УДК 519.173:004.92

Т.М. Басюк
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

ОСНОВНІ ПІДХОДИ ДО ПОБУДОВИ ПРОГРАМНИХ ЗАСОБІВ ВІЗУАЛІЗАЦІЇ ДАНИХ

Проаналізовано методології проектування інформаційних систем та наведено основні підходи до побудови програмних засобів візуалізації даних.

In article methodology of designing of information systems are analysed and the basic approaches to construction of data visualization software are presented.

Вступ

За останні роки комп’ютерні засоби моделювання та візуалізації перетворились із інструментів вирішення різного роду задач на потужний апарат дослідження суспільних явищ та технічних проблем. Вони з успіхом використовуються у таких галузях, як системний аналіз, автоматизація проектування, організація роботи обчислювальних засобів та комп’ютерних мереж [1, 2].

Різноманітність галузей, у яких використовуються програмні засоби моделювання й візуалізації, пояснюється неможливістю або трудомісткістю проведення реальних експериментів на окремих групах досліджуваних об’єктів. З огляду на це, проблема створення нових програмних систем переводиться в площину використання вже наявних програмних компонентів та їх адаптації до нових задач [3, 4].

Що стосується візуалізації даних, наданих у вигляді матричних структур, із подальшим їх перетворенням у графові моделі, то сьогодні накопичений відносно невеликий досвід їх вирішення. Основними завданнями при цьому є як фундаментальні (пояснюються недосконалістю існуючих методів візуалізації), так і технічні (пов’язані із складністю вибору та забезпеченням взаємодії різних програмних та апаратних засобів) [5, 6]. Отже, актуальним завданням є виокремлення основних аспектів побудови прикладного програмного забезпечення, яке надавало б розробникам можливість із створення високорівневих засобів візуалізації даних, заданих у матричній формі.

Огляд літературних джерел

Сучасні засоби візуалізації будуються у напрямку розроблення гнучкої структури прикладної програми, яка може змінюватись залежно від вимог, що висуваються користувачем. Забезпечення цієї модифікації накладає деякі обмеження на розробників, найважливішим серед яких є неможливість часткової перебудови системи відповідно до нових вимог. Це завдання є вкрай важливим, оскільки його вирішення дасть змогу скоротити процес як розроблення, так і модифікації програм цього класу.

Незважаючи на актуальність завдання, сьогодні накопичений відносно невеликий досвід її вирішення, особливо щодо комп’ютерної реалізації. У роботах [5, 7, 8] описано основні моделі побудови програмного забезпечення. Проте у жодному з наведених досліджень авторами не було проаналізовано та описано весь процес побудови та можливої модифікації розроблених програмних засобів залежно від зміни постановки задачі.

Постановка задачі

З огляду на зазначену складність, важливим завданням є розроблення основних підходів до побудови програмних засобів, застосування яких дало змогу значно спростити процедуру проектування програмних засобів візуалізації матричних структур. Крім того, запропоновані рішення повинні

передбачати можливість відносно легкої модифікації програмного забезпечення цього класу. При цьому важливими поняттями, які необхідно врахувати, має бути синтез моделей роботи системи.

Основні результати досліджень

На сучасному етапі розвитку інформаційних технологій значно ускладнюється процедура побудови та швидкої модифікації інформаційних систем, які виникають у різних галузях людської діяльності. Не є винятком засоби візуалізації матричних структур, які характеризуються:

- складністю опису, що вимагає забезпечення складного процесу моделювання й аналізу даних та процесів;
- сукупністю взаємодіючих компонентів, які мають свої локальні завдання й мету функціонування. Наприклад, підпрограми моделювання графічної форми зображення, аналітичної обробки розташування вершин, пошуку в матричній структурі ознак належності до того чи іншого типу графу;
- відсутністю прямих аналогів, що обмежує можливість використання яких-небудь типових проектних рішень і прикладних систем;
- необхідністю функціонування в неоднорідному середовищі (у різних операційних системах та апаратних платформах);
- істотною часовою протяжністю проекту, обумовленою обмеженими можливостями колективу розробників та різним ступенем готовності окремих частин системи.

З врахуванням наведених чинників пропонується під час розроблення інформаційної системи візуалізації матричних структур виходити з модульного принципу побудови. Застосування модульного принципу дасть змогу створити програмний продукт шляхом створення нових або модифікації існуючих компонентів (у разі зміни постановки задачі), що значно пришвидшить процедуру створення й адаптації та дасть змогу розподілити роботу між окремими виконавцями [8].

Що стосується методології проектування, то сьогодні для побудови інформаційних систем широко застосовуються *каскадна та спіральна моделі*. Проектування згідно із каскадною моделлю полягає в розділенні процесу розроблення на етапи (каскади), коли з одного етапу на інший переходять лише після завершення роботи на попередньому (рис. 1).

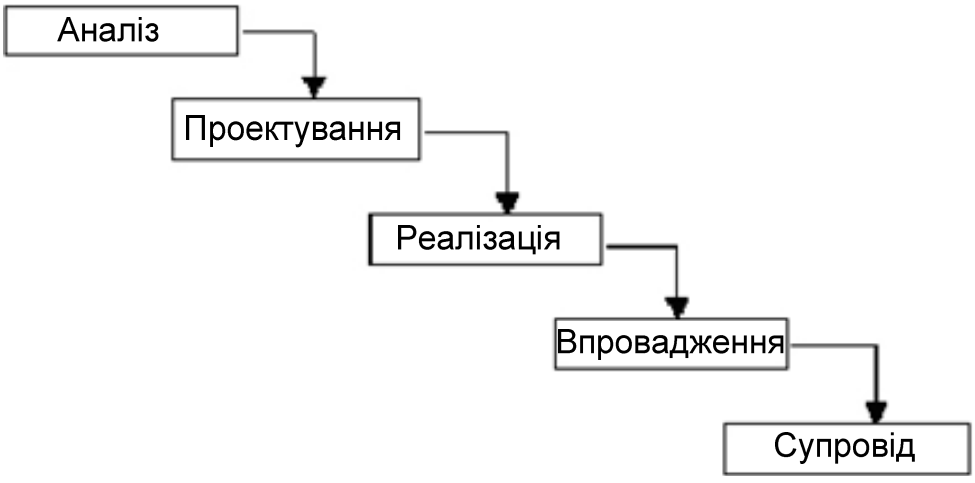

Рис. 1. Каскадна методологія проектування

Перевагами каскадного підходу є те, що на кожному етапі формується завершений набір проектної документації, достатньої для продовження роботи іншою групою виконавців, а виконувані в логічній послідовності етапи робіт дають змогу чітко планувати терміни завершення та відповідні затрати.

Цей підхід є зручним у випадку побудови інформаційної системи, для якої на початку розроблення можна точно сформулювати всі вимоги, яким вона повинна відповідати. Проте у разі застосування каскадної методології виникають непередбачені деталі, викликані тим, що в реальному

процесі створення програмного забезпечення буває необхідно повернутися до попередніх етапів для уточнення чи перегляду раніше прийнятих рішень. У результаті реальний процес створення набуватиме вигляду, наведеного на рис. 2.

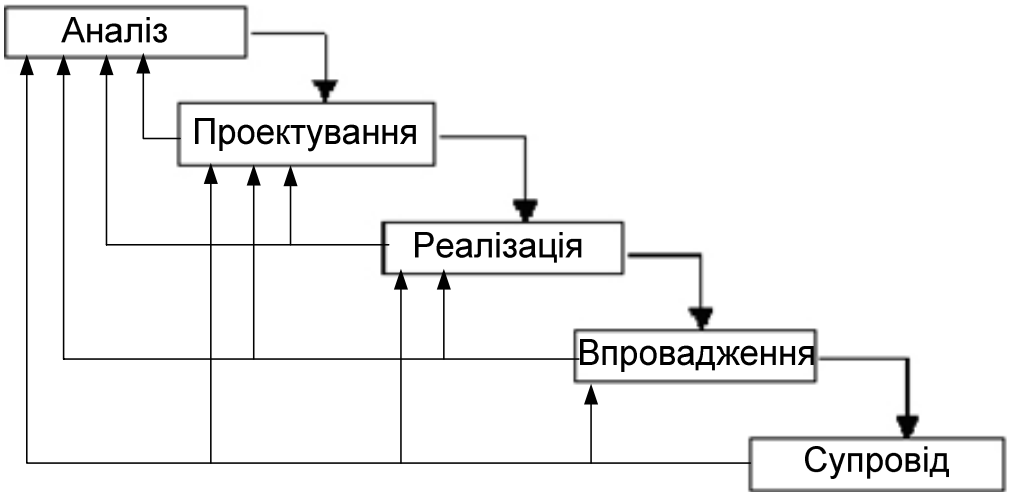

Рис. 2. Модифікована каскадна методологія проектування

Отже, значним недоліком каскадного підходу є *істотне запізнення* в отриманні результатів. Погодження здійснюється тільки після завершення кожного з етапів робіт, що значно знижує гнучкість та швидкість розроблення, а у випадку неточного викладення вимог чи їх зміни користувачі отримають систему, яка не відповідає їх вимогам. Що стосується *спіральної методології*, то її особливістю є акцентування уваги на початкових етапах: аналізі та проектуванні. На цих етапах розробляють програми-прототипи, які повинні продемонструвати функції проектованої системи (рис. 3).

Рис. 3. Спіральна методологія проектування

Кожен виток спіралі відповідає створенню фрагмента чи версії програмного забезпечення. На ньому уточнюються цілі та характеристики проекту, визначається його якість та плануються роботи, пов’язані із наступним витком спіралі. Тобто, здійснюється процес послідовної конкретизації проекту, який і доводиться до реалізації.

Незважаючи на позитивність спіральної методології, *основним її недоліком* під час розв’язання задач візуалізації є визначення моменту переходу на наступний етап, що переважно і створює певну затримку при проектуванні системи, оскільки цей перехід здійснюється на основі плану чи статистичних даних навіть у випадку незакінченості роботи на поточному етапі.

Що стосується візуалізації матричних структур, то застосування наведених методологій побудови є вкрай незручним як для реалізації уніфікованого ядра, так і для розроблення прикладних

бібліотек обчислювальних модулів. Оскільки їх розроблення має проходити із врахуванням різних вимог до конструювання, що накладає певні обмеження на методологію проектування [3].

З огляду на зазначені особливості оптимальним з погляду реалізації є використання об'єктно-орієнтованого підходу для побудови прикладних програмних систем певного класу. Оскільки механізми інкапсуляції, наслідування та поліморфізму, що ним передбачені, можуть бути конструктивно застосовані як для реалізації уніфікованого ядра, так і для розроблення прикладних бібліотек обчислювальних модулів [9, 10].

Популярність використання об'єктно-орієнтованого підходу багато в чому обумовлена концептуальною цілісністю і кращою формою структуризації програмного забезпечення, розроблювального на його основі. У результаті з використанням цього підходу досягають швидшого і надійнішого розроблення програм, а також, можливості гнучкої модифікації існуючого програмного забезпечення при створенні нових програмних варіацій. Ці обставини є вирішальними при побудові інформаційних систем як галузі знань, що динамічно розвивається.

Застосування об'єктно-орієнтованого підходу для реалізації цієї задачі містить етапи, наведені на рис. 4.

Рис. 4. Етапи проектування системи візуалізації

У статті зупинимось на описі та дослідженні першого етапу. Він є базовим і полягає в аналізі вимог та попередньому проектуванні системи. На початку розроблення системи після аналізу вимог необхідно розбити її на модулі. Під модулями прийнято розуміти набір класів та окремих об'єктів, підсистем та подій, взаємозв'язаних між собою. Зв'язок (інтерфейс) між підсистемами забезпечує форму всіх взаємодій між ними, проте не специфікує внутрішню структуру чи особливості реалізації підсистеми. З огляду на це кожна підсистема може розроблятися незалежно від решти підсистем. Підсистеми визначають послідовний спосіб розгляду прикладної задачі, для вирішення якої розробляється певна система. Наприклад, система файлів є підсистемою операційної системи, яка забезпечує набір взаємозв'язаних абстрактних операцій, що реалізуються у вигляді окремого модуля.

Підсистеми повинні визначатися так, щоб більша частина їх взаємодій залишалась всередині підсистеми і тим самим зменшувала глобальні потоки даних та залежності між ними. Існує два типи взаємодій між підсистемами: клієнт-сервер та міжпрограмна взаємодія. У першому випадку клієнт дає запит серверу на виконання певної операції, а сервер повертає результат. У випадку міжпрограмної взаємодії обидві підсистеми викликають одна одну та обмінюються даними у міру необхідності. Для реалізації прикладної системи відображення даних оптимальною є реалізація на міжпрограмному рівні, оскільки дає змогу простіше та швидше (минаючи канали зв'язку) реалізувати процедури створення графічних примітивів.

Особливістю цього етапу є синтез моделей, які дають змогу спроектувати та протестувати розроблену систему залежно від вимог, які до неї висуваються. Вимогами можуть бути завдання обслуговування клієнтів банку, пошуку інформації в інформаційній мережі чи реалізація комп'ютерного верстання книжкової продукції. Застосовування моделей є вкрай важливим, оскільки вони дають змогу спростити розроблення, аналіз і реалізацію системи на комп'ютері внаслідок акцентування на основних особливостях розроблюваного продукту. Побудова моделей є поширеним способом вивчення складних об'єктів і явищ для перевірки працездатності розроблювальної системи та її модифікації як на початку проектування, так і на інших етапах розроблення.

Розробляючи систему візуалізації даних, її представляють у вигляді двох взаємозалежних моделей: функціональної та об'єктної. Вони дають змогу адекватно представити структуру проектованої системи та визначити всі функціональні залежності між об'єктами загалом.

Об'єктна модель представляє проектований додаток у вигляді сукупності об'єктів, кожен з яких відповідає певним особливостям та виконує специфічні функції. В ній зображаються об'єкти, важливі для розроблюваного додатка, які визначають прагматику досліджуваної системи. Під об'єктами розуміють поняття, абстракції або будь-які елементи з чітко окресленими границями, які мають значення у контексті розглядуваної задачі. Окреслення об'єктів переслідує дві мети: визначення розглядуваної задачі та основ для її реалізації на комп'ютері. Отже, ціллю розробки об'єктної моделі є опис елементів, що становлять проектовану систему та встановлення залежностей між ними [8]. Об'єктну модель програмного продукту візуалізації графів наведено на рис. 5.

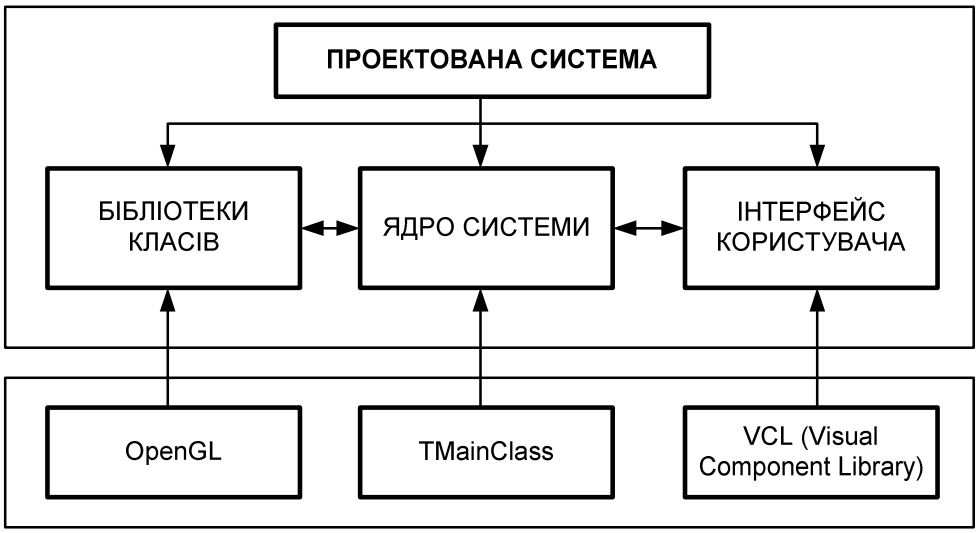

Рис. 5. Об'єктна модель системи відображення даних

Спроектована модель складається з трьох основних частин: графічного інтерфейсу користувача, ядра системи та бібліотеки класів, що реалізує основні процедури обробки зображень. *Графічний інтерфейс користувача* об'єднує в собі всі елементи та компоненти програми, які впливають на взаємодію користувача з прикладною програмною системою та виконує функції: навігації між блоками системи, відображення інформації про поточний стан системи, зворотний зв'язок з користувачем, підтримку прийняття рішень при використанні системи. Особливою

функцією графічного інтерфейсу є організація процесу введення вхідної інформації, її фільтрування та збереження на зовнішніх носіях. Графічний інтерфейс програми створюється із використанням бібліотеки візуальних компонентів (Visual Component Library, VCL) [11]. Це є об’єктно-орієнтована бібліотека, яка застосовується при створенні програм за допомогою засобів візуального програмування. VCL містить велику кількість готових елементів керування, придатних для застосування в різних сферах, таких як робота з базами даних, взаємодія з операційною системою, програмування мережевих додатків.

Ядро системи забезпечує функціонування додатка та інтегрує в собі такі процеси, як моделювання, візуалізація, підтримка нових типів об’єктів та алгоритмів. Ядро функціонально дає змогу маніпулювати окремими об’єктами та створювати конвеєри моделювання та візуалізації з окремих алгоритмів. Застосування такого підходу дає змогу не лише створювати результуюче зображення, але й зберігати історію його створення. Володіючи даними історії створення, ядро системи може в автоматичному режимі модифікувати об’єкти зображення у випадку зміни початкових даних.

Важливою операцією, яку виконує ядро системи, є опрацювання граничних ситуацій, яке здійснюється шляхом визначення реакції кожного об'єкта й всієї системи на певні зовнішні дії. Виділяють такі граничні ситуації: ініціалізація, термінація й обвал. Зокрема, процес ініціалізації полягає в приведенні системи у фіксований початковий стан: оголошення всіх початкових змінних та параметрів задачі та формування ієрархії класів. Це є початкова фаза роботи системи, під час якої доступною є лише частина можливостей. Процес термінації полягає у вивільненні всіх апаратних та програмних ресурсів, зайнятих під задачі системи. Забезпечення цього процесу є вкрай важливим, оскільки за його допомогою забезпечується безпомилкова робота апаратного забезпечення при завершенні роботи прикладної програми. Процес обвалу являє собою незапланована термінацію системи. Ця ситуація може виникнути у результаті помилок користувача (при введенні некоректних вхідних даних), недостатньої кількості ресурсів системи (брак місця на зовнішніх носіях інформації) чи сторонніх впливів (вимкнення електроенергії). З огляду на зазначені чинники у ядрі системи передбачені відповідні алгоритми та методи перехоплення, що мінімізують вплив зазначених факторів на роботу програми.

Бібліотека класів – реалізує основні процедури обробки зображень. Дані та алгоритми бібліотеки класів виконані у вигляді ієрархії. Їх застосування дає змогу реалізувати основні методи опрацювання сформованих зображень відповідно до потреб користувача. Класи бібліотеки, крім основних геометричних перетворень, підтримують додаткові (об’єднання, накладання, позиціонування). Як стандарт під час розроблення бібліотеки класів застосовують бібліотеку Open Graphics Library (OpenGL) – відкриту графічна бібліотеку, яка визначає незалежний від мови програмування кросплатформений програмний інтерфейс для написання додатків, що використовують комп’ютерну графіку [12]. Ця бібліотека містить близько 250 функцій для відображення різних графічних схем із застосуванням примітивів. OpenGL орієнтується переважно на уніфікацію реалізації програмного коду незалежно від апаратної платформи. На базовому рівні ця бібліотека описує поведінку функцій, що її формують. Реалізація стандарту дає змогу використовувати наявні функції апаратного забезпечення, а у випадку неможливості їх забезпечення – емулювати їх програмно. Основним принципом роботи бібліотеки є отримання набору векторних графічних примітивів з подальшим математичним їх опрацюванням та побудовою растрового зображення на екрані монітора.

Запропонована об'єктна модель представляє статичну структуру проектованої системи. Однак її знання є недостатнім для розуміння й оцінки роботи, оскільки необхідно мати засоби для опису змін, що відбуваються з об'єктами та їх зв'язками під час роботи кожної підсистеми. З огляду на це будується функціональна модель. Для її побудови необхідно визначити вхідні і вихідні значення параметрів об’єктів моделі та побудувати схему функціональних залежностей між її елементами.

Функціональна модель складається з окремих об’єктів та взаємозв’язків між ними (рис. 6).

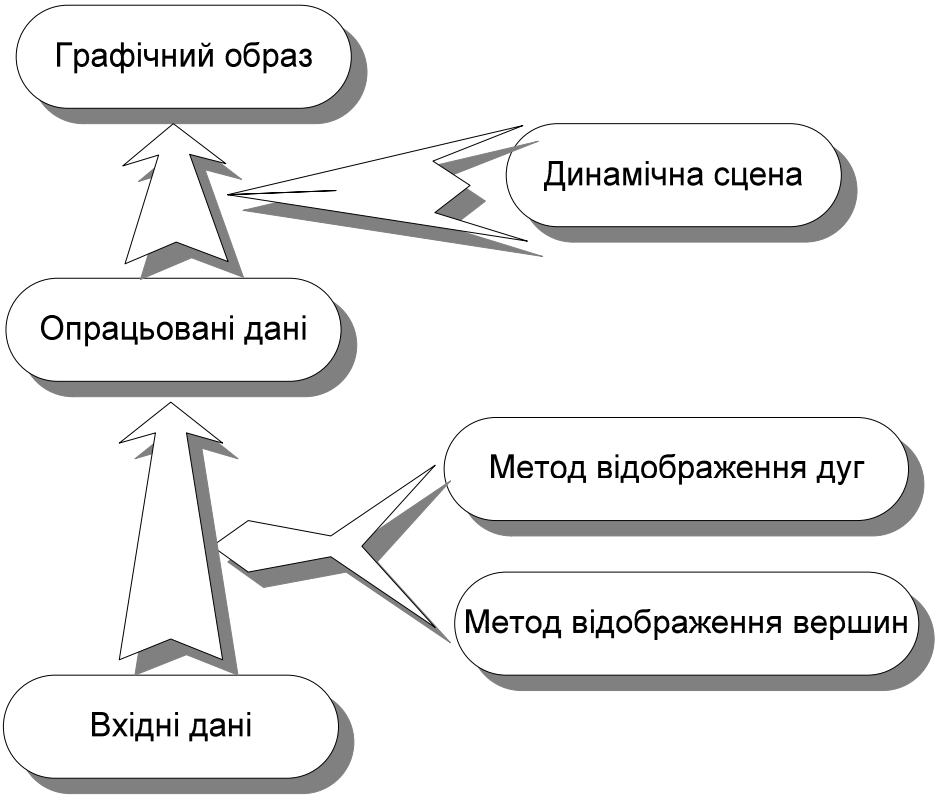

Рис. 6. Функціональна модель побудови зображення

У розробленій моделі стан об'єктів характеризується сукупністю поточних значень атрибутів та зв'язків. Під час роботи вони взаємодіють між собою, у результаті чого змінюються їх стани. Одиницею впливу є події, кожна з яких змінює стан одного або декількох об'єктів у системі або спричиняє виникнення нових подій. Робота системи характеризується послідовністю подій, що в ній відбуваються. Вони можуть бути незалежними (введення матриці суміжності) або передувати іншим (візуалізація графу здійснюється після введення матриці) і передають інформацію з одного об'єкта в інший. Особливістю створеної функціональної моделі є наявність певних взаємозв'язків, що відтворюють процес візуалізації і наводять процедуру створення об'єкта візуалізації у визначені моменти часу. З огляду на зазначені особливості, представлені моделі дають змогу одержати повне уявлення про функціональність системи, що значно спрощує подальше її програмування, налагодження, супровід та модифікацію.

Важливим етапом проектування прикладної програми візуалізації даних є реалізація принципів керування програмним забезпеченням. Як видно із функціональної моделі, під час аналізу всі взаємодії в системі подаються у вигляді подій. Існує два класи методів керування ними: зовнішні та внутрішні. При зовнішньому керуванні в кожний момент часу діє одна із процедур (це є найпростіший принцип). Внутрішнє керування пов'язане з потоками управління в процесах. На відміну від зовнішніх подій, внутрішні принципи передачі контролюються програмою й можуть бути за необхідності структуровані. Створення системи візуалізації даних вимагає проектування додатка з використанням принципів зовнішнього керування із застосуванням методики інтерактивного інтерфейсу (рис. 5). Основною особливістю цього типу систем є потреба у виділенні об'єктів, що формують інтерфейс (забезпечення їх реалізації за допомогою VCL) з використанням засобів середовищ об'єктно-орієнтованого програмування (механізмів наслідування та поліморфізму), які дають змогу утворити цілісний каркас програмного продукту та відносно легко його модифікувати.

Перспективи досліджень

Розроблення інформаційної системи візуалізації даних може бути зведена до об'єктного аналізу прикладної задачі, вибору наукових даних та специфікації методів, а також до визначення

об’єктів та реалізації спеціальної бібліотеки для їх представлення та маніпулювання ними. Бібліотека повинна бути побудована у вигляді ієрархії класів, наслідуваних від класів ядра TMain. Важливо, що механізми об’єктно-орієнтованого проектування підтримують повторне використання раніше розроблених класів як для створення нових класів, так і для їх використання в нових алгоритмах з новими типами об’єктів. Зокрема, розробляти певну прикладну програму необхідно відповідно до таких підходів:

- ідентифікація класів наукових даних та алгоритмів, які беруть участь у постановці та розв’язанні задач шляхом вибору ключових даних прикладної області;
- визначення семантики вибраних класів та множини їх відношень, таких як „загальний–специфічний” і розроблення класифікації об’єктів (ієрархії наслідуваних класів) для даних та алгоритмів;
- встановлення залежностей використання між класами об’єктів як вхідних, так і вихідних зв’язків шляхом семантичного аналізу та їх розподілу на прості та множинні зв’язки;
- написання класів, які визначають специфічні атрибути, властивості та функціонування алгоритмів, що виконують основні операції;
- тестування різних методів та алгоритмів роботи, що ґрунтуються на вибраному системному класі.

Нові класи, будучи інтегрованими в систему, повинні бути реалізовані за допомогою описаних підходів, які дають змогу уніфікувати користувацький інтерфейс та процедури введення-виведення. Автоматично візуалізовані, класи геометричних об’єктів повинні реалізовувати методи геометричного та топологічного опису. Ці методи дають змогу виконувати стандартні процедури відображення на рівні абстрактного класу TMain.

Висновок

Проаналізовано основні підходи до проектування програмних засобів візуалізації даних на основі об’єктно-орієнтованого підходу. Застосування цієї методології дає змогу відносно легко побудувати систему на основі запропонованих моделей та забезпечити механізми її модифікації шляхом розширення функцій чи змін алгоритмів візуалізації. Завдяки оригінальній структурі запропоновані підходи дають змогу реалізувати широкі функціональні можливості для створення спеціальних інтегрованих систем візуалізації та математичного моделювання.

Крім того, зазначені підходи є простими у використанні, високофункціональними під час маніпулювання науковими даними та забезпечують як високу гнучкість, так і легку розширюваність при створенні складних спеціалізованих систем.

1. Блюменау Д.И. Информация и информационный сервис. – Л.:Наука, 1989. – 420 с. 2. Галузинський Г.П., Гордієнко І.В. Сучасні технологічні засоби обробки інформації: Навч. посібник. – К.: КНЕУ, 1998. – 224 с. 3. Басюк Т.М. Критерії відображення графів в процесі візуалізації // Наукові записки Української академії друкарства. – 2004. – № 7. – С. 60–63. 4. Басюк Т.М. Аналіз та класифікація методів візуалізації // Поліграфія і видавнича справа. –2003. – № 40. – С. 109–114. 5. Касьянов В.Н., Евстегнеев В.А. Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ, 2003. – 1104 с. 6. Дистель Р. Теория графов. – Новосибирск: Изд-во Института математики, 2002. – 336 с. 7. Гайсарян С.С. Объектно-ориентированные технологии проектирования прикладных программных систем. – М.: ЦИТ, 2001 – 342 с. 8. Гамма Э., Хэлм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования: Пер. с англ. А. Сликина. – СПб.: Питер, 2003. – 386 с. 9. Басюк Т.М. Синтез об’єктної та функціональної моделей // Комп’ютерні технології друкарства. – 2005. – №13. – С.79–85. 10. Семенов В.А., Крылов П.Б., Морозов С.В., Тарлапан О.А. Объектно-ориентированная архитектура для приложений научной визуализации и математического моделирования // Программирование. – 1999. – № 8. – С. 24–32. 11. Архангельский А.Я. Приемы программирования в Delphi на основе VCL. – М.: Бином, 2006. – 944 с. 12. Евченко А. OpenGL и DirectX: программирование графики. – СПб.: Питер, 2006. – 352 с.