

Диодни множачи Диодни множачи припадају групи електронских множача, а састављени су од диодних генератора неких фиксних функција и рачунских појачивача. Најчешће се користе диодни генератори функција који дају следеће фиксне функције: $F(X)=X^2$, $F(X)=\ln X$, $F(X)=e^x$

Генератори функција У аналогној рачунарској техници, физички уређај који може да оствари математичку операцију $y = f(x)$ назива се генератор функција.

23. Дигитални рачунар као средство за симулацију

Структура класичног дигиталног рачунара састоји се од процесора, меморије и улазно-излазних. 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.
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
3. Анализа резултата симулације врши се уз помоћ експертних система коришћењем одговарајућих база знања. 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 софтверкој реализацији малих рачунара већ данас омогућавају јефтину и ефикасну примену симулације у малим и средњим предузећима.

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.

24. Хибридни рачунар као средство за симулацију

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. Код хибридног рачунара аналогни рачунар изводи имплицитне операције и извршава интеграљење док дигитални рачунар извршава алгебарске функције.

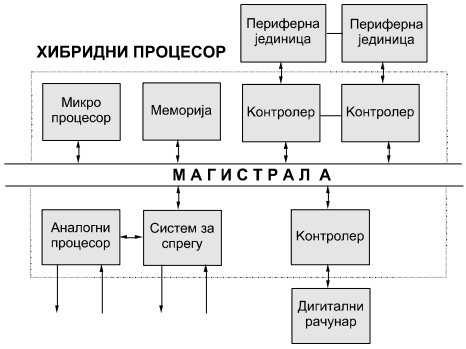
За време паралелне обраде, обрада свих података почиње и завршава се у истом тренутку. Ова се обрада обавља на аналогном процесору, који сачињава више оперативно независних подпроцесора.

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.



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.

25. Симулација континуалних система – Формални модел

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.

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
δ - функција преноса $\delta : U \times S \to S$
λ - функција излаза $\lambda : U \times S \to Y$
S_0 - скуп почетних стања (почетних услова)

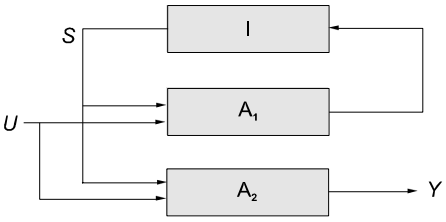
Процес симулације континуалних система може се описати једначинама *континуалног аутомата* :

$$S(t) := I \times A_1 \cdot \{U(t), S(t)\}$$

$$Y(t) := A_2 \cdot \{U(t), S(t)\}$$

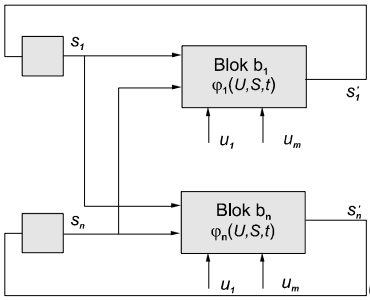
Преносна функција, у горњим једначинама, раздељена је у две врсте оператора: оператор A_1 и A_2 су алгебарске функције, док оператор *I* представља интеграцију.

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.



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_1 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: $U = [y_1(t),...,y_m(t)]$ и $S = [s_1(t),...,s_n(t)]$. Овакав модел представља систем диференцијалних једначина првог реда.

Функција једне или више променљивих - блок

Функција једне променљиве φ 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:

$$\varphi : X \to Y$$

Функција више променљивих φ је пресликавање облика

$$\varphi : X \times X \times X \times ... \times X \to Y.$$

Блок се може представити уређеном тројком

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

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

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)*

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:

скуп улаза $U = X - Y$

скуп излаза $R = Y - X$

скуп веза $C = X \cap Y$.

26. Симулација континуалних система помоћу аналогних рачунара

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 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:

$$S(t) := I \times A_1 \cdot \left\{ U(t), S(t)' \right\}$$

$$Y(t) := A_2 \cdot \left\{ U(t), S(t)'' \right\}$$

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

27. Симулација континуалних система помоћу дигиталних рачунара

Основна разлика између аналогног и дигиталног рачунара је 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. 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 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 $t_n = t_0 + nh,\ n = 0,1,..., N$. 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, монотоно 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 симулационом алгоритму н-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:

$$S(t_{n+1}) = I^* \times A_1 \cdot \{U(t_n), S(t_n)\}$$

$$Y(t_n) = A_2 \cdot \{U(t_n), S(t_n)\};\ \ n = 0,1,..., N$$

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 $s_i'(t_{n-r})$, $r = 0,1,...k$. 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 $n+1 \to n$.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.

28. 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.*

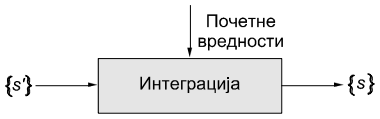
Свака таква метода је једна апроксимација стварне интеграције дискретном алгебарском шемом, те је стога најважније питање, питање тачности и стабилности изабране методе. Посебну тешкоћу овде представља чињеница да је појава грешке у тесној вези са изабраним континуалним системом за симулацију, а не само у вези са одабраном методом за решавање. Проблем нумеричке интеграције састоји се у томе да се дефинише неки оператор *F* тако да једначина

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

апроксимира на датом интервалу $\left(t_n, t_n + h\right)$ 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 *к*-корачном интеграционом формулом. Наведена једначина може се назвати *дискретни блок за интеграцију* и може се представити сликом 6.4. Величина интервала *h* не мора бити константна, већ се може мењати.



Слика 6.4 Дискретни блок за интеграцију

У једнокорачне интеграционе методе спадају:

1. *Ојлерова (*Eulier*) метода* $s_{n+1} = s_n + hs'_n + O\left(h^2\right)$

2. *Метода трапеза* $s_{n+1} = s_n + (h/2)\left(s'_n + s'_{n+1}\right) + O\left(h^3\right)$

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

$$\tilde{s}_{n+1} = s_n + hs'_n$$

$$s_{n+1} = s_n + \frac{h}{2}\left(s'_n + \tilde{s}'_{n+1}\right) + O(h^3)$$

Свака од ових метода формира диференцну једначину, помоћу које се може израчунати низ $\{s_n\} = \{s_0, s_1,..., s_n\}$, чији елементи представљају приближне вредности решења $s(t_0)$, $s(t_1)$.... 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 интеграли. 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},... итд.

Општа формула за вишекорачну методу интеграције има облик:

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

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.

29. 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 $t_n = t_0 + nh,\ n = 0,1,...N$. Коначна, ненулта вредност $h = t_{n+1} - t_n$ назива се корак интеграције. Корак интеграције може али не мора бити константан.

- $f^*(t)$-дискретизована функција $f(t)$(по времену и по нивоима ε)
- ε-најмања вредност са којом може да се престави 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:

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

Ако је ΔT (T или h) довољно мало, тада приближно важи

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

Што је мањи интервал ΔT, 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).

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. При великим интервалима интеграције губи се 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.

Превођење континуалног модела у модел са дискретним 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.

30. 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:

$$y_1 = f_1(x_1)$$

$$x_1 = f_2(y_1)$$

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

$$y = f(x)$$

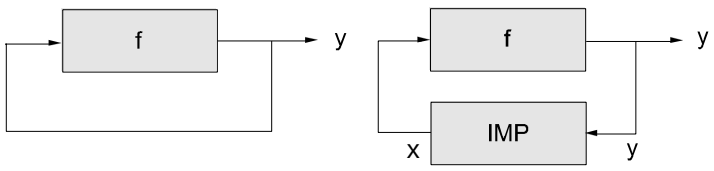
$$x = IMP(y)$$

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

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

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)

31. Рачунарска реализација симулатора

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

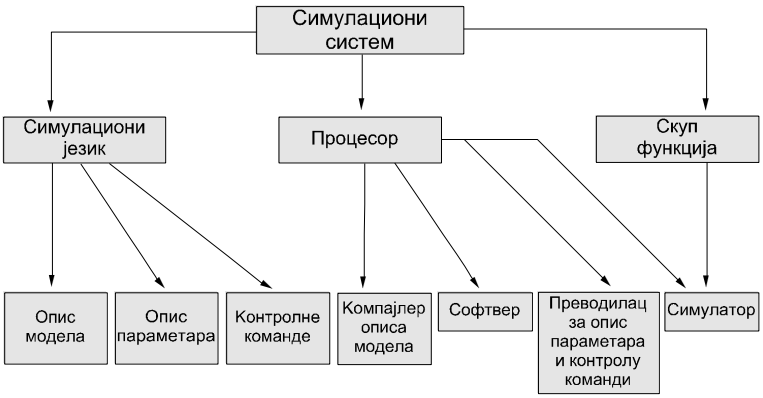
Симулациони језик служи за опис:

- 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.



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
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.

32. Процесор симулатора

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. Преводилац 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.

33. Симулација дискретних догађаја – Формални Модел

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 *догађај* наступа у одређеном тренутку времена.

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,

δ - квази-преносна функција;

задаје се следећим функцијама:

δ^{Φ} - пресликава $S \to S$;

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

δ^{ex} - пресликава $U \times S \times T \to S$;

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

λ - излазна функција; пресликава $S \to Y$

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

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.

34. 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

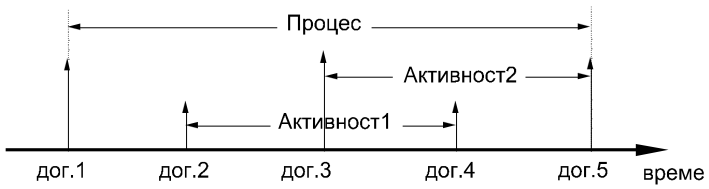
- промене атрибута појединих објеката система (опслужилац у систему постаје слободан или заузет).

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.



35. Развој симулације дискретних догађаја

Кључни елементи развоја симулацоних програма у симулацији система са дискреним догађајима су: механизам помака времена, приступи генерисању догађаја и основне стратегије симулације.

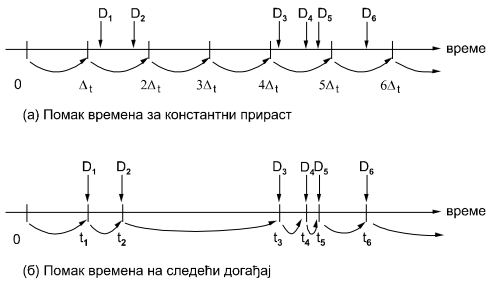
Механизам помака времена
У симулацији система са дискретним догађајима користе се два основна механизма помака времена: помак времена за константни прираст и помак времена на наредни догађај.

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.

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.



Слика 7.2 Механизми померања времена у симулацији дискретних догађаја

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.

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.

Генерисање догађаја

У рачунарској имплементацији симулационоог 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.
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.

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.

36. Стратегија распоређивања догађаја у СДД

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.

37. Процедура сканирања активности у СДД

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*. 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.

38. Стратегија интеракција процеса у СДД

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

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 активностти.

- 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.

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 i SIMULA. Због комбиновања добрих особина распоређивања догађаја и сканирања активности, интеракција процеса представља логичан избор за пројектовање модерних симулационих система.

39. Вештачка интелигенција и симулација

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.

Вештачка интелигенција је научна дисциплина посвећена "опремању" рачунара интелектуалним способностима као што су опажање, резоновање и учење. 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. 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.

Разлике: 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. Симулација захтева активности које и даље захтевају интензиван људски рад, а то су: формулисање проблема и анализа резултата симулације

40. 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 емулације. 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* і *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.

Основу досадашњих истраживања у области ВИ представља систем физичких симбола који су поставили *Newell* і *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, са мање детаља. *Newell* і *Simon* су увели систем физичких симбола којима се симулира човеков мисаони апарат, а сам мисаони процес представљен је као манипулација физичким симболима. 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 ревођење 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. (Има још у књизи...)

41. Основни концепти Вештачке Интелигенције

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.

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).

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

42. Основни елементи Вештачке Интелигенције

Хеуристичко претраживање

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. 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.

Представљање знања

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.

Логичко закључивање

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. .

Језици и алати вештачке интелигенције

Вештачка интелигенција је настала као експериментална наука са циљем да развије рачунарске програме који би опонашали интелигентно (људско) понашање. 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.

43. Основна подручја Вештачке Интелигенције

Системи вештачке интелигенције могу се класификовати у три основна подручја: експертни системи, природни језици и роботика.

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 домашаја конвенционалних рачунарских програма.

Друго подручје, названо системи природних језика, обухвата програме који препознају природни језик корисника.

Трећи тип програма ВИ обухвата једноставне системе за перцепцију вида, говора и додира. Рачунарски визуелни системи могу да интерпретирају визуелне сцене или да доносе закључке о квалитету или физичкој орјентацији одређених објеката.

44. Експертни системи

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. 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.

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, преводећи 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. 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.

45. 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. 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 резоновања" 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.
♦ 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 омогућују 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.

46. Експертни 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. 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. Проблеми који се решавају коришћењем ЕС-а најчешће су слабо структуирани, те не подлежу математичком моделирању и формализацији.
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.

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

47. Структруа експертних система

Основни саставни делови сваког ЕС, без обзира на величину, сложеност, начин коришћења и подручје примене су: база знања, механизам закључивања и радна меморија. У већини случајева, ЕС имају и интерфејс према кориснику.
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.

48. Представљање 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. Највише су се примењивале следеће три технике за представљање знања:

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

Задовољавају три критеријума за представљање знања:Могућност проширивања, Једноставност и Експлицитност.

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. Правило представља логичку релацију између елемената проблемског подручја и може се изразити на следећи начин: "ако" X "тада" Y што значи : "ако важи предпоставка 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).

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. 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. Продукциона правила омогућују да се прикаже и неизвесност у доношењу одлука. 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.

Семантичке мреже

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 дређени однос. 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. 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).

Оквири

Оквир је комплексна структура помоћу које се може представити неки појам или објекат. 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. Сваки атрибут има низ својстава, која не морају 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. 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 подоквире (subframes), који поред свих атрибута оквира у којима се налазе и које од њих наслеђују, могу да имају и неке специфичне атрибуте.

49. 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. 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. Механизам закључивања током тражења решења проблема може да захтева од корисника да унесе додатне податке уколико је то потребно.

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-maching), и може се вршити на разне начине. Две најпознатије стратегије претраге базе знања су уланчавање унапред и уланчавање уназад.

Уланчавање унапред полази од премиса правила. У сваком цилкусу, 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) конфликата. 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. Један конкретан узорак задовољеног правила у складу је са тачно једним подскупом скупа података у радној меморији који задовољавају премисе правила.

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. 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 унзад механизам закључивања увек полази од неког претпостављеног решења (циља), механизми закључивања који користе ову стратегију често се називају "механизми вођени циљевима" за разлику од механизама са претраживањем унапред који су "вођени подацима".

50. Симулација заснована на знању

У традиционалном приступу, 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. 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".

С друге стране, појам "симулација у великом", 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.

Комплексност проблема, присутна у реалним апликацијама, омогућава коришћење ВИ у симулацији на два начина. 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.

51. Симулациони процес и експертни системи

Могуће примене експертних система у појединим фазама симулационог процеса:

Изградња модела - То су: експертска помоћ новим корисницима, побољшање форми представљања реалних система и смањење обима израчунавања у симулацији.

Процена параметара - Процена параметара модела и улазних података.

Валидација и верификација - Валидација и верификација модела су експертски послови за које се могу формулисати прагматична правила за: отклањање грешака, давање савета, аутоматско тестирање модела, анализу погрешних резултата.

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. Спецификација симулационог модела најчешће захтева детаљно знање о ограничењима и релацијама које проистичу из одређеног домена или система у оквиру неког домена.

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.

52. Развој експертског 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.

Уграђивање правила

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:

- 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.

Област акција ентитета

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.

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. 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.

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 и може се у општем облику приказати као: ПРЕ <акција> УСЛОВ <услов>.

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. Овај тип правила се може представити у општем облику као: НАКОН <акција> AND <услов> CALL <функција(аргумент)>.

Област атрибута ентитета

Атрибут, као и сваки податак, треба заштитити од неконзистетних вредности. Две врсте неконзистентних вредности често се везују за атрибуте: неконзистентан тип и опсег вредности. Правило о типу ..., Правило о опсегу вредности .... и Правило о зависности атрибута ограничава опсег дозвољених вредности атрибута, али се опсег вредности одређује на основу услова који га ставља у зависност од вредности другог атрибута

53. Стратегија трофазне симулације као продукциони систем

Интеракција процеса је специфична методологија симулације, настала комбинацијом распоређивања догађаја и сканирања активности. Методологија интеракције процеса се 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:

фаза А - ажурирање времена,
фаза В - извршавање безусловних догађаја,
фаза С - извршавање условних догађаја.