ЗМІСТ

ВСТУП

Актуальність імітаційного моделювання зросла разом з потребою підвищення ефективності методів системного аналізу. В першу чергу, як машинне моделювання, воно має велику кількість переваг над фізичними експериментами. Першим з них буде простота повторення умов, потрібних для проведення експерименту, а також легкість його переривання при потребі. Існують дві основні проблеми, для вирішення яких найдоцільніше використати машинні експерименти: дослідження системи та її оптимізація. Для отримання якісних результатів експерименту важливо правильно вибрати число проведення дослідів так, щоб ресурсні затрати були мінімальними, а результати відповідали заданій точності. Іншими словами, варто зробити число спроб мінімальним, але водночас таким, яке б не спотворило здобутої інформації.[7] Питання умовного кошторису ресурсів постає особливо гостро, коли ми говоримо про моделювання саме соціальних систем, бо зазвичай агентами в ньому постають люди, які за своєю природою є унікальними “екземплярами” свого “класу”.

Мета даної роботи - дослідити способи імітаційного моделювання соціальних систем, ознайомитись з методами розробки та верифікації імітаційних моделей, створити власну практичну модель системи, яка була б максимально наближеною до справжньої моделі поведінки людей.

СУТЬ ДИПЛОМНОЇ РОБОТИ

Розділ 1. Теоретичні основи імітаційного моделювання

1.1 Генератори випадкових чисел

Імітація є найбільш корисною для дослідження стохастичних систем, які напряму залежать від переплетення впливу багатьох випадкових величин, тому варто насамперед поговорити про методи та способи побудови генераторів випадкових величин (звідси ГВВ). [5]

Існує низка способів генерування випадкових чисел, але на сьогодні найбільш прийнятним вважається спосіб застосування рекурсивних формул, коли на підставі *i*-того випадкового числа обчислюється *i*+1-те випадкове число. Розглянемо ГВВ рівномірно розподілених на інтервалі (0;1). До ГВВ є 5 основних вимог:

1. Числа мають бути незалежними і рівномірно розподіленими на інтервалі (0;1);
2. Для роботи ГВВ обсяг пам'яті повинен бути малим.
3. Відтворення послідовності випадкових чисел має бути можливим;
4. Повинна генеруватися велика кількість чисел, які б не повторювалися;
5. Швидкодія;

Спробуємо відтворити широко поширений конгруентний метод генерування з наступним рекурсивним рівнянням:

$$z_{i+1} = (az_i + b)(mod\ c), i = 1, \ldots$$

$$\zeta_{i+1} = \frac{z_{i+1}}{c},$$

де *a, b, c* - параметри генератора, а z_0 - його початкове значення. *(mod c)* - ділення за модулем *c*. З формули зрозуміло, що число *c* не буде меншим за z_i, тому при діленні z_i на *c* виходить випадкова величина ζ_i, яка є в інтервалі (0;1).

Для імітування важлива якомога довший період ГВВ. З нашої рекурсивної формули випливає, що в той момент, коли в послідовності з’являється число, яке вже було згенероване - тоді повторюватиметься і вся послідовність. Кількість унікальних чисел в одному циклі власне називається періодом ГВВ.[6]

Щоб згенерувати випадкову величину *r*, що розподілена за заданим законом F(x), використовують також наступні методи, представлені на рисунку 1.1:

- оберненої функції;
- табличний;
- такий, що базується на функціональних властивостях законів розподілу.

Варто зазначити, що якість та методи ГВВ мають прямий вплив на результати моделювання. Погано побудований ГВВ може створити ряд помилок, що спричинить не тільки деяку похибку у результатах імітації, але й у самому алгоритмі імітації. Така похибка може означати, що алгоритм моделювання більше не відповідає тому, як функціонує реальна система.

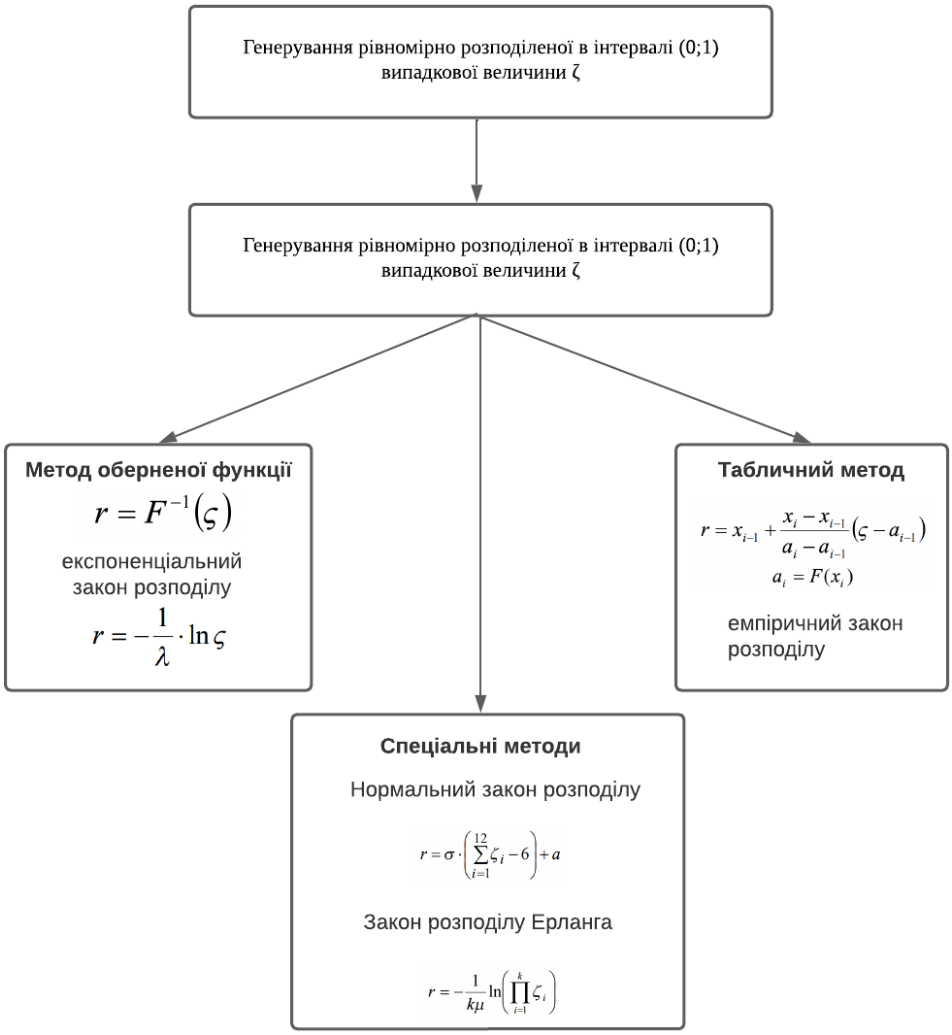


Рисунок 1.1 Методи генерування випадкового числа *r*

1.2 Алгоритми імітації функціонування систем

Алгоритмом імітації називають алгоритм, що відтворює функціонування системи за допомогою комп’ютерної програми. Схема роботи алгоритму представлена на рисунку 1.2.

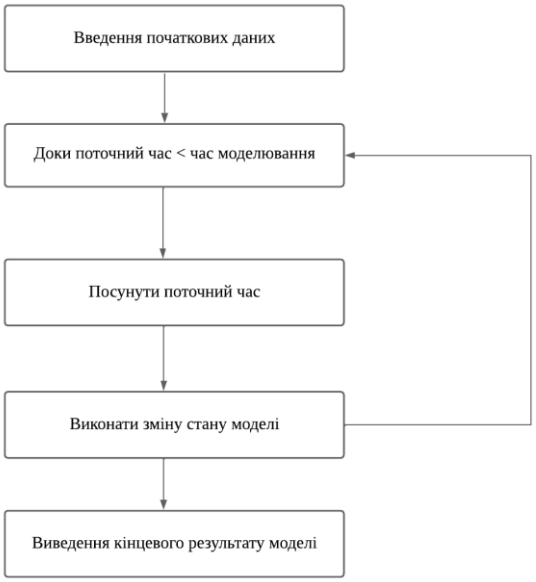


Рисунок 1.2 Схема роботи алгоритму

Такі кроки, як просування модельного часу, зміна модельного стану, залежно від часу та обробка даних, отриманих під час імітації є також функціями, побудованими в залежності від специфіки моделі. Коротко розглянемо кожен з них.

Існують три алгоритми просування модельного часу: за принципом Δt, за принципом послідовного проведення об’єктів по моделі та за принципом найближчої події. Принцип Δt є найпростішим для розуміння: час моделювання поділяється на рівні інтервали Δt = const, такі, щоб в одному інтервалі виникала лише одна подія. Інакше логіка алгоритму моделювання системи спотворюється. Цей принцип є найпростішим і підходитиме для більшості систем, але його недолік у тому, що він потребує більших витрат комп’ютерного часу, ніж інші.

За принципом послідовного проведення об’єктів по моделі алгоритм просування часу не будується. Кожний об’єкт проводиться по моделі з моменту його надходження у модель до моменту виходу з моделі. Такий алгоритм імітації дозволяє моделювати систему тільки в моменти виникнення подій, тому використовується дуже рідко, оскільки часто призводить до занадто складних алгоритмів.

За принципом найближчої події модельний час просувається від моменту виникнення однієї події до моменту виникнення іншої, і після кожного просування часу реалізуються зміни стану моделі, відповідні до події, що виникла. Специфіка моделей, які використовують саме цей принцип, така, що вони змінюють свій стан лише в деякий момент імітації, коли виникає певна подія, а не завжди.[3] Наприклад, подія “вступ абітурієнта в університет” збільшує кількість наявних студентів, а подія “випуск бакалаврів” - зменшує і тд. За рахунок того, що моделювання системи пропускається в моменти, коли не відбувається жодної події, цей принцип є найекономнішим.

Щодо способів зміни стану моделі, їх також є три: орієнтований на дії, орієнтований на події та процесно-орієнтований. Орієнтований на дії, наприклад, потребує визначення усіх дій, що є можливими в системі, умов їхнього початку та кінця. Тобто якщо умова початку виконується - виконується дія, якщо після цього виконується умова кінця - дія зупиняється. Щоправда, для того, щоб була виконана кожна дія, сканування умов здійснюється для всієї множини дій при кожному просуванні модельного часу, що є ресурсозатратним, а тому має обмежене застосування в імітації.

При процесно-орієнтованому підході описують процес проходження об’єктів уздовж моделі, використовуючи кінцевий набір операторів. Послідовність операторів потім транслюється у відповідну послідовність подій і далі моделювання здійснюється як при підході, орієнтованому на події, який буде розглянуто наступним.

При способі, орієнтованому на події, визначаються і описуються події, що

відбуваються в моделі. В такій моделі імітація - це виконання упорядкованої в часі послідовності подій, що є логічно взаємопов’язані. Наведемо приклад системи масового обслуговування (звідси СМО) з одним оператором та обмеженою чергою.[2] Стан системи визначатиметься станом оператора та станом черги і змінюватиметься з подією “поява нового запиту” та “завершення обробки запиту”.

Подія “поява нового запиту” складається з наступних умов та дій:

- якщо я вільне місце в черзі - зайняти його;
- інакше - інкрементувати число втрачених клієнтів;
- генерувати наступне входження клієнта в СМО.

Подія “завершення обробки запиту” складається з таких дій:

- інкрементувати число обслуговуваних клієнтів;
- якщо черга не пуста, зайняти оператора і зменшити чергу на одиницю, запам’ятати момент виходу запиту з каналу у момент часу - до поточного моменту часу додати тривалість обслуговування запиту у каналі;
- інакше звільнити оператора, запам’ятати момент виходу запиту з каналу у момент часу, що більший за час моделювання (тобто у найближчий час вихід запиту із каналу не очікується).

Після того, як “поява нового запиту” виконалася, потрібно запам’ятати моменти виникнення наступних подій: “поява нового запиту” та “завершення обробки запиту”. З них вибирається мінімальний і встановлюється відповідність між ним та подією, що виникла. Тоді вона виконується і після цього просувається модельний час. При виконанні “завершення обробки запиту” запам’ятовується або наступне завершення, або позначається, що звільнення каналу не відбудеться до наступної події. Отже, модельний час просувається від виконання однієї події до іншої, які в свою чергу є послідовно впорядкованими в часі.[6]

Алгоритм програми, що імітує СМО, взявши за основу цикл рисунку 1.2, виглядатиме наступним чином:

1. Ввід вхідних даних: поточний час, час моделювання, стан оператора, момент появи нового запиту та момент завершення обробки запиту.
2. Перевірка чи поточний час не більший часу моделювання.

3. Пошук найменшого з моментів (момент поява нового запиту та момент завершення обробки запиту) та запам’ятовування, якій події він відповідає.
4. Просув час в найближчий момент події.
5. Виконання події, яка відповідає моменту найближчої події.
6. Виведення результатів моделювання: кількість обслуговуваних та втрачених клієнтів протягом часу моделювання.

Способи обробки інформації про роботу моделі зазвичай статистичні. Імовірність відмови в обслуговуванні оцінюють відношенням втрачених клієнтів до усіх, що потрапляли в систему. Середня довжина очікування в черзі оцінюють середнім значенням динамічної випадкової величини за формулою:

$$L_{\text{середнє}} = \frac{\sum_i L_i \Delta t_i}{T_{\text{моделювання}}},$$

де L_i - довжина черги i-того випадку, Δt_i - інтервал часу, протягом якогось спостерігалось L_i, а $T_{\text{моделювання}}$ - час спостереження за моделлю.

Якщо при моделюванні було використано спосіб просування модельного часу за принципом Δt, то можна припустити, що $T_{\text{моделювання}} = n * \Delta t$, тоді формула набуде наступного вигляду:

$$L_{\text{середнє}} = \frac{\sum_i L_i}{n},$$

де n - кількість інтервалів часу моделювання.

Щоб підрахувати середнє значення часу очікування запитів у черзі, сумарний час очікування усіх вимог ділять на їхню кількість:

$$Q_{\text{середнє}} = \frac{\sum_i L_i \Delta t_i}{N_{\text{клієнтів}}},$$

де L_i - кількількість запитів, що очікують обслуговування, Δt_i - часу, протягом якого спостерігалось L_i, а $N_{\text{клієнтів}}$- загальна кількість обслуговуваних клієнтів за час спостереження за моделлю.

В складніших випадках використовують масив черги вимог для розрахунку середнього часу очікування в черзі. Розмір цього масиву є максимальним числом клієнтів у черзі. Зазвичай черги систем масового обслуговування організовуються чергою LIFO: коли новий клієнт приходить у чергу, він займає наступне за останнім непустим елементом масиву місце, а коли клієнт залишає чергу, то звільняється останнє непусте місце масиву.

Нижче на рисунку 1.3 наведена схема підрахунку середнього часу обслуговування в СМО за допомогою масиву значень.

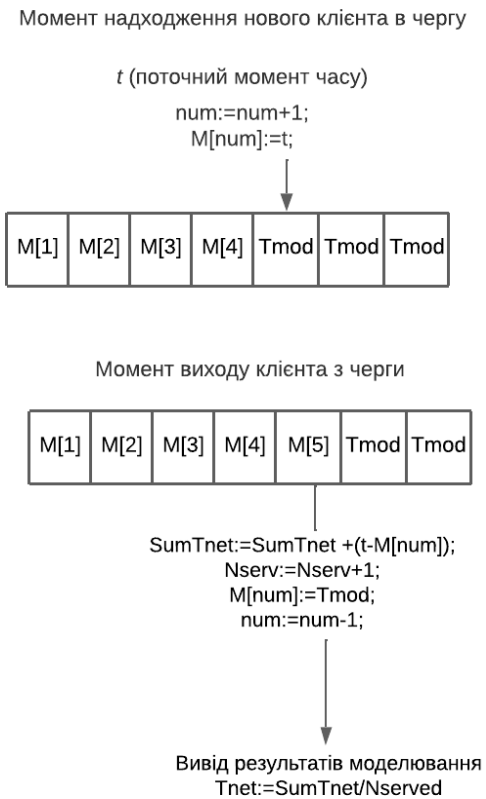


Рисунок 1.3 Схема підрахунку середнього часу обслуговування в СМО

Таким способом також зручно рахувати середній час обслуговування клієнта в СМО. У виділеному сегменті масиву записується час початку обслуговування і при виході клієнта з системи, обчислюється як різниця між входом та виходом з мережі.

$$T_{net} = \frac{\sum_{i=1}^{N_{serv}} (t - t_{enter})}{N_{serv}},$$

де i - номер клієнта, N_{serv} - кількість обслужених клієнтів, t_{enter} - час входження клієнта в систему.

1.3 Імітаційне моделювання мережі масового обслуговування

Для того, щоб розробити алгоритм імітації моделі мережі масового обслуговування, потрібно описати елементи моделі, їхні стани та множину подій. Визначимо елементи мережі масового обслуговування: вхідний потік, власне система масового обслуговування та зв'язок.

Вхідний потік можна визначити як інтервал надходження вимог (нових клієнтів) у мережу. Цей інтервал може бути як випадковою, так і детермінованою величиною і навіть номером СМО, до якої вимога надходить. Стан потоку можна описати моментом часу надходження наступної вимоги у систему. Інтервал часу надходження вимог може бути і випадковою величиною з певним відомим нам законом розподілу. Тоді для його побудови вибирають відповідний генератор випадкових величин, як це було описано у розділі 1.1.

Система масового обслуговування - це сукупність паралельно з'єднаних пристроїв та черг перед ними. Її можна описати двома величинами: кількістю пристроїв та обмеженням на максимальний розмір черги. Кожен пристрій можна також описати кількістю часу, потрібного для обслуговування вимоги та станом. При описі стану також використовують дві величини: стан в обов'язково поточний момент часу та момент виходу з пристрою вимоги, яку обслужили. Є три види стану: вільний, зайнятий та заблокований. Який момент виходу вимоги записати, якщо пристрій вільний, а отже вихід не очікується? Загальноприйнятою величиною є нескінченність. Ще одним випадком використання нескінченності - стан "заблокований". При цьому стані момент виходу залежить від перебігу подій в сусідніх елементах і не залежить від самого пристрою. Чергу описують зазвичай максимальною довжиною, але ще можна додати її тип (LIFO, FIFO чи пріоритизація). Стан черги описують кількістю клієнтів в ній та можливо їхнім порядком та пріоритетом, в залежності від типу черги.

Зв'язок можна розуміти як маршрут вимог між СМО. Він описується двома СМО-точками (від якої рухається вимога та до якої рухається вимога) і станом

(заблокований або незаблокований). Оскільки маршрути зазвичай мають розгалуження, точкою призначення можуть бути дві і більше СМО. В такому разі вони вказуються разом з імовірністю потрапляння вимоги на дану гілку маршруту. Також блокування зв'язку задається умовою, рівень складності якої залежить від типу моделі.

Події, що виникають у моделі, бувають двох видів: надходження та вихід вимоги до/з системи. Найзручніше в такому випадку використати принцип найближчої події для просування модельного часу.

Якщо визначити усі елементи, про які ми говорили вище, як об'єкти, то використовуючи об'єктно-орієнтований підхід можна скласти універсальну модель СМО. Одним з важливих етапів побудови моделі для розробника є її верифікація. Використовуючи модель на рисунку 1.4 наведемо декілька, здавалось, очевидних, але обов'язкових для перевірки прикладів.

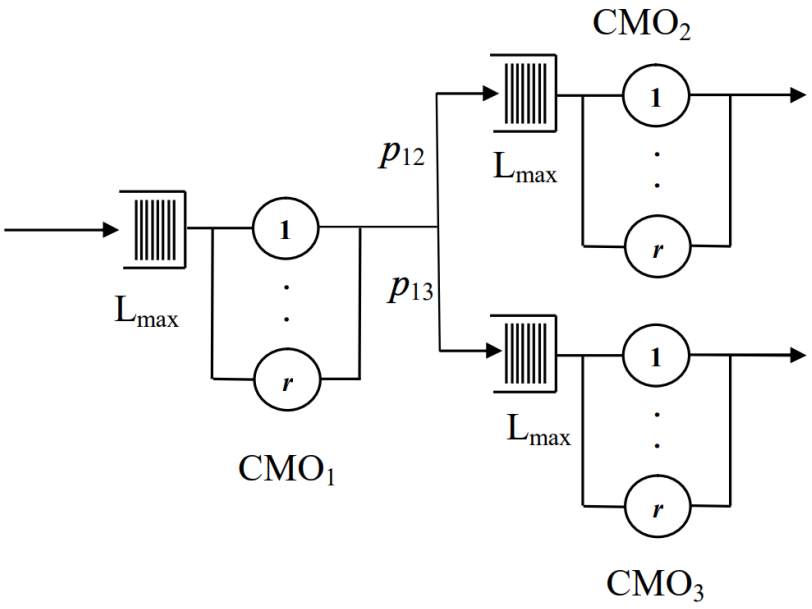


Рисунок 1.4 Модель системи масового обслуговування

Якщо зменшити інтервал часу, з яким нові вимоги входять у мережу, кількість вимог в чергах, а також завантаження пристроїв повинні зменшуватись.

При збільшенні середнього часу обслуговування вимоги на пристрої ж, навпаки збільшуватись.

Розглянувши модель з рисунку 1.4, можна очікувати наступних результатів: якщо додати декілька пристроїв до СМО3, то середнє очікування в черзі мало б зменшитися, а середня кількість зайнятих пристроїв - збільшитися. Причиною цього стало б те, що до СМО3 зменшилася б кількість нових запитів, що надходять.

Якщо закрити маршрут СМО1-СМО3, то можна очікувати, що СМО2 буде набагато частіше відмовляти в обслуговуванні запитів, так само як і збільшиться середня зайнятість пристроїв та час очікування в черзі.

Зміна вхідних значень повинна призводити до логічної зміни вихідних результатів моделі, тому за результатами верифікації моделі можна зробити висновок про відповідність функціонування моделі функціонуванню реальної системи.

Розділ 2. Практична реалізація імітаційного моделювання

2.1 Мова програмування GPSS

GPSS (General Purpose Simulation System – система моделювання загального призначення) – це мова імітаційного моделювання систем, в основному соціального призначення, такі як СМО. Для просування модельного часу ця мова використовує принцип найближчої події. Система в цій мові моделюється як операції входу в систему і передається від одного сервісу до іншого у вигляді блоків.

Основна мета GPSS полягає в тому, щоб створювати моделі систем з подійно-орієнтованим підходом, де різні події відбуваються впевненими проміжками часу і впливають один на одного. GPSS надає можливість моделювати такі складні системи, як виробничі процеси, транспортні мережі, логістичні системи, банківські та фінансові операції, і багато інших. Під час імітації збережені дані формують звіт - важливий елемент результатів симуляції, який надає інформацію про різні аспекти моделі та її виконання. Звіт у GPSS містить числові дані, статистику, графіки та інші відомості, які допомагають аналізувати та оцінювати результати симуляції. Це включає загальні показники, такі як час моделювання, кількість подій, час очікування, час обробки тощо. Якщо в моделі використовуються ресурси, такі як машини, робочі місця або інше обладнання, звіт може містити статистику про їхнє використання, час простою ресурсів та ін. Також GPSS може виводити графіки, які показують залежності між часом та іншими характеристиками системи. Це дозволяє візуалізувати динаміку моделі та виявити можливі проблеми або ефективність системи.

Для того, щоб описати імітаційну модель мовою GPSS, варто побудувати її схему, на якій елементи СМО з’єднані між собою. Описом моделі виступає сукупність операторів, які визначають процеси обробки вимог, а саме виникнення транзактів (надходження нових запитів в систему), обслуговування в пристроях, очікування в черзі та виведення з СМО.

Розглянемо основні оператори мови GPSS:

GENERATE 864000 - визначає час моделювання в секундах.

FUNC1 FUNCTION RN1,C4

0,0/0.2,0.3/0.4,1.2/1.6,1.7

- функція з назвою “FUNC1”. RN1 - аргумент, який вказує на ГВВ, що використовується. C4 - аргумент, який вказує на те, що функція є неперервною і заданою чотирма точками: (0;0), (0.2; 0.3), (0.4, 1.2), (1.6; 1.7).

QUEUE KLIENT,2 – черга з назвою “KLIENT”. Числовий аргумент 2 означає, що черга збільшиться на 2. Обернений оператор - **DEPART QAA**.

ADVANCE – імітує очікування запиту протягом вказаного часу.

ENTER SOME_STORAGE,3 – зайняття запитом 3 місця в накопичувачі SOME_STORAGE. Обернений оператор - **LEAVE SOME_STORAGE,3**.

SOME_STORAGE STORAGE 10 – накопичувач з назвою “SOME_STORAGE” ємністю 10.

TERMINATE 1 – вихід запиту з системи.

ASSIGN 2,WORK – зміна параметрів транзактів. У цьому прикладі другий параметр транзакта отримає значення WORK.

TEST E V7,OPT1,OPT2 – якщо виконується рівність, перейти до блоку OPT1, інакше до OPT2. Якщо ж потрібен безумовний перехід - оператор **TRANSFER**

LOOP 4,MIT – використання циклу, в якому щоразу перевіряється параметр 4.

SAVEVALUE DOHID,3 – лічильник з назвою “DOHID”, що збільшується на 3.

2.2 Моделювання служби замовлення таксі.

Розглянемо приклад з навчального посібника І. В. Стеценка “Моделювання систем” (2010)[1].

Постановка задачі:

“Служба замовлення таксі має 5 каналів для одночасного прийняття замовлень по телефону. Час між спробами виклику таксі розподілений за законом Ерланга другого порядку із середнім 180 секунд. Абонент затрачає 30 секунд для набирання номера і, якщо застає всі канали служби замовлення зайнятими або після з’єднання з’ясовує, що черга на обслуговування перевищує 10 замовлень (в такому випадку замовлення не приймаються), то через 60 секунд він повторює набирання номера. Після п’яти спроб абонент припиняє набирання. Служба замовлення таксі має у своєму розпорядженні 30 машин таксі для обслуговування замовлень. Час, витрачений на проїзд до клієнта, залежить від відстані до нього. Ймовірності можливих відстаней розподіляються таким чином: 2 км – з імовірністю 0,1, 8 км - з імовірністю 0,2, 9 км – з імовірністю 0,25, 11 км - з імовірністю 0,17, 12 км – з імовірністю 0,23, 20 км – з імовірністю 0,05. Вартість проїзду до клієнта клієнтом не сплачується. Швидкість руху машин рівномірно розподілена в інтервалі 45±5 км/год. Час обслуговування клієнта рівномірно розподілений в інтервалі 50±20 хвилин. Вартість попереднього замовлення складає 2 гривні, вартість проїзду 1 км складає 2 гривні. Метою моделювання є визначення такої кількості операторів-телефоністів та водіїв таксі, при якій максимізується прибуток служби замовлення”

Щоб реалізувати надходження нового запиту згідно з умови, потрібно використати два оператори: GENERATE 90,FN$DIS і ADVANCE 90, FN$DIS. Функція DIS задаватиме експоненціальний розподіл. Дві затримки середнього значення 90 відповідають закону Ерланга другого порядку із середнім значенням 180.

Набір номеру до телефоніста можна задати простою затримкою транзакта, в середньому він триває 30 секунд: ADVANCE 30. Успіхом вважаємо, що абонент додзвонився до телефоніста і зайняв його (за умовою задачі - одного з 5) розмовою на 30 секунд. Отримуємо фрагмент, зображений на рисунку 2.1:

```
TEL  STORAGE  5
     ENTER    TEL
     ADVANCE  30
     LEAVE    TEL
```

Рисунок 2.1 Фрагмент коду моделювання додзвону абонента

Оскільки у клієнта є 5 спроб додзвонитися до оператора, для реалізації нам потрібен цикл. У ньому ми передбачимо два розвитки подій: оператор або відповів, або не прийняв дзвінок: TRANSFER BOTH,VIDP,VIDM. Коли оператор не приймає дзвінок і підрахована кількість спроб додзвону дорівнює тій, що задана в умові - клієнт виходить з системи (TERMINATE). Для статистичного звіту корисно буде підрахувати кількість втрачених клієнтів (SAVEVALUE NEOBSL+,1).

Таким чином отримуємо наступний фрагмент програми, представлений на рисунку 2.2:

```
TEL      STORAGE     5
DODZVON  ADVANCE     30
         TRANSFER    BOTH,VIDP,VIDM
VIDP     ENTER       TEL
         ADVANCE     30
         LEAVE       TEL
VIDM     ADVANCE     60
         LOOP        1,DODZVON
         TERMINATE
```

Рисунок 2.2 Фрагмент коду обробки відповіді оператора та відмови в обслуговуванні

Клієнтові важлива довжина черги. Її можна перевірити наступним чином: TEST L Q$KLIENT,10,VIDM, де L - less than, Q$KLIENT - черга з назвою “KLIENT”, 10 - максимально припустиме для клієнта значення довжини черги, а VIDM - блок, до якого виконується перехід при виконанні умови. Для статистичних даних підраховуємо дохід SAVEVALUE DOHID+,2000. Для переходу до

обслуговування клієнта, якщо він не відмовився до цього, нам не потрібна умова, тому використовуємо TRANSFER ,OBSL. Отже, остаточно фрагмент програми, що моделює обслуговування клієнтів у операторів, приймає вигляд, як на рисунку 2.3:

```
TEL      STORAGE     5
DODZVON  ADVANCE     30
         TRANSFER    BOTH,VIDP,VIDM
VIDP     ENTER       TEL
         ADVANCE     30
         LEAVE       TEL
         TEST L      Q$KLIENT,10,VIDM
         SAVEVALUE   DOHID+,2000.
VIDM     ADVANCE     60
         LOOP        1,DODZVON
         TERMINATE
```

Рисунок 2.3 Фрагмент коду, що моделює обслуговування клієнтів у операторів

Блок, який описує обслуговування клієнта власне водієм таксі включає в себе наступні етапи:

- затримку в черзі Q$KLIENT
- зайняття однієї з 30 машини таксі (заданих TAXI STORAGE 30)
- затримка часу, що імітує добирання таксі до клієнта
- затримка часу, що імітує власне поїздку
- звільнення машини
- підрахунок прибутку за клієнта
- інкрементування кількості обслужених клієнтів
- вихід клієнта з системи

Щоб задати час обслуговування з розподілом, вказаним в умові, визначимо функцію з назвою DOBSL, яка буде неперервною і задається двома точками.

Оскільки час добирання машини до клієнта невідомий, зате відома відстань, згенерована ГВВ та швидкість машини, задана розподілом, змінна TIME обчислюється динамічно.[4] Прибуток таксі від поїздки також задається змінною COST і розраховується як добуток відстані поїздки на вартість одного метра.

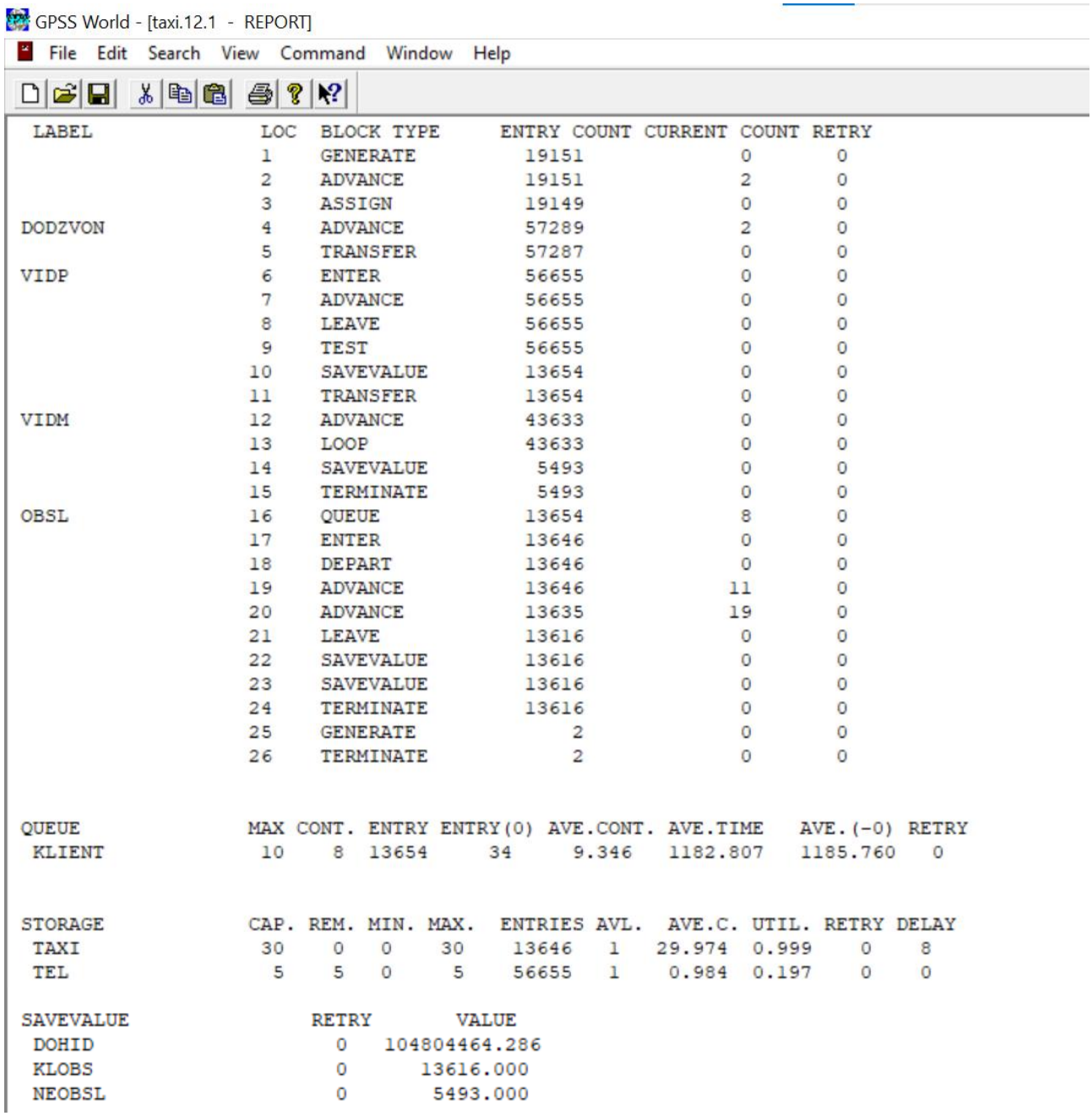
Фрагмент програми, що відповідає за час моделювання протягом 240 годин реалізовується за допомогою операторів GENERATE 864000 та TERMINATE 1 для виходу з програми.[8]

Рисунок 2.4 показує повний код програми, яка моделює поставлену задачу в підрозділі 2.2, а рисунок 2.5 показує фрагмент звіту, згенерований GPSS про результати моделювання.

File Edit Search View Command Window Help

```
TAXI       STORAGE   30                                          ;к-сть машин
TEL        STORAGE   5                                           ;к-сть операторів

DIS        FUNCTION  RN1,C24                                     ;ф-я експоненціального закону розподілу
0.,0./.100,.104/.200,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.750,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8

DOBSL      FUNCTION  RN2,C2                                      ;ф-я часу облуговування клієнта (сек)
0.,1800/.9999,4200\

DSPEED     FUNCTION  RN3,C2                                      ;ф-я швидкості руху машини (м/с)
0.,11.111/.9999,13.888

DIST       FUNCTION  RN4,D6                                      ;ф-я задання відстані до клієнта
0.1,5000/0.3,8000/0.55,9000/0.72,11000/0.95,12000/1.00,20000

TIME       VARIABLE  FN$DIST/FN$DSPEED                          ;час руху машини до клієнта
COST       VARIABLE  FN$DOBSL#FN$DSPEED#0.2                    ;вартість обслуговування клієнта (коп/м)

           GENERATE  90,FN$DIS                                   ;генерація клієнта, що хоче таксі
           ADVANCE   90,FN$DIS                                   ;затримка ерлангового розподілу
           ASSIGN    1,5                                         ;5 спроб набирання номера

DODZVON    ADVANCE   30                                          ;набирання номера
           TRANSFER  BOTH,VIDP,VIDM                              ;якщо оператор вільний - відповідь, інакше відмова

VIDP       ENTER     TEL                                         ;зайняти оператора
           ADVANCE   30                                          ;тривалість розмови (сек)
           LEAVE     TEL                                         ;звільнити оператора
           TEST L    Q$KLIENT,10,VIDM                            ;якщо черга на машину більше 10 людей, то відмова
           SAVEVALUE DOHID+,200                                  ;підрахувати дохід за минуле замовлення
           TRANSFER  ,OBSL                                       ;перейти до обслуговування

VIDM       ADVANCE   60                                          ;затримка після невдалого дзвінка
           LOOP      1,DODZVON                                   ;цикл додзвону до оператора

           SAVEVALUE NEOBSL+,1                                   ;підрахувати необслужених клієнтів
           TERMINATE                                             ;вивід з системи клієнта, що не додзвонився

OBSL       QUEUE     KLIENT                                      ;очікування машини
           ENTER     TAXI                                        ;таксі виконує замовлення
           DEPART    KLIENT                                      ;декрементувати кількість очікуючих клієнтів
           ADVANCE   V$TIME                                      ;таксі прямує до  клієнта
           ADVANCE   FN$DOBSL                                    ;таксі обслуговує клієнта
           LEAVE     TAXI                                        ;звільнити таксі
           SAVEVALUE DOHID+,V$COST                               ;підрахувати дохід за замовлення
           SAVEVALUE KLOBS+,1                                    ;підрахувати к-сть обслуговуваних клієнтів
           TERMINATE                                             ;вивід з системи клієнта, якого було обслужено
           GENERATE  864000                                      ;час моделювання 240 годин
           TERMINATE 1                                           ;
```

Рисунок 2.4 Повний код програми, яка моделює поставлену задачу в підрозділі 2.2



LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	19151	0	0
	2	ADVANCE	19151	2	0
	3	ASSIGN	19149	0	0
DODZVON	4	ADVANCE	57289	2	0
	5	TRANSFER	57287	0	0
VIDP	6	ENTER	56655	0	0
	7	ADVANCE	56655	0	0
	8	LEAVE	56655	0	0
	9	TEST	56655	0	0
	10	SAVEVALUE	13654	0	0
	11	TRANSFER	13654	0	0
VIDM	12	ADVANCE	43633	0	0
	13	LOOP	43633	0	0
	14	SAVEVALUE	5493	0	0
	15	TERMINATE	5493	0	0
OBSL	16	QUEUE	13654	8	0
	17	ENTER	13646	0	0
	18	DEPART	13646	0	0
	19	ADVANCE	13646	11	0
	20	ADVANCE	13635	19	0
	21	LEAVE	13616	0	0
	22	SAVEVALUE	13616	0	0
	23	SAVEVALUE	13616	0	0
	24	TERMINATE	13616	0	0
	25	GENERATE	2	0	0
	26	TERMINATE	2	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
KLIENT	10	8	13654	34	9.346	1182.807	1185.760	0

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
TAXI	30	0	0	30	13646	1	29.974	0.999	0	8
TEL	5	5	0	5	56655	1	0.984	0.197	0	0

SAVEVALUE	RETRY	VALUE
DOHID	0	104804464.286
KLOBS	0	13616.000
NEOBSL	0	5493.000

Рисунок 2.5 Фрагмент звіту, згенерований GPSS про результати моделювання.

2.3 Моделювання маршруту приміського сполучення.

Розглянемо ще один приклад з навчального посібника І. В. Стеценка “Моделювання систем” (2010).[1]

Постановка задачі:

“На маршруті приміського сполучення працюють два мікроавтобуси (А і В), кожний з яких має n місць. Мікроавтобус А користується більшою популярністю, ніж автобус В, оскільки водій мікроавтобуса А їздить акуратніше і швидше. Тому пасажир, який підійшов до зупинки, сідає в мікроавтобус В тільки у випадку, коли автобуса А немає. Мікроавтобус відправляється на маршрут, якщо всі місця в ньому зайняті. Пасажири підходять до зупинки через 0,5 хвилин. Припускається, що всі пасажири їдуть до кінця маршруту. На проходження маршруту мікроавтобус А витрачає 20±5 хвилин, а мікроавтобус В – 30±5 хвилин. Після того, як пасажири звільнили автобус (протягом часу 5±1 хвилин), він їде у зворотному напрямку тим же чином. Плата за проїзд складає 2 гривні. Авто підприємство стільки ж втрачає (недоотримує), якщо пасажир, прийшовши на зупинку, не стає у чергу і обирає інший маршрут. Визначити виручку автопідприємства за день від маршруту, якщо мікроавтобуси працюють 10 годин на добу.”

Щоб змоделювати надходження нового пасажира з середнім значенням 30 секунд, використаємо оператор GENERATE 30. Для моделювання двох варіантів розвитку подій “вибір автобуса А” або “вибір автобуса В” використаємо оператор TRANSFER BOTH,GET_BUS_A,GET_BUS_B.[8]

При успішному виборі займаємо місце в автобусі оператором ENTER. Опісля варто перевірити чи не готовий автобус до відправки. Якщо кількість зайнятих місць є рівною 25 - переходимо до відправки. При вході клієнта в систему для статистичного звіту корисно підраховувати загальну їх кількість. Для цього використовується блок SAVEVALUE CLIENTS+,1, який інкрементує CLIENTS

щоразу, коли до блоку надходить транзакт.[9] Виходить фрагмент коду, представлений на рисунку 2.6:

```
GET_BUS_A ENTER     BUSA                    ;зайняти місце в автобусі
          TEST E    BUSA,25,GO_BUS_A        ;перевірити чи він не готовий до відправки
          SAVEVALUE CLIENTS+,1              ;записати число пасажирів

GET_BUS_B ENTER     BUSB                    ;зайняти місце в автобусі
          TEST E    BUSB,25,GO_BUS_B        ;перевірити чи він не готовий до відправки
          SAVEVALUE CLIENTS+,1              ;записати число пасажирів
```

Рисунок 2.6 Фрагмент коду, що моделює вибір транзакта

Коли виконується умова для відправки, описуємо курсування автобуса по маршруту оператором ADVANCE 1200,FN$TIME_DELTA, де 1200 - час проходження маршруту автобусом в секундах, а FN$TIME_DELTA - функція розподілу часового відхилення. Після проходження маршруту варто підрахувати дохід: 2 гривні за кожного з 25 пасажирів. Коли автобус завершив маршрут, доцільно звільнити місця оператором LEAVE та вивести пасажирів з системи оператором TERMINATE. За умовою завдання це відбувається протягом 5 хвилин (оператор ADVANCE 300,FN$TIME_DELTA). Отримуємо наступний фрагмент, представлений на рисунку 2.7:

```
GO_BUS_A  ADVANCE 1200,FN$TIME_DELTA          ;час курсування автобуса по маршруту
          SAVEVALUE DOHID_A+,50               ;підрахувати дохід за курс
          LEAVE     BUSA,25                   ;звільнити усі місця в автобусі
          TERMINATE 25                        ;вивести пасажирів з системи
          ADVANCE 300,FN$TIME_DELTA           ;час на звільнення місць

GO_BUS_B  ADVANCE 1800,FN$TIME_DELTA          ;час курсування автобуса по маршруту
          SAVEVALUE DOHID_B+,50               ;підрахувати дохід за курс
          LEAVE     BUSB,25                   ;звільнити усі місця в автобусі
          TERMINATE 25                        ;вивести пасажирів з системи
          ADVANCE 300,FN$TIME_DELTA           ;час на звільнення місць
```

Рисунок 2.7 Фрагмент коду, що моделює курсвання автобуса по маршруту

За умовою завдання час моделювання - 10 годин або 36000 секунд. Рисунок 2.8 показує повний код програми, яка моделює поставлену задачу в підрозділі 2.3, а рисунок 2.9 показує фрагмент звіту, згенерований GPSS про результати моделювання.

```
BUSA      STORAGE   25                      ;кількість місць в автобусі A
BUSB      STORAGE   25                      ;кількість місць в автобусі B

TIME_DELTA FUNCTION RN2,C2                  ;ф-я розподілу відхилення часу
0.,1800/.9999,4200

GENERATE 30                                 ;генерація клієнта, що прийшов на станцію

CHOICE    TRANSFER BOTH,GET_BUS_A,GET_BUS_B ;якщо автобус A вільний - вибираємо його, якщо ні - автобус B


GET_BUS_A ENTER     BUSA                    ;зайняти місце в автобусі
          TEST E    BUSA,25,GO_BUS_A        ;перевірити чи він не готовий до відправки
          SAVEVALUE CLIENTS+,1              ;записати число пасажирів
|
GET_BUS_B ENTER     BUSB                    ;зайняти місце в автобусі
          TEST E    BUSB,25,GO_BUS_B        ;перевірити чи він не готовий до відправки
          SAVEVALUE CLIENTS+,1              ;записати число пасажирів

GO_BUS_A  ADVANCE 1200,FN$TIME_DELTA        ;час курсування автобуса по маршруту
          SAVEVALUE DOHID_A+,50             ;підрахувати дохід за курс
          LEAVE     BUSA,25                 ;звільнити усі місця в автобусі
          TERMINATE                         ;вивести пасажирів з системи
          ADVANCE 300,FN$TIME_DELTA         ;час на звільнення місць

GO_BUS_B  ADVANCE 1800,FN$TIME_DELTA        ;час курсування автобуса по маршруту
          SAVEVALUE DOHID_B+,50             ;підрахувати дохід за курс
          LEAVE     BUSB,25                 ;звільнити усі місця в автобусі
          TERMINATE                         ;вивести пасажирів з системи
          ADVANCE 300,FN$TIME_DELTA         ;час на звільнення місць


GENERATE 36000                              ;час моделювання 10 годин
TERMINATE 1                                 ;завершити моделювання
```

Рисунок 2.8 Повний код програми, яка моделює поставлену задачу в підрозділі 2.3

NAME	VALUE
BUSA	10000.000
BUSB	10001.000
CHOICE	2.000
GET_BUS_A	3.000
GET_BUS_B	6.000
GO_BUS_A	9.000
GO_BUS_B	14.000
TIME_DELTA	10002.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	2399	0	0
CHOICE	2	TRANSFER	2399	2349	0
GET_BUS_A	3	ENTER	25	0	0
	4	TEST	25	0	0
	5	SAVEVALUE	0	0	0
GET_BUS_B	6	ENTER	25	0	0
	7	TEST	25	0	0
	8	SAVEVALUE	0	0	0
GO_BUS_A	9	ADVANCE	25	25	0
	10	SAVEVALUE	0	0	0
	11	LEAVE	0	0	0
	12	TERMINATE	0	0	0
	13	ADVANCE	0	0	0
GO_BUS_B	14	ADVANCE	25	25	0
	15	SAVEVALUE	0	0	0
	16	LEAVE	0	0	0
	17	TERMINATE	0	0	0
	18	ADVANCE	0	0	0
	19	GENERATE	2	0	0

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
BUSA	25	0	0	25	25	1	24.865	0.995	2349	0
BUSB	25	0	0	25	25	1	24.604	0.984	2349	0

CEC XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
2402	0	72000.000	2402	0	1		

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
2403	0	108000.000	2403	0	19		
6	0	2567474.160	6	9	10		
17	0	2718894.720	17	9	10		
12	0	2835263.760	12	9	10		
8	0	2843835.600	8	9	10		

Рисунок 2.9 Фрагмент звіту, згенерований GPSS про результати моделювання.

ВИСНОВКИ

Імітаційне моделювання є актуальним і ефективним інструментом для дослідження та аналізу різноманітних соціальних, економічних і технічних систем. Його використання дозволяє зрозуміти та передбачити поведінку системи в різних умовах, визначити її слабкі та сильні сторони, а також оптимізувати її функціонування. Імітаційне моделювання дозволяє досліджувати складні системи, де взаємодіють багато змінних і факторів. Воно дозволяє розглядати систему в цілому, враховуючи взаємодію її складових частин та оцінювати їхні впливи на загальну динаміку системи. Воно дозволяє прогнозувати поведінку системи в різних сценаріях, тестувати різні рішення та стратегії і визначати їхні наслідки. Це допомагає у плануванні та прийнятті рішень, зокрема у виробничих, логістичних, фінансових та соціальних сферах. Імітаційне моделювання дозволяє знаходити шляхи їх вирішення та оптимізації. Це може бути пов'язано з покращенням роботи виробничих процесів, визначенням оптимальних розкладів або управлінням ресурсами, наприклад знизити витрати на експерименти та тестування рішень в реальному світі. Таким чином, актуальність імітаційного моделювання полягає в його можливості аналізувати складні системи, прогнозувати їхню поведінку, виявляти проблеми та оптимізувати функціонування системи. Це важливий інструмент для прийняття обґрунтованих рішень у різних сферах діяльності.

У цій роботі було розглянуто способи імітаційного моделювання соціальних систем, ознайомлено з методами розробки та верифікації імітаційних моделей, та створено дві практичні моделі систем масового обслуговування, які є наближеними до справжньої моделі поведінки людей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стеценко, І.В. Моделювання систем: навч. посіб. [Електронний ресурс, текст] / І.В. Стеценко ; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2010. – 399 с. ISBN 978-966-402-073-9
2. Буртняк, І.В. Імітаційне моделювання: метод. рекомендації [Електронний ресурс, текст] / І.В. Буртняк ; М-во освіти і науки України, Івано-Франківськ
3. Томашевський В.М. Моделювання систем. – К. : Видавнича группа BHV, 2005. – 352с.
4. Стеценко І.В., Батора Ю.В. Інформаційна технологія визначення оптимальних параметрів управління транспортним рухом через світлофорні об’єкти міста // Математичні машини і системи. – Київ, 2007. - №.3,4 – С.211-217.
5. Стеценко І.В.: Методичні вказівки до курсової роботи з дисципліни «Моделювання систем». – ЧІТІ, 2000.
6. Імітаційне моделювання систем та процесів: Електронне навчальне видання. Конспект лекцій / В. Б. Неруш, В. В. Курдеча. – К.: НН ІТС НТУУ «КПІ», 2012. – 115 с.
7. Тимченко А.А. Основи системного проектування та системного аналізу об’єктів. Основи системного підходу та системного налізу об’єктів нової техніки: Навч. посібник/За ред.. Ю.Г.Леги. – К.:Либідь, 2004. – 288с.
8. Luis Beltran Palma Ttito - Introducción a GPSS - https://www.youtube.com/watch?v=dqiBl1WylSw&ab_channel=LuisBeltranPalmaTtito
9. Ranji Raj - General Purpose Simulation Software (GPSS) | Hairdresser's Shop - https://www.youtube.com/watch?v=UElJEW_cO10&ab_channel=RANJIRAJ