УДК 681.3.06

А.М. Ковальчук, к.т.н., доц.
В.Г. Левицький, к.т.н., доц.
І.І. Самолюк, аспір.
В.М. Янчук, к.т.н., доц.
Житомирський державний технологічний університет

АВТОМАТИЗОВАНА РОЗРОБКА МАТЕМАТИЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ КОМПОНЕНТНОЇ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ

Представлено компоненту клієнт-серверну архітектуру, орієнтовану на автоматизацію процесу розробки математичного програмного забезпечення. Розглянуто особливості використання двох основних типів програмних компонент: компонент математичного аналізу даних та компонент побудови інтерфейсу користувача. Визначено можливість практичної реалізації запропонованої схеми на прикладі використання сучасних клієнт-серверних технологій.

Постановка проблеми. В наш час рівень розвитку обчислювальної техніки та мов програмування дозволяє за короткий термін створювати програмне забезпечення (ПЗ) для вирішення широкого кола задач. Однак якість і вартість програмного забезпечення безпосередньо залежить від кількості розробників та часу, витраченого на розробку. Задача зменшення вартості та підвищення швидкості проектування та реалізації ПЗ шляхом автоматизації процесу створення програм є актуальною та необхідною.

Однією з головних проблем, які виникають у процесі автоматизації даної задачі ПЗ є питання створення архітектури ПЗ, придатної для автоматизованої розробки. Зазначимо, що для вирішення поставленої задачі також необхідно обмежити область використання (ОВ) програмного забезпечення, яке розробляється. Така вимога необхідна для проведення детальної структурної декомпозиції задач з ОВ програмної системи з метою виявлення та врахування головних особливостей описання та аналізу задачі, специфіку представлення результатів тощо.

Мета і завдання роботи. Метою роботи є автоматизована розробка математичного програмного забезпечення. Основне завдання дослідження: розробка компонентної клієнт-серверної архітектури для автоматизованого формування програмного забезпечення.

Наукова новизна роботи. Розроблено нову для предметної області математичного програмного забезпечення процедуру автоматизованого формування програмного забезпечення на основі запропонованої компонентної клієнт-серверної архітектури.

Аналіз досліджень і публікацій. Дана робота є логічним продовженням досліджень, які проводились колективом авторів раніше [1–6]. Основна її відмінність полягає в об'єднанні та узагальненні запропонованих ідей з індивідуальних підходів до розробки спеціалізованого ПЗ на область математичного програмного забезпечення.

Існуючі підходи до розробки та створення програмного забезпечення за часом та умовами формування засобів вирішення задач ОВ можна розділити на дві великі групи:

1) створення монолітної програмної системи за допомогою традиційних засобів розробки, яка не містить зв'язків з іншими програмними засобами;
2) створення програмних систем, що формуються в процесі виконання із множини підсистем або компонент за правилами, які визначені раніше.

При розробці математичного ПЗ не існує обмежень на використання першого чи другого способу формування системи. За першою схемою будь-яка автоматизація вимагатиме повторної генерації машинного коду всієї програми. Другий спосіб позбавлений такого недоліку, тому що він дозволяє модифікувати або змінювати окремі компоненти програмної системи або навіть правила їх поєднання. Однак, незважаючи на привабливість такого способу формування ПЗ, він вимагає набагато більше зусиль від розробників саме на стадії формування архітектури програмної системи.

Можливими способами реалізації другої схеми формування ПЗ, на жаль, є використання дещо прив'язаної до операційної системи технології динамічного зв'язування бібліотек або позбавленої даного недоліку технології клієнт-сервер.

Клієнт-серверний підхід не є принципово новим при створенні програмного забезпечення [7]. Однак не всі потенціальні можливості зазначеної технології використовувалися повною мірою, особливо при створенні складних програмних систем, призначених, наприклад, для числового аналізу математичних задач. Серед головних причин не досить ефективного використання можливостей технології можна виділити такі: 1) відсутність незмінного стандарту для реалізації механізмів високорівневої взаємодії між клієнтом та сервером; 2) складність проектування архітектури програмного забезпечення, яка надасть можливість без особливих затрат праці розширювати функціональні можливості системи за рахунок підключення нових компонент.

Концептуальна схема математичного програмного забезпечення. Складові частини програмного забезпечення (тут та далі компоненти) умовно за призначенням можуть бути поділені на дві основні групи: компоненти, які відповідають за взаємодію користувачем та компоненти, що проводять математичні обчислення. Окремо слід виділити підсистему (будемо називати її контейнер), яка поєднує всі підсистеми в одне ціле. В термінах технології клієнт-сервер контейнер виконує функції тонкого клієнта, який керує процесом взаємодії компонент серверів. Зазначимо, що за запропонованою схемою клієнт має повну інформацію про те, які компоненти інтерфейсу користувача взаємодіють з серверами математичних обчислень, однак він не має відомостей про взаємодію серверів між собою. Завданням клієнта є пошук та завантаження необхідних серверів, а також надання серверу математичних обчислень, відомостей про всі наявні сервери інтерфейсу користувача. Крім розподілу за типами серверів існує також розподіл за видами задач, які вирішують математичні компоненти. Як правило, такий вид компонент є орієнтованим на вузьку предметну область, наприклад вирішення задач наближення даних, числове вирішення систем диференціальних рівнянь, побудову компіляторів тощо. Виходячи з цього, ми суттєво звужуємо область пошуку серверів інтерфейсу користувача, необхідних для вирішення окремої задачі. Такий підхід надає також можливість повторного використання компонент інтерфейсу користувача та агрегацію математичних компонент при вирішенні складних прикладних задач. Для підтримки функціонування запропонованої архітектури розроблено ряд програмних інтерфейсів, як клієнта так і серверів, за допомогою яких компоненти можуть бути поєднані у єдину систему, яка вирішує поставлені проектувальником завдання. Згідно з запропонованим описом було створено спрощену схему взаємодії компонент за клієнт-серверною архітектурою для математичного програмного забезпечення (рис. 1).

Використання шаблонів задач для адаптивного формування програмної системи. Запропонована архітектура накладає обмеження на процедуру роботи користувача в програмній системі. Кожен крок процесу формування програмної системи вимагає автоматизованої побудови множин візуальних компонент (V) та компонент обробки даних з множини попередніх дій користувача (R) та з множини попередніх візуальних компонент. Очевидно, що перший крок не має попередньої інформації про дії користувача, тому початковими даними є множини V та L. Як закон, за яким буде зроблений наступний крок в ітеративному процесі формування ПЗ, використано множини шаблонів задач (T) – структури, які містять формалізований опис можливих елементів V, R, L та відношень між ними.

Шаблони задач за їх функціональним призначенням можна поділити на декілька груп. Формально множину компонент для розв'язку математичних задач L можна представити як таку, що складається з елементів множини видів математичних задач $F = \{F_1, F_2, \ldots, F_n\}$, де кожен елемент зв'язаний з певною, індивідуальною для кожного елемента F_i підмножиною видів аналізу $A = \{A_1, A_2, \ldots, A_m\}$. Будь-який з елементів A_j може бути представлений у вигляді підмножини числових методів $M = \{M_1, M_2, \ldots, M_p\}$, за допомогою яких проводиться числовий аналіз. В загальному випадку для кожного числового методу існує унікальна послідовність маніпуляцій для того, щоб описати задачу, провести її аналіз, отримати та переглянути результати досліджень. Проведення таких маніпуляцій вимагає введення в інтерфейс користувача програмної системи відповідної множини візуальних засобів або

використання внутрішньої мови. Таким чином, програмна система, що орієнтована на розв'язання великої кількості видів математичних задач повинна мати величезну кількість візуальних засобів або надскладну внутрішню мову. Інтерфейси, що мають велику кількість візуальних засобів (такі як Mathcad) чи занадто ускладнену мову (Matlab та MATEMATICA), незручні у використанні. Для вирішення даної проблеми була використана ідея адаптивного розподілу управління процесом вирішення математичної задачі між лінгвістичним та візуальним інтерфейсом програмної системи. В подальшому розвитку даної ідеї було запропоновано розбити весь процес аналізу математичної задачі на декілька етапів, кожен з яких відповідає за вирішення окремої частини задачі. Кількість етапів (тобто рівень деталізації процесу описання, аналізу, представлення результатів та надання допомоги) залежить від виду задачі, що розв'язується.

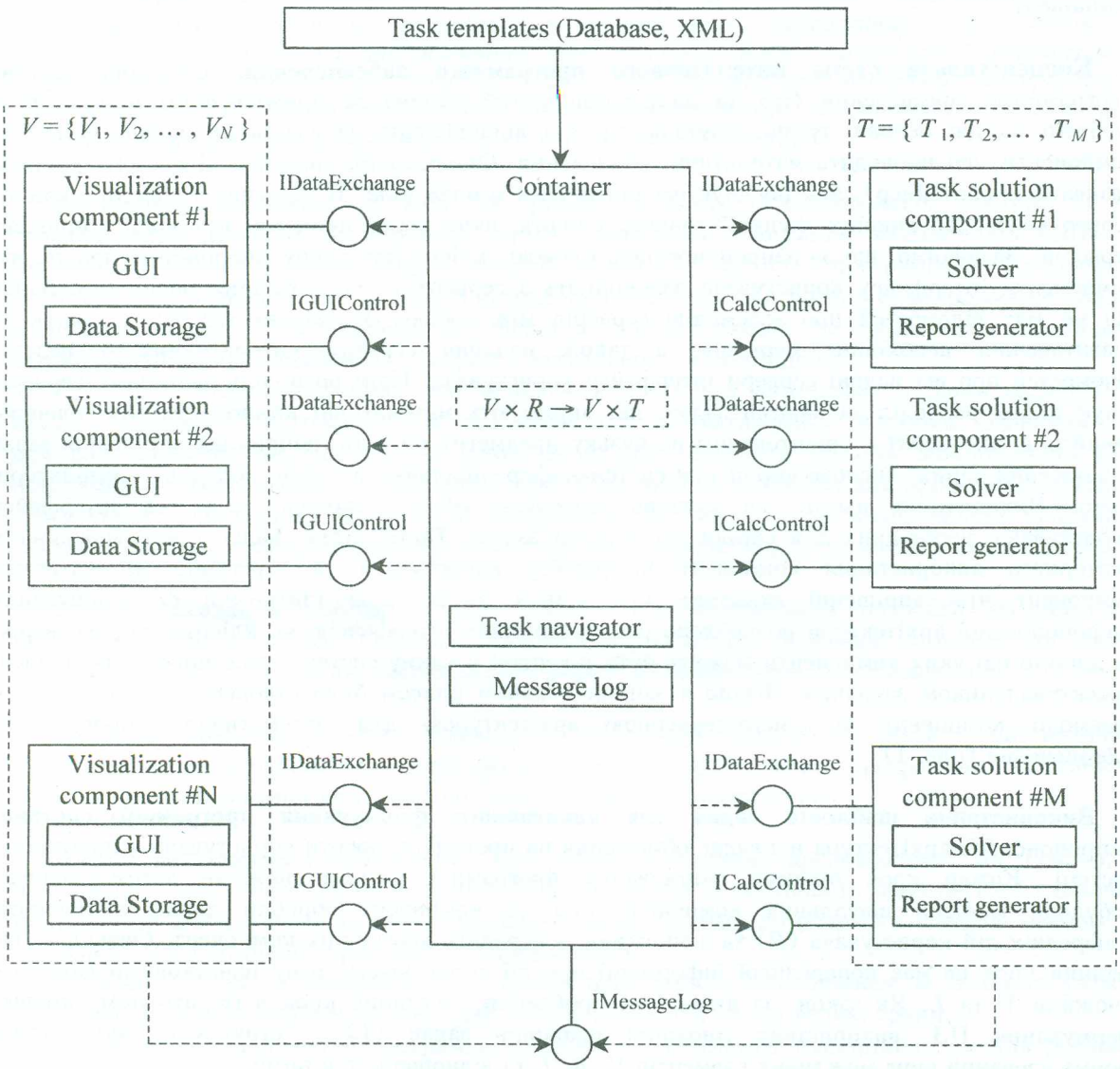

Рис. 1. Спрощена схема взаємодії компонент за клієнт-серверною архітектурою для математичного програмного забезпечення

Отже для кожного елемента з множини числових методів M інтерфейс користувача може бути представлений у вигляді множини інтерфейсів I, де елементи I – це інтерфейси користувача на кожному етапі аналізу задачі. Таким чином, інтерфейс користувача для i-ї математичної задачі j-го виду аналізу, k-го методу аналізу може бути представлений у вигляді $I_{jik} = \{F_i, A_j, M_k, I, R\}$, де R – множина попередніх маніпуляцій користувача. Множину інтерфейсів I визначаємо як $\{V, L, Q\}$, де V – підмножина візуальних інтерфейсів, L – підмножина мовних інтерфейсів, а Q – непуста скінчена множина станів ІК. Всі представлені в

програмній системі види аналізу об'єднані в підмножини $G = \{G_1, G_2, \ldots, G_p\}$, яким відповідають однакові підмножини візуальних інтерфейсів.

Процес формування математичного ПЗ використовує також ряд множин, що вміщують інформацію довідникового характеру щодо ієрархії задач, видів та методів аналізу, візуальних інтерфейсів і підсистем формування ІК.

1. Ієрархія математичних задач і відповідних елементів їх візуального представлення складає множину кортежів $D_F = \{<IDM, P, O, N, M, I>\}$, де IDM – унікальний номер, що ідентифікує математичну задачу ($F_{IDM} \in F$); P – номер, що вказує на ідентифікатор батьківської задачі більш високого рівня ієрархії ($F_P \in F$); O – порядковий номер задачі серед задач такого ж рівня ієрархії; N – назва задачі; I – графічне представлення задачі.

2. Ієрархія видів аналізу для математичних задач і відповідних елементів їх візуального представлення складає множину кортежів $D_A = \{<IDA, IDM, P, O, N, I, G, S>\}$, де IDA – унікальний номер, що ідентифікує вид аналізу для математичної задачі ($A_{IDA} \in A$); IDM – номер, що ідентифікує задачу ($F_{IDM} \in F$); P – номер, що вказує на ідентифікатор батьківського виду аналізу більш високого рівня ієрархії ($A_P \in A$); O – порядковий номер виду аналізу серед задач такого ж рівня ієрархії; N – назва виду аналізу; I – графічне представлення виду аналізу; G – номер підмножини видів аналізу, яким відповідають однакові підмножини візуальних інтерфейсів; S – підмножина станів ІК ($S \subset Q$).

3. Ієрархія математичних методів проведення аналізу для математичних задач і відповідних елементів їх візуального подання складає множину кортежів $D_M = \{<MM, IDA, MS, N, I>\}$, де MM – унікальний номер математичного методу ($M_{MM} \in M$); IDA – номер, що ідентифікує вид аналізу для математичної задачі ($A_{IDA} \in A$); MS – номер підмножини математичних методів, які використовуються в даному методі; N – назва математичного методу; I – графічне подання математичного методу.

4. Ієрархія взаємного використання математичних методів складає множину кортежів $D_{MS} = \{<MM, MS>\}$, де MM – номер математичного методу ($M_{MM} \in M$); MS – номер підмножини математичних методів, до складу якої входить даний метод.

5. Зв'язки станів і множин підсистем формування ІК складають множину кортежів $D_{SK} = \{<SI, K, O, N, I>\}$, де SI – стан ІК ($SI \in Q$); K – номер, що визначає підмножину підсистем формування ІК; O – пріоритет підмножини підсистем формування ІК в даному стані; N – назва підмножини підсистем формування ІК; I – графічне представлення (зображення) підмножини підсистем формування ІК.

6. Візуальні інтерфейси підсистем формування ІК складають множину кортежів $D_V = \{<KI, V>\}$, де KI – унікальний номер підсистеми формування ІК; V – візуальний інтерфейс користувача.

Процедура взаємодії між клієнтськими та серверними компонентами. Шаблон задачі забезпечує точку входу для динамічного формування програмного забезпечення. Використовуючи формальні методи з [1–2] контейнер використовує шаблони задачі для генерації множини елементів управління задачею, яку далі будемо називати навігатором задачі. Дана множина елементів управління керує станами компонент обробки даних та компонентами відображення даних. Таким чином користувачі отримують можливість управління системою та можливість як обчислити, так і наповнити контейнери компонентами відображення даних.

Інша важлива функція контейнера – управління потоками даних між компонентами обробки даних та компонентами відображення даних. Складність практичної реалізації взаємодії між множиною різних за призначенням та будовою компонент можна подолати за рахунок використання стандартного (спроектованого та реалізованого заздалегідь) набору програмних інтерфейсів. Всі програмні інтерфейси для взаємодії компонент повинні бути реалізовані у кожній без виключення компоненті.

Пропонується реалізувати підтримку двох видів інтерфейсів IDataExchange та IGUIControl кожною компонентою представлення даних. Інтерфейс IGUIControl реалізує методи управління функціональними властивостями компонент. Кожна математична задача може мати власний специфічний спосіб представлення даних, тому інтерфейс IGUIControl обов'язковий для всіх компонент, що потребують адаптації до особливості поточної математичної задачі. Така вимога створює значний потенціал при розробці компонент, здатних використовуватися для різних математичних задач.

Інтерфейс IDataExchange реалізує методи обміну даними, що необхідні контейнеру в процесі функціонування потоку даних компонентами в обох напрямках. Таким чином, дані передаються від однієї компоненти до іншої під управлінням контейнера. Наприклад результати роботи одної компоненти обробки можуть бути передані компоненті представлення даних або іншій компоненті обробки під управлінням контейнера. Таке архітектурне рішення було прийняте для того, щоб позбавити розробників компонент складного процесу вибору потрібної компоненти з підмножини наявних компонент представлення для наступного кроку формування програмної системи.

Крім інтерфейсу IDataExchange всі компоненти обробки даних, підтримують інтерфейс ICalcControl, що необхідний для управління процесом математичних обчислень. Інтерфейс IDataExchange надає можливість призупиняти, переключати та поновлювати математичні обчислення, тобто реалізує методи управління ходом математичних обчислень.

Інший важливий інтерфейс в запропонованій архітектурі – інтерфейс IMessageLog, що підтримує зберігання різних повідомлень від компонент. Цей інтерфейс належить контейнеру та використовується, в процесі керування системою обробки помилок.

Висновки та перспективи подальших досліджень. В роботі було запропоновано компонентну клієнт-серверну архітектуру математичного програмного забезпечення з елементами функціональної відкритості та адаптивності. Створення даної архітектури є результатом детальної структурної декомпозиції та розробки формального опису ОВ ПЗ. Необхідність даної розробки обумовлена тим, що створення окремої програмної системи для кожного методу дослідження математичної задачі не відповідає сучасним вимогам програмної інженерії та вимагає значних матеріальних та людських ресурсів. Запропонована архітектура ПЗ завдяки можливості модифікації ПЗ без додаткової модифікації програмного коду у перспективі забезпечує:

1) відкритість системи для розширення функціональних можливостей;
2) перенесення великої кількості предметно-орієнтованої функціональності з середовища контейнера на периферійний рівень (на рівень компонент), що дозволяє значно розвантажити контейнер;
3) автоматизацію формування ПЗ при підключенні компонент для вирішення нових математичних задач;
4) проведення розподілених обчислень;
5) можливість підключення до системи компонент незалежно від мови програмування, що була використана для їх створення;
6) створення віртуальної лабораторії, де для розв'язку однієї складної задачі буде залучено декілька комп'ютерів, зв'язаних між собою локальною мережею або мережею Internet.

ЛІТЕРАТУРА:

1. *Ковальчук А.М., Левицький В.Г.* Компонентна архітектура адаптивного інтерфейсу користувача програмного інструмента розробки комп'ютерних практикумів // Вестник Херсонского государственного технического университета. – 2002. – № 14. – С. 291–295.
2. *Ковальчук А.М., Левицький В.Г.* Розробка адаптивного інтерфейсу користувача програмної системи числового аналізу математичних задач // Вісник ЖІТІ. – 2002. – № 20. – С. 111–119.
3. *Kolodnytskyy M., Kovalchuk A., Kuryata S., Levitskyy V.* The Mathematical Software Implementation for Computational Algebra and Number Theory // Proceedings of the 4th Asian Symposium on Computer Mathematics, December 17–21, Chiang Mai, Thailand, 2000. – P. 291–294.
4. *Ковальчук А.М., Левицький В.Г., Янчук В.Г.* Вивчення математичних моделей міграції радіонуклідів у лісових екосистемах // Вестник Херсонского государственного технического университета, 2002. – № 14. – С. 82–87.
5. *Kolodnytskyy M., Kovalchuk A., Kuryata S., Levitskyy V., Samolyuk I.* Interactive Visual Software System for Problem Solving and Teaching in Finite Group Theory // Proceedings of the 23rd International Conference on Information Technology Interfaces, Pula, Croatia, 2001. – P. 409–416.

6. *Левицький В.Г.* Автоматизація процесу конструювання трансляторів за допомогою інструментального засобу "FANCYCC 2.0" // Праці 2-ї національної науково-практичної конференції студентів та аспірантів "Системний аналіз та інформаційні технології", 28-30 червня 2000 р. – Київ, 2000. – С. 164-167.
7. *Richter J., Clark J.* Programming Server-Side Applications for Microsoft® Windows® 2000, Microsoft Press, 2000. – 736 p.

КОВАЛЬЧУК Андрій Михайлович – кандидат технічних наук, доцент кафедри АіКТ Житомирського державного технологічного університету.

Наукові інтереси:
- комп'ютерні інформаційні технології;
- архітектура програмних систем;
- графічні системи та методи організації графічних інтерфейсів користувача прикладних програмних систем;
- використання обчислювальної техніки в навчальному процесі.

ЛЕВИЦЬКИЙ В'ячеслав Георгійович – кандидат технічних наук, доцент кафедри ПЗОТ Житомирського державного технологічного університету.

Наукові інтереси:
- комп'ютерні інформаційні технології;
- використання обчислювальної техніки в навчальному процесі;
- побудова компіляторів.

САМОЛЮК Ігор Іванович – кандидат технічних наук, доцент кафедри АіКТ Житомирського державного технологічного університету.

Наукові інтереси:
- комп'ютерні інформаційні технології;
- математичне моделювання;
- теорія автоматичного керування.

ЯНЧУК Валентин Миколайович – кандидат технічних наук, доцент кафедри АіКТ Житомирського державного технологічного університету.

Наукові інтереси:
- комп'ютерні інформаційні технології;
- інформаційні системи, бази даних;
- математичне моделювання екологічних процесів.

Подано 01.11.2003