MATHEMATICAL AND COMPUTER MODELING

УДК 004 (94+41)

ВОЛОДИМИР СОКОЛОВ

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ІНТЕГРАЛЬНИХ ОБ'ЄКТІВ

В роботі представлено архітектуру програмного забезпечення, що складається з об'єктів, здатних до безпосередньої взаємодії шляхом утворення активних динамічних сполук, та бути атомарними об'єктами або інтегральними. Визначено, що будь-яка сполука об'єктів, як і атоми, має валентність – здатність до утворення сполук, та може бути інкапсульована в клас сполуки. Сформульовано принципи, на яких базується запропонована архітектура. Такі як наявність атомарного базису архітектури, інкапсуляція конекторів всередині компонентів, інтеграція як основний принцип побудови ієрархії компонентів, заборона успадкування класів компонентів на атомарному та вищих рівнях, дескриптивність структур компонентів. Показано, що утворення нових похідних класів як сполук атомів можна вважати інтеграцією, степінь якої залежить від рівня інтеграції складових об'єктів. Ієрархія класів на основі інтеграції без успадкування дозволяє використовувати уніфікований механізм як створення класів, так і взаємодії об'єктів. В результаті досліджень розроблено єдиний механізм побудови інтегральних об'єктів завдяки дескриптору, який містить повний опис структури об'єкта, що дозволяє використовувати універсальну функцію синтезу сполук всіх інтегральних об'єктів. Обґрунтовано, що розв'язання будь-якої задачі можна розглядати як процес синтезу дескриптора інтегрального об'єкту. Розглянуто питання статичної та динамічної поведінки інтегральних об'єктів, розроблено двох етапний підхід до створення інтегрального об'єкта шляхом утворення початкового прото-об'єкту з подальшою побудовою повного об'єкту, що дає можливість відкладеного та часткового створення інтегрального об'єкту під час його використання в програмі. Визначено, що під час виконання програми структура інтегрального об'єкту функціонально еквівалентна повній дезінтеграції об'єкту до атомарного рівня, що дає можливість явно перетворювати інтегральні об'єкти в атомарні сполуки. Розглянуто ізомерні класи об'єктів, що можуть змінювати структуру сполуки під час виконання, залишаючи незмінною зовнішній інтерфейс. В якості опису архітектури обрано графічну форму, формулу сполуки, яка відображає склад інтегрального об'єкта та степінь інтеграції, атомарну формулу сполуки, яка відображає реальний склад інтегрального об'єкта з атомів, та структурна формула, що відображає порядок з'єднання об'єктів у сполуці. Показано, що для такої архітектури можна застосовувати операції над конфігурацією як статично, так і динамічно.

Ключові слова: архітектура програмного забезпечення, інтегральні об'єкти, валентність об'єктів, формула сполуки, ізомерні класи.

Постановка проблеми. Архітектура програмного забезпечення (ПЗ), що створюється на основі об'єктно-орієнтованого підходу, використовує систему моделей ПЗ, наприклад, у вигляді діаграм універсальної мови моделювання (Unified Modeling Language, UML). При цьому питання архітектури верхнього загальносистемного рівня відпрацьовуються достатньо глибоко, але відображення проектних рішень на програмну реалізацію часто виконується програмістами неоднозначно внаслідок складності розуміння архітектури. Слабким місцем цього процесу є степінь автоматизації відображення моделі архітектури на її реалізації. Архітектуру ПЗ можна розділити на три основні рівні: системний, модельно-орієнтований, реалізації. Системна архітектура оперує вимогами, декомпозицією на підсистеми та компоненти. Модельно-орієнтована архітектура визначає вид компонент у залежності від обраної парадигми або моделі (наприклад, об'єктна-орієнтована, сервіс-орієнтована, процедурна).

Архітектура реалізації оперує програмними елементами, з яких складається система. Саме архітектура реалізації є найменш дослідженою і цілком залежить від здібностей програміста, хоча саме нею визначається складність ПЗ. Спрощення архітектури реалізації завдяки уніфікації проектних рішень суттєво впливає на весь процес розробки ПЗ. Тому розробка ефективної архітектури реалізації є актуальною проблемою.

Парадигма об'єктно-орієнтованого програмування (ООП) базується на відомих принципах і використовує шаблони проектування, частина з яких є архітектурними, та універсальну мову моделювання (Unified Modeling Language, UML) для аналізу та проектування об'єктно-орієнтованих програм. Однак, питання способів взаємодії об'єктів та архітектури реалізації відпрацьовані недостатньо.

Великі діаграми класів не дають уявлення проектувальнику про склад класу, наприклад, 10 рівня успадкування. Внаслідок цього складність проектування і витрати на реалізацію та виконання зростають. Перспективним є шлях створення програм на мета-рівні з можливістю автоматичної генерації потрібного коду. Для таких систем важливим є уніфікація способів взаємодії, єдиний механізм утворення складних ієрархічних структур програм та можливість рефакторингу архітектури без великих витрат. Розуміння структури класу, об'єкту та залежностей між ним напряму впливає на складність програмування. Цим підтверджується актуальність проблеми розроблення простих та ефективних архітектур.

Аналіз останніх досліджень і публікацій. Вирішенню проблеми розроблення архітектур програмного забезпечення присвячено багато публікацій, зокрема стосовно ООП. Створено багато шаблонів проектування, стандартних рішень та продовжується пошук нових, що також підтверджує необхідність розроблення простих та ефективних архітектур.

В [1] розглядаються питання перспектив подальшого розвитку та впровадження шаблонів проектування ПЗ, їх переваги та недоліки, а також умови застосування. У [2] розглянуто засоби формалізованого проектування та синтезу абстрактних типів даних, алгебраїчних класів і об'єктно-орієнтованих програм, за їхніми уявленнями в алгебрах алгоритмів. Описано метод синтезу як частини самого інструментарію, так і багатопотокової програми мовою Java. Необхідність розробки архітектур та стандартизація їх опису розглядається в [3], зокрема, представлено огляд сучасних стандартів опису архітектур. В роботі [4] розглядається проблема наскрізного функціоналу як такого, що розподілений по ПЗ за межами модулів і пропонуються архітектурні рішення цієї проблеми шляхом винесення наскрізного функціоналу в окремий клас. Питання інтеграції різних архітектур розглянуто в [5], а також зміни архітектури ПЗ, невідповідність задокументованої архітектури її реалізації та пропонується формальний опис пошарової архітектури, метод інтеграції модульної, пошарової та об'єктно-орієнтованої архітектур.

З огляду на ці публікації можна зробити висновок, що основні напрями досліджень пов'язані з проблемами повторного використання коду, багатошаровістю та ієрархічністю під час розробки архітектур ПЗ.

Метою статті є зменшення складності процесу розробки об'єктно-орієнтованих програм завдяки розробленню архітектури на основі інтегральних об'єктів.

Опис архітектури. Будемо скорочено називати архітектуру програмного забезпечення на основі інтегральних об'єктів – АІО. При описі АІО розглядається клас обчислювальних задач, в яких потрібно для заданих вхідних даних обчислити результат. Відповідно до шаблону проектування MVC (Model – View– Controller) акцент робиться на модель. Крім того, АІО базується на технології активних динамічних сполук об'єктів (АДС) [6], якою використовується ООП.

Основними елементами архітектури, як відомо, є компоненти, конектори та конфігурація [3]. Для опису АІО, крім словесного використовуватимемо формальний опис, графічну форму та формули.

Компоненти. Як компоненти АІО, використовуються об'єкти класів, які мають спеціальну будову. Такими компонентами є атомарні та інтегральні об'єкти. Особливістю архітектури є те, що конектори включені до складу компонентів та забезпечують представлення вхідних та вихідних даних і безпосередню взаємодію об'єктів.

Атоми є об'єктами атомарних класів, вимоги до яких описано в [7]. Атом реалізує одну функцію, має нуль або більше входів і один вихід. Функцію атома реалізує ядро, яке активується при звертанні до виходів. Вхідні та вихідні конектори забезпечують утворення сполук шляхом з'єднання вихідних конекторів одних об'єктів з вхідними конекторами інших об'єктів. З огляду на це можна говорити про валентність атома, яка визначає його властивість до утворення сполук та може мати кількісну оцінку у вигляді дробу, наприклад 2/1, що означає вхідну валентність 2 та вихідну валентність 1. Конектори мають тип даних.

Функціонально атомарний клас можна описати як

$$A: x, y \to z,$$

де A – ім'я класу;

x, y – входи;

z – вихід.

Валентність можна позначити як $A(2/1)$.

Графічно атомарний об'єкт a класу A можна представити як показано на рис. 1.

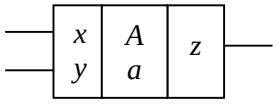

Рисунок 1 – Графічний опис атомарного об'єкта

Інтегральні об'єкти є екземплярами інтегральних класів, які є контейнерами, що містять сполуку об'єктів. Наприклад (див. рис. 2):

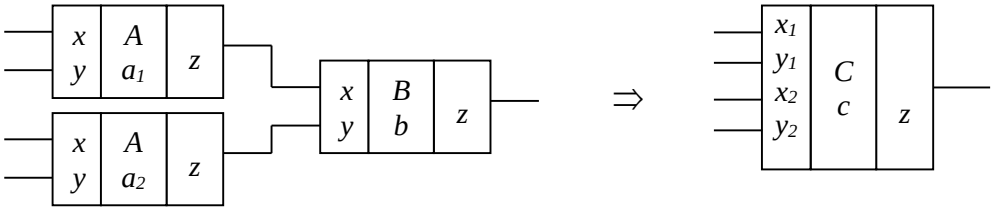

Рисунок 2 – Приклад утворення інтегрального об'єкта

Інтегральний клас C реалізує функцію, яка є композицією функцій складових об'єктів

$$C: x_1, y_1, x_2, y_2 \to z.$$

Валентність класу $C(4/1)$. Формула сполуки, яку реалізує клас C можна представити як

$$C^1 = A_2 B. \qquad (1)$$

де верхній індекс інтегрального класу визначає степінь інтеграції, а нижній індекс класу означає кількість об'єктів цього класу у складі сполуки (у випадку 1 вона не пишеться).

Степінь інтеграції атомів дорівнює нулю, а для інтегральних класів визначається максимальним степенем інтеграції складових класів (степінь інтеграції класу на 1 більше максимального степеня базових класів). Тому, у загальному випадку формула сполуки інтегрального класу містить кількість об'єктів кожного класу та степені їх інтеграції, наприклад

$$D^2 = C_2^1 B. \qquad (2)$$

Інтегральний клас можна вважати похідним від базових класів. Розкладаючи формулу сполуки інтегрального об'єкта рекурсивно можна отримати еквівалентну атомарну формулу, в якій фігурують виключно атомарні класи та сумарна кількість атомарних об'єктів кожного класу. Така формула відображає реальний склад інтегрального об'єкта. Наприклад, для (2), враховуючи (1), атомарна формула має вигляд

$$D^1 = A_4 B_3.$$

Формула сполуки дає кількісну оцінку складу інтегрального класу. Можливе існування двох або більше класів з однаковою формулою, але з різною структурою. Такі класи можна вважати ізомерними. Для повного опису класу доцільно використовувати структурну формулу.

Структурна формула має відображати повну інформацію про будову інтегрального класу і повинна містити [7]:

1. Множину базових класів *Classes*;
2. Множину об'єктів *Objects*, кожний з яких є екземпляром класу з *Classes*;
3. Схему сполуки *Connection*, яка задає структуру сполуки;
4. Множину входів *Inputs*, що є входами об'єктів;
5. Множину виходів *Outputs*, що є результуючими виходами об'єктів;

$$S = \langle\, Classes,\ Objects,\ Inputs,\ Outputs \,\rangle$$

Кожний інтегральний клас має містити дескриптор інтегрального класу (ДІК), до складу якого входить структурна формула, ім'я класу та покажчик степені інтеграції. ДІК зберігається в одному екземплярі і доступний всім об'єктам класу. Дескриптор кожного інтегрального об'єкту містить ім'я об'єкту. Таким чином, дескрипторам притаманні всі метадані, необхідні для створення інтегральних об'єктів та доступні під час виконання програми. ДІК може представлятися різними способами, наприклад, у форматі XML.

Внаслідок того, що в АІО не використовується статичне успадкування компонентів, а замість неї застосовуються динамічні сполуки та інтеграція, то виникає задача створення та знищення інтегрального об'єкту під час виконання програми. З іншого боку, функцію інтегрального об'єкта реалізують ядра базових об'єктів без будь-якого іншого додаткового програмного коду, тому на функцію ядра інтегрального об'єкту можна покласти створення та знищення інтегрального об'єкту.

Створення інтегрального об'єкту може відбуватися двома способами. Перший спосіб передбачає безпосереднє повне створення базових об'єктів та їх сполуки під час створення об'єкту (функція синтезу сполуки зі складу ядра викликається конструктором класу). Інший спосіб передбачає створення інтегрального об'єкту в два етапи: на першому етапі створюється лише оболонка інтегрального об'єкту (прото-об'єкт), який містить тільки вхідні та вихідні конектори; на другому етапі, при активації об'єкта (звертання до його виходів) функція синтезу, використовуючи дескриптор, створює базові об'єкти та їх сполуку. При такому способі отримуємо відкладене створення інтегрального об'єкта до моменту його першого використання та ієрархічне створення базових інтегральних об'єктів таким самим способом, причому створення інтегрального об'єкту може бути неповним для даної активації, якщо для обчислення активованого виходу потрібна лише частина сполуки.

Знищення інтегрального об'єкту може відбуватися трьома способами: повне знищення (виклик функції знищення зі складу ядра деструктором класу), знищення сполуки до стану прото-об'єкта та знищення прото-об'єкта.

Така реалізація створення та знищення інтегральних об'єктів надає гнучкості в управлінні пам'яттю та забезпечує можливість створення універсальної для всіх інтегральних об'єктів функції ядра, робота якої залежить лише від метаданих дескрипторів.

Дані. Атомарні дані можуть представлятися компонентами, які транслюють вхідні значення у вихідні без перетворень. У випадку констант компонент не має входу, а тільки вихід. Функція ядра атому даних є тривіальною. Інтегральні класи даних не можуть мати схему сполуки, але можуть об'єднуватись в структурні типи, наприклад (див. рис. 3):

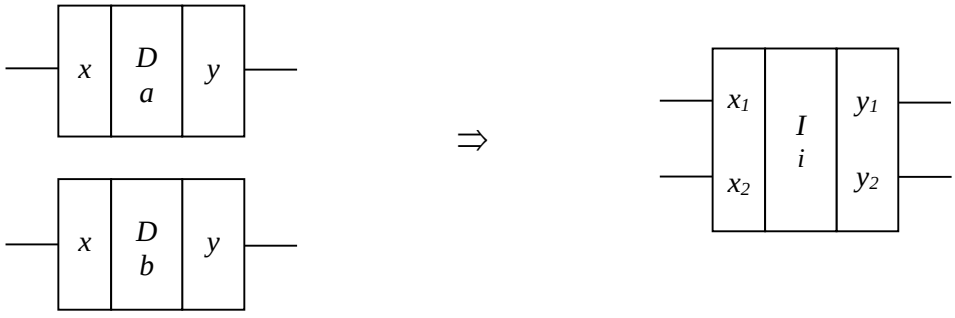

Рисунок 3 – Приклад утворення структур даних

Вихідна валентність об'єктів даних дорівнює вихідній валентності, крім випадку використання констант, коли вихідна валентність менше вхідної. Степінь інтеграції об'єктів даних визначає рівень вкладеності структур даних.

Конфігурація. Конфігурація визначає спосіб з'єднання компонентів архітектури для їх взаємодії. Об'єкт, який є елементом сполуки, містить у складі ядра функцію з'єднання вхідного конектора з вихідним конектором іншого об'єкту шляхом зв'язування з ним за посиланням. Об'єкт не може бути ініціатором з'єднання. Ініціатор з'єднання має знаходитись ззовні об'єкта, наприклад, функція синтезу сполуки у складі ядра інтегрального об'єкту.

Один вхід об'єкту може з'єднуватись лише з одним виходом іншого об'єкту, в той час як з одним виходом об'єкту можуть з'єднуватись декілька входів інших об'єктів.

В АІО розглядаються лише такі схеми сполук, що не мають циклічних зв'язків (можуть бути представлені ациклічним графом).

В загальному випадку інтегральний об'єкт може інтегрувати декілька не зв'язаних між собою сполук.

Сполука інтегрального об'єкта може не мати властивостей, що притаманні атомам, такі як повна функціональна залежність виходів від входів та відсутність транзитивних залежностей виходів якщо, наприклад, схема сполуки містить виходи, що не є виходами кінцевих об'єктів.

Можна виділити наступні операції над конфігурацією:

– розпізнавання класу сполуки – операція співставлення зі зразком сполуки або фрагменту сполуки з наявними інтегральними класами;

– ідентифікація класу – клас ідентифікується структурною формулою; два класи є еквівалентними, якщо їх структурні формули співпадають з точністю до імен об'єктів і конекторів;

– інтеграція – утворення нового класу для заданої сполуки, збільшення степені інтеграції;

– повна інтеграція – рекурсивна заміна об'єктів сполуки на об'єкти максимальної степені інтеграції відомих класів; використовує операцію розпізнавання класу сполуки;

– дезінтеграція – виведення сполуки зі складу інтегрального об'єкту, зменшення степені інтеграції;

– повна дезінтеграція – рекурсивна дезінтеграція інтегрального об'єкту до сполуки, об'єктами якої є виключно атоми (мінімізація степені інтеграції)

– розгортання прото-об'єку – ієрархічне створення сполуки до атомарного рівня з прото-об'єкту;

– згортання до прото-об'єкту – ієрархічне знищення сполуки інтегрального об'єкту до рівня прото-об'єкту.

Наведені операції дозволяють виконувати перетворення компонентів архітектури як статично, так і динамічно.

Архітектурні розрізи. ПЗ, що створено за АІО, можна розглядати з точки зору процесів створення ПЗ та його виконання. В процесі створення ПЗ для розв'язання певної задачі мають створюватись базові атомарні класи та інтегральні класи, які можуть повторно використовуватись як самостійно, так і як базові для інших інтегральних класів. Такий підхід є стандартним підходом до побудови програмних систем, зокрема і в ООП. Основна операція – інтеграція – виконується проектувальником вручну та зводиться до побудови ДІК. Тобто фактично весь процес програмування можна розглядати як процес створення атомів та ДІК з використанням універсальної функції синтезу сполук. Накопичення та повторне використання цих основних артефактів є основною діяльністю в АІО.

Динамічна складова розглядає процес виконання ПЗ, побудованого за АІО. Так як створення інтегральних об'єктів відбувається динамічно та ієрархічно, то вся функціональність визначається та реалізується функціями атомів та схемою їх з'єднання. Створення, знищення та функціонування інтегральних об'єктів може відбуватися з

використанням паралельних процесів. Операція повної дезінтеграції до атомарного рівня є способом перетворення архітектури в процесі виконання для оптимізації програми. Застосування поетапного створення та знищення інтегральних об'єктів з використанням прото-об'єктів також дозволяє оптимізувати витрати ресурсів.

Архітектурні принципи. Таким чином, в ході досліджень було визначено архітектурні принципи, на яких базується АІО:

– атомарний базис – нижнім рівнем компонентів АІО є атомарні об'єкти, які реалізують функціональність всіх інтегральних об'єктів;

– інкапсуляція конекторів – кожний об'єкт містить вхідні та вихідні конектори, які забезпечують взаємодію між об'єктами шляхом сполучання вхідних конекторів одних об'єктів з вихідними конекторами інших об'єктів;

– інтеграція – основний спосіб утворення нових класів у вигляді сполуки атомарних та інших інтегральних об'єктів;

– заборона успадкування – атомарні та інтегральні класи не можуть успадковувати інші класи, але допускають успадкування субатомарних класів; замість успадкування використовується інтеграція;

– дескриптивність – кожний об'єкт АІО повинен мати дескриптор класу та об'єкту, який містить метадані про об'єкт, що забезпечує виконання операцій над архітектурою.

Висновки. Створено архітектуру, що базується на єдиних принципах взаємодії компонентів та утворення ієрархічних структур, використовує єдині механізми динамічних операцій над архітектурою, що дозволяє зменшити як складність ПЗ, що проектується, шляхом рахунок уніфікування способів побудови системи, так і витрати ресурсів під час виконання ПЗ завдяки можливості оптимізації структури та паралельності операцій.

Перспективними напрямами досліджень АІО є створення системи числових показників складності структур інтегральних об'єктів, формалізація операцій над конфігурацією та дослідження ізомерних об'єктів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] С.В. Бевз, А.Г. Гречко, та С.Я. Вишневський, "Перспективи впровадження шаблонів проектування у розробку програмного забезпечення", *Вісник Хмельницького національного університету,* №4, с.208-210, 2013.

[2] А.Е. Дорошенко, та В.А. Иовчев, "Средства проектирования объектно-ориентированных программ на основе алгебры алгоритмики", *Проблеми програмування*, № 2-3. с.241-250, 2012.

[3] О.Є. Коваленко, "Стандартизація формального опису системної архітектури ситуаційних центрів" на *X науково-практичній конференції Системи підтримки прийняття рішень*, Київ, 2015, с.111-114.

[4] В. М. Медведєва, "Інтеграція засобів аспекто-орієнтованого підходу у об'єктно-орієнтовану мову програмування", *Восточно-Европейский журнал передовых технологий*, № 2 (2), с.19-28, 2016.

[5] М.Т. Фісун, "Модель програмного забезпечення на основі інтеграції модульної, пошарової та об'єктно-орієнтованої архітектур", *Наукові праці Чорноморського державного університету імені Петра Могили*, том 90, № 77, с. 205-215, 2008.

[6] В.В. Соколов,"Технологія програмування активних динамічних сполук об'єктів" на *V науково-технічній конференції Пріоритетні напрямки розвитку телекомунікаційних систем та мереж спеціального призначення,* Київ, 2010, с. 232.

[7] В.В. Соколов. "Застосування функціональної та реляційної моделей в об'єктно-орієнтованому програмуванні", *Information Technology and Security,* т.5, №1, с. 54-63, January-June 2017.

Стаття надійшла до редакції 10 вересня 2017 року.

REFERENCE

[1] S.V. Bevz, A.G. Grechko, and S.Y. Vyshnevs'kyi, "Study of implementation design patterns in software development", *Proc. Vinnytsya national technical university*, No 4, pp. 208-210, 2013.

[2] A.E. Doroshenko, and V.A. Iovchev, "Tools for designing object-oriented programs based on algebra of algorithms", *Problems of programming*, No 2-3. pp. 241-250, 2012.

[3] O.Y. Kovalenko, "Standardization of the formal description of the system architecture of situational centers", in *Proc. XI Scientific and Practical Conference of Decision Support Systems*, Kyiv, 2015, pp. 111-114.

[4] V.M. Medvedeva, "Integration of the means of an aspect-oriented approach in object-oriented programming language", *East European Magazine of Advanced Technology*, No 2 (2), pp.19-28, 2016.

[5] M.T. Fisun, "The software model based on the integration of modular, layered, and object-oriented architectures", in *Proc. Black Sea State University named after Petro Mohyla*, vol. 90, No 77, pp. 205-215, 2008.

[6] V. Sokolov, "Programming technology of active dynamic connections ofobjects" in *Proc. V scientific conference Priority directions of development of telecommunication systems and networks for special purposes*, Kyiv, 2010, p. 232.

[7] V. Sokolov, "Application of functional and relational models in object-oriented programming", *Information Technology and Security*, Vol. 5, Iss. 1, pp. 54-63, January-June 2017.

ВЛАДИМИР СОКОЛОВ

АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ ИНТЕГРАЛЬНЫХ ОБЪЕКТОВ

В работе представлена архитектура программного обеспечения, состоящая из объектов, способных к непосредственному взаимодействию путем образования активных динамических соединений, которые могут быть атомарными или интегральными объектами. Определено, что любое соединение объектов, как и атомы, имеет валентность – способность к образованию соединений, и может быть инкапсулировано в класс соединения. Сформулированы принципы, на которых базируется предложеная архитектура, такие как наличие атомарного базиса архитектуры, инкапсуляция коннекторов внутри компонентов, интеграция как основной принцип построения иерархии компонентов, запрет наследования классов компонентов на атомарном и высших уровнях, дескриптивность структур компонентов. Показано, что образование новых производных классов в виде соединений атомов можно считать интеграцией, степень которой зависит от уровня интеграции составляющих объектов. Иерархия классов на основе интеграции без наследования позволяет использовать унифицированный механизм как создания классов, так и взаимодействия объектов. В результате исследований разработан единый механизм построения интегральных объектов за счет дескриптора, который содержит полное описание структуры объекта, что позволяет использовать универсальную функцию синтеза соединений всех интегральных объектов. Обосновано, что решение любой задачи можно рассматривать как процесс синтеза дескриптора интегрального объекта. Рассмотрены вопросы статического и динамического поведения интегральных объектов, разработан двухэтапный подход к созданию интегрального объекта путем образования начального прото-объекта с последующим построением полного объекта, что дает возможность отложенного и частичного создания интегрального объекта при его использовании в программе. Определено, что при выполнении программы структура интегрального объекта функционально эквивалентна полной дезинтеграции объекта до атомарного уровня, что позволяет явно преобразовывать интегральные объекты в атомарные соединения. Рассмотрены изомерные классы объектов, которые могут изменять структуру соединения во время выполнения, оставляя неизменной внешний интерфейс. В качестве описания архитектуры выбрано графическую форму, формулу соединения, которая отражает

состав интегрального объекта и степень интеграции, атомарную формулу соединения, которая отражает реальный состав интегрального объекта из атомов, и структурную формулу, отражающую порядок соединения объектов в соединении. Показано, что для такой архитектуры можно применять операции над конфигурацией как статически, так и динамически.

Ключевые слова: архитектура программного обеспечения, интегральные объекты, валентность объектов, формула соединения, изомерные классы.

VOLODYMYR SOKOLOV

ARCHITECTURE OF SOFTWARE BASED ON INTEGRATED OBJECTS

The paper presents an architecture of software consisting of objects capable of direct interaction by the formation of active dynamic connections, which can be atomic or integral objects. It has been determined that any combination of objects, like atoms, has valency- the ability to form connections, and can be encapsulated into a class of connection. The principles on which the proposed architecture is based are formulated, such as the existence of an atomic basis on architecture, the encapsulation of connectors in components, integration as the basic principle of constructing a component hierarchy, the prohibition of inheritance of component classes at the atomic and higher levels, and the descriptiveness of component structures. It is shown that the formation of new derivative classes in the form of atomic connections can be considered an integration, whose degree depends on the level of integration of the constituent objects. The hierarchy of classes on the basis of integration without inheritance allows using a unified mechanism for both class creation and interaction of objects. As a result of the research, a universal mechanism for constructing integrated objects was developed through a descriptor that contains a complete description of the structure of the object, that allows to use the universal function of synthesizing the connections of all integral objects. It is substantiated that the resolving of any task can be considered as a process of synthesis of a descriptor of an integrated object. The questions of static and dynamic behavior of integrated objects are considered, a two-stage approach to the creation of an integral object was developed by creating an initial proto-object with the subsequent construction of a complete object, which enables the delayed and partial creation of an integrated object when used in the program. It is determined that during program execution the structure of an integrated object is functionally equivalent to the complete disintegrated of an object to an atomic level, which allows to explicitly transform integral objects into atomic connections. Isomeric classes of objects are considered which can change the structure of the connection at run time, leaving the external interface unchanged. As a description of the architecture, a graphic form, a formula of connection that reflects the composition of the integral object and the degree of integration, an atomic formula of connection that reflects the actual structure of the integrated object from the atoms, and a structural formula that reflects the order of connecting objects in the connection are selected. It is shown that for such architecture it is possible to apply operations over configuration both statically and dynamically.

Keywords: software architecture, integrated objects, valence of objects, formula of connection, isomer classes.

Володимир Володимирович Соколов, кандидат технічних наук, доцент кафедри кібербезпеки та застосування автоматизованих інформаційних систем та технологій, Інститут спеціального зв'язку та захисту інформації національного технічного університету "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна.

E-mail: vsokolov@i.ua.

Владимир Владимирович Соколов, кандидат технических наук, доцент, доцент кафедры кибербезопасности и применения автоматизированных информационных систем и технологий, Институт специальной связи и защиты информации Национального технического университета "Киевский политехнический институт имени Игоря Сикорского", Киев, Украина.

Volodymyr Sokolov, candidate of technical sciences, associate professor, associate professor at the cybersecurity and application of automated information systems and technologies academic department, Institute of special communication and information protection National technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

УДК 004.942::510.22

ГРИГОРІЙ КРАВЦОВ,
ОЛЕКСАНДР ДОЛГОРУКОВ,
ВОЛОДИМИР КОШЕЛЬ

ТЕОРЕТИКО-МНОЖИНИЙ ПІДХІД ДО ВИЗНАЧЕННЯ СУТНОСТІ ПОНЯТЬ "КЛАСИФІКАЦІЯ", "ТАКСОНОМІЯ" Й "ОНТОЛОГІЯ"

Проаналізовано визначення поняття "класифікація" і "таксономія". Встановлено відображення класифікацією системи мереологічних і таксономічних поділів. Мереологічний поділ орієнтований на членування поняття про предмет за відношенням "ціле-частина". Тоді як при таксономічному поділі між поняттям і членами поділу повинно існувати відношення "рід-вид". Водночас виявлено синонімічність використання понять "класифікація" і "таксономія". Це призводить до тлумачення поняття "таксономія" як синоніму до "класифікація". Така некоректність обумовлена тим, що "таксономія" у тлумаченні ієрархічної структури є одним з різновидів класифікації як утворення зі сукупності таксономічних поділів. Тому доцільно вважати множину "таксономія" підмножиною множини "класифікація". Крім цього, виокремлено та проаналізовано визначення поняття "онтологія". Акцентовано увагу на використанні формальних онтологій при побудові семантичної глобальної мережі. За основу такого тлумачення взято їх відображення множиною визначень. Таке відображення утворює таксономію з класів і підкласів, а також множину взаємозв'язків між ними. При використанні онтологій доцільно вказувати на її вид. Оскільки просту онтологію можна тлумачити як таксономію. Тоді використання складної онтології обумовлене встановленням і декларуванням її можливостей. З огляду на це, вперше надано логіко-математичний доказ співвідношення понять "таксономія", "класифікація" та "онтологія". Згідно з результатами досліджень між онтологіями, таксономіями і класифікаціями існує наступний зв'язок з точки зору теорії множин: множина таксономій є підмножиною класифікацій, множина класифікацій є підмножиною онтологій. Отримані результати є новими для науки та суперечать загально існуючим поглядам. Однак, застосування наукового методу на основі теорії множин підтверджує коректність результатів досліджень.

Ключові слова: класифікація, таксономія, онтологія, множина, відношення, мереологічний поділ, таксономічний поділ, теоретико-множинний підхід.

Вступ. Відповідно до [1]: «Класифікація – спосіб упорядкування, структурування деякої множини об'єктів, розбиття її на певні підмножини шляхом артикулювання, виділення певної ознаки об'єктів вихідної множини як підстави їх структурування за цією ознакою. Такого роду ознака називається підставою класифікації і повинна цілком визначатися. Класифікація множини об'єктів є однією з первинних і, разом з тим, фундаментальних форм як емпіричного, так і теоретичного пізнання. Відомими прикладами класифікацій в науці є класифікації видів тваринного та рослинного світів (наприклад, Ліннея, Бюффона, Ламарка); соціального світу (наприклад, К. Маркса), духовного світу (наприклад, І. Канта, Г. Гегеля).