РОЗРОБКА КОМПАКТНИХ ПРОГРАМНИХ ЗАСОБІВ БЛОЧНО-ОРІЄНТОВАНОГО РОЗПОДІЛЕНОГО МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ

Анопрієнко О. Я., Єрмоленко І. О., Потапенко В. А.
Кафедра ЕОМ ДонДТУ
anoprien@cs.dgtu.donetsk.ua

Abstract

Anoprienko A. Ermolenko I., Potapenko V. Development of compact software for block-oriented distributed simulation of dynamic systems. *Simulation program, developed using Java programming language, is described. System allows effective simulation on distributed network environment. Some experiments and implementation features of a program are described.*

Вступ

Сучасна еволюція методів та засобів комп'ютерного моделювання характеризується двома найбільш вагомими тенденціями [1, 9]: **по-перше**, швидким розвитком графічних інтерфейсів з розвинутими можливостями багатовіконної візуалізації результатів та наочного візуального синтезу моделей, чому найбільше відповідає методика блочно-орієнтованого моделювання [14, 17, 18]; **по-друге**, поширенням паралельного та розподіленого моделювання з максимальним використанням можливостей технологій Інтернет [2, 7, 12, 13, 15, 16].

При цьому типовим є конфлікт між простотою вивчення і застосування конкретної системи моделювання та її функціональністю і універсальністю [19]. Найбільш вдало це протиріччя переборено у блочно-орієнтованій системі MATLAB/Simulink - одному з найсучасніших та найширше розповсюджених на сьогодні засобів інженерного та наукового моделювання [6]. Однак орієнтація на використання тільки цієї системи має в багатьох випадках суттєві недоліки:

- Simulink є комерційним продуктом фірми MathWorks, який потребує обов'язкового ліцензування при його використанні, що коштує за українськими мірками немало і для богатьох користувачів є сьогодня просто недосяжним;
- для ефективного використання система потребує значні комп'ютерні ресурси, і в типовому робочому варіанті, наприклад, остання версія Simulink (разом з MATLAB) займає понад 100 Мбайт дискової пам'яті, а в максимальному - більш як 700 Мбайт;
- в більшості конкретних випадків користувачу достатньо лише малої частини функціональних можливостей системи, при цьому всі інші (понад 80 %) ресурси опиняються зайвими;
- система є недостатньо гнучкою при необхідності використання її для мережевого розподіленого моделювання, або як одного з елементів комплексного спеціалізованого моделюючого середовища, наприклад, з використанням сучасних технологій Інтернет.

В зв'язку з цим актуальною є розробка більш компактної та гнучкої моделюючої системи, яка за своїми основними можливостями була б спроможна наблизитися до

кращих властивостей Simulink, але при цьому не потребувала б коштовного ліцензування і була повністю "прозорою" для користувача в разі необхідності її модифікації у будь якому напрямку. Розробка такої полнофункціональної працюючої системи в університетських умовах за відносно короткий час є дуже нетривіальною задачею і стала можливою лише в останні роки, дякуючи новітнім технологіям програмування. В першу чергу при цьому треба назвати мову Java [5], розроблену у середині 90-х років і орієнтовану на ефективне багатопроцесне об'єктно-орієнтоване програмування у мережевому середовищі.

На кафедрі ЕОМ ДонДТУ за останні десятиріччя було накопичено суттєвий досвід використання та розробки різноманітних засобів моделювання [3, 8, 10, 11, 12, 20], який дозволив зробити ще один значний крок у розробці нового покоління моделюючих середовищ. Стаття присвячена саме такій системі моделювання нового покоління, яка була розроблена на кафедрі ЕОМ у 1999 році з використанням об'єктно-орієнтованої технології та мережевих і інтерфейсних можливостей мови Java.

Вимоги до сучасних систем моделювання динамічних процесів

Сучасна система моделювання, для того, щоб бути конкурентно здатною, повинна володіти рядом обов'язкових властивостей:

Відкрита архітектура. Деякі програми моделювання щоб задовольняти широкому колу задач мають дуже великі бібліотеки компонентів. З одного боку, це дозволяє легко виконувати багато завдань, але, з іншого боку, користувачу буває важко розібратися з безліччю елементів, і він іноді просто губиться при спілкуванні з такою програмою. До того ж, завжди знайдуться задачі, які не можуть бути охоплені конкретною множиною елементів, і в такому випадку користувач вимушений відмовлятися від роботи з такою програмою. Виходячи з цього, можна зробити висновок, що однією з вимог до сучасної системи моделювання повинна бути можливість створення нових елементів на базі вже існуючих. Звичайно, процес створення нового елемента повинен бути максимально автоматизованим та інтуїтивно зрозумілим користувачу.

Можливість роботи у мережі. У сучасних умовах жодну велику технологічну проблему не можна вирішити без переробки значних обсягів інформації та без використання такого потужного, оперативного і зручного засобу об'єднання зусиль та ресурсів, як комп'ютерна мережа. Комп'ютерна мережа - це та глобальна ідея, що сьогодні об'єднує розрізнених володарів комп'ютерів, фахівців та вчених і дозволяє отримати вагомий синергетичний ефект від сумісної праці багатьох. З усього вищезгаданого можна зробити висновок, що сучасна система моделювання не можлива без мережевої підтримки.

Незалежність від архітектури комп'ютера та операційної системи. Програма для мережевого застосування, що має надію придбати популярність у користувачів, повинна однаково стабільно працювати на комп'ютерних системах різноманітних апаратних та програмних платформ. В особливості це стосується програм-симуляторів, які повинні дозволяти виконувати роботу в гетерогенній мережі, де останній час усе частіше разом використовуються операційні системи Unix та MS Windows.

Можливість обміну інформацією з іншими системами моделювання. Дуже часто дані, отримані після моделювання, необхідно обробляти або використовувати в інших програмах та навпаки. Так, наприклад, система навігації може використовувати дані про динаміку корабля, отримані з програми моделювання [3]. Тому однією з основних

вимог до системи моделювання є можливість інтенсивного обміну інформацією з іншими програмами та засобами.

Структура системи моделювання

Нова програма моделюваня була розроблена з урахуванням усіх названих вище вимог на мові програмування Java з використанням об'єктно-орієнтованої технології.

Програма має три основні складові частини:

- частина, що відповідає за моделювання;
- інтерфейс;
- моделі елементів.

Моделююча частина є основою, "двигуном" програми. Вона відповідає за перевірку правильності укладання схеми, сортування елементів схеми у порядку їх обчислення, перетворення блочного запису до математичної форми, а також за безпосередньо процес моделювання. Моделююча частина в свою чергу складається з чотирьох пакетів: Basic, Engine, Calculator, Functor.

У пакеті **Basic** містяться базові класи для всієї іншої програми, а також описуються такі базові поняття як елемент схеми (клас Node), спроможність класу моделювати себе (інтерфейс Simulable), спроможність виводити інформацію про змінні класу на екран (інтерфейс Printable). Елемент схеми будується на базі таких понять, як вхідна сторона елементу (інтерфейс InputSide) та вихідна сторона (інтерфейс OutputSide). Самі елементи можуть бути трьох видів:

- елементи-джерела (абстрактний клас Producer),
- елементи-приймальники (абстрактний клас Consumer)
- елементи-посередники (абстрактний клас Mediator).

Інерційні елементи, такі як інтегратори та блоки затримки, повинні реалізовувати інтерфейс Heavy. Між елементами встановлюватися зв'язки (Connection), а групи елементів можуть бути об'єднані в схему (абстрактний клас ChipMediator).

Така структура була обрана з умови забезпечення можливості модернізації системи. На базі даного набору класів, використовуючи стандартну схему створення елементів, можна легко доповнити існуючу бібліотеку новими, необхідними користувачу блоками.

Пакет **Calculator** поширює можливості програми і дозволяє записувати рівняння не тільки у блочно-орієнтованому вигляді, але й в більш звичному - математичному. Даний пакет також складається з множини класів. Так, інтерфейс Calculable є базовим і задає здібність об'єкту до обчислення. Цей інтерфейс реалізують абстрактні класи UnaryOperation та BinaryOperation, що є у свою чергу батьками для унарних та бінарних операцій, відповідно. З унарних операцій реалізовані унарний плюс (UPlus) та унарний мінус (UMinus). До бінарних операцій відносяться додавання (Plus), віднімання (Minus), множення (Multiply), ділення (Divide). У класі Tokenizer реалізований синтаксичний аналізатор рядків. Обчислення виразів виконується у класі Expression. Якщо у рядку, що аналізується, буде виявлена помилка, то виникне виключення CalculatorException.

Пакет **Functor** є поширенням пакету Calculator і дозволяє використати в математичних виразах знаки математичних функцій (такі, як синус та косинус). Клас ExtendedExpression є нащадком класу Expression з пакету калькулятор. Він

перевизначає деякі методи таким чином, щоб правильно інтерпретувалися назви математичних операцій. На даний момент реалізовані дві операції: синус та косинус, але кількість операцій може бути легко збільшена. Якщо у введеному рядку є помилка, або задана назва неіснуючої функції, то виникає виключення FunctionException.

Пакет **Engine** відповідає за моделювання схеми. Клас Simulator виконує перевірку правильності побудови схеми, а саме наявність та коректність розстановки зв'язків між елементами. Клас TopoSort виконує топологічне сортування графу, та встановлює послідовність, у якій буде вироблятися моделювання елементів, а також слідкує за наявністю алгебраїчних циклів у схемі. При наявності алгебраїчних циклів виникає виключення CyclicDependencyException. Клас SimSpace виробляє установку часу та кроку моделювання, а також запускає безпосередньо процес моделювання.

Усі реалізовані **моделі елементів** можуть бути поділені на чотири основні групи [9]:

- генератори сигналів;
- інерційні блоки;
- безінерційні блоки;
- приймальники сигналів.

Генератори сигналів - це блоки без входу, що видають на виході залежний від часу сигнал, а також блоки, що читають вхідну інформацію з файлу або одержують інформацію з зовнішніх джерел, наприклад, з другого комп'ютера. *Інерційні блоки* - це блоки, вихідний сигнал яких в заданий момент часу не залежить від вхідного сигналу в тій же часовій точці. Типові інерційні блоки - інтегратори та блоки з затримкою у часі. *Безінерційні блоки* - це блоки, вихідний сигнал яких в заданий момент часу залежить від вхідного сигналу (суматори, блоки перемноження та ін.). Відмінність між безінерційними та інерційними блоками суттєва для алгоритмів із замкнутими контурами. *Приймальники сигналів* - це блоки без виходу, що виконують деякі дії з інформацією на вході, наприклад, відображають її на екрані, зберігають в файл, або передають на другий мережевий комп'ютер.

Моделі елементів будуються на базі класів, розташованих у пакеті Basic. При цьому елементи-джерела повинні бути нащадками класу Producer, елементи-посередники – класу Mediator, елементи-приймальники – класу Consumer. Крім того, інерційні елементи повинні реалізовувати інтерфейс Heavy. Основна частина елементів, впроваджених в даній програмі, відповідає набору елементів моделюючої програми ISRSIM, розробленої наприкінці 80-х років в інституті системної динаміки та регулюючої техніки (ISR) Штутгартського університету [9]. Даний набір елементів був обраний таким чином, щоб при необхідності на їх базі можна було побудувати усі інші елементи. Крім того, були додані деякі специфічні блоки, що збільшують можливості програми: блоки-приймальники та елементи, які здійснюють обмін даними у мережі.

Інтерфейс системи

Від того, наскільки зручним є інтерфейс програми, також залежить, чи буде використуватися програма взагалі. У даному проекті була зроблена спроба реалізувати зрозумілий, дружній користувачеві інтерфейс. При розробці інтерфейсу враховувалося те, що користувач буде самостійно поширювати можливості програми, тобто розробляти і додавати нові елементи, компонувати їх у бібліотеки та інше.

Тому інтерфейс був розроблений таким чином, щоб без особливих труднощів можно було вносити до нього зміни: додавати нові елементи, змінювати зображення й назви вже існуючих, змінювати кількість та назви параметрів елементів і таке інше. Усі дії, пов'язані зі створенням моделей, їх симулюванням та переглядом результатів, виконуються за допомогою графічного інтерфейсу програми. Створення моделі при цьому йде шляхом перетягування потрібних елементів на робоче поле та установки їх параметрів. Інтерфейс програми дозволяє вибирати необхідні бібліотеки, брати з них потрібні елементи, встановлювати зв'язки між елементами, а також автоматизувати процес створення нових елементів.

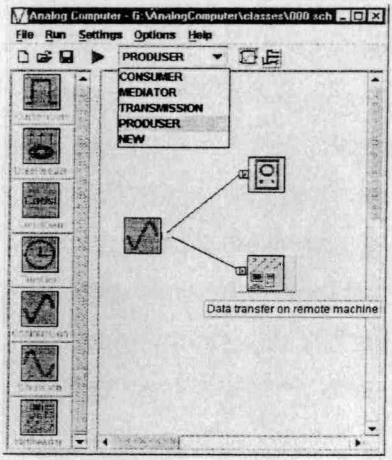

Рис. 1 - Приклад зовнішнього вигляду інтерфейсу розробленої системи

Класи, що реалізують інтерфейс, знаходяться в пакеті Interface. До цього пакету входять декілька класів виключень, що генеруються при виникненні помилок під час виконанні програми. Для роботи з графічними зображенням елементів використовується клас Element, а для відображення його контактів - клас Pin. Сполучення між елементами реалізуються за допомогою класу Junction. Усі створені елементи та зв'язки покладаються на набірне поле (клас PinBoard). Для організації бібліотек елементів використовується клас ElementList. Для перетворення схеми з графічної форми і для передачі її в симулятор використовується клас PinBoardTranslator. Робота з макроелементами здійснюється в класі MacroReader. Створення рядків меню та панелі інструментів відбувається у класі ProgramActions. Головним файлом програми є клас AnalogComputer, і робота програми починається з його запуску.

Програма була розроблена за принципами подійно орієнтованого програмування. Так, в програмі відстежуються події від усіх елементів, що знаходяться на робочому полі програми (такі, як меню, панелі інструментів, елементи, набірне поле і тому подібне). Кожна подія обробляється відповідним чином, а система очікує далі виникнення нової події. Вихід з програми здійснюється вибором відповідного пункту меню або шляхом закриття головного вікна програми.

Розробка спеціалізованих бібліотек

Важливою особливістю створеної програми є можливість розширення існуючої бібліотеки. На базі вже існуючих елементів користувач може розробляти нові спеціалізовані бібліотеки. Дана особливість істотно розширює сферу застосування програми та збільшує коло її користувачів.

Для розробки нового елементу необхідно для нього скласти рівняння, що виражає залежність його вихідних значень від вхідних та від параметрів. Отримані рівняння

вирішуються відносно старших похідних і після цього записуються у блочному вигляді з використанням вже існуючих елементів. Для елемента, що розробляється також вказуються його входи, виходи та параметри.

Таким чином була створена, наприклад, бібліотека для моделювання роботи елементів трансмісії гірничих машин, яка має наступні складні елементи (відповідні системи рівнянь, враховуючи обмежені розміри публікації, не наводяться):

- Diesel – модель двигуна внутрішнього горіння;
- ElectricalEngine1 – модель електродвигуна (ММ Пінчука-Вейца);
- ElectricalEngine2 – модель електродвигуна (ММ Рівіна);
- ElectricalEngine3 – модель електродвигуна очисного комбайну;
- FlyWheel – модель маховика;
- HydroMuff – модель гідромуфти;
- HydroTransformer – модель гідротрансформатора;
- Reductor – модель редуктора.

Використання системи для моделювання у мережевому середовищі

Важливою особливістю розробленої програми є можливість роботи у мережі. Дана спроможність дозволяє розподіляти складні обчислення по декількох комп'ютерах і завдяки цьому скорочувати час моделювання. Також можна здійснювати обмін даними з іншими системами.

Робота у мережі реалізована за допомогою механізму сокетів. Для передачі результатів моделювання іншому комп'ютеру необхідно до схеми, що досліджується, внести деякі зміни: виходи відповідних елементів з'єднати з елементами NetWriter. Як параметр даному елементу задається номер сокету, через який буде робитись зв'язок. У цьому випадку даний комп'ютер буде виступати як сервер, тобто програма буде чекати підключення клієнта, і лише після цього запустить процес моделювання. Щоб скористуватися даними, що передаються сервером, потрібно у схемі, що набрана на комп'ютері клієнта, використати елементи NetReader, параметрами яких будуть відповідні ім'я хост-комп'ютера та номер сокета, через котрий здійснюється з'єднання.

Для проведення експериментів по мережевому моделюванню розроблена система була використана спільно з моделлю інтегрованої навігаційної системи Eline Navigation System, яка призначена для виконання наступних завдань [2, 3]:

- визначення місцеположення судна;
- відображення й оцінка ситуації руху;
- планування траєкторії судна;
- управління рухом судна.

Інтегрована навігаційна система використовується штурманом, який взаємодіє з системою шляхом трекбола та клавіатури, доповнюючи або приймаючи результати моделювання. Головним джерелом інформації для всіх задач інтегрованої навігаційної системи є апріорні знання, представлені у базах знань комп'ютера. Вони складаються з бази даних електронної карти, правил судноводіння та динамічних моделей судна, що керується, та інших суден.

Розроблена система (в цьому експерименті позначена як AnalogComputer) використовувалася для моделювання поведінки керуємого судна. Динаміка змінних стану судна була при цьому описана наступними рівняннями:

$$\dot{y}_s = v\left(\psi_s + \psi_{dr}\right) = v\psi_s + vk_{dr}r_s, \qquad (1)$$

$$\dot{\psi}_s = r_s, \qquad (2)$$

$$\dot{r}_s = -\frac{1}{T}r_s + \frac{K}{T}\delta, \qquad (3)$$

де v - швидкість судна у продольному напрямі,
y_s - координата у вхідної точки R,
Ψ_s - кут повороту,
r_s - швидкість повороту.

Залежність між кутом руля δ та швидкістю повороту r_s описується рівнянням (3). Наступні три параметри вимірюються певними засобами:

- постійна часу T,
- статистичний приріст K,
- коефіцієнт дрейфу k_{dr}.

Завдяки дрейфу судна напрямок руху та поперечна ось не рівні. Ця розбіжність моделюється кутом Ψ, що викликаний дрейфом. Кут Ψ розраховується пропорційно швидкості повороту.

У ході моделювання цієї задачі навігаційна система з певною періодичністю передавала системі моделювання кут повороту руля δ, а назад отримувала швидкість r_s та кут Ψ_s повороту судна.

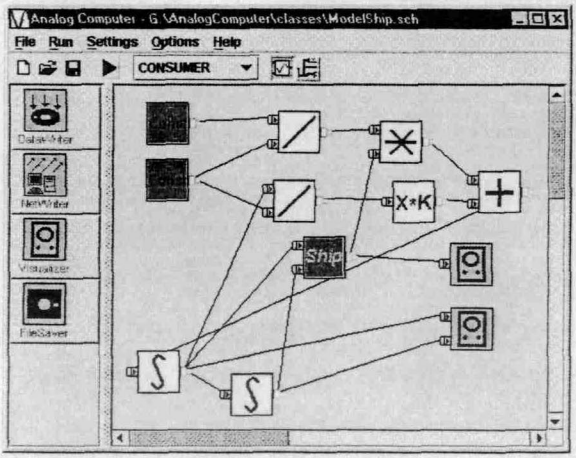

Рис. 2 - Фрагмент схеми моделювання динаміки судна

Спільна робота двох систем здійснювалася наступним чином. В системі навігаційного моделювання та в системі AnalogComputer були створені потоки, відповідальні за

мережевий обмін, який здійснювався в мережі Internet відповідно протоколу TCP/IP. Після запуску моделі судна в навігаційній системі кожні 250 мс здійснювався мережевий обмін. Як вхідні параметри підсистема AnalogComputer одержувала від навігаційної системи значення кута повороту руля судна, що моделювалося. У свою чергу навігаційна система одержувала значення кута відхилення від півночі в діапазоні від 0 до 360 градусів за годинною стрілкою. У залежності від значення кута відхилення від півночі Ψ обчислювалось збільшення ΔX і ΔY для координат судна X та Y. Наприклад, якщо кут Ψ був у межах від 0° до 45° градусів, то збільшення ΔX і ΔY розраховувались наступним чином:

$$\Delta X = V * \Delta\Psi * \Delta t \qquad (4)$$
$$\Delta Y = V * \Delta t$$

де $\Delta\Psi$ - модуль різниці між поточним і попереднім значенням Ψ у радіанах,
Δt - інтервал обміну - 250 мс,
V - швидкість судна.

Таким чином, здійснювався обмін впродовж всього періоду моделювання. Блок-схема проведеного експерименту по розподіленому моделюванню зображена на рис. 3.

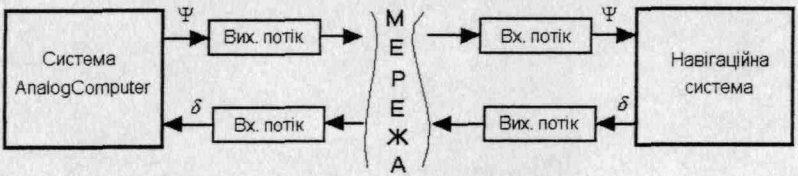

Рис. 3 - Структура розподіленого моделювання

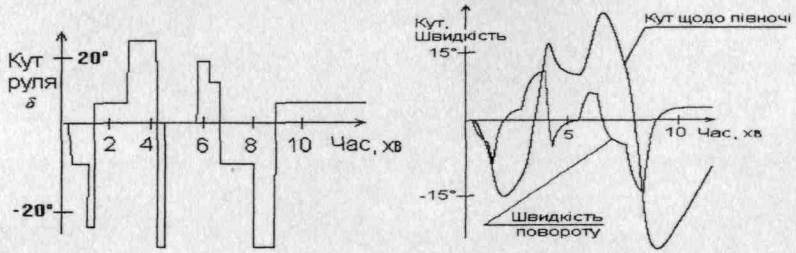

Рис. 4 - Візуалізація вхідних (ліворуч) та вихідних (праворуч) змінних моделювання системою AnalogComputer

Експерименти підтвердили можливість ефективного використання в процесі моделювання протоколу обміну TCP/IP. При цьому необхідно підкреслити такі його характеристики:

- *Надійність.* Якщо сегмент TCP, дрібна одиниця передачі даних TCP, буде загублений або ушкоджений, реалізація TCP виявить це і повторно передасть необхідний сегмент.
- *Універсальність логічного з'єднання.* Перед початком передачі даних TCP установлює з віддаленою машиною зв'язок, користуючись URL, що теоретично дозволяє використовувати для розподіленого моделювання кожний комп'ютер,

з'єднаний з Інтернет.

• *Безупинний потік даних.* TCP забезпечує механізм передачі, що дозволяє пересилати довільну кількість байтів. Після встановлення з'єднання сегменти TCP забезпечують для рівня програм емуляцію безупинного потоку даних.

Висновки

Таким чином, розроблена компактна система моделювання динамічних систем є гнучким, достатньо функціональним та досить простим у використанні моделюючим засобом, що підтвердили експерименти по використанню системи для організації розподіленного моделювання. При цьому потреби в ресурсах є майже на порядок меншими, ніж це має місце в разі використання MATLAB/Simulink. Основними напрямками подальшого розвитку мають бути:

- підвищення обчислювальної потужності системи та ефективності використання нею комп'ютерних ресурсів шляхом оптимізації її ядра та взаємодії з іншими елементами моделючого середовища, наприклад, з паралельними ЕОМ та програмами на C++;
- розробка більш функціональних та різноманітних засобів графічної візуалізації результатів, наприклад, з використанням технології VRML [14, 17];
- розробка різноманітних спеціалізованих бібліотек моделюючих блоків, орієнтованих на конкретні предметні області, а також розробка спеціалізованих варіантів системи, оптимізованих для використання в тих чи інших умовах.

Література

1. Аноприенко А.Я. От вычислений к пониманию: когнитивное компьютерное моделирование и опыт его практического применения на примере решения проблемы Фестского диска // Научные труды Донецкого государственного технического университета. Серия "Информатика, кибернетика и вычислительная техника" (ИКВТ-99). - Донецк: ДонГТУ. - 1999. - С. 36-47.
2. Аноприенко А.Я., Кривошеев С.В. Применение методов параллельного программирования в интегрированной навигационной системе для судов внутреннего и смешанного плавания // Тезисы докладов международной научно-технической конференции "Интеллектуальные многопроцессорные системы" 1-5 сентября 1999 г. - Таганрог. - 1999. - С. 75-77.
3. Аноприенко А.Я., Кривошеев С.В., Потапенко В.А. Моделирование процесса обработки информации в интегрированной навигационной системе // Тези доповідей міждержавної науково-методичної конференції "Комп'ютерне моделювання 30 червня - 2 липня 1999 р., м. Дніпродзержинськ. - Дніпродзержинськ. - 1999. - С. 114-115.
4. Аноприенко А.Я., Святный В.А. Универсальные моделирующие среды // Сборник трудов факультета вычислительной техники и информатики. Вып.1. - Донецк: ДонГТУ. - 1996. - С. 8-23.
5. Гослинг Д., Арнольд К. Язык программирования Java / Пер. с англ. - Спб.: Питер. - 1997. - 304 с.
6. Гультяев А.К. MATLAB 5.2. Имитационное моделирование в среде Windows: Практическое пособие. СПб.: КОРОНА принт. - 1999. - 288 с.
7. Святний В.А. Проблеми паралельного моделювання складних динамічних систем // Наукові праці Донецького державного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка: - Донецьк: ДонДТУ, 1999. - С. 6-14.

8. Святный В.А., Аноприенко А.Я. Опыт реализации системы моделирования динамических процессов на параллельной ЭВМ в среде сетевого графического интерфейса // Тезисы докладов Всесоюзной научно-технической конференции "Перспективы развития и применения средств вычислительной техники для моделирования и автоматизированного исследования". - Москва. - 1991. - С. 190-191.
9. Святный В.А., Цайтц М., Аноприенко А.Я. Реализация системы моделирования динамических процессов на параллельной ЭВМ в среде сетевого графического интерфейса // Вопросы радиоэлектроники, серия "ЭВТ", вып. 2. - 1991. - С. 85 - 94.
10. Anoprienko A., Bazhenov L., Bräunl T. The development of the interface subsystem for the massive paralel simulation environment // 11. Simposium in Dortmund "Simulationstechnik". November 1997. - Braunschweig: Vieweg. - 1997. - S. 672-677.
11. Anoprienko A., Feldmann L., Lapko V., Svjatnyj V., Braeunl T., Reuter A., Zeitz M. Massive parallel models of net dynamic objects. Proceedings of the 1995 EUROSIM Conference, EUROSIM-95, Vienna, Austria, 11-15 September 1995, ELSEVIER. - 1995. - P. 237 - 242.
12. Anoprienko A., Svjatnyi V., Bräunl T., Reuter A., Zeitz M. Massiv parallele Simulationsumgebung für dynamische Systeme mit konzentrierten und verteilten Parametern // 9. Simposium in Stuttgart "Simulationstechnik", Oktober 1994. Vieweg. - 1994. - S. 183-188.
13. Büchel G., Hartung G. A practical course in design and simulation of digital systems using the Web // Global Congress on Engineering Education. - Cracow, Poland, 6-11 September, 1998. - P. 164-168.
14. Diessner M. Visualisierung simulierter Prozesse im W3 // 9. Simposium in Stuttgart "Simulationstechnik", Oktober 1994. Vieweg. - 1994. - S. 581-586.
15. Dorwarth H., Schumann M., Seibt F. Komponenten und Konzepte für Simulationsumgebungen im WWW // 9. Simposium in Stuttgart "Simulationstechnik", Oktober 1994. Vieweg. - 1994. - S. 593-598.
16. Kazitov M.V., Nelayev V.V. Active virtual laboratory on the Internet as an effective tool for learning // Global Congress on Engineering Education. - Cracow, Poland, 6-11 September, 1998. - P. 269-272.
17. Kötter H.-F., Krämer B., Völker N. 3D-Visualisierung sicherheitskritischer Vorgänger in der Lehre mittels Java und VRML // Proc. Simulation und Visualisierung '99. 4-5 März 1999. - Magdeburg: SCS. - 1999. - S. 19-32.
18. Leister W., Larsen A. Grafische Methoden auf dem Web - Eine Übersicht // Proc. Simulation und Visualisierung '99. 4-5 März 1999. - Magdeburg: SCS. - 1999. - S. 359-376.
19. Reger K. Konfigurierbarkeit universeller Simulationssysteme // 11. Simposium in Dortmund "Simulationstechnik". November 1997. - Braunschweig: Vieweg. - 1997. - S. 541-546.
20. Svjatnyi V., Feldmann L., Lapko V., Anoprienko A., Reuter A., Bräunl T., Zeitz M. Massive parallel simulation of dynamic systems // Zeszyty naukowe. - 1997. - №1. - P. 207-229.