Імітаційне моделювання.

Методи імітаційного моделювання широко використовуються для дослідження складних об'єктів і систем в різноманітних галузях людської діяльності.

Переваги застосування імітаційного моделювання найбільш помітно виявляються в разі моделювання виробничих і технологічних процесів, процесів матеріально-технічного забезпечення виробництва, в логістиці, під час проведення бізнес-планування, екологічних та соціологічних досліджень. Часто моделювання закінчують ще до того, як будуть отримані конкретні результати. Визначення моментів часу, в який зацікавлені сторони розуміють, що ж насправді відбувається в системі, вже може бути вирішенням проблеми. Тому навіть не завжди потрібно проводити статистичну обробку результатів експерименту.

Методи проектування імітаційних моделей.

Перш ніж почати побудову моделі, потрібно мати певну схему її проектування, за якою визначають основні принципи і методи розробки імітаційної моделі. Сукупність правил виявлення і застосування системних принципів і методів визначає методологію проектування. Її можна розглядати на різних рівнях деталізації в залежності від обраних способів розробки програмних реалізацій імітаційної моделі. За допомогою обраних програмних засобів визначають і можливі методи їх застосування. Наприклад, вибір за основу імітаційної моделі мереж Петрі або СМО заздалегідь визначає метод побудови її у вигляді формальних схем цих мереж.

Варіантний метод.

Найпростіший і широко застосовуваний метод.

Проектувальник послідовно крок за кроком створює імітаційну модель, спираючись тільки на свій досвід і інтуїцію. В процесі проектування розглядають кілька варіантів кожної частини системи, що моделюється для її відображення в імітаційної подмодели. Найбільш доцільний варіант вибирається з урахуванням рішень, прийнятих щодо інших частин системи, що моделюється.

Це послідовна схема проектування, відповідно до якої вибір варіанта імітаційної моделі є суб'єктивним і залежить від рівня знань проектувальника про систему.

Застосування варіантного методу рідко призводить до допустимим проектним рішенням і не відповідає загальній схемі системного аналізу імітаційного моделювання. Така схема передбачає виконання ітераційної процедури, під час якої проектувальник не один раз повертається до вже розробленого частинам імітаційної моделі і коригує їх, поки не буде впевненим, що модель відповідає цілям моделювання, або відмовиться від неї.

Ітераційний метод.

Суть цього методу полягає в тому, що шляхом численних ітерацій спроектована спочатку імітаційна модель перетворюється в таку, яка відповідає цілям моделювання. Цей метод є методом «проб і помилок», що передбачає циклічні зміни, в результаті чого отримують модель, яка задовольняє вимогам точності і адекватності. Весь хід проектування та остаточний результат в значній мірі залежать від вибору початкової імітаційної моделі. Проектувальник повинен вибрати початкову модель, використовуючи метод аналогії, який базується на знанні характеристик компонентів системи, технологічних засобів і прийнятих рішень в подібних умовах.

Вибір виходить імітаційної моделі дуже впливає на результати проектування і може зробити його неможливим або занадто дорогим. Якщо результати порівняння моделі і реальної системи незадовільні, то перш ніж вносити зміни в модель, необхідно сформулювати ряд гіпотез, за допомогою яких можна визначити причину невідповідності. Гіпотези доцільно формулювати для декількох рівнів уявлення імітаційної моделі:

- Опис структури;
- Алгоритмів поведінки;
- Параметрів та вхідних даних.

Ієрархічні методи.

Незалежно від того, який метод використовується - варіантний або ітераційний, - існують 2 принципово відмінних підходи до проектування імітаційних моделей. Відповідно до першого підходу - проектування здійснюється за схемою зверху вниз (ієрархічне, багаторівневе або спадне проектування), відповідно до іншого - від низу до верху (висхідне проектування). Перший підхід передбачає розподіл системи на підсистеми з дотриманням принципу цілісності системи і називається декомпозицією. Другий підхід з позиції розгляду структури системи є зворотним до першого і називається композицією. Він передбачає розгляд структури системи з метою створення моделі, який починають з її елементів і підсистем, а потім переходять до системи в цілому.

Спадне проектування.

В основі лежить принцип послідовної деталізації або декомпозиції. Він полягає в поступовому уточненні абстрактного опису системи, в процесі якого на кожному етапі побудови моделі задається певний рівень деталізації відображення системи.

На першому етапі проектування будується найбільш загальна однорівнева імітаційна модель системи, за допомогою якої оцінюються тільки основні показники її роботи. На наступному етапі деякі блоки моделі описуються більш детально. Таким способом під час переходу від більш високого рівня опису кожного з блоків моделі до більш низького можна досягти більшої точності та адекватності моделі системи в цілому.

В процесі побудови імітаційної моделі під час переходу з одного рівня опису на інший слід дотримуватися одного з головних принципів декомпозиції ієрархічних систем - обсяг інформації, яка передається з рівня й детальніші описи моделі на рівень менш деталізований, повинен бути менше; час роботи блоку на рівні з більшою деталізацією має бути меншим, ніж час роботи блоку на рівні з меншою деталізацією.

Висхідне проектування.

Загальна схема заснована на поступовому відображенні елементів системи в моделі, починаючи з найнижчого рівня системи з подальшим переходом до більш високого. Такий підхід має істотний недолік: розглядаючи окремі елементи системи і намагаючись відобразити їх якомога детальніше в моделі, проектувальник може не бачити систему в цілому. Це може привести до того, що під час побудови моделі найвищого рівня імітаційна модель може виявитися функціонально неповною, оскільки не були враховані взаємозв'язку між різними рівнями системи.

Найбільш ефективним методом проектування імітаційних моделей є такий, що об'єднує в собі в певній пропорції спадний і висхідний проектування.

Основна цінність імітаційного моделювання полягає в тому, що воно базується на методології системного аналізу і дає можливість досліджувати проектовану або аналізовану систему з використанням технології операційного дослідження, яка включає такі взаємопов'язані етапи:

1. Формулювання проблеми і змістовна постановка задачі.
2. Розробка концептуальної моделі.
3. Розробка та програмна реалізація імітаційної моделі.
4. Перевірка правильності і достовірності моделі.
5. Організація і планування проведення експерименту.
6. Прийняття рішень за результатами моделювання.

Формулювання проблеми і змістовна постановка задачі.

Імітаційне моделювання доцільно проводити, наприклад, для виконання таких завдань:

- вибір і обґрунтування технологічного маршруту виготовлення виробів;
- оцінювання страхових запасів на ділянках комплектації складського виробництва;
- дослідження роботи автоматизованого складу;
- аналіз роботи системи збору, передачі та обробки інформації;
- аналіз часу доступу користувачів до інформаційної бази;

- вибір комплексу технічних засобів для обробки інформації в організації або на виробництві;
- вибір організаційної структури управління цехом підприємства;
- вибір складу технологічного обладнання на ділянці за критерієм мінімізації часу обробки.

Проблема відрізняється від завдання тим, що точні методи її вирішення невідомі. Проблема завжди комплексна і складається з декількох завдань. Тому, формулюючи проблему, для вирішення якої розробляється модель, потрібно в першу чергу визначити цілі моделювання, потім вивчити об'єкт моделювання, визначити межі, в яких буде проводитися дослідження.

Після завершення цього етапу на змістовному рівні описуються основні характеристики системи, вхідні і вихідні змінні, їх взаємозв'язок, зовнішні впливи на систему, визначаються основні критерії функціонування системи та обмеження. Подальше уточнення і формалізацію моделі здійснюють на етапі створення концептуальної моделі.

Розробка концептуальної моделі.

Концептуальною називається абстрактна модель, яка виявляє причинно-наслідкові зв'язки, властиві досліджуваного об'єкта в межах, визначених цілями дослідження. Це формальний опис об'єкта моделювання, яке відображає концепцію (погляд дослідника на проблему).

Концептом називають деяку структуру, яка необхідна для визначення об'єктів. Об'єкти можна визначити через перелік їх атрибутів (властивостей). З позицій аналізу системи - це найважливіший етап, так як неправильно сформульована концепція призведе до побудови неправильної моделі.

Під час розробки концептуальної моделі необхідно:

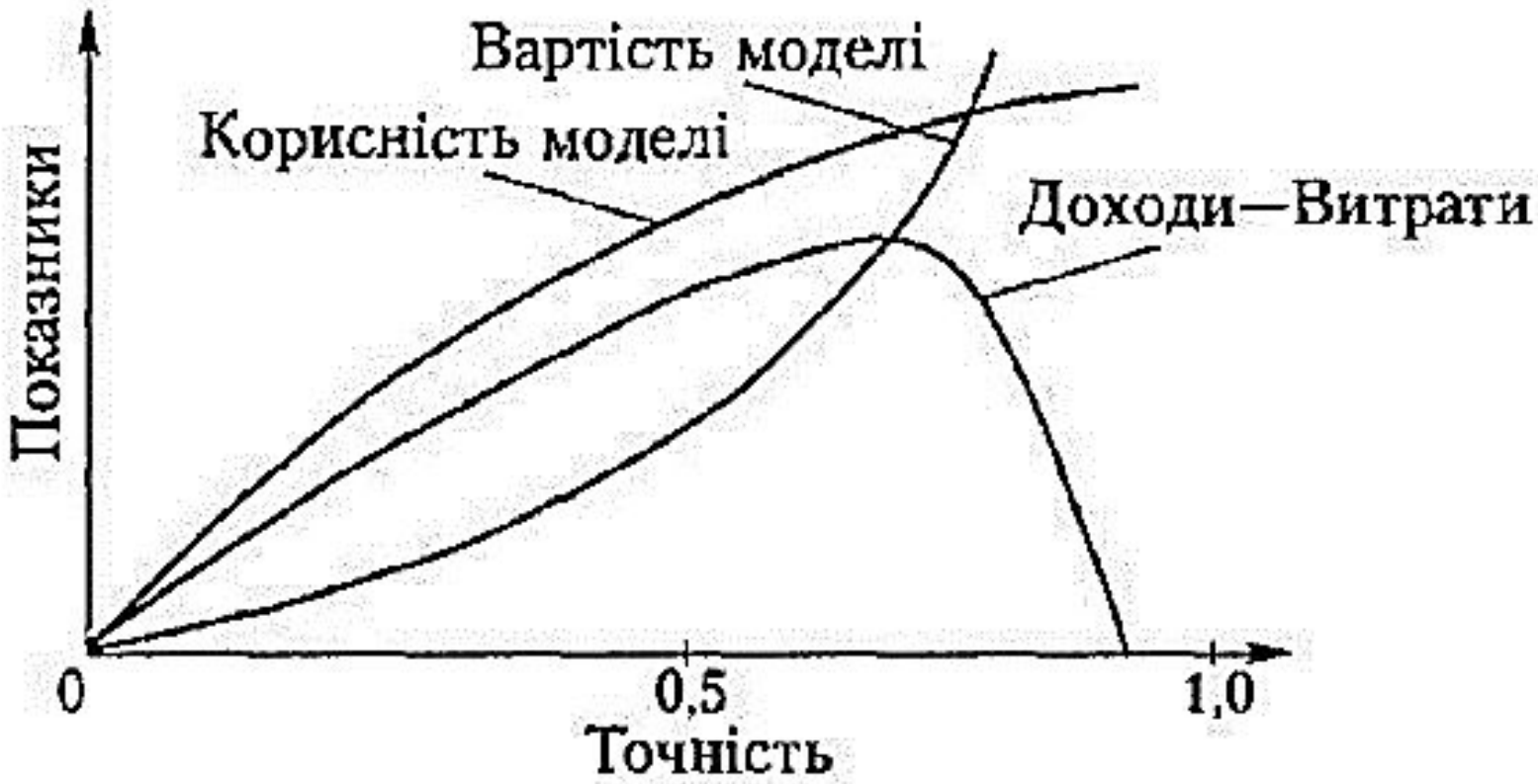
- визначити цілі моделювання;
- сформулювати цільові функції (критерії якості), що моделюється;
- вибрати ступінь деталізації зображення моделі;
- описати вихідні, вхідні змінні і параметри моделі;
- додати функціональні залежності, які описують поведінку змінних і параметрів;
- описати обмеження на можливі зміни величин;
- розробити структурну схему концептуальної моделі і скласти опис її функціонування.

Під час побудови концептуальної моделі об'єкта істотним є виділення і опис станів об'єкта.

Стан системи визначається безперервними або дискретними значеннями характеристик елементів системи.

Вибір ступеня деталізації опису об'єкта моделювання.

Під час імітаційного моделювання важливим є визначення ступеня точності опису реального процесу для отримання вірогідної інформації шляхом моделювання, тобто вибір необхідного рівня деталізації опису процесів функціонування системи. Цей рівень залежить від цілей моделювання, заданих обмежень і можливості отримувати вхідні дані з необхідною точністю. Більш детальна модель буде точніше, але вона буде складніше і дорожче, так як для її побудови, перевірки, документування та використання необхідні великі витрати. З іншого боку, найдетальніша модель точніше і тому буде частіше використовуватися. Значить, потрібно встановлювати рівновагу між вимогами дослідження і вартістю моделі. На рис. показано, як змінюються показники моделі в залежності від обраної точності.


Вартість моделі
Корисність моделі
Доходи—Витрати
Показники
0
0,5
1,0
Точність

Для оцінки рівня деталізації моделі можна використовувати показники ступеня деталізації. Одним з них є ставлення реального часу до модельного. Іншим важливим показником є тимчасова роздільна здатність моделі, яка визначається як найкоротший інтервал часу між послідовними подіями, які фіксуються під час моделювання, тобто найменша ідентифікована порція інформації в моделі.

Від обраного рівня деталізації моделі залежить і її стійкість, тобто здатність точно відтворювати поведінку системи, конфігурація якої або вхідні дані для якої не були передбачені на етапі побудови моделі. Стійкість зростає зі збільшенням ступеня деталізації моделі. Приклад: під час побудови моделі комп'ютера з метою визначення середнього часу виконання програм, які надходять до нього, можна використовувати СМО з одним пристроєм. Ця модель буде відтворювати вхідний потік програм із заданим розподілом ймовірності часу надходження їх в комп'ютер. Час виконання цих програм буде задано деякою функцією розподілу ймовірностей. Ця модель буде дуже загальної, але вона дасть можливість визначити час виконання програм комп'ютером.

Більш детальної буде модель, в якій надходять в комп'ютер програми будуть вимагати під час перебування в комп'ютері деяких ресурсів (оперативної пам'яті, зовнішніх пристроїв, процесорного часу, файлів, програмних модулів). Споживання цих ресурсів програмами теж буде відтворювати процес виконання програм комп'ютером. Ця модель вимагає більш детальних даних про роботу цих програм, тобто робочого навантаження на комп'ютер, і буде точніше відтворювати його роботу. Таку модель можна побудувати як мережу СМО, але витрати на її побудову будуть великими.

Найбільш деталізованої буде модель, якщо надходять в комп'ютер програми подати у вигляді комп'ютерних команд. У цьому випадку модель комп'ютера повинна відтворювати всі його команди, тобто повністю відображати роботу комп'ютера на рівні мікросхем. Ця модель буде найбільш точною і найбільш дорогий, але вона дасть можливість відповісти не тільки на питання про повну загальну середню часу виконання програм в комп'ютері, але і на ті, які пов'язані з роботою комп'ютера. Наприклад, як зміняться параметри роботи комп'ютера в разі заміни процесора на інший з іншою системою команд.

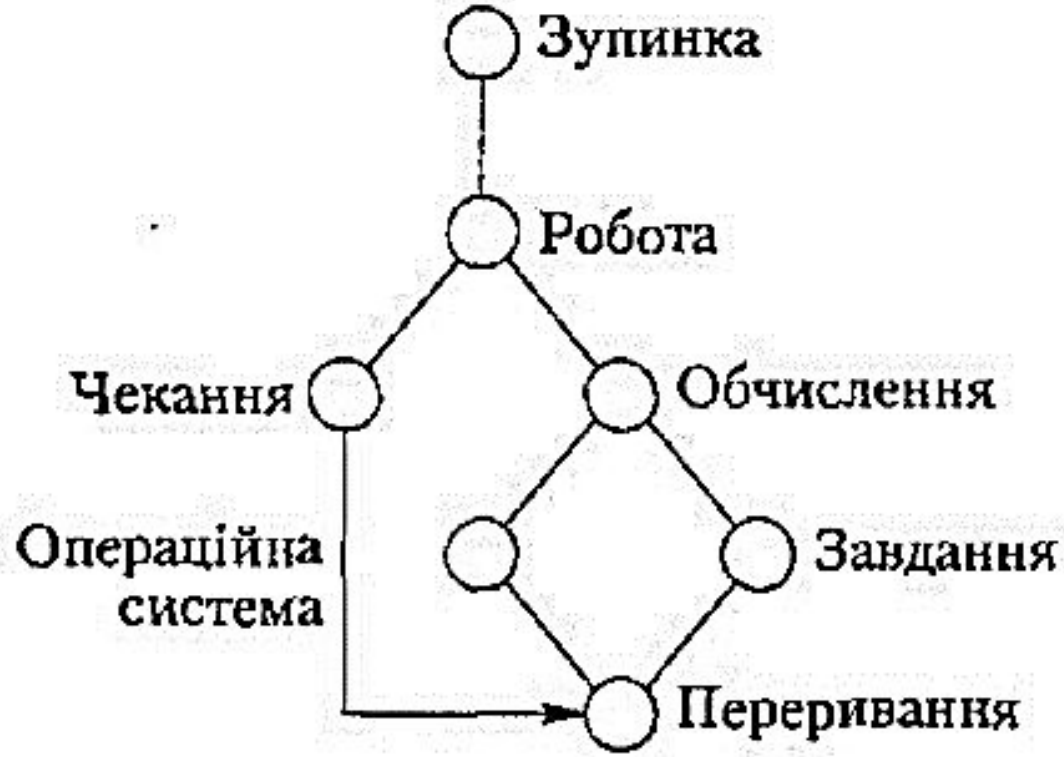
Опис змінниих моделі.

На етапі розробки концептуальної моделі визначаються вимоги до вхідних даних. Для цього використовуються різні методи вимірювань, якщо моделируемая система реально існує. Частина даних можна отримати з технічної та конструкторської документації системи, офіційних звітів, статистичних збірників, довідників. Вхідні і внутрішні змінні вибираються відповідно до ступеня деталізації різних частин моделі з урахуванням можливих змін і доступності початкових даних. У разі побудови стохастичною імітаційної моделі доводиться приймати рішення, які далі доцільно використовувати в моделі - емпіричні, теоретико-імовірнісні або такі, що були отримані на основі результатів статистичного аналізу.

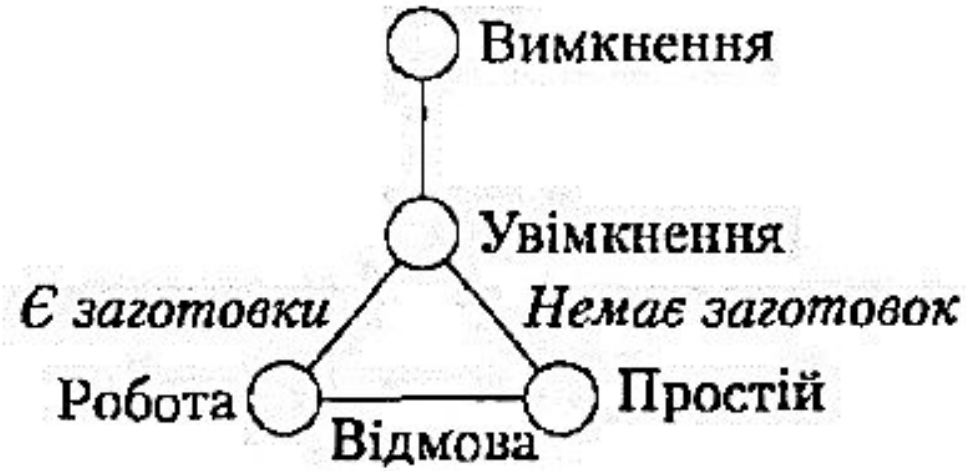
Найскладнішим є збір статистичних даних про зовнішні впливи на систему. Побудова концептуальної моделі - ітераційний процес і кількість ітерацій залежить від складності моделі системи. Спочатку будується узагальнена модель. Накопичуючи дані про систему, розробник будує більш детальну модель. Потім концептуальна модель обговорюється з користувачем.

Формалізоване зображення концептуальної моделі.

Формалізациія – такий опис моделі, яка передбачає використання математичних методів дослідження, тобто по завершенні цього етапу розробляється логіко-математичний опис модельованої системи з урахуванням динаміки її функціонування. Таким чином, формалізація - це відображення системи або процесу в точних поняттях. Зобразити структуру концептуальної моделі можна за допомогою діаграм станів системи. Простий приклад такої діаграми, що відображає роботу центрального процесора, наведено на рис. На діаграмі можна позначити причини зміни станів елементів системи. На наступному рис. Показана спрощена діаграма станів верстата. Причини зміни стану можуть задаватися аналітично як функції часу. Так, час напрацювання верстата на відмову можна задати у вигляді функції розподілу ймовірностей.


Зупинка
Робота
Чекання
Обчислення
Операційна система
Завдання
Переривання

а


Вимкнення
Увімкнення
Є заготовки
Немає заготовок
Робота
Простій
Відмова

б

Для завдання концептуальної структури можуть використовуватися формальні математичні моделі, такі як мережі Петрі, мережі СМО, потокові діаграми та ін. Уже на цьому етапі на вибір структури може істотно впливати вибір програмних засобів реалізації імітаційної моделі. Сучасні програмні засоби імітаційного моделювання мають різні візуальні компоненти, за допомогою яких можна побудувати концептуальну модель.

Побудова концептуальної моделі - найвідповідальніший етап моделювання. Неправильна концепція, покладена в основу моделі, неправильні припущення про взаємозв'язки змінних і параметрів призводять до того, що виконання подальших етапів побудови імітаційної моделі виявляється безглуздим і часто призводить до невиправданих витрат.

Після побудови концептуальної моделі перевіряють її відповідність об'єкту моделювання, використовуючи метод зворотного перетворення. Розглядається побудована модель, здійснюється перехід до прийнятих припущень, апроксимації та спрощень і повернення до реальних процесів і явищ, що моделюється.

Побудована концептуальна модель передається на обговорення експертам, які повинні визначити її відповідність цілям моделювання з точки зору вирішення поставленої проблеми.

Вибір способів реалізації імітаційної моделі.

На цьому етапі виконуються роботи, пов'язані з підготовкою і реалізацією імітаційної моделі на комп'ютері. Розробляється логічна схема моделі, яка перетворюється потім в програму. Подальша формалізація концептуальної моделі відбувається на етапі розробки структури імітаційної моделі, але слід мати на увазі, що відображення структури моделі залежить від обраних способів програмування.

Програмна реалізація імітаційної моделі може бути створена за допомогою:

- алгоритмічних мов загального призначення;
- спеціалізованих мов моделювання;
- пакетів прикладних програм для моделювання;
- засобів автоматизації програмування імітаційних моделей;
- діалогових і візуальних систем моделювання;
- інтелектуальних систем моделювання.

Структуру моделі можна зобразити з використанням елементів програмних засобів створення імітаційної моделі. Перш, ніж почати розробку структурної схеми імітаційної моделі, необхідно вибрати мову програмування, яким буде реалізована модель.

Мова, яким буде здійснена програмна реалізація імітаційної моделі, слід вибирати в залежності від складності імітаційної моделі і типу комп'ютера. Існує багато мов моделювання, орієнтованих як на суперкомп'ютери, так і на персональні комп'ютери. Спочатку необхідно визначити, як буде реалізована імітаційна модель: алгоритмічною мовою загального призначення, об'єктно-орієнтованою мовою з візуальним середовищем або з використанням спеціалізованих способів моделювання.

Для реалізації простої імітаційної моделі можна використовувати алгоритмічний мову загального призначення, що дає можливість отримати більш швидкодіючу модель.

У разі реалізації складної імітаційної моделі, в якій багато різних компонент, перевага варто віддавати спеціалізованих засобів моделювання.

У разі використання алгоритмічних мов з метою уніфікації програмних модулів і принципів структурного і модульного програмування, необхідно розробити засоби:

- управління процесом моделювання в модельному часу;
- збір статистичних даних про роботу моделі;
- генерування випадкових величин з різними законами розподілу ймовірностей для моделювання стохастичних впливів;
- діагностики помилок під час використання програми моделювання.

На наступному етапі потрібно оцінити можливості побудови програмної реалізації імітаційної моделі за допомогою цього способу і проведення експериментів з моделлю. Для цього треба відповісти на такі питання:

- які існують засоби генерування випадкових чисел і змінних?
- як здійснюється збір статистичних даних моделювання?
- які можливості засобів щодо налагодження моделі?
- наскільки засіб відповідає можливості відображення структури модельованої системи?
- які можливості способу щодо зміни структури моделі?
- наскільки просто вносити зміни в програмну реалізацію моделі?

- чи легко обробляти отриманий статистичний матеріал за результатами прогонів моделі?
- чи існує інтерфейс з іншими програмними засобами (графічними, анімаційними, статистичними, оптимізаційними пакетами)?
- наскільки зручний інтерфейс між користувачем і програмою?
- чи дає засіб можливість формувати звіт про роботу моделі в зручному для користувача вигляді?
- чи можна використовувати засоби планування проведення експериментів і оптимізації для прийняття рішень?

Розробка структурної схеми імітаційної моделі та опис її функціонування.

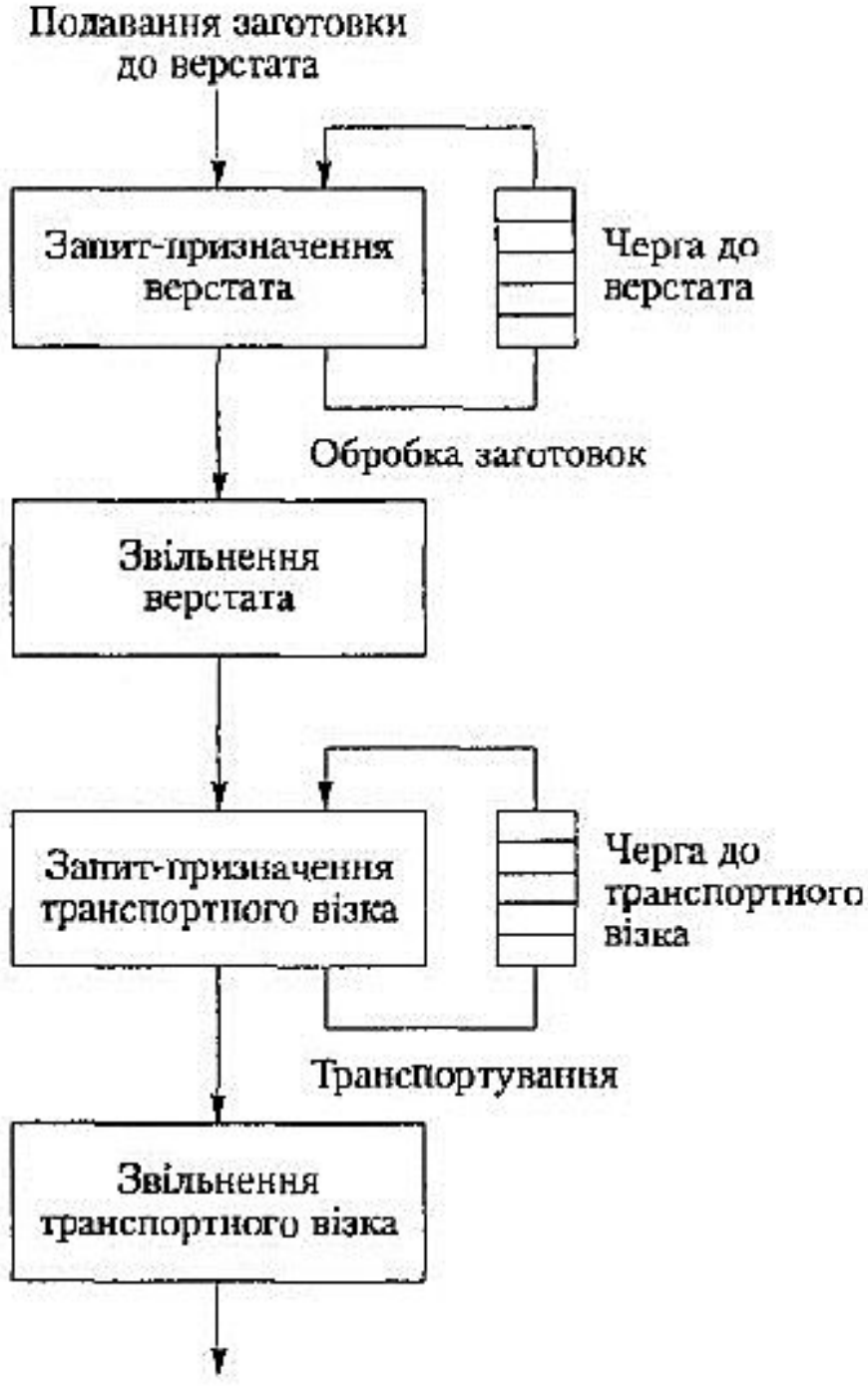
На цьому етапі об'єкт остаточно формалізується у вигляді абстрактної системи. Для цього необхідно описати структуру системи і сукупність математичних описів всіх її елементів, зовнішніх впливів і взаємодій між усіма елементами в процесі функціонування системи.

Структура системи визначається з урахуванням побудованої концептуальної моделі і обраних засобів моделювання. У разі реалізації імітаційної моделі за допомогою алгоритмічної мови загального призначення на структуру імітаційної моделі істотно впливає обраний підхід до реалізації імітаційного алгоритму. У разі вибору об'єктно-орієнтованої мови описують класи і підкласи для статичних і динамічних об'єктів, задають інтерфейси, методи класів і властивості об'єктів.

Якщо модель буде реалізована мовою моделювання GPSS, то структура моделі подається у вигляді блок-схеми, що складається з блоків різних типів. Набір блоків в блок-схемі визначає набір операторів мови, які описують структуру модельованої системи і логіку її функціонування. У такій схемі блоки відображають виконувані над транзактами операції, а стрілки між блоками - маршрути руху транзактов.

Якщо імітаційної моделлю системи є СМО, то її структуру можна зобразити у вигляді схеми, на якій є генератори вимог, що надходять в систему, буфери або черги, пристрої для обслуговування, що реалізують затримку вимог в системі. Черги до деякого ресурсу створюються через його зайнятості. На цій схемі можуть бути зображені програмні блоки, які реалізують необхідні функції в реальній системі. Якщо імітаційна модель будується як мережу Петрі, то структура мережі зображується у вигляді орієнтованого графа.

Якщо програмна реалізація імітаційної моделі здійснюється мовою, орієнтованому на процеси, то структуру моделі можна зобразити у вигляді ресурсів і шляхів проходження потоків повідомлень через ці ресурси. Структурну схему можна побудувати детальніше, якщо виділити всі типи подій для потоків повідомлень і дій, що належать ресурсів. Фрагмент такої схеми показаний на рис.


Подавання заготовки до верстата
Запит-призначення верстата
Черга до верстата
Обробка заготовок
Звільнення верстата
Запит-призначення транспортного візка
Черга до транспортного візка
Транспортування
Звільнення транспортного візка

Стрілки, що показують дії, пов'язані з потоком заготовок (повідомлень). Основними подіями в такій схемі є запит, призначення і звільнення ресурсів, таких як верстат і транспортна візок. Якщо ресурс вже зайнятий, до нього організовується чергу заготовок. На цій же схемі можуть бути позначені умови зміни потоків повідомлень. Таку структурну схему імітаційної моделі називають діаграмою або картою подій.

Програмна реалізація імітаційної моделі.

Програмну реалізацію імітаційної моделі рекомендується будувати за модульним принципом. Це дає можливість вдосконалити модель за допомогою ітераційного методу, додаючи до неї модуль за модулем. Структура програми моделі повинна відповідати структурі імітаційної моделі. Кожен модуль повинен супроводжуватися коментарем.

Для оцінювання правильності функціонування програмної реалізації імітаційної моделі проводяться пробні експерименти (тестування моделі), в яких широко використовуються засоби відрегулювати системи моделювання.

Типова помилка під час налаштування моделі пов'язана з неузгодженістю пропускної спроможності окремих елементів системи, тобто повідомлення надходять в деякі елементи моделі частіше, ніж вони встигають обслуговуватися. Тому доцільно на деяких ділянках моделі, в яких можуть накопичуватися повідомлення, задавати обмежувальні умови на довжину черги.

Після закінчення настройки функціонування програмної реалізації імітаційної моделі необхідно перевірити її працездатність у всьому діапазоні зміни вхідних змінних. Всі значення змінних в моделі повинні бути зведені до обраному об'єкті модельного часу. Для остаточного тестування моделі на контрольних прикладах необхідно залучати тих людей, які не брали участь в програмуванні моделі або майбутніх її користувачів.

Автоматизація програмування.

1. **Комп'ютерна інженерія**. 1968 г. Software Engineering.

Дисципліна, об'єктом дослідження якої є великі комп'ютерні системи та проблеми, які виникають під час їх створення. У наш час продовжують розвиватися різні методи розробки складного програмного забезпечення. В рамках комп'ютерної інженерії робиться спроба визначити абстрактну систему понять цього процесу. Кожен новий підхід передбачає свою систему, яка схожа на інші, але має деякі нюанси.

2. Язык SDL.

Specification and Description Language. Мова специфікацій і описів. Під специфікацією розуміють точне формальне визначення системи або її частини, під описом - неформальну специфікацію, яка ілюструє той чи інший аспект системи. Описи використовують на початкових етапах розробки системи або для документування системи. На етапах детального проектування використовують специфікації, за якими передбачається виконання автоматичного генерування програмного коду. Той факт, що для різних етапів розробки системи використовується одна мова, є суттєвою перевагою SDL.

Мова SDL як спосіб аналізу систем широко використовується в телекомунікаційних стандартах. Його основними складовими є структурна модель і розширений кінцевий автомат. Вони орієнтовані на здійснення специфікацій подієво-орієнтованих систем.

3. Метод OOSE.

Об'єктно-орієнтований програмний інжиніринг OOSE (Object-Oriented-Software Engineering). Основне завдання цього підходу - наблизити комп'ютерну інженерію до типового промислового процесу (наприклад, будівництво). Основний принцип підходу - об'єктна орієнтованість, як аналізу, проектування, програмування, так і опис процесу розробки програмного забезпечення в цілому. У OOSE пропонується компактне опис структури комп'ютерної інженерії, основою якої є поняття архітектури. Це поняття включає основні концепції і технології згідно з постановою як об'єктно-орієнтовані, певний набір напівформальних моделей з графічними нотаціями, наданими для опису розроблюваної системи.

4. Метод Буча.

За допомогою методу описуються об'єктна модель і візуальні засоби, а також сам процес розробки програмного забезпечення з визначенням цілей, видів діяльності і передбачуваних результатів. Розглядаються організаційні питання створення програмного забезпечення та вплив на них об'єктно-орієнтованого підходу. Метод заснований на єдиній моделі класів. Її пропонується використовувати на різних етапах розробки отримання єдиної специфікації системи, яка змінюється від однієї фази розробки до іншої.

В якості основних понять використовуються методологія і метод.

Методологія - це набір методів, які використовуються на протязі всього життєвого циклу створення програмного забезпечення, з'єднаних єдиної філософської концепцією. Такий концепцією в методі Буча є об'єктно-орієнтована погляд на світ.

Метод - чітко визначений процес створення набору моделей за допомогою специфіковані нотацій; ці моделі описують різні аспекти програмного забезпечення. Ділиться на 3 частини: нотації, процес, прагматика.

Основа методу - можливість розглядати розроблювану систему з різних поглядів. Результат такого розгляду називається моделлю системи. Типи моделей: логічна, фізична, статична, динамічна.

Нотація - це графічна мова для опису моделей. Ця частина методу є формальною.

Процес - це опис цілей, видів діяльності і результатів для різних фаз об'єктно-орієнтованого аналізу і проектування. Процес не формалізується як набір процедур, а розділяється на частини, для яких описуються інтерфейсні характеристики.

Прагматика - це та специфіка об'єктно-орієнтованого підходу, яка проявляється в організаційних питаннях створення програмного забезпечення: управління проектом, персоналом, ризиками, версіями системи, конкретні програмні засоби підтримки розробки програмного забезпечення. Ця частина методу найбільш формальна.

5. Язык UML (Unified Modeling Language).

Поняття UML, на яких засновані засоби структурної декомпозиції проекту і розроблюваної системи:

Пакет - простір імен проекту, який складається з безлічі сутностей, виражених за допомогою понять і діаграм UML.

Модель - це тип пакету, який є певним закінченим чином системи, що описує її з певного погляду.

Погляд - певний спосіб бачення системи, з урахуванням якого будується певна модель системи. Погляд включає в себе набір графічних нотацій: і їх семантику. Виділяються статичний, випадків використання, взаємодії, звичайно-автоматичний, активностей, фізичний, керований.

Підсистема - це вид пакета, який описує певну частину системи, виділену в єдине ціле по реалізаційних або функціональним міркувань. Структура підсистеми ділиться на 2 частини - декларативну і реалізаційну. Перша визначає зовнішню поведінку підсистеми і може включати в себе випадки використання, інтерфейси і ін. Реалізаційна частина описує, яким чином реалізується декларативна частина.

6. Методологія ROOM.

Методологія ROOM (Real-Time Object-Oriented Modeling) - це об'єктно-орієнтована розробка систем реального часу. Її розвиток пов'язаний з канадською компанією ObjectTime Limited, яка на основі цієї методології випустила програмний продукт ObjectTime. Передбачені 2 рівня подачі розроблюваної системи:

- рівень схем;
- рівень деталізації.

Виділення цих рівнів направлено на автоматичну кодогенерацію. Ця методологія істотно відрізняється від UML, де пропонуються тільки погляди на систему (view). Для рівня схем методологія ROOM пропонує набір графічних нотацій, що дозволяє зобразити структуру системи (класів і об'єктів) і опис її поведінки.

7. Метод RUP.

Корпорація IBM Rational Corp. створила надбудову над UML під назвою RUP (Rational Unified Process), яка дає можливість систематизувати процес створення програмного забезпечення на основі UML, пропонуючи використовувати повний набір програмних продуктів.

Одним з основних понять методу є фаза. Фаза - це етап розробки системи. Розглядаються таку силу-силенну фаз:

1. Початок - фаза визначення меж проекту, оцінювання реальності його виконання (терміни, плани, кошти, люди, ризики).
2. Деталізація - фаза створення архітектурного прототипу системи, визначення вимоги до проекту, його вартості і терміну виконання, складання детального плану роботи.
3. Конструювання - фаза реалізації проекту.
4. Передача - передача системи замовнику.

Розробка системи виконується в 4 фази за допомогою робочих процесів (workflow), які є послідовністю дій, пов'язаних загальною специфікою. Передбачається виконання таких робочих процесів:

1. Бізнес - моделювання (Business Modeling).
2. Винесення вимог (Requirements).
3. Аналіз і проектування (Analysis and Design).
4. Реалізація (Implementation).
5. Тестування (Testing).
6. Впровадження системи (Deployment).

Крім того, виділяють такі допоміжні процеси:

- Управління проектом (Project Management).
- Створення конфігурації змін і управління ними (Configuration and Change Management)
- Створення конфігурації середовища (Environment).

Програмні генератори імітаційних моделей.

З метою спрощення процесу створення імітаційних моделей останнім часом розробляються діалогові або інтерактивні системи моделювання, які за допомогою інтерфейсу транслюють висловлювання на природній мові в програмні реалізації імітаційних моделей. Перші спроби побудови таких систем робилися з використанням мови Simula, але кошти автоматичного програмування не були створені через складність реальних модельованих систем. Найбільшого успіху досягли під час розробки спеціалізованих систем моделювання для вузьких областей. У той же час, бажано, щоб вони не враховували специфічні особливості певної галузі. Цього можна досягти завдяки уніфікації опису імітаційних моделей, базуючись на єдиній математичній основі для всіх форм зображення об'єкта, що моделюється.

Технологія створення засобів автоматичного генерування імітаційних моделей передбачає:

- вибір ефективного і коректного конструктивного безлічі елементів моделі;
- розробку засобів специфікації, вибір і налаштування елементів моделі;
- наявність засобів конструювання моделі з обраних елементів.

Виділимо мову імітаційного моделювання GPSS. За допомогою блоків цієї мови можна побудувати дискретно-подієві моделі загального вигляду. Створення генератора імітаційних моделей, в якому як проміжний код використовується код за допомогою мови GPSS, дає можливість в разі потреби змінювати або розширювати сгенерированную програмну реалізацію імітаційної моделі. Що стосується формальної структури генератора імітаційних моделей, то відомо, що найширший клас структур імітаційних моделей покривають стохастичні мережі СМО, які використовуються під час моделювання інформаційних, технологічних, транспортних, ділових процесів, систем обслуговування загального вигляду.

Вибір класу структур моделей визначає обмеження на використання об'єктів або систем конкретної предметної області, тобто їх необхідно зображати як дискретні об'єкти класу стохастичних мереж СМО.