УДК 519.85:658.846

С.В. БЕВЗ, А.Г. ГРЕЧКО, С.Я. ВИШНЕВСЬКИЙ
Вінницький національний технічний університет

ПЕРСПЕКТИВИ ВПРОВАДЖЕННЯ ШАБЛОНІВ ПРОЕКТУВАННЯ У РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Досліджено можливості впровадження шаблонів проектування у розробку програмного забезпечення, їхній вплив на програмні продукти, визначено переваги та недоліки їхнього використання. Шаблони проектування дозволяють створити масштабовану, динамічну, продуману структуру проекту.

Ключові слова: шаблони проектування, програмні продукти, структурні шаблони проектування, породжуючі шаблони проектування, поведінкові шаблони проектування.

S.V. BEVZ, A.G. GRECHKO, S. YA. VYSHNEVS'KYI
Vinnytsya national technical university

STUDY OF IMPLEMENTATION DESIGN PATTERNS IN SOFTWARE DEVELOPMENT

Implementation of design patterns in to the software development, their impact on software and advantages and disadvantages in using it is investigated. Design patterns can create scalable, dynamic, assertive project structure. Design patterns are one of the most important aspects of system architecture, they are used to create dynamic systems, that can be used in many different situations, that can dynamically change some components, without changing all the system. Design patterns make it easier to reuse successful designs and architectures. Expressing proven techniques as design patterns makes them more accessible to developers of new systems. Design patterns help you choose design alternatives that make a system reusable and avoid alternatives that compromise reusability. Design patterns can even improve the documentation and maintenance of existing systems by furnishing an explicit specification of class and object interactions and their underlying intent.

Keywords: design patterns, software, structural design patterns, generating design patterns, behavioural design patterns.

Вступ

Сучасні системи автоматизованого створення програмного забезпечення орієнтовані на активне використання шаблонів проектування, призначених для автоматизації рутинних процесів розробки програмних продуктів. Їх використання дозволяє впроваджувати принципи абстрактних технологій в процеси розробки узагальнених схем реалізації алгоритмів з подальшою їх деталізацією. Запропонована Крістофером Олександром мова патернів для реалізації архітектурного проектування знайшла своє широке відображення в об'єктно-орієнтованому програмуванні, що дозволяє накопичення та активне використання сучасних глосаріїв шаблонів проектування [1]. Тому актуальною є задача дослідження особливостей сучасних програмних шаблонів та розробка рекомендацій щодо перспектив їх ефективного використання в процесі створення власних програмних продуктів.

Метою роботи є підвищення ефективності створення програмних застосувань шляхом активного впровадження патерних технологій.

Об'єктом дослідження постають процеси автоматизованого проектування програмних продуктів.

Під предметом дослідження розуміємо шаблони проектування програмного забезпечення.

Головною задачею вбачаємо формування системи критеріїв оцінювання можливостей використання шаблонів проектування та визначення умов їх ефективного застосування у процесі створення сучасних програмних додатків.

Визначення моделі шаблонів проектування

У системах створення програмного забезпечення під шаблоном проектування (або патерном) розуміють відтворювану архітектурну конструкцію, що являє собою засіб вирішення задачі проектування в межах деякого часто виникаючого контексту [2]. Система базових шаблонів акумулює відкриті програмні зразки, які не можуть бути прямо перетворені у вихідний код програми. Шаблони в об'єктно-орієнтованому програмуванні презентують наявні відносини і взаємодії між класами та об'єктами [3]. Ідіоми є вузькоспеціалізованими засобами програмного проектування, орієнтованими під конкретну платформу, що обмежує їх загальне використання. Високорівневі архітектурні шаблони носять універсальний характер. Алгоритми за своєю природою також є шаблонами, але не проектування, а обчислення, оскільки розв'язують логічні та обчислювальні завдання поставленої задачі. Патерни проектування покликані автоматизувати процеси повторного використання вдалих проектних і архітектурних рішень.

Загальну стркуктуру шаблону проектування подамо у вигляді моделі (рис.1). Таким чином, патерн складається з чотирьох базових елементів

- імені, яке узагальнює проблеми проектування, наявні шляхи пошуку розв'язків та їх наслідки; привласнення патернам імен забезпечує реалізацію принципів абстрактного програмування та складання каталогу розроблених шаблонів;
- завдання, що визначає коло охоплених задач і включає перелік умов застосування шаблонів;
- рішення, котре включає опис елементів дизайну, функцій шаблонів та відносин між ними;
- результатів, що визначають наслідки застосування шаблону, переваги, недоліки та перспективи використання патерну в процесі розробки власного програмного продукту.

Аналіз перспектив використання шаблонів проектування

Наявні шаблони проектування можна класифікувати на структурні, породжуючі та поведінкові.

Структурні шаблони проектування акумулюють складні структури, забезпечують їх подальшу деталізацію, описують інтерфейси базових об'єктів та їх реалізацію, сприяють оптимізації процесу програмування. Серед структурних шаблонів розглянемо шаблон проектування Фасад, який надає уніфікований інтерфейс високого рівня та спрощує подальше використання підсистем, агрегує класи, що реалізують функціональність системи, та забезпечує низькорівневий доступ до класів. Приклад використання шаблону Фасад наведено на рис. 2.

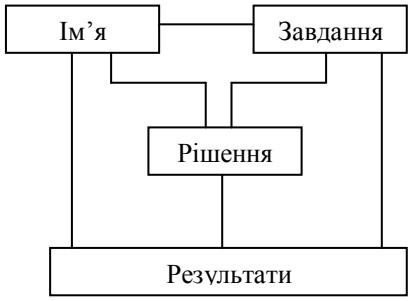

Рис. 1. Модель шаблону проектування

До головних функцій шаблону Фасад за [2] віднесемо:

- визначення інтерфейсу підсистем;
- делегування запитів клієнтів обраним об'єктам підсистеми;
- створення нових методів, які об'єднують виклики системи і надають логіку їх опрацювання;
- приховування процесів обробки підсистеми;
- оптимізацію параметрів робочих методів шляхом підстановки розрахованих значень.

Підсистеми шаблону реалізують закритий для зовнішніх компонентів функціонал та опрацьовують запити клієнтів. Одна підсистема може мати довільну кількість Фасадів. Клієнт генерує запити Фасаду у прихованому режимі.

Шаблон проектування Фасад рекомендовано використовувати в процесі реалізації архітектурної компоновки програми для забезпечення клієнт-серверної взаємодії при об'єднанні підсистем сервера в один Фасад для створення зручного клієнтського інтерфейсу. Тоді весь функціонал складної системи об'єднується в єдиний компонент. Крім того, Фасад зручно застосовувати для реалізації системи безпеки, що дозволяє уніфікувати набір дозволених операції клієнту та регулювати доступ клієнта до базового функціоналу підсистем.

Породжуючі шаблони проектування абстрагують процеси ініціювання, забезпечують незалежність систем від способу створення, композиції та ідентифікації об'єктів [2]. Породжені шаблоном класи реалізують принципи спадкування та інкапсуляції функцій. Для прикладу розглянемо шаблон проектування Одинак (Singleton), який відноситься до складу твірних шаблонів. Даний шаблон гарантує для класу наявність одного екземпляру і забезпечує глобальну точку доступу до нього [1]. Створений екземпляр розширюється шляхом успадкування функцій та дозволяє клієнтам працювати зі своїм функціоналом, не змінюючи власний код. Діаграму класів шаблону проектування Одинак наведено на рис.3.

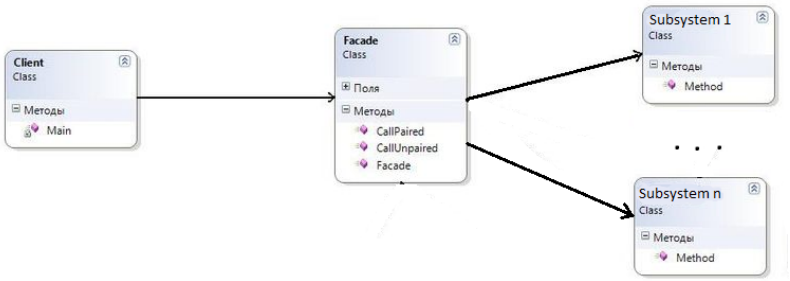

Рис. 2. Діаграма класів шаблону проектування Фасад

Шаблон Одинак утримує два поля: UniqueInstance та SingletonData. UniqueInstance реалізує посилання об'єкта самого на себе та формує єдине посилання системи на Одинака. Головний метод шаблону Instance() повертає уже створене посилання на об'єкт Одинака, чим забезпечує одноразове створення об'єкта. Таким чином, Одинак реалізує посилання усіх об'єктів на єдиний екземпляр класу, що дозволяє скоротити процеси копіювання запитів із забезпеченням єдиного визначеного функціоналу.

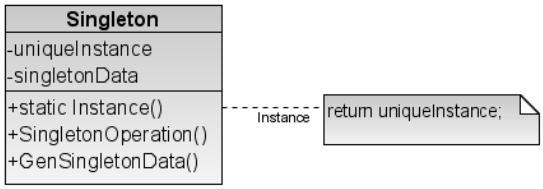

Рис. 3. Діаграма класів шаблону проектування Одинак

Поведінкові шаблони проектування визначають алгоритми та способи реалізації взаємодії різних об'єктів і класів [3]. Їх використовують з метою забезпечення гнучкості програмного застосування. Як приклад розглянемо поведінковий шаблон Стратегія (Strategy), призначений для ідентифікації сімейства алгоритмів, інкапсуляції їх властивостей та забезпечення можливості їх взаємозамінності. Головним завданням шаблону є вибір оптимального алгоритму роботи об'єкта за типом клієнта чи типом оброблюваних даних. Задачі, для яких рекомендовано використання шаблону Стратегія, орієнтовані на здійснення вибору алгоритмів поведінки кожного екземпляру

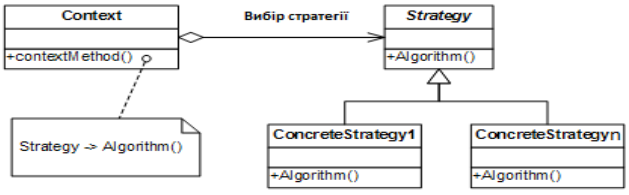

Рис. 4. Діаграма класів шаблону проектування Стратегія

класу та об’єкту на стадії виконання. Інтерфейс шаблону дозволяє інкапсулювати в собі набір алгоритмів і опрацьовувати їх у закритому для клієнтів режимі. Діаграму класів шаблону Стратегія показано на рис. 4.

Клас Context пересилає класу Strategy запит клієнта. Клас Strategy визначає порядок реалізації різних алгоритмів. Конкретні класи Concrete Strategy реалізують обрані алгоритми. Кожний похідний клас реалізує один варіант роботи алгоритму. Вибір класом Context певних об’єктів Strategy забезпечує динамічну зміну алгоритмів у процесі їх реалізації. Наслідком використання шаблону є можливість відмови від застосування перемикачів та і/або умовних операторів. Проте включення шаблону не звільняє клієнта від необхідності самостійно розібратися в особливостях реалізації можливих стратегій для здійснення вибору найбільш вдалої з них. Результати аналізу перспектив використання розглянутих шаблонів зведені в табл. 1.

Таблиця 1

Перспективи використання шаблонів проектування

Назва шаблону	Умови для використання	Переваги	Недоліки
Фасад	Коли компоненти системи зв’язуються між собою, реалізуючи багатокомпонентну архітектуру; у клієнт-серверних програмах, коли операції складних підсистем сервера об’єднуються в один Фасад з метою спрощення клієнт-серверної взаємодії.	Складна система мінімізується в один об’єкт, який використовує клієнт. Шаблон надає більшу безпеку для самої системи.	Використання шаблону передбачає введення додаткового компонента до і так складної системи та створення його логіки, яка буде поєднувати усі операції системи.
Одинак	Коли потрібно забезпечити існування лише одного екземпляру об’єкта за весь час роботи програми; коли багатьом компонентам системи необхідний один і той самий об’єкт або ресурс.	Зберігаються вільні ресурси за рахунок створення лише одного об’єкта у всій системі. Простота підтримки потокобезпечності. Скорочення процесів копіювання запитів для створення єдиного функціоналу.	Велика увага приділяється створенню функціоналу, який забезпечує коректну роботу об’єкта.
Стратегія	Коли потрібно змінювати вибраний алгоритм незалежно від об'єктів-клієнтів, які його використовують під час роботи програми.	Можливість позбутися умовних операторів. Клієнт може обирати найбільш влучну стратегію залежно від вимог щодо швидкодії і пам'яті.	Збільшення кількості об'єктів. Високі вимоги до клієнта, який має знати особливості реалізації кожної стратегії для вибору найбільш вдалої.

Висновки

Використання шаблонів проектування дозволяє автоматизувати процеси створення вихідного коду програмного продукту та оптимізувати роботу програми. Проведений аналіз різних класів шаблонів дозволяє систематизувати особливості, переваги та недоліки їхнього використання та окреслити коло задач, для розв’язання яких перспективним є впровадження розглянутих шаблонів. Таким чином, використання структурних шаблонів проектування оптимізує роботу клієнт-серверних додатків та складних програмних застосування багатоланкової архітектури. Породжуючі шаблони дозволяють забезпечити незалежність системи від способу створення, композиції та подання об’єктів. Поведінкові шаблони проектування визначають сімейства алгоритмів та реалізують взаємодію між різними об’єктами і класами.

Література

1. Гамма З. Приемы объектно-ориентированного проектирования - Паттерны проектирования / З. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – М. : ПИТЕР, 2001. – 344с.

2. Самойленко О.А. Конструювання комплексної системи інформаційної підтримки управлінських рішень / О.А. Самойленко, В.С. Степашко // Збірник праць. – К. : МННЦ ІТС, 2009. – С. 211–219.

3. Гради Буч Объектно-ориентированное проектирование / Гради Буч – К. : Диалектика и М. : И.В.К., 1992.

References

1. Gamma Z. Pryyomy' ob"ektno orientirovanogo programirovaniya - Patterny' proektirovania / Z.Gamma, R. Helm, R. Johnson, J. Vlyssydes. – M. : St. Peterburg, 2001. – 344 s.[in Russian]

2. Samojlenko A.A. Konstruyuvannya kompleksnoi systemy' informacaijnoi pidtry'mky' y'pravlins'ky'h rishen' / A.A. Samojlenko, V.S . Stepashko // Proceedings. – Kyiv : MNNC ITS. In 2009. – C. 211 – 219. [in Ukrainian]

3. Gradi Butch, Ob"ektno - orientirovanogoe proektirovanie. – Kiev: Dialektika i N. : IVK, 1992.[in Russian].

Рецензія/Peer review : 21.5.2013 р. Надрукована/Printed :19.6.2013 р.
Рецензент: д.т.н. Перевозніков С.І.