МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ЗАСТОСУВАННЯ МЕТОДІВ ІНСЕРЦІЙНОГО МОДЕЛЮВАННЯ ДО АНАЛІЗУ БІЗНЕС ПРОЦЕСІВ УНІВЕРСИТЕТУ

Дипломна робота

на здобуття ступеня вищої освіти магістр

Виконала: студентка 2 курсу 261М групи
Спеціальності: 126 Інформаційні системи та технології
Приймак Каріна Сергіївна
Керівник: доктор фізико-математичних наук, професор Песчаненко Володимир Сергійович
Рецензент: кандидат педогогічних наук, доцент Гончаренко Тетяна Леонідівна

Херсон – 2019

ЗМІСТ

ВСТУП

У даній роботі пропонується метод інсерційного моделювання – напрямку, який розвивається на протязі останнього десятиліття як підхід до побудови загальної теорії взаємодії агентів та середовищ у складних розподілених багатоагентних системах.

Актуальність проблеми

Сьогоднішній світ важко уявити без комп'ютерів, так само, як без електрики, телебачення та інших досягнень технічного прогресу. Комп'ютери вже не тільки вирішують науково-технічні і інженерні завдання - комп'ютеризація охоплює все більше сфер соціального життя, обслуговування, економіки та побуту.

А ще півстоліття тому уявити собі сучасний рівень комп'ютеризації людського життя могли тільки фантасти і найбільш освічені вчені, що визначали стратегічні напрямки розвитку комп'ютерної науки, без якої всі ці досягнення були б неможливі.

Одним з таких видатних вчених був Віктор Глушков. Його внесок у світову науку в 1997 році оцінено медаллю Міжнародного комп'ютерного товариства «Піонер комп'ютерної техніки», яку він отримав (посмертно) за «заснування першого в СРСР Інституту кібернетики НАН України, створення теорії цифрових автоматів і роботи в області макроконвеєрних архітектур обчислювальних систем».

Сучасні комп'ютерні системи характеризуються великою складністю, до їх розробки залучаються великі колективи людей. Прикладами можуть служити телекомунікаційні системи, вбудовані системи управління автомобілем або автоматизованим підприємством. Тому проблема забезпечення правильності в прийнятті рішень щодо організації взаємодії компонентів системи, відповідності її реалізації

задумам творців і вимогам замовників стоїть дуже гостро. Адже виправлення помилки після закінчення розробки обходиться досить дорого. Тому дуже важливо забезпечити правильність проектування системи починаючи з ранніх стадій. О. Летичевський ввів нову технологію розробки програмного забезпечення складних розподілених систем – інерційне моделювання. Вона базується на автоматно-алгебраїчних моделях взаємодії агентів і середовищ і забезпечує якісно новий рівень надійності процесу розробки складних систем.

Метою даного дослідження є аналіз економічних процесів університету засобами інсерційного програмування. В ході виконання роботи буде представлено дослідження котре базується на основі методів інсерційного моделювання та алгебраїчного програмування, як інструмент дослідження бізнес процесів університету.

Об'єктом дослідження виступають економічні процеси університету.

Предметом дослідження являються методи алгебраїчного програмування та інсерційного моделювання в аналізі економічних процесів університету.

Завдання роботи

Для досягнення поставлених цілей необхідно виконати наступні завдання:

1. Розробити формальну модель економічних процесів університету.
2. Реалізувати модель засобами алгебраїчного програмування та IMS.
3. Порівняти отриманні данні при моделюванні з реальними

Методи дослідження

В ході наукового дослідження будуть використані наступні методи дослідження: синтез, аналіз, порівняння, моделювання, методи верефікації.

Наукова новизна

Результати дисертації є новими і полягають у використанні методів інсерційного моделювання та алгебраїчного програмування до аналізу економічних процесів університету. В ході виконання роботи буде представлено підхід до аналізу та візуалізації економічних процесів університету.

Практична цінність

Представлені результати дозволяють науково обґрунтовано вирішувати важливі завдання, що виникають при моделюванні економічних процесів університет, проаналізувати бізнес процеси університету, запропонувати підхід до аналізу фінансової моделі університету на стійкість та адекватність.

Апробація. Матеріали дипломної роботи апробовані на міжнародній конференції ICTERI - 2019, Formalization and Algebraic Modeling of University Economics.

РОЗДІЛ 1 . МЕТОДИ ТА ЗАСОБИ ВЕРИФІКАЦІЇ ЕКОНОМІЧНИХ МОДЕЛЕЙ

1.1. Властивості

При моделюванні дослідник повинен бути впевнений в правильності моделі, у відповідності моделі реальному прототипу. Точність математичного моделювання залежить від того, наскільки добре математична модель відображає властивості об'єкта.

На точність моделювання впливають такі особливості:

- спрощення моделі;
- помилки при побудові моделі;
- використання елементів з низькою точністю, з лінійною апроксимацією;
- наявність в моделі вироджених кінцевих елементів;
- некоректні зв'язку;
- некоректні параметри моделей;
- некоректні властивості елементів;
- некоректні початкові і граничні умови;
- похибки методу розрахунків.

Верифікація моделі - перевірка її істинності, адекватності. У відношення до дескриптивних моделей. Верифікація моделі зводиться до співставлення результатів розрахунків по моделі відповідними даними дійсності - фактами і закономірностями економічного розвитку. Верифікації імітаційної моделі є перевірка відповідності її поведінки припущеннях експериментатора.

Перевірка моделі (model checking) - один з підходів до вирішення проблеми верифікації. В якості мов специфікації для вираження властивостей систем при цьому підході використовуються темпоральні логіки. Завдання перевірки моделі полягає у визначенні виконуваності

для системи, заданої формальним чином (у вигляді формальної моделі), властивість, представленої формулою темпоральної логіки.

Верифікація імітаційної моделі є перевірка відповідності її поведінки припущеннях експериментатора. Коли модель організована у вигляді обчислювальної програми для комп'ютера, то спочатку виправляють помилки в її записі на алгоритмічній мові, а потім переходять до верифікації. Це перший етап дійсної підготовки до імітаційного експерименту. Підбираються деякі вихідні дані, для яких можуть бути передбачені результати прорахунку. Якщо виявиться, що ЕОМ видає дані, що суперечать тим, які очікувалися при формуванні моделі, значить, модель невірна, тобто не відповідає закладеним в неї очікуванням. У зворотному випадку переходять до наступного етапу перевірки працездатності моделі - її валідації.

Валідація моделі (model validation) - перевірка відповідності даних, одержуваних в процесі машинної імітації, реальному ходу явищ, для опису яких створена модель. Вона проводиться тоді, коли експериментатор переконався на попередній стадії (верифікації) в правильності структури (логіки) моделі, і полягає в тому, що вихідні дані після розрахунку на комп'ютері зіставляються з наявними статистичними відомостями про моделюючу систему.

У більш загальному вигляді верифікація - це підтвердження на основі наданних об'єктивних доказів того, що встановлені вимоги були виконані. Якщо образно, то верифікація - процедура зіставлення того, що зроблено (або ще поки робиться), з тим, що було задумано (наказано) зробити, тобто зіставлення закінченого або проміжного результату з вхідними вимогами - "погляд назад".

Валідація - підтвердження на основі наданням об'єктивних доказів того, що вимоги, призначені для конкретного використання або застосування, виконані. Образно кажучи, валідація - це процедура зіставлення того, що задумано зробити (або ще поки робиться), з тим,

що необхідно споживачеві для конкретного застосування, тобто зіставлення планованого або проміжного результату діяльності з поточними вихідними вимогами - "погляд вперед".

Верифікація є інструментом валідації, її частиною. Верифікація триває аж до моменту кодування програми, а валідація здійснюється безпосередньо після. Тому в практиці моделювання з використанням ЕОМ верифікація та валідація моделей завершується після проведення обчислювального експерименту і підтвердження його результатами відповідності як реальним процесам досліджуваного об'єкта, так відповідності конкретним умовам застосовності (або вимогам). Однак в більшості випадків процеси верифікації, валідації, тестування і реалізації перетинаються за часом.

Недетермінізм в суттєвих модифікаціях часто означає, що поведінка цільової системи залежить від ще не визначеної частини стану і, оскільки перевірити правильність поведінки в цьому випадку все одно не можна, тестова система повинна виключати такі тестові впливи поки не накопичить інформацію про стан цільової системи, достатню щоб винести вердикт про правильність або помилковість поведінки цільової системи.

1.2. Огляд найбільш популярних сучасних програмних систем імітаційного моделювання

Імітаційне моделювання - це метод дослідження, при якому вивчаєма система замінюється моделлю, що з достатньою точністю описує реальну систему, і з нею проводяться експерименти з метою отримання інформації про цю систему. Існує чотири види імітаційного моделювання: динамічне моделювання, системна динаміка, дискретно - подієве моделювання і агентне моделювання.

Динамічне моделювання застосовується для моделювання динамічних систем (механічні або фізичні процеси, системи управління) і вони описуються алгебраїчними або диференціальними рівняннями.

Системна динаміка дозволяє зрозуміти структуру і динаміку складних систем і головним чином використовується в довгострокових, стратегічних моделях зі зворотним зв'язком.

Дискретно - подієве моделювання використовується для побудови моделі, що відображає розвиток системи в часі, коли стани змінних змінюються миттєво в конкретні моменти часу.

Агентне моделювання - відносно нове і його основою є поняття «агент» - якась сутність, що володіє активністю, автономною поведінкою, яка може приймати рішення відповідно до деякого набороу правил, взаємодіяти з середеовищем, а також самостійно змінюватися. Агенти можуть представляти пішоходів, автомобілі або роботів в фізичному просторі, клієнта або продавця на середньому рівні, або ж конкуруючі компанії на високому [1]. Агентне моделювання застосовується для імітації інтелектуальних, децентралізованих і розподілених систем.

Існує цілий ряд програмних інструментів, орієнтованих на перераховані вище види моделювання:

1. Динамічні системи (Maple, Matlab).
2. Системна динаміка (PowerSim, iThink).
3. Дискретно - подієве моделювання (Arena, GPSS World і т.д.).
4. Мультиагентні системи (AnyLogic).

Тут слід зазначити, що в даний час дана класифікація багато в чому стає умовною, оскільки сучасні інтегровані засоби моделювання охоплюють як динамічні системи, так і системну динаміку, дискретно - подієве моделювання та мультиагентні системи (наприклад, AnyLogic). Але таких програмних засобів на ринку ще дуже мало, а найбільш

представницької є група систем імітаційного моделювання, орієнтований на дискретні системи.

Найбільш підходящим видом імітаційного моделювання стосовно до обчислювальних систем і мереж передачі даних є дискретно - подієве моделювання і агентне моделювання.

Розробка імітаційних моделей виконується за такими етапами:

1. Розробка концептуальної моделі;
2. Підготовка вихідних даних;
3. Вибір засобів моделювання;
4. Розробка програмної моделі;
5. Перевірка адекватності і коректування моделі;
6. Планування машинних експериментів;
7. Моделювання;
8. Аналіз результатів моделювання.

При виборі засобів імітаційного моделювання слід враховувати всі можливості, що надаються ними, які можна об'єднати в такі групи:

- основні характеристики;
- сумісне програмне забезпечення;
- анімація;
- статистичні можливості;
- звіти з вихідними даними та графіками;
- послуги, що надаються замовникам і документація.

Нижче будуть розглянуті основні на дані момент програмні інструменти моделювання систем.

Програма Maple досі є одним з лідерів серед універсальних систем символьних обчислень. Вона надає користувачеві зручне інтелектуальне середовище для математичних досліджень будь-якого рівня і користується особливою популярністю в науковому середовищі. Відзначимо, що символьний аналізатор програми Maple є найбільш сильною частиною цього ПО, тому саме він був запозичений і

включений в ряд інших CAE-пакетів, таких як MathCad і MatLab, а також до складу пакетів для підготовки наукових публікацій Scientific WorkPlace і Math Office for Word .

Maple надає зручне середовище для комп'ютерних експериментів, в ході яких пробуються різні підходи до задачі, аналізуються приватні рішення, а при необхідності програмування відбираються фрагменти, що вимагають особливої швидкості. Пакет дозволяє створювати інтегровані середовища за участю інших систем і універсальних мов програмування високого рівня. Коли розрахунки проведені і потрібно оформити результати, то можна використовувати засоби цього пакета для візуалізації даних і підготовки ілюстрацій для публікації. Для завершення роботи залишається підготувати друкований матеріал (звіт, статтю, книгу) прямо в середовищі Maple, а потім можна приступати до чергового дослідження. Робота проходить інтерактивно - користувач вводить команди і тут же бачить на екрані результат їх виконання. При цьому пакет Maple зовсім не схожий на традиційну середу програмування, де потрібно жорстка формалізація всіх змінних і дій з ними. Тут же автоматично забезпечується вибір відповідних типів змінних і перевіряється коректність виконання операцій, так що в загальному випадку опис змінних і сувора формалізація записів не потріба.

Пакет Maple складається з ядра (процедур, написаних на мові C і добре оптимізованих), бібліотеки, написаної на Maple-мові, і розвиненого зовнішнього інтерфейсу. Ядро виконує більшість базових операцій, а бібліотека містить безліч команд - процедур, що виконуються в режимі інтерпретації. Інтерфейс Maple заснований на концепції робочого поля (worksheet) або документа, що містить рядки введення-виведення і текст, а також графіком.

MATLAB - один з найпотужніших на сьогоднішній день пакетів обробки даних. Назва розшифровується як Matrix Laboratory. Система MatLab відноситься до середнього рівня продуктів, призначених для символьної математики, але розрахована на широке застосування в сфері CAE (тобто сильна і в інших областях). MatLab - одна з найстаріших, ретельно опрацьованих і перевірених часом систем автоматизації математичних розрахунків, побудована на розширеному поданні та застосуванні матричних операцій. Це знайшло відображення і в самій назві системи - MATrix LABoratory, тобто матрична лабораторія. Однак синтаксис мови програмування системи продуманий настільки ретельно, що дана орієнтація майже не відчувається тими користувачами, яких не цікавлять безпосередньо матричні обчислення.

Незважаючи на те що спочатку MatLab призначалася виключно для обчислень, в процесі еволюції (а зараз випущена вже версія 7), на додаток до прекрасних обчислювальних засобів, у фірми Waterloo Maple за ліцензією для MatLab було придбано ядро символьних перетворень, а також з'явилися бібліотеки, які забезпечують в MatLab унікальні для математичних пакетів функції. Наприклад, широко відома бібліотека Simulink, реалізуючи принцип візуального програмування, дозволяє побудувати логічну схему складної системи управління з одних тільки стандартних блоків, не написавши при цьому ні строчки коду. Після конструювання такої схеми можна детально проаналізувати її роботу.

В системі MatLab також існують широкі можливості для програмування. Її бібліотека C Math (компілятор MatLab) є об'єктної і містить понад 300 процедур обробки даних на мові C. Усередині пакета можна використовувати як процедури самої MatLab, так і стандартні процедури мови C, що робить цей інструмент найпотужнішою підмогою при розробці додатків (використовуючи компілятор C Math, можна вбудовувати будь-які процедури MatLab в готові програми). Пакет Powersim є прекрасним засобом створення безперервних моделей. Однак

з точки зору дискретного моделювання він недостатньо ефективний. Powersim підходить користувачам, яким потрібна побудова безперервних моделей, і які хочуть вивчити досить складну систему позначень Systems Dynamics. Ця система реалізована в Powersim у вигляді наступних конструкцій. Перша, звана рівнем, складається з таких компонентів, як гроші, складські запаси, шкідливі викиди і т. Д. Друга конструкція - потік. Він об'єднує елементи, які переміщаються між рівнями. Допоміжні атрибути, що нагадують осередки формул в електронних таблицях, і константи дозволяють модифікувати потік.

У процесі побудови моделі розробник розміщує блоки і визначає змінні, які складають математичну основу кожного блоку. У цьому йому допомагає діалогове вікно Define Variable, в якому міститься список всіх допустимих змінних для кожного блоку і поле для опису змінних, щоб користувачі легше могли зрозуміти цю модель. Пакет Powersim виділяється серед інших пакетів здатністю обробляти масиви і підтримувати колективну роботу, а також тим, що містить бібліотеку з великим числом функцій.

Масиви зручні і для створення моделей, в конструкції яких рівні змінюють свій стан, а розробник хоче простежити за цими змінами. Powersim включає в себе більше 150 функцій, розділених на 16 груп, в тому числі фінансову, математичну, статистичну, графічну і історичну. Подібно до інших пакетів, Powersim використовує при виконанні моделей засоби анімації. Ключові параметри, діаграми і таблиці можна виводити безпосередньо на екран моделювання, спрощуючи тим самим перегляд результатів. Функція Multiuser Game надає можливість кільком користувачам одночасно запускати модель, щоб спільно над нею працювати. Це особливо корисно для робочих колективів, які проводять тестування. Powersim містить багато стандартних засобів Windows-додатків, такі як меню і інструментальні лінійки, і підтримує технології Dynamic Data Exchange (DDE) і Object Linking and Embedding (OLE).

Недоліки пакета Powersim: • обмежена підтримка дискретного моделювання.

Powersim призначений для побудови безперервних і частково дискретних моделей. Основна мета мови Powersim полягає в побудові опису або математичної моделі уявної або реальної системи. Будь-яка модель складається з безлічі взаємозалежних елементів, що описуються змінними. Елементи моделі і зв'язку між ними визначають структуру моделі. Для визначення імітаційних моделей в Powersim існує редактор діаграм, в якому всі змінні представляються графічними об'єктами (піктограмами), з'єднаними між собою за допомогою стрілок, що позначають потоки і зв'язку. Кожна зв'язок відображає деяку залежність між змінними, з'єднаними даної зв'язком. Точне визначення виду залежності визначається рівнянням, записаному на мові Powersim. Powersim дає можливість бачити на одній і тій же діаграмі структуру і рівняння моделі, а також її поведінку. Для відображення поведінки моделі в ході моделювання існують анімаційні засоби і динамічні об'єкти, які можна розміщувати на діаграмі довільним чином. Поведінка моделі визначається з імітаційних експериментів (імітацій) з моделлю і може бути використано не тільки для аналізу самої моделі, а й для поліпшення розуміння поведінки модельованої системи в різних ситуаціях. Мова імітаційного моделювання Powersim може бути використаний для побудови моделей як простих, так і складних систем. Відомо, що для складних систем характерна множинність опису. Тому для них не можна побудувати єдину, істинно вірну модель, а можна лише описати їх поведінку за допомогою тих чи інших моделей, що відображають характерну поведінку модельованих систем в конкретних ситуаціях. Проте, Powersim є досить потужним інструментом, що дозволяє не тільки швидко і наочно будувати і аналізувати системно-динамічні моделі, а й демонструвати в доступній формі результати моделювання широкого кола людей, які не обов'язково є фахівцями в

галузі математичного моделювання. Powersim відноситься до того сімейства мов імітаційного моделювання (Dynamo, Stella / iThink, Vensim, Rusim), який досить швидко і ефективно дозволяє оволодіти технікою імітаційного моделювання представникам не тільки природних, а й гуманітарних наук і в цьому сенсі є більш доступним в освоєнні, ніж чисто технічні середовища розробки імітаційних моделей, такі як GPSS, GASP, SIMSCRIPT, SIMULA, SLAM, SIMULINK (MATLAB), РДО і ін., що володіють більш широкими можливостями, але припускають наявність у користувача не тільки хорошої підготовки в області математичного моделювання, але і програмування.

Програмний продукт iThink був розроблений спеціально для моделювання системної динаміки, компанією ISee systems, inc. iThink - це мова візуального програмування для моделювання системної динаміки, представлений Баррі Ричмондом в 1985 році. Програма дозволяє користувачам запускати моделі, створені у вигляді графічних уявлень системи, використовуючи чотири фундаментальні будівельні блоки.

Пакет Ithink - один з найбільш потужних продуктів, розглянутих нами. С точки зору безперервного моделювання він відстає від Powersim, проте краще підтримує дискретне моделювання. Крім того, пакет Ithink забезпечений чудовими навчальною програмою і документацією, а також великою кількістю блоків для складання моделі. Пакет випускається в двох версіях - Basic і Authoring. Версія Authoring, яка порівнювалася з іншими пакетами, дозволяє розробнику включати в модель лінійки з двигунами і інші засоби управління моделлю, а також вводити діаграми та інші зображення прямо в модель, щоб користувачі могли контролювати процес моделювання і відразу бачити його результати. Подібно Powersim, пакет Ithink використовує систему позначень Systems Dynamics, яка в основному орієнтована на безперервне моделювання. Для реалізації цієї системи служать

конструкції чотирьох типів: станції, аналогічні рівням в пакеті Powersim, потоки, конвертери, що нагадують допоміжні атрибути з Powersim, і з'єднувачі, відповідні зв'язків.

Щоб створювати дискретні моделі, Ithink використовує три спеціальні станції:

- черги, в яких елементи обробляються за принципом першим прийшов - першим обслужений;
- сховища, які перед початком обслуговування накопичують задану кількість елементів і зручні при пакетній обробці;
- транспортери, які передають елементи між станціями.

Моделі, побудовані за допомогою Ithink, складаються з рівнів і ієрархій. Користувач будує опис моделі на високому рівні за допомогою середовищ моделювання процесів, кожна з яких дозволяє створити модель однієї підсистеми. Завершивши опис, розробник переходить на наступний щабель деталізації і вводить в кожну підмодель необхідні конструкції. Між підмоделями встановлюються зв'язки, що вказують на взаємодію підсистем. Побудувавши модель, забезпечену необхідним числом ієрархічних рівнів, розробник переходить в режим моделювання, щоб визначити математичні зв'язки між станціями, потоками і іншими конструкціями.

Подібно Powersim, пакет Ithink пропонує розробнику список допустимих 14 змінних для визначення математичних зв'язків. Ithink забезпечує проведення аналізу чутливості моделі шляхом її багаторазового запуску з різними вхідними параметрами. Результати кожного прогону виводяться в окремому рядку вихідний діаграми. Вихідними даними є основні види розподілів, що застосовуються для статистичного аналізу або діаграми. При виконанні моделі Ithink використовує кошти анімації, що переміщують розташовані на різних рівнях станції відповідно до логіки моделі. Результати моделювання виводяться у вигляді тимчасових діаграм або діаграм розкиду. Щоб

задати діаграму або таблицю, розробнику потрібно лише вибрати використовувані величини і вказати необхідні параметри. Хоча вибір форматів для виведення результату в Ithink не так широкий, як в Extend, з цієї точки зору він перевершує як Powersim, так і Process Charter. Компанія High Performance Systems забезпечила пакет Ithink керівництвом по моделюванню під назвою «Системне мислення» і спеціальною інформацією, призначеної для користувачів, що працюють в різних галузях бізнесу. Недоліки пакета Ithink: • підтримує меншу кількість функцій, ніж Powersim.

Arena, розроблене компанією Systems Modeling Corporation програмне забезпечення для імітаційного моделювання, дозволяє створювати рухомі комп'ютерні моделі. Основа технологій Arena - мова моделювання SIMAN і система Cinema Animation. SIMAN, вперше реалізована в 1982 р. - надзвичайно гнучка і виразна мова моделювання. Вона постійно вдосконалюється шляхом додавання нових можливостей. Для відображення результатів моделювання використовується анімаційна система Cinema animation, відома на ринку з 1984 р. Процес моделювання має таку структуру: спочатку користувач крок за кроком будує в візуальному редакторі системи Arena модель, потім система генерує по ній відповідний код на SIMAN, після чого автоматично запускається Cinema animation.

Arena забезпечена зручним об'єктно-орієнтованим інтерфейсом і володіє дивовижними можливостями по адаптації до всіляких предметних областей. В цілому система виключно проста у використанні. Імітаційна модель Arena включає такі основні елементи: джерела і стоки (Create і Dispose), процеси (Process) і черги (Queue). Джерела - це елементи, від яких в модель надходить інформація або об'єкти. Швидкість надходження даних або об'єктів від джерела зазвичай задається статистичною функцією. Сток являє собою пристрій для прийому інформації або об'єктів. Поняття черги близьке до поняття

сховища даних - це місце, де об'єкти очікують обробки. Час обробки об'єктів (продуктивність) в різних процесах може бути різним. В результаті перед деякими процесами можуть накопичуватися об'єкти, які очікують своєї черги. Часто метою імітаційного моделювання є мінімізація кількості об'єктів в чергах. Тип черги в імітаційній моделі може бути конкретизований. Черга може бути схожа на стек - прийшли останніми в чергу об'єкти першими відправляються на подальшу обробку (LIFO: last-in - first-out). Альтернативою стеку може бути послідовна обробка, коли першими на подальшу обробку відправляються об'єкти, що прийшли першими (FIFO: first-in - first-out). Можуть бути задані і більш складні алгоритми обробки черги. Процеси - це аналог робіт у функціональній моделі. В імітаційній моделі може бути задана продуктивність процесів.

Середовище імітаційного моделювання Arena:

1. Документування, анімація і демонстрація змін і динаміки складних процесів і систем
2. Аналіз бізнес-процесів, пов'язаних з обслуговуванням клієнтів або документообігом, зовнішніх і внутрішніх процедур в страхуванні, фінансової і банківської сфери
3. Аналіз потоків в простих виробничих процесах
4. Моделювання складних дій по роботі з клієнтами
5. Детальний аналіз складних виробничих процесів, що включають інтенсивні операції з транспортування матеріалів (із застосуванням автонавантажувачів, робокар і персоналу).

Таким чином, Arena - це система імітаційного моделювання, що дозволяє створювати рухомі (імітаційні) комп'ютерні моделі, використовуючи які можна адекватно описати і прогнозувати реальні процеси.

Мова імітаційного моделювання GPSS являє собою набір команд і операторів для опису об'єкта моделювання.На основі мови імітаційного

моделювання GPSS розроблений ряд систем імітаційного моделювання. У даній роботі розглядається одна з таких систем - GPSS World.Основне призначення мови моделювання GPSS і розробленої на його основі системи GPSS World - імітаційне моделювання дискретних динамічних систем.Система GPSS World - це потужне середовище комп'ютерного моделювання загального призначення, розроблена для професіоналів в області моделювання. Це комплексний моделюючий інструмент, який охоплює області як дискретного, так і безперервного комп'ютерного моделювання, що володіє найвищим рівнем інтерактивності і візуального представлення інформації.

GPSS World є найбільш сучасною реалізацією мови GPSS, доповненої допоміжною мовою PLUS. GPSS World включає в себе 53 типів блоків і 25 команд, велика кількість системних числових атрибутів. Крім того, 12 типів операторів складають мову PLUS - Programming Language Under Simulation. Ефективність PLUS багато в чому забезпечується великою бібліотекою процедур.GPSS World є об'єктно-орієнтованою мовою. У сукупність його об'єктів входять об'єкти «Модель», «Процес моделювання», «Звіт» та текстові об'єкти.

• Об'єкт «Модель» - головним чином містить оператори моделі, а також набір вбудованих налаштувань. Крім того, включає в себе закладки та циркулярний список синтаксичних помилок.

• Об'єкт «Процес моделювання» створюються при трансляції операторів об'єкта «Модель». Після цього для зміни його стану застосовуються команди. Ці команди можуть входити в об'єкт «Модель» або передаватися об'єкту «Процес моделювання» в інтерактивному режимі.

• Об'єкт «Звіт». Однією з найсильніших сторін GPSS завжди були стандартні звіти. По суті без зусиль з боку розробника моделі по завершенню моделювання автоматично створюється звіт про всі об'єкти GPSS, що містяться в моделі.

• Текстовий об'єкт - це спосіб представлення звичайного текстового файлу в GPSS World. В основному вони використовуються спільно з командами INCLUDE для підключення деякого набору операторів, що використовується в різних моделях. Крім того, закріпивши команду INCLUDE за гарячою клавішею, можна інтерактивно передавати об'єкту «Процес моделювання» цілі списки керівників команд.

При імітаційному моделюванні в сучасній практиці в якості інструментального засобу отримала широке поширення система моделювання AnyLogic. AnyLogic розроблена на основі сучасних концепцій в області інформаційних технологій і результатів досліджень в теорії гібридних систем і об'єктноорієнтованого моделювання. Це комплексний інструмент, який охоплює в одній моделі основні нині напрями моделювання: дискретно-подієвого, системної динаміки, агентне.

AnyLogic є інструментальним засобом імітаційного моделювання і був створений російською компанією The AnyLogic Company. На даний момент, це єдиний засіб, що включає в себе весь набір підходів до імітаційного моделювання. Всього існує кілька версій програми, серед яких є безкоштовна - Personal Learning Edition, яка доступна для освітніх цілей і самоосвіти. Даний засіб має сучасний російськомовний інтерфейс і дозволяє будувати навіть найскладніші моделі. Потужні бібліотеки і інструменти дозволяють вирішувати широкий спектр завдань аж до стратегічних моделей розвитку великих компаній. До всього цього є можливість програмування на мові Java, що значно підвищує гнучкість і функціональність моделювання. Таким чином, за допомогою AnyLogic є можливість повністю імітувати весь бізнес-цикл [4].

Можливості, які доступні в AnyLogic [5]:

- Повністю об'єктно-орієнтована платформа, підтримка операційних систем Windows, Linux, Mac OS;

- Можливість використовувати в моделюванні агентні, системно-динамічні і дискретно-подієві підходи;
- Зручне графічне середовище, що дозволяє значно прискорити процес створення моделей;
- Підтримка створення бібліотек для багаторазового використання написаних модулів;
- Програмування на мові Java, створення призначених для користувача бібліотек і робота з базами даних;
- Багатий набір функцій розподілу дозволяє створювати складні стохастичні моделі;
- Можливість створення інтерактивної анімації для поліпшення наочності моделей;
- Велика експериментальна база, вбудована підтримка моделювань Монте Карло і передових форм оптимізації дає велику різноманітність підходів моделювання;
- Детальна теоретична довідка з прикладами російською мовою, що дозволяє швидко освоїти моделювання.

Імітаційне моделювання - це розробка і виконання на комп'ютері програмної системи, що відбиває структуру і функціонування об'єкта, що моделюється або явища в часі. Програмний комплекс, отриманий при цьому, називають імітаційної моделлю цього об'єкта або явища. Тобто імітаційна модель - це спрощена подоба реальної системи, або існуючої, або тієї, яку припускають створити [1, 2]. Можна сказати, що імітаційне моделювання - це окремий випадок математичного моделювання. Існує клас об'єктів, для яких з різних причин не розроблені аналітичні моделі, або не розроблені методи рішення отриманої моделі, або розроблені моделі не підходять для проведення результативних обчислювальних експериментів. У цьому випадку математична модель замінюється імітаційною моделлю.

На відміну від системної динаміки і дискретно-подієвих моделей агентні моделі децентралізовані. Тут не обирається поведінка системи в цілому, поведінка агентів визначається на індівідуальному рівні, а динаміка системи виникає як результат діяльності багатьох агентів. Агентне моделювання являється підходом більш потужним і універсальним, тому що воно дозволяє врахувати будь-які складні структури і поведінки. Інша важлива перевага агентного моделювання в тому, що розробка моделі можлива в відсутності знання про глобальні залежності: потрібно визначати індивідуальну логіку поведінки учасників процесу для того, щоб побудувати агентну модель і вивести з неї глобальну поведінку. Агентну модель простіше підтримувати: уточнення зазвичай робляться на локальному рівні і не вимагають глобальних змін.

Мультиагентні системи складаються з безлічі штучних агентів, які працюють спільно. Агент - це еволюція поняття "об'єкт".

Існує багато визначень агента. Загальним в них є те, що агент - деяка сутність, яка володіє активністю, автономною поведінкою, може приймати рішення відповідно до деякого набору правил, може взаємодіяти з оточенням і іншими агентами і може змінюватися. Можна сказати, що агент - це об'єкт, що володіє певними властивостями.

Наведемо основні властивості, притаманні агенту:

• автономність: агенти функціонують без прямого втручання в їх дії, вони можу самостійно контролювати свій стан і реагувати на зміни, що відбуваються;

• методи спілкування: агенти взаємодіють один з одним за допомогою деякого мови;

• реактивність: агенти здатні сприймати навколишнє середовище;

• активність: агенти володіють цілеспрямованою поведінкою і здатні самі проявляти ініціативу:

• інтелектуальна поведінка: агент здатний до навчання, здатний знаходити оптимальні способи поведінки;

• індивідуальна картина світу: кожен агент по своєму сприймає навколишнє середовище;

• мобільність: здатність до передачі коду агента.

Для того щоб агент міг вести себе певним чином, він повинен мати спеціальні «пристрої»: ефектори - органи, які впливають на середовище, рецептори - отримують інформацію від впливу середовища і процесор - він буде обробляти інформацію.

Побудова агентних моделей вимагає визначення безлічі агентів і основ їхньої поведінки, визначення взаємовідносин між агентами і теоретичних основ цих відносин, вибору платформи для агентного моделювання (5). Визначення агентів з точним завданням їх поведінки і взаємодії з іншими агентами - це основа для розробки адекватних агентних моделей.

РОЗДІЛ 2. ВИКОРИСТАННЯ ІНСЕРЦІЙНОГО МОДЕЛЮВАННЯ ДЛЯ ВЕРИФІКАЦІЇ ЕКОНОМІЧНИХ МОДЕЛЕЙ

2.1. Агенти та середовища

Інсерційне моделювання являє собою напрямок, який розвивається на протязі останнього десятиліття як підхід до побудови загальної теорії взаємодії агентів і середовищ в складних розподілених багатогоагентних системах. Основні поняття інсерційного моделювання (середовище, агенти, функція занурення) були введені в роботах [14, 20, 21], опублікованих у 90-х роках.

Інсерційне програмування - це програмування на базі моделі поведінки агентів в середовищах [7]. В основі моделі лежать поняття розміченої транзіціонной системи і відношення бісимуляційній еквівалентності.

Інсейрційне моделювання займається побудовою моделей і вивченням взаємодії агентів і середовищ в складних розподілених багатоагентних системах.

Агент - це транзіційна система, стан якої визначаються з точністю до бісимуляціонної еквівалентності.

Одночасні процеси або агенти є основними об'єктами теорії взаємодії. Агенти - це об'єкти, які можна визнати окремими від решти світу або оточення.

Вони існують у часі та просторі, змінюють свій внутрішній стан та можуть взаємодіяти з іншими агентами та середовищем, здійснюючи спостережувані дії та змінюючи своє місце серед інших агентів (мобільність). Агентами можуть бути об'єкти в реальному житті або моделі компонентів інформаційного середовища в комп'ютеризованому світі. Поняття агента формалізує такі об’єкти, як програмні компоненти, програми, користувачі, клієнти, сервери, активні компоненти розподілених баз знань тощо. Більш конкретне поняття агента використовується і в так званому агентному програмуванні, інженерній

дисципліні, присвяченій розробці інтелектуальних інтерактивних систем.

Середовище - це агент, який володіє функцією занурення. Більш точно, середовище - це набір <E, C, A, Ins>, де E - це безліч станів середовища, C - безліч дій середовища, A - безліч дій агентів, що занурюються в середу, Ins: E x F (A) → E - функція занурення. Тут F (A) - повна алгебра поведінки агентів з безліччю дій A. Таким чином, всяке середовище E допускає занурення будь-якого агента з безліччю дій A. Таким чином, будь-яке середовище E допускає занурення будь-якого агента з безліччю дій A.

Агенти є об'єктами, які можна розпізнати окремо від решти світу або навколишнього середовища. Вони існують в часі і просторі, змінюють свій внутрішній стан і можуть взаємодіяти з іншими агентами і середовищами, виконуючи дії і змінюючи місце серед інших агентів (мобільність). Агенти можуть бути об'єктами в реальному житті або моделями компонентів інформації середовище в комп'ютеризованому світі.

Основне поняття теорії взаємодії - це поведінка агентів або процесів, що взаємодіють один з одним у межах певного середовища. Звичайна точка зору полягає в тому, що середовище для даного агента - це набір усіх інших агентів, що оточують цього. У теорії взаємодії агентів та середовищ, розробленої в [15] та представленій на лекціях, поняття середовища формалізується як агент, що надається функцією вставки, що описує зміну поведінки середовища після вставки агента в нього . Після вставки одного агента середовище готова прийняти інший, і розглядається як агент, воно може бути самостійно вставлено в інше середовище вищого рівня. Тому багатоагентні та багаторівневі середовища можуть бути створені за допомогою функцій вставки.

Функція вставки сприймає поведінку агента та поведінку середовища як аргументи і повертає нову поведінку цього середовища.

І агенти, і середовища характеризуються своєю поведінкою, представленою як елементи алгебри безперервної поведінки.

Неформально основні положення парадигми інсерційного моделювання можна сформулювати наступним чином:

1. Держава є ієрархія середовищ і агентів, занурених у ці середовища.
2. Агенти і середовища є сутності, що еволюціонують у часі і володіють спостережуваною поведінкою.
3. Середовище для агента - це сукупність всіх інших агентів оточуючих його.
4. Занурення агента в середовище змінює поведінку цього середовища і породжує нове середовище, готове до занурення в неї нових агентів (якщо для них є місце в цьому середовищі).
5. Середовище, що розглядається як агент, може бути занурене в середовище верхнього рівня. Університет можна розглядати як агент, що занурений в середовище держави.
6. Агенти і середовища можуть моделювати інші агенти і середовища на різних рівнях абстракції.

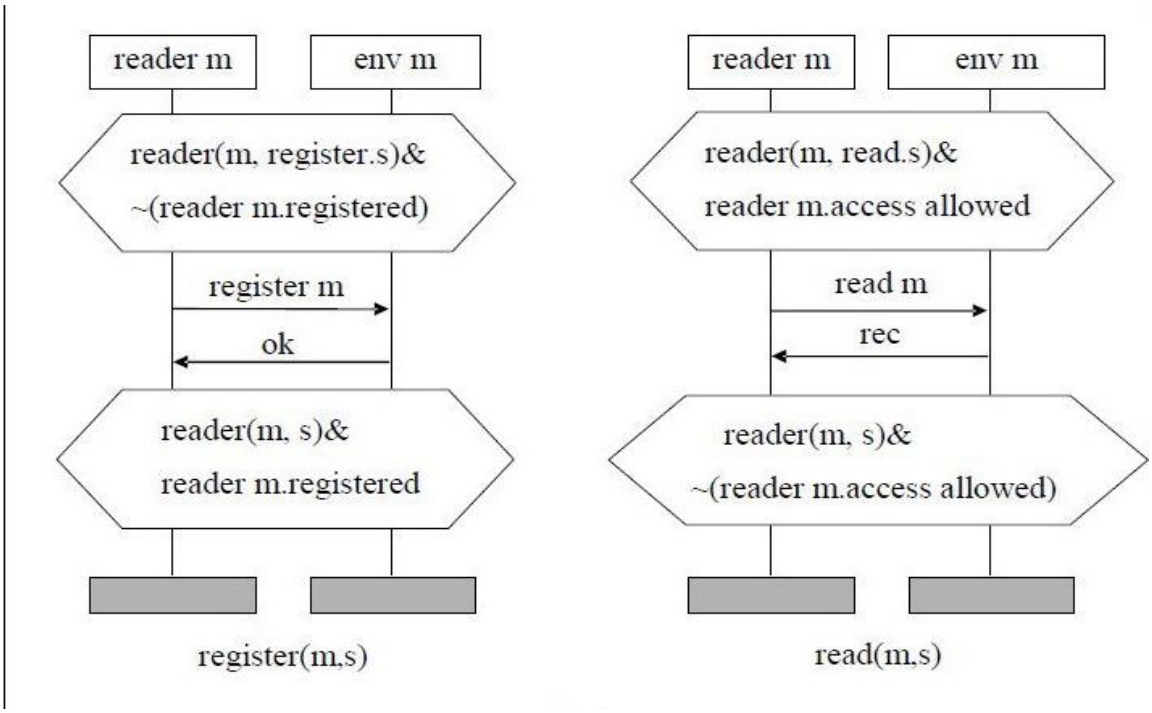
2.2. Мова базових протоколів

Базові протоколи застосовуються для представлення інсерційних моделей розподілених взаємодіючих систем. Будемо припускати, що в якості базової мови використовується мова багатосортного обчислення предикатів першого порядку. Специфікуєма система являє собою середовище з зануреними в неї агентами. Тому в атрибутну розмітку станів середовища входить не тільки інформація про стан середовища, але також і інформація про стани агентів. Передбачається, що агенти (принаймні, після занурення у середовище) мають унікальні імена і є типізованими. Відношення слідування пропозицій базової мови, що позначається знаком «|= », може бути задано шляхом інтерпретації

формул базової мови на деякій баготосортності алгебраїчної системи або аксіоматично через відношення виводимості в деякій несуперечливої системі аксіом. Базова мова використовується для опису властивостей станів специфікуємої системи (точніше їх розміток). При цьому передбачається, що всі істинні речення базової мови істини на всіх станах системи (вимога цілісності або безпеки). Звідси випливає, що, якщо на стані *s* виконується умова *a* (*s* |= *a*) і |= *a* → *b*, то також *s* |= *b*.

Кожен базовий протокол являє собою вираз виду $\forall x$ (α→< u > β), де x - список (типізованих) параметрів, α і β - формули базової мови, u - процес протоколу (цілком визначена поведінка). Формула α називається передумовою, а формула β - постумовою базового протоколу.

Рис. Приклад графічної інтерпретації базового протоколу



Сам базовий протокол може розглядатися як формула темпоральной логіки, що виражає той факт, що, якщо (для відповідних значень параметрів) стан системи має розмітку, що задовольняє умові α, то, процес u може бути ініційований і, після його завершення, розмітка буде задовольняти умові β. Система, яка задовольняє цій умові, називається реалізацією системи базових протоколів.

Для кожної системи P базових протоколів в якості її реалізації буде визначена система S (P), породжена цими протоколами. Система S (P) є атрибутного системою, що використовує формули базової мови в якості розмітки станів. Поведінка системи S (P) визначається за допомогою спеціальної композиції поводжень. Ця композиція називається частково послідовною композицією і застосовується рекурсивно до процесів базових протоколів. Для визначення частково послідовної композиції застосовується відношення перестановки на безлічі дій системи S (P). У разі перестановки всіх дій двох базових протоколів, їх композиція вироджується в паралельну, а в разі, коли ніякі дії не перестановочні, - в послідовну композицію процесів.

Для обчислення функції розмітки станів в кінці застосування протоколу використовується предикатний трансформер - перетворення, визначене на формулах базової мови. Умова, що характеризує розмітку стану до застосування протоколу, і його постумовою предикатний трансформер перетворює в нову розмітку (трансформоване постумовою).

Для визначення базових протоколів використовуються дві мови. Перша - це базова логічна мова з деякими модифікаціями для перед- і постумовою. Друга мова - це мова процесів базових протоколів. Загальною частиною двох мов є мова дій.

2.3. Алгебра поведінки

2.3.1. Перехідні системи

Сама алгебра поведінки влаштована досить просто. Вона є двухосновну алгебру $<U, A>$, першою компонентою якої є безліч U поводжень, а другою - безліч дій A. Сигнатура алгебри поведінки складається з двох операцій, одних відносин і трьох констант. Перша операція $a.u$ називається префіксинг. Її аргументами є дія a і поведінк u.

Результат є нова поведінка. Друга операція є операція недетермінірованного вибору $u + v$. Це бінарна операція, визначена на множині поведінок. Вона коммутативна, асоціативна і ідемпотентна. Константи алгебри поведінок - успішне завершення Δ, невизначена поведінка $\perp$ і тупикова поведінка 0, яке є нулем (нейтральним елементом) недетермінірованного вибору. На безлічі поведінок визначено бінарне відношення апроксимації $\subseteq$, яке є відношенням часткового порядку з найменшим елементом $\perp$. Операції префіксінга і недетермінірованного вибору монотонні і безперервні відносно цього віднощення.

Для опису динаміки систем використовуються перехідні системи. Існує кілька видів перехідних систем, які отримуються шляхом збагачення звичайної перехідної системи додатковими структурами. Звичайна система переходу визначається як кортеж

$$<S, T>, T \subseteq S^2$$

де S - сукупність станів, а T - перехідне відношення, позначене також як $s \to s'$. Якщо немає ніяких додаткових структур, можливо, єдиною корисною конструкцією є транзитивне закриття перехідного відношення, позначене як $s \xrightarrow{\alpha} s'$ і виражаючи досяжність в просторі стану S.

Маркована система переходу визначається як потрійна

$$<S, A, T>, T \subseteq S \times A \times S$$

де S - це знову набір станів, A - це сукупність дій (альтернативна термінологія: мітки або події), T - сукупність мічених переходів. Приналежність до перехідного відношення позначається $s \xrightarrow{\alpha} s'$. Це головне поняття в теорії взаємодії. Розглянути зовнішню поведінку системи та її внутрішнє функціонування можна використовуючи поняття мічених переходів. Два стани вважаються еквівалентними, якщо ми не можемо відрізнити їх, спостерігаючи тільки зовнішню поведінку, тобто дії, вироблені системою під час її функціонування. Ця еквівалентність фіксується поняттям подібності.

Змішана версія

$$<S, A, T>, T \subseteq S \times A \times S \cup S^2$$

об'єднує немічених переходи $s \to s'$ з міченими переходами $s \xrightarrow{\alpha} s'$. В цьому випадку ми говоримо про неспостережувані або приховані і

спостережувані переходи. Однак, як це буде продемонстровано пізніше змішана версія може бути технічно зведена до маркованих системам, іноді легше визначити змішану система, а потім зменшити її до маркованої.

Приписувана система переходів:

$$< S , A ,U ,T ,\phi > , \phi : S \rightarrow U$$

Цей вид перехідної системи використовується, коли слід маркувати не тільки переходи, але й стани. Функція φ називається функцією мітки стану. Зазвичай набір міток стану структурується як $U = D^R$, де множина R називається набором атрибутів, а множина D - набором значень атрибутів.

Отримані скориговані системи переходів розрізняють три види підмножин

$$S_0, S_\Delta, S_\perp \subseteq S$$

у безлічі S системних станів. Це початкові стани, стани успішного завершення та невизначені (розбіжні) стани відповідно. Значення цих коригувань полягає в наступному: з початкових станів система може запускатися, в станах успішного завершення система може завершити роботу, невизначені стани використовуються для визначення відношення апроксимації на множині станів, у невизначених станах поведінка системи може бути вдосконалена (розширена). Стани успішного завершення необхідно відрізняти від мертвих станів блокування, тобто станів, з яких немає переходів, але вони не є ні станами успішного завершення, ні невизначеними станами. Властивість не мати переходів позначається як $s \not\rightarrow$.

Існує два види недетермінізму, притаманного перехідним системам. Перший - це наявність двох переходів $s \xrightarrow{\alpha} s'$ і $s \xrightarrow{\alpha} s''$ для деякого стану s при $s' \neq s''$.

Цей недетермінізм означає, що після виконання дії система може обрати наступний стан недетермінованим. Другий вид недетермінізму - це можливість різного коригування одного і того ж стану. Тобто стан може бути одночасно станом успішного зупинення, а також невизначеним або початковим.

Позначена система переходу (без прихованих переходів) називається детермінованою, якщо для довільних переходів із s $\stackrel{\alpha}{\rightarrow}$ s' ∧ s випливає, що $s' = s''$ і $S_\Delta \cap S_\perp = \emptyset$.

2.3.2. Послідовні та паралельні композиції

Існує багато композицій, що охоплюють базову алгебру процесів або алгебру поведінки. Більшість з них визначаються незалежно від представлення агента як перехідної системи. Ці операції зберігають схожість і можуть розглядатися як операції над поведінкою. Ще одна корисна властивість цих операцій - безперервність. Найбільш популярними операціями є послідовні та паралельні композиції.

Послідовна композиція визначається за допомогою наступних правил виводу і рівнянь.

$$\frac{u \stackrel{a}{\rightarrow} u'}{(u; v) \stackrel{a}{\rightarrow} (u'; v)}$$

$$((u+\Delta); v) = (u; v) + v$$

$$((u + \perp); v) = (u; v) + \perp$$

$$(u; 0) = 0$$

Ці визначення слід розуміти наступним чином. Спочатку ми розширюємо сигнатуру алгебри поведінки, додаючи нову бінарну операцію ((); ()).Потім додайте тотожності для цієї операції і переконайтеся, що нові рівняння в вихідній сигнатурі не з’являються (консервативність розширення). Тоді визначається перехідне відношення на множині класів еквівалентності розширених виразів поведінки (повинна бути показана незалежність щодо вибору представника). Ці класи тепер стають станами перехідної системи, а значення вираження визначаються як його поведінка.

Паралельна композиція поведінки передбачає, що визначена комбінація дій. Вона розглядається як асоціативно-комутативна

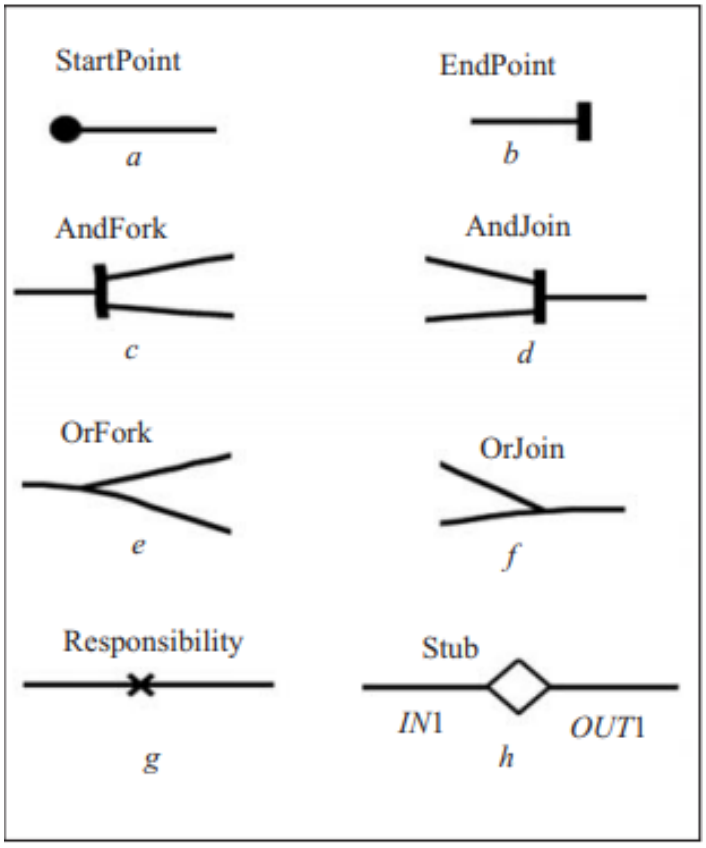
оперпція a × b з аннулятором ∅. Правила і тотожності для паралельної композиції:

$$\frac{u \xrightarrow{a} u',\ v \xrightarrow{b} v',\ a \times b \neq \emptyset}{u \| v \xrightarrow{a \times b} u' \| v'},$$

$$\frac{u \xrightarrow{a} u',\ v \xrightarrow{b} v'}{u \| v \xrightarrow{a} u' \| v,\ u \| v \xrightarrow{b} u \| v',\ u \| (v+\Delta) \xrightarrow{a} u',\ (u+\Delta) \| v \xrightarrow{b} v'}$$

$$(u+\Delta) \| (v+\Delta) = (u+\Delta) \| (v+\Delta) + \Delta$$
$$(u + \bot) \| v = (u + \bot) \| v + \bot$$
$$u \| (v + \bot) = u \| (v + \bot) + \bot$$

Рис.



Кожен шлях має початкову вершину, причому їх може бути кілька. Кінцеві максимальні шляху закінчуються кінцевими вершинами, представленими жирними короткими лініями, ортогональними шляхах, які вони завершують. З початкової вершини виходить, а в кінцеву входить тільки один шлях. Шляхи можуть розгалужуватися і зливатися (fork and join). Існують два види розгалуження і злиття: паралельні і альтернативні. Паралельні розгалуження (AndFork) і злиття (AndJoin)

позначені жирними лініями, ортогональними шляхах. У парлельного розгалуження існує один вхідний шлях і кілька вихідних. Паралельне злиття може мати кілька вхідних шляхів, але тільки один вихідний. У альтернативних розгалуження (OrFork) і злиття (OrJoin) така ж структура, як і у паралельних (розгалуження має один вхідний шлях і кілька вихідних, злиття - кілька вхідних і один вихідний шлях). Альтернативні розгалуження і злиття представлені точками на шляхах.

Крім розгалужень і злиттів на шляхах є два види символів: зобов'язання, позначене хрестиком (Responsibility), і стаб, позначений ромбом (Stub). Символи зобов'язань мають по одному вхідному і вихідному шляху, символ стаб може мати по кілька вхідних і вихідних шляхів. З кожним зобов'язанням асоційований деякий локальний опис системи локальних описів, які, в осносном, відносяться до базової системи, хоча в деяких випадках - і до управління.

РОЗДІЛ 3. РОЗРОБКА ЕКОНОМІКО-МАТЕМАТИЧНОЇ МОДЕЛІ УНІЕРСИТЕТУ

3.1. Формальна специфікація економічної моделі університету засобами інсерційного моделювання

Виходячи із умови моделювання, ми можемо виділити 5 типів агентів котрі можуть взаємодіяти між собою в процесі симуляції моделі.

В моделі можливо виокремити 5 типів агентів:

1. Уряд
2. Університет
3. Викладачі
4. Студенти:
   4.1. Студенти бюджетної форми навчання
   4.2. Студенти контрактної форми навчання

Агентів в сердовищі інсерційного моделювання можна представити наступним чином, розглянемо приклад формалізації агента “Університет” засобами інсерційного моделювання:

University: obj (
Score:(ScoreType)→real
…..
)

Оскільки університет має два тип рахунків: спеціальний та генеральний, ми можемо цей факт відобразити(формалізувати) засобами інсерційного моделювання наступним чином, створивши спеціальний перелічувальний тип ScoreType:

ScoreType:(
Total,
Special
)

Агентів та взаємозв'язок між ними у вигляді процесів, представлено на діаграмі взаємодії агентів.

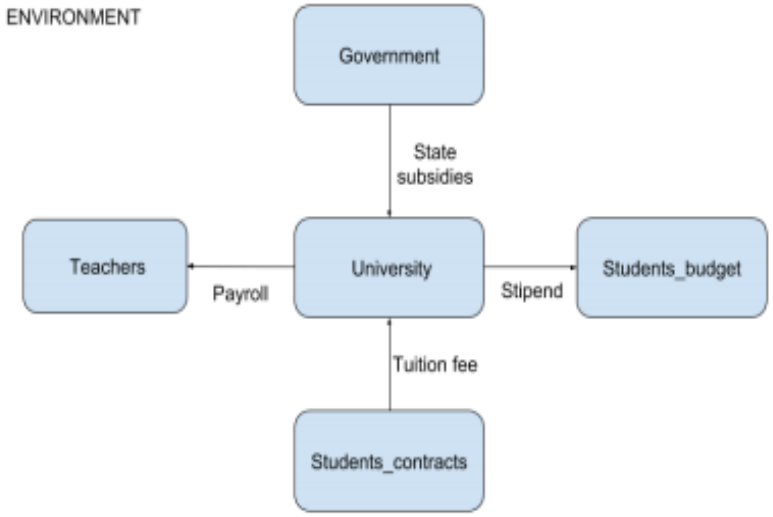


Рис. 1. Діаграма взаємодії агентів в економічній моделі університету

Як зазначено на рис. 1, ми можемо виділити 6 основних дій;

1. Держава фінансує університет.
2. Університет виплачує заробітні плати викладачам та співробітникам.
3. Університет виплачує стипендію студентам.
4. Студенти контрактної форми навчання платять за навчання.
5. Інші витрати університета: комунальні послуги, незаплановані витрати.
6. Виплати відпускних.

Надалі розглянемо формалізації основних дій засобами інсерційного моделбвання. В Таблиця №1 наведенно приклади формалізації, але спочатку представимо графічну інтерпретацію дії за допомогою мови базових протоколів Рис.2

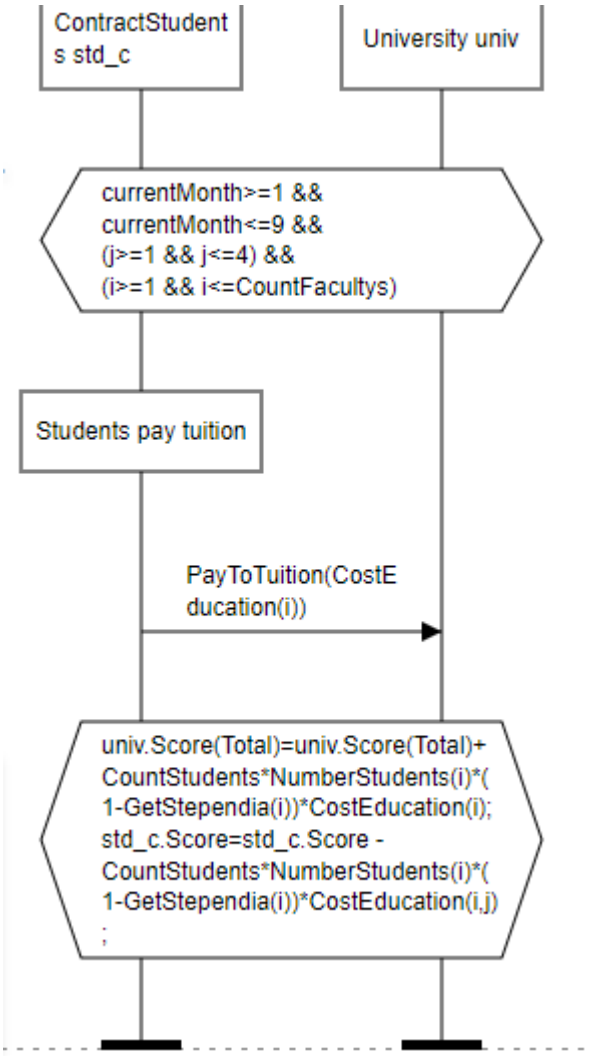


Рис. 2. Формалізація дії “оплата за навчання”.

В даній дії беруть участь два типи агентів: студенти контрактної форми навчання та університет - цей факт відображений на інстанціях даної діаграми. Оскільки плата за навчання на різних кафедрах на різних курсах різний нам потрібно цей факт відобразити в моделі. Створимо функціональний тип:

CostEducation:(int,int)→real,

...

CostEducation(1,2)→1350

CostEducation(2,3)→1350

CostEducation(3,4)→1450

де

1. Перший параметр - номер кафедри.
2. Другий параметр - курс

CostEducation(1,2)→1350 - дана форма запису формально означає, що на кафедрі початкової освіти на 2 курсі, вартість навчання вартує 1350 грн. Надалі розглянемо формалізацію дій у алгебраїчному вигляді:

Таблиця 1 "Приклади формалізації"

Формалізація дії засобами мови базових протоколів	**Опис дії**
bp1= Operator((currentMonth>9 && currentMonth<=12)→ (<"university pay vacation for teachers">) (**teachers.Score**=teachers.Score+VacationPay; **univ.Score(Total)**=univ.Score(Total)-0.5*VacationPay/t; **univ.Score(Special)**=univ.Score(Special)-0.5*VacationPay/t))	Виплата відпускних коштів, для викладачів.
bp2= Operator(**Forall(i:int)** (1 && i>=1 && i<=CountFacultys)→ (<"University pays scholarships">) (**std_s.Score**=std_s.Score + CountStudents*NumberStudents(i)*GetSte	Виплата стипендій студентам контрактної форми навчання

pendia(i)*ScholarshipSize(i); **univ.Score(Special)**=univ.Score(Special) - (CountStudents*NumberStudents(i)*GetStependia(i)*ScholarshipSize(i))*0.5; **univ.Score(Total)**=univ.Score(Total) - (CountStudents*NumberStudents(i)*GetStependia(i)*ScholarshipSize(i))*0.5)))	
bp3= Operator(**Forall(T:TypeTeachers)** (currentMonth>=1 && currentMonth<=9)→ (<”Teacher salary”>) **(teachers.Score**=teachers.Score+CostSalary(t); **univ.Score(Total)**=univ.Score(Total)-0.5*CostSalary(t); **univ.Score(Special)**=univ.Score(Special) - 0.5*CostSalary(t))))	Виплата заробітної плати викладичам та іншим співробітникам
bp4= Operator((1)→(<”Payment of utility services”>) (**univ.Score(Total)**=univ.Score(Total)- univ.Score(Total)*CoefUnplannedExpenses))	Незаплановані витрати

bp5= Operator((1)→(<”Payment of utility services”>) **(univ.Score(Total)**=univ.Score(Total)-univ.Score(Total)*CoefUtilityServices))	Оплата комунальних послуг
bp6= Operator((currentMonth == 1)→ (<”Investment to university ”>) **(gov.Score** = gov.Score - gov.Score*InvestmentRatio; **univ.Score(Total)**=univ.Score(Total)+gov.Score*InvestmentRatio))	Інвестування університета урядом країни
bp7= Operator((currentMonth == 1)→ (<”Distribution between accounts”>) **(univ.Score(Total)**=univ.Score(Total)-univ.Score(Total)*DistributionRatio; **univ.Score(Special)**=univ.Score(Special)+univ.Score(Total)*DistributionRatio))	Внутрішній розподіл коштів між рахунками уніерситету
bp8= Operator(**Forall(T:TypeTeachers)** (currentMonth == 8)→ (<”Calculation vacation”>) **(VacationPay=**VacationPay+CostSalary(T)+VacationPay+CostSalary(T)*0.3))	Розрахунок велечини відпускних коштів.
bp9= Operator(	Інкрементаці

(currentMonth<lastMonth→ (<”Next month”>) (**currentMonth** = currentMonth + 1))	я місяця
bp10= Operator((currentMonth==lastMonth→ (<”Lasr month”>) (**1**))	Останній місяць в моделі

На найвищому рівні модель можна інтерпретувати наступною UCM діаграмою, Рис.3. На UCM діаграмі видно, що підпроцеси: tutionFee, slaryForTeachers, stipend - це паралельні підпроцеси.

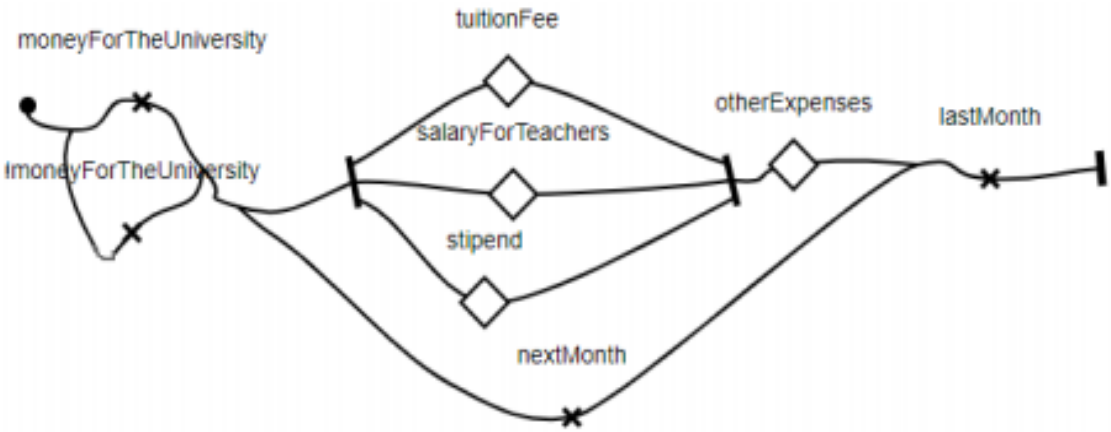


Рис.3 UCM діаграма

Надалі розглянемо алгебраїчну інтерпритацію поведінки моделі та їх формальний опис:

So =moneyForTheUniversity . EB1 +
!moneyForTheUniversity.EB1,
EB1 =(
{tutionFee || slaryForTeachers || stipend};
otherExpenses;
EB2

),

EB2 = nextMonth . So + lastMonth . Delta

1. TutionFee - процес оплати за навчання; студенти, що навчаються за контрактом, оплачують університету вартість навчання.

2. SlaryForTeachers - оплата праці викладачам.

3. Stipend- виплата стипендії.

4. OtherExpenses - процес, що представляє собою незаплановані витрати університету (бізнес поїздки, організація конференцій тощо).

3.2. MSC діаграма, як інструмент представлення та аналізу бізнес процесів університету

Наступним кроком після розробки специфікації формальної економічної моделі університету є її кодування та її відладка.

На рис представлена MSC діаграма даної моделі. Даний тип діаграми широко застосовується для представлення вимог, специфікації систем з метою їх аналізу, а також для операцій з вимогами. Поштовхом до розвитку такого типу діаграм став розвиток та популяризація так графічних мов як, UML, SDL, MSC котрі стандартизовані Міжнародним союзом електрозв'язку ITU.

На відміну від мов програмування данні мови не призначені для кодування системи, а використовуються для відображення архітектури розробляємої системи, ці мови більш високого рівня абстракцій, ніж мови програмування.

Середовище MSC. Стандартна семантика діаграм MSC визначається як алгебра процесу семантика [7-8], MSC відображає є сукупність слідів процесу над безліччю використовуваних подій в MSC.

Взагалі кажучи даний тип діаграми досить інформативно ілюструє поведінку агентів в економічній системі та повідку моделі взагалі. Взагалі кажучи даний тип діаграми можно розглядати, як досить потужний інструменти формалізації та візуалізації бізнес процесу Надалі розглянемо більщ детально отриману MSC діаграму нашої моделі Рис. 1

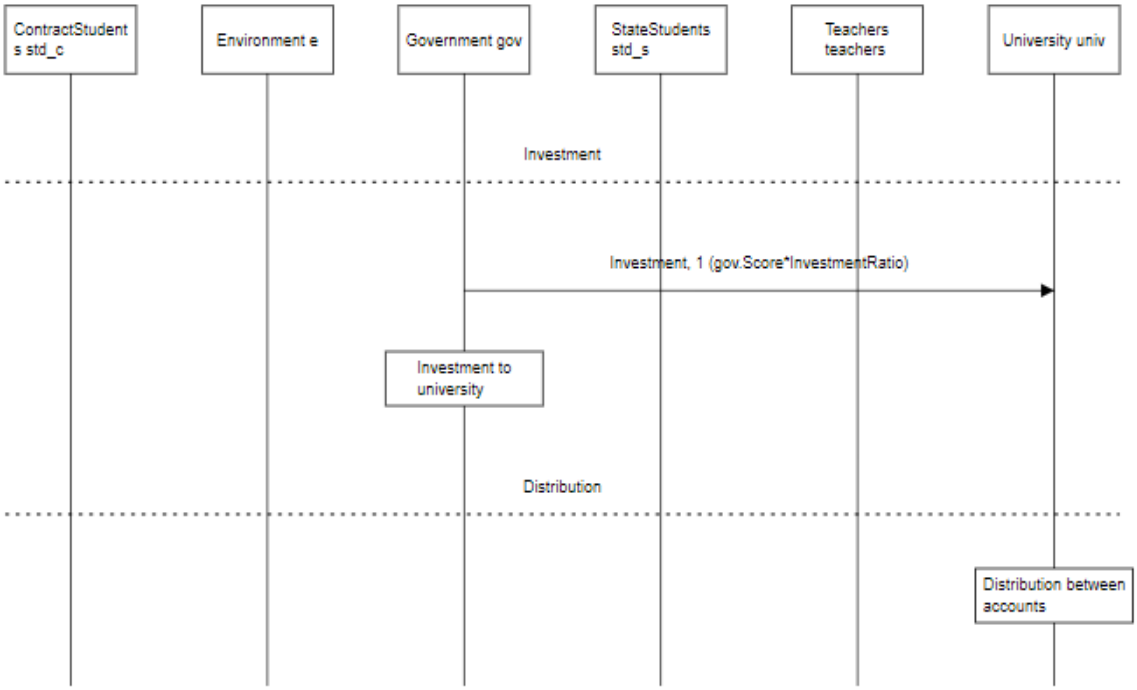


Рис. 1. MCS діаграма

Спираючись на розроблену раніше специфікацію в даній моделі взаємодіють 5 типів агентв:

1. Держава (Government#gov)
2. Університет (University#univ)
3. Викладачі (Teachers#teachers)
4. Студенти:

4.1. Студенти бюджетної форми навчання (StateStudents#std_s)

4.2. Студенти контрактної форми навчання (ContractStudents#stc_c)

Цей факт відображений на MSC діаграмі у вигляді відповідних інстанцій, що еквівалентно списку агентів у специфікації, що описани у попередньому розділі. Оскільки держава (Government#gov) інвестує університет (University#univ) на першому місяці симуляції даної моделі,

це відображено у вигляді процесу Investment(InvestmentAmount), котрий приймає параметр - InvestmentAmount, що є величиною інвестування університету. Цей факт можливо відобразити наступним чином:

proc(B)=(
(Government#gov:instance).(
(Government#gov: Investment
(InvestmentAmount) to University#univ))
(University#univ:instance).(
(University#univ: Investment(InvestmentAmount)
from Government#gov))
)

Кожний місяць, університет формує надходження та видатки, до видатків можна віднести такі витрати: оплата заробітної плати, виплата стипендій студентам, оплату комунальних полу, а також інші незаплановані витрати. До надходжень можно віднести: інвестування університету державою, а також надходження від інших структурних підрозділів університет, оплата за навчання студентів контрактної форми навчання

В ході симуляції моделі кожний тип операції та список агентів котрі взаємодіють в даній операції відображається на MSC діаграмі.

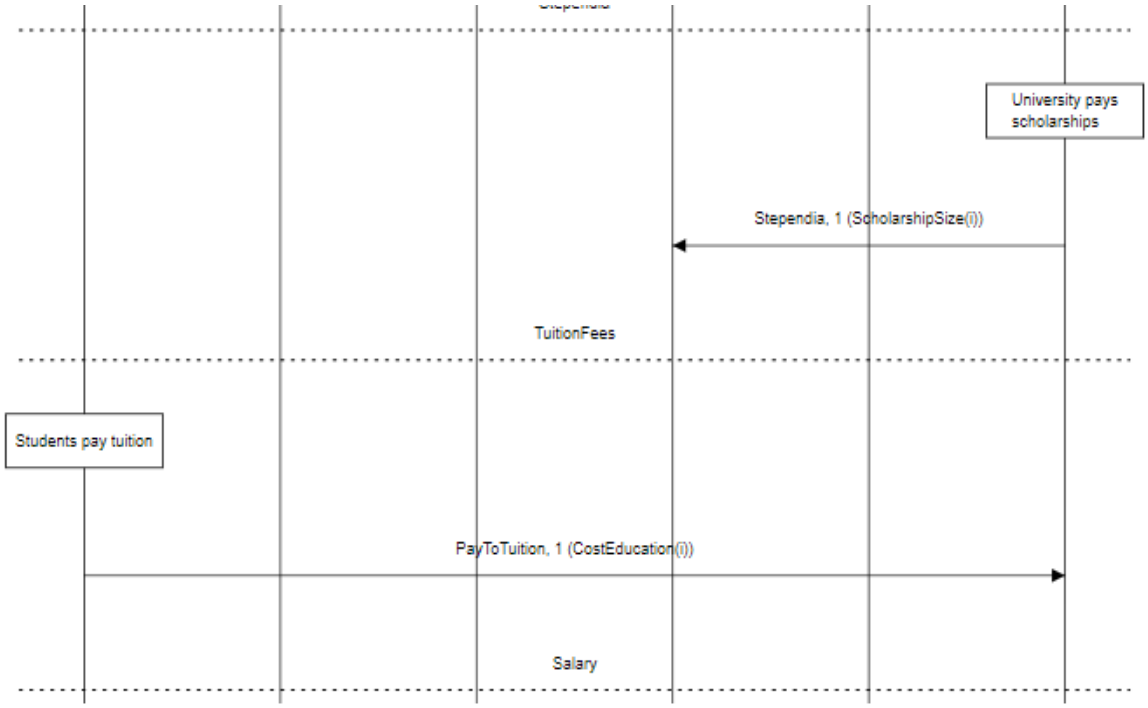


Рис. 2. MSC діаграма

На Рис. 2. відображено процес два процеси видатки у вигляді оплати стипендій студентам (Stependia(ScholarshipSize(i))), де ScholarshipSize(i)-функціональний тип, котрий повертає значення величини стипендій відповідно до кафедри, надходження у вигляді оплати за навчання(PayToTuittion(CostEducation(i))), де CostEducation(i)-функціональний тип, котрий повертає значення величини оплати за навчання відповідно до кафедри.

proc(B)=(

(University#univ: instance).(

(University#univ: Stependia(ScholarshipSize(i) to StateStudents#std_s)

(StateStudents#std_s: instance).(

(StateStudents#std_s :Stependia(ScholarshipSize(i) from University#univ))

(ContractStudents#stc_c:instance).(

(ContractStudents#stc_c :PayToTuittion(CostEducation(i) to University#univ))

…

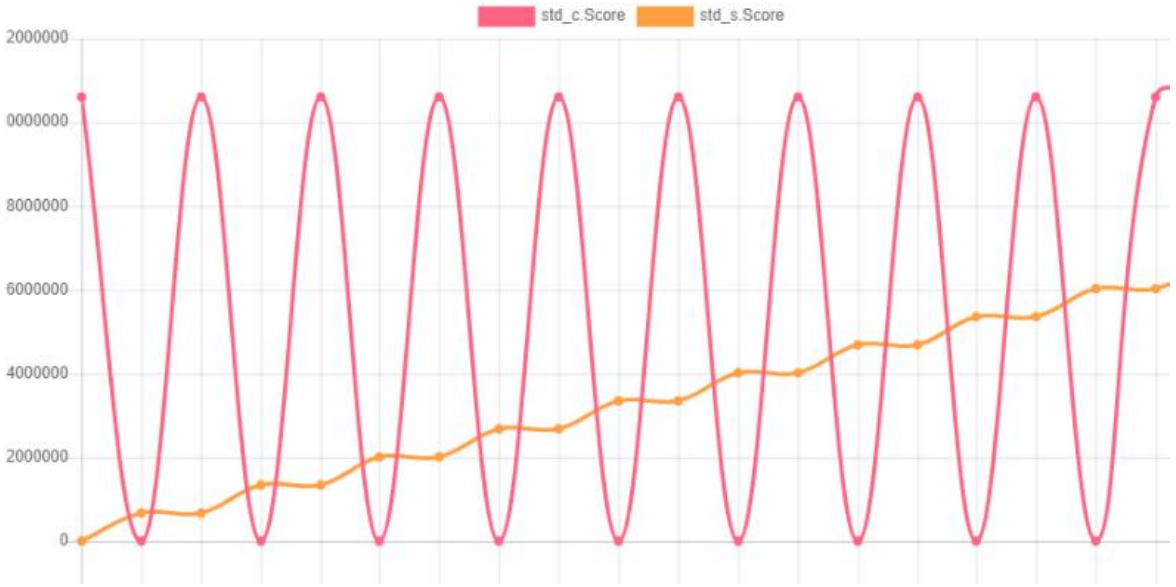
)

Як зазначалось раніше, MSC відображає є сукупність слідів процесу над безліччю використовуваних подій і відображає кожний процес та агентів котрі взаємодіють між собою за допомогою цього процесу, ми можемо використовувати даний тип діаграми для аналізу та формального представлення поведінки економічної моделі, а також використовувати як механізм формального представлення та аналізу бізнес процесів університету.

3.3. Візуалізація та представлення конкретної модель. Результати моделювання

Надалі розглянемо результати симуляції та графіки, котрі ми побудували в системі інсерційного моделювання. На Рис. 1 відображена траєкторія оплати студентів контрактників (червона), та траєкторія стипендій (жовта).

Оскільки під студентами контрактної форми навчання ми розуміємо всіх студентів цієї форми навчання в університеті , то в величину 10605456 грн. закладено оплату всіх студентів контрактної форми навчання. Студенти сплачують за навчанння кожний місяц (вличину 10605456 грн.) це відображено на Рис.#. Траєкторія оплати студентів контрактної форми за навчання. Траєкторія має вигляд синусоїдальної форм.



Оскільки в величину котру інвестує держава в університет закладено тільки видатки на студентів бюджетної форми навчання, ця величина розрахована тільки на нормальне функціонування університету для студентів денної бюджетної форми навчання. При ситуації коли в університету навчаються студенти тільки бюджетної форми навчання цієї величини повинно бути достатньо для функціонування університету протягом року.

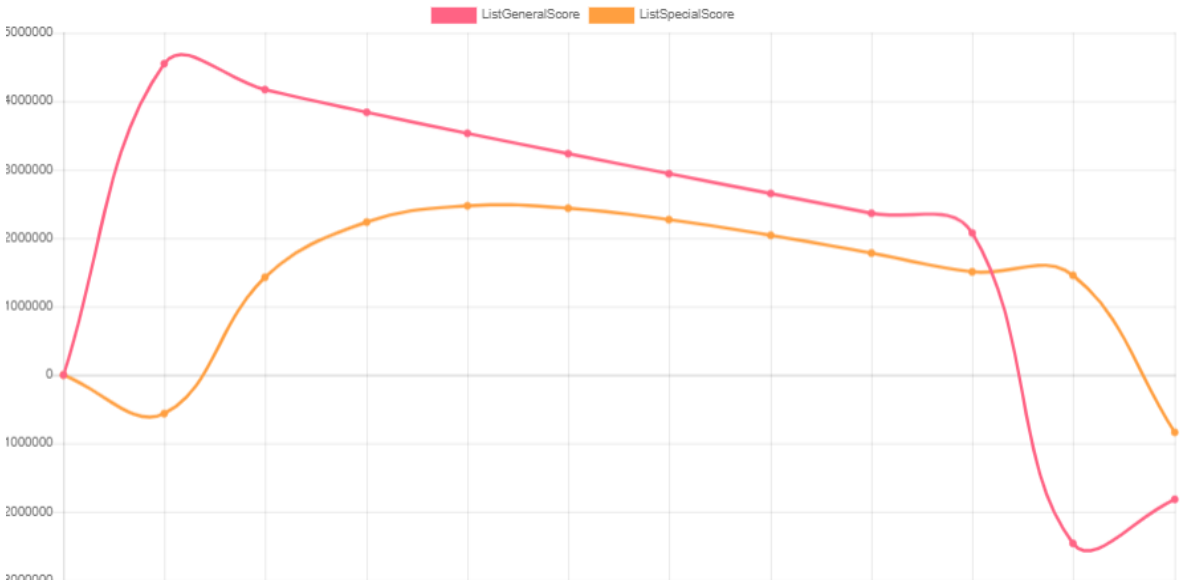
У величина коштів котру сплачують студенти контрактної форми навчання повинно бути закладено оплату заробітної плати викладачам, витрати на комунальні послуги та інше, відповідно до кількості студентів контрактної форми навчання. Тобто величини котру сплачують студенти контрактної форми навчання повинно бути достатньо на нормальне функціонування університету відповідно до кількості студентів контрактної форми навчання.

Щоб перевірити ситуацію чи використовує університет кошти котрі спрямовані на оплату видатків студентів бюджетної форми навчання для оплати видатків студентів контрактної форми навчання, потрібно розглянути решту бюджету університут на останній місяць.

1. Якщо університет не використовує кошти бюджетної форми навчання для оплати видатків студентів контрактної форми

навчання залишок бюджету на останній місяць буде додатній.

2. Якщо університет використовує кошти бюджетної форми навчання для оплати видатків студентів контрактної форми навчання залишок бюджету на останній місяць буде від'ємний.



Розглянувши графік, котрий відображений на Рис.# Траєкторія бюджету університету червона (генеральний рахунок), жовта (спеціальний рахунок), ми можемо бачити, що вже на 10 місяці бюджет університету стає від'ємний, це означає, що оплати за навчання студентами контрактної форми навчання не вистачає.

Використовуючи даний підхід ми можемо досліджувати поведінку фінансової моделі, підбирати такі значення ендогенних змінних котрі ми досліджуємо, щоб модель була адекватна, використовувати такий підхід як інструмент прогнозування.

3.4. Символьна модель економіки університету

На відміну від раніше описаної конкретної моделі, символьне

моделювання дає нам можливості розробляти та перевірити нашу модель на такі важливі властивості як надійність та стабільність.

На відміну від конкретних моделей символьні моделі не дають можливості створювати діаграми, а дають можливість відображати функціональні залежності у вигляді формул, що може характеризувати величини нашої моделі та досліджувати властивості надійності та стабільності. Розробка символьної моделі для економіки університету дасть змогу проаналізувати поведінку системи задаючи параметри цієї моделі у вигляді певних нерівностей, та проаналізувати, при яких значеннях дана система виде себе адекватно. При розробці моделі точне значення деяких екзогенних змінних може бути невідомим, ми можемо представи її у вигляді системи нерівностей, після чого перевірити адекватність та стабільність моделі.

Таким чином, для цього проекту було обрані наступні парметри:

1) середня ціна за навчання (AverageTuitionFee)
2) кількість студентіві (CountScudenst)
3) кількість студентів бюджетної форми навчання (CountStateStudents)
4) кількість студентів контрактної форми навчання (CountContractStudents)

Віжобразмио вище перелічені параметри моделі у вигляді системи нерівностей:

1. 2000<AverageTuitionFee<1500
2. CountScudenst>=5000
3. 5000>CountStateStudents>= 3000
4. 2000>=CountContractStudents>=1000

Після того як ми позначили невідомі параметри як змінні, ми можемо оцінити їх у певних межах. У процесі моделювання для цих прикладів ми отримуємо такі формули:

1. AverageTuitionFee = F1 (p1, p2,…),

2. CountScudenst = F2 (s1, s2, ...),

3. CountStateStudents = F3 (l1, l2, ...),

4. CountContractStudents = F4(f1,f2,...)

де p1, p2,…; s1, s2, ...; l1, l2, ...f1...fn - невідомі параметри даної моделі.

Доводячи, що ці параметри входять в описані інтервали, можна підтвердити або спростувати адекватність даної моделі. Використовуючи таких підхід до аналізу економічних моделей ми можемо аналізувати та досліджувати такі властивості, як стабільність, адекватність. Крім цього таки підхід може використовуватись як інструмент прогнозування, тобто ми можемо задавати невідомі нам параметри за допомогою системи нерівностей та аналізувати при яких сценаріях модель будет давати стабільну та адекватну поведінку.

ВИСНОВКИ

Під моделюванням розуміється процес побудови, вивчення і застосування моделей. Поточна роль імітаційного моделювання - допомога при плануванні удосконалень, оптимізація, порівняння альтернатив, проектування нового, тобто стратегічний і тактичний рівні. Тут імітаційне моделювання дійсно може допомогти значно підвищити ефективність, заощадити суттєві кошти або навіть попередити катастрофічні наслідки помилкових рішень. Вибір рівня абстракції і методу моделювання - центральне питання в побудові моделі.

Традиційні підходи імітаційного моделювання розглядають об'єкти моделювання як щось середнє арифметичне або як пасивні заявки або ресурси в процесі. Ці методи не враховують індивідуальних здібностей кожного їх модельованих об'єктів. У той же час саме в силу цих здібностей може змінюватися динаміка всієї системи в цілому. Агентне моделювання позбавлене цих недоліків, але розглядає об'єкти як активні, взаємодіючі між собою елементи, здатні виявляти індивідуальні властивості. З цієї причини можна вважати агентне моделювання найкращим традиційним підходом. При мультиагентному моделюванні є можливість впливати на об'єкти моделювання, а значить можна знайти такі керуючі впливи на агентів, які можуть привести до бажаної динаміці що складається з складний систем.

Вході написання дипломної роботи було проаналізовано досить велику кількість типової літератури з математико-економічного моделювання, імітаційного моделювання, формальних методів економіко-математичного моделювання.

В першій частині докладно описано можливості сучасних систем економічного моделювання та властивості котрі можуть бути дослідженні за допомогою цих програмних систем.

В другому розділ описана теорія інсерційного моделювання, як

засіб дослідження економіко-математичних моделей.

В третьому розділі представлено підхід в основі якого лежить інсерційне моделювання та алгебраїчне програмування, як засіб всебічного дослідження економіко - математичних моделей. Запрограмована модель котра відображає властивості моделі “Економіка університет”, запропоновано підхід до аналізу та представлення бізнес процесів університету на основі MCS - діаграм та інсерційного моделювання вцілому. Запропоновано підхід до дослідження таких властивостей, як стійкість та адекватність котрий базується на основі побудови символьної моделі.

Використання інсерційного моделювання достатньо ефективною може використовуватися для аналізу бізнес процесів різного рівня установ. Матеріали дипломної роботи апробовані на міжнародній конференції ICTERI - 2019