Рефакторинг програмного забезпечення інформаційних систем при використанні породжувальних паттернів проектування

В. В. ЗАВГОРОДНІЙ, К. М. ЯЛОВА

Дніпродзержинський державний технічний університет

Приведено результати визначення ефективності використання породжувальних паттернів проектування при розробці програмного забезпечення веб-орієнтованих інформаційних систем на прикладі Інтернет-магазину. Встановлено якісний вплив застосування шаблонів шляхом визначення ряду метрик та характеристик якості програмного коду.

Представлены результаты определения эффективности использования порождающих паттернов проектирования при разработке программного обеспечения веб-ориентированных информационных систем на примере Интернет-магазина. Установлено качественное влияние использования шаблонов путем определения ряда метрик и характеристик качества программного кода.

Results of determination of use's efficiency of creational patterns are provided. A software development of the web-oriented information systems on the example of the E-commerce shop is chosen as domain of the article. High-quality influence of templates' use by determination of metrics and characteristics of program code's quality is set.

Вступ. Сучасні вимоги автоматизації бізнесу вимагають створення програмних продуктів (ПП) для кінцевого споживача у найкоротші строки, при цьому мало хто із замовників піклується про рівень якості отриманого кінцевого ПП. А між тим, ключовим фактором забезпечення ефективного застосування ПП є ретельне оцінювання та досягнення високих значень саме показників якості програмного забезпечення (ПЗ). Постійне підвищення складності функцій, реалізованих програмами в інформаційних системах (ІС), призводить до збільшення їх обсягу та трудомісткості створення. Відповідно до змін складності програм зростає кількість виявлених та залишених у них дефектів та помилок, що відображається на їх якості [1]. Багато зі стандартів оформлення та критеріїв якості програмного коду, які визначають специфічні для мов програмування згоди та правила, мають на меті полегшити майбутній супровід ПЗ, покращити читання коду і спростити відлагодження та оновлення програм. У багатьох об'єктно-орієнтованих системах можна використовувати існуючі зразки, шаблонні моделі, формалізовані описи ефективних типових рішень у проблемній області. Перші спроби систематизації найбільш вдалих проектних рішень в інформаційних технологіях (ІТ) були здійснені американськими науковцями Erich Gamma, Richard Helm, Ralph Johnson, John Ylissides та Martin Fowler. Результати їх роботи представлено в каталозі паттернів проектування та каталозі архітектурних шаблонів для розробки корпоративних додатків. На основі їх ідей та способів представлення мікро-архітектур шаблонів набули розвитку такі методи покращення програмного коду, як: рефакторинг, реінженірінг, оптимізація програмного коду, розробка ПЗ шляхом тестування і т.д. Питання ефективного застосування паттернів при проектуванні та реалізації ІС для різноманітних проблемних областей опрацьовують такі науковці, як: Міхєєва Т.І. (інтелектуальні транспортні системи), Мірютов А.А. (ефективна архітектура ІС), Норенков І.П. (підтримка прийняття рішень на основі паттернів проектування), Мутілін В.С. (тестові сценарії), Ніконенко А.А. (комп'ютерна лінгвістика) та інші. При цьому залишається не достатньо висвітленим питання оцінки ефективності використання паттернів проектування з точки зору рефакторингу програмного коду ІС.

Постановка завдання. Метою даної роботи є створення підходу застосування породжувальних паттернів проектування, що використовуються для рефакторингу програмного коду автоматизованих веб-орієнтованих ІС. Обґрунтування доцільності використання шаблонів проектування шляхом визначення значень метрик та характеристик якості програмного коду на прикладі ПЗ Інтернет-магазину, розробленого із використанням технології ASP.NET MVC.

Результати дослідження. При проектуванні складноорганізованої ІС необхідно виявити об'єкти предметної області, віднести їх до класів, визначити інтерфейси класів, ієрархію спадкування та схему композиції, встановити регламент відношення між класами. При цьому необхідно уникнути або, принаймні, звести до мінімуму необхідність перепроектування системи [2]. Оскільки сучасні ІС у своїй основі мають, як правило, стандартизоване апаратне забезпечення та операційну систему, а відрізняються програмним кодом [3], то дана робота буде акцентувати свою увагу на якісних характеристиках розроблюваного ПЗ. До методів покращення якості програмного коду належить рефакторинг – процес зміни внутрішньої структури програми, що не торкається її зовнішньої поведінки. Рефакторинг необхідно застосовувати постійно при розробці коду, при цьому він виконується для покрашення зрозумілості коду, оптимізації його структури, забезпечення можливості розвитку і підтримки програми. Одним із підходів рефакторингу можна вважати застосування паттернів проектування.

На сьогоднішній день в об'єктно-орієнтованому аналізі та проектуванні розроблено багато різноманітних паттернів, які можна класифікувати наступним чином: архітектурні паттерни (описують фундаментальні способи структурування програмних систем); паттерни проектування (дають змогу описати систему в термінах класів); паттерни аналізу (використовуються для організації процесу об'єктно-орієнтованого моделювання); паттерни тестування (описують загальні схеми процесу тестування програмних систем) та паттерни реалізації

(використовуються при написанні програмного коду на заданій мові програмування).

В даній роботі будуть розглядатися породжувальні паттерни проектування та визначатися можливість їх ефективного застосування при розробці ПЗ ІС.

Ідея створення паттернів проектування базувалася на припущені, що досвід розв'язання інформаційних проблем – безцінний, і будь-яку задачу інформаційного характеру вже було розв'язано, або існує спосіб приведення поточної задачі до розв'язаної раніше. Рішення проблеми систематизації накопиченого досвіду в об'єктно-орієнтованому проектуванні, а також його надання широкому загалу розробників і взяли на себе паттерни проектування. Паттерн проектування – це багаторазово застосована архітектурна конструкція, що описує взаємодію класів і об'єктів та надає розв'язання загальної проблеми проектування в рамках конкретного контексту [4]. Загальні переваги використання паттернів проектування наступні:

- паттерни дозволяють підсумувати досвід експертів і зробити його доступним рядовим розробникам;
- імена паттернів утворюють свого роду словник, який дозволяє розробникам краще розуміти один одного;
- якщо в документації системи зазначено, які паттерни в ній використовуються, це дозволяє архітекторам швидше зрозуміти систему;
- паттерни спрощують реструктуризацію системи незалежно від того, чи використовувалися паттерни при її проектуванні;
- правильно вибрані паттерни проектування дозволяють зробити програмну систему більш гнучкою, її легше підтримувати і модифікувати, а код такої системи у більшій мірі відповідає концепції повторного використання.

Паттерни проектування поділяються на: породжувальні, структурні та паттерни поведінки. Породжувальні паттерни – це паттерни проектування, які призначені для створення об'єктів, дозволяючи системі залишатися незалежною як від самого процесу породження, так і від типів об'єктів що породжуються. Наведемо результати використання породжувальних паттернів проектування при розробці ПЗ Інтренет-магазину із використанням технології ASP.NET MVC.

Наведені приклади застосування виконано на прикладі магазину побутової техніки. Необхідно зауважити, що оскільки паттерн проектування – це шаблон архітектурного рішення, то контент сайту не впливає на кінцевий результат роботи. Паттерн Abstract Factory (Абстрактна фабрика) – це породжувальний паттерн, що надає інтерфейс для створення сімейств класів, пов'язаних між собою.

В рамках архітектури ПЗ для Інтернет-магазину цей паттерн найдоцільніше використати для сімейств класів, що відтворюють взаємозв'язок між торгівельними марками та побутовою технікою зазначеної марки. В рамках паттерну Abstract Factory визначаються наступні складові частини:

- AbstractFactory (HouseHoldFactory) – абстрактна фабрика, що задає інтерфейс для операцій створення об'єктів-продуктів та містить сигнатуру методів, призначених для створення об'єктів-продуктів;
- ConcreteFactory (SamsungFactory, PhilipsFactory, LGFactory, ToshibaFactory і т.д.) – конкретні фабрики, які реалізують операції створення об'єктів-продуктів. Кількість конкретних фабрик залежить від загальної кількості торгівельних марок у магазині. Містить реалізацію методів, що створюють екземпляри визначеного продукту;
- AbstractProduct (TVset, Refrigerator і т.д.) – клас, що задає інтерфейс для типів об'єктів-продуктів. Кількість абстрактних продуктів залежить від загальної кількості видів товарів у магазині;
- ConcreteProduct (SumTV, SumR, ToshTV, ToshR, LGTV і т.д.) – класи конкретних продуктів, які визначають об'єкти-продукти та реалізують інтерфейс Abstract Factory. Кількість конкретних продуктів залежить від загальної кількості товарів у магазині всіх видів та торгівельних марок. Для наочності представлення мікро-архітектури (набір програмних компонент і їх взаємозв'язків) ПЗ системи на рис. 1 наведено фрагмент діаграми класів.

Зазвичай, в ході виконання програми створюється один екземпляр класу ConcreteFactory. При необхідності створити новий екземпляр зазначеного класу товарів відповідний Контролер звернеться до спеціального об'єкту-фабрики із запитом створення необхідного екземпляру класу. З метою визначення ефективності використання шаблонів проектування при розробці ПЗ ІС будемо використовувати характеристики якості програмного коду, які можуть бути кількісними та ознаковими [5].

До ряду кількісних характеристик відносяться певні чисельні величини – метрики, що дають змогу отримати чисельне значення властивостей ПЗ. Найбільш широкого застосування набули наступні метрики інформаційної складності: метрика Холстеда; цикломатичне число Маккейба; NORM (Number of Remote Methods) – кількість віддалених методів, які викликаються; RFC (Response of Class) – кількість методів, які можуть бути викликані екземпляром заданого класу; базові метрики *LOC* (Lines of Code) – кількість рядків коду та *NOC* (Number of classes) – кількість класів і інші. В рамках метрики Холстеда використовуються чотири основоположні поняття: кількість унікальних операторів – *nO*, кількість унікальних операндів – *nOprnd*, загальна кількість операторів – *NO*, загальна кількість операндів – *NOprnd*. На їх основі визначаються:

- довжина програми: $N = NO + NOprnd$, (1)
- словник програми: $n = nO + nOprnd$, (2)
- складність програми: $D = \left(\frac{nO}{2}\right)\left(\frac{NOprnd}{nOprnd}\right)$, (3)
- обсяг програми: $V = N \log_2 n$, (4)
- трудомісткість: $E = DV$. (5)

До ознакових характеристик якості програмного коду відносяться: гнучкість – можливість внесення змін до коду без особливих зусиль та ризиків; прозорість – властивість, що характеризує легкість розуміння коду; крихкість – властивість програми пошкоджуватися в багатьох місцях при внесені однієї зміни та структурованість – ступінь логічного розбивання на керовані блоки і інше. Для визначення ефективності застосування паттерну Abstract Factory було здійснено розрахунок метрик якості коду при традиційній об'єктно-орієнтованій реалізації та для реалізації із вико-

ристанням зазначеного паттерна. Результати розрахунків стосовно блоку коду (рис. 1) наведено в таблиці 1.

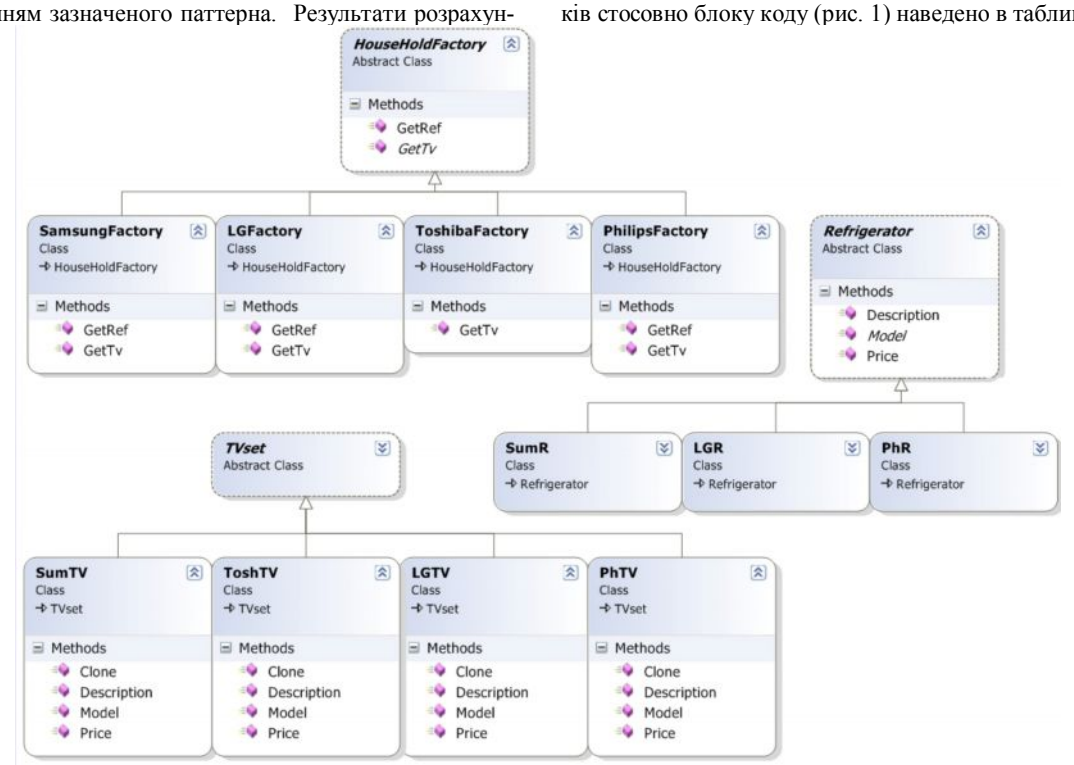

Рис. 1. Фрагмент діаграми класів із використанням паттерну bstractFactory

Позитивний вплив на значення метрик якості коду із застосування паттерну проектування AbstractFactory вбачається у видаленні дубльованого фрагменту коду, що містить складові оператору розгалуження (C#):

```
switch type
{
case type.equals("SamsungTV") tv= new SamsungTV ();
case type.equals("ToshibaTV") tv = new TosibaTV ();
case type.equals("LGTV") tv = new LGTV();
...
}
```

та заміну його на реалізацію паттерну.

Таблиця 1. Порівняльні значення метрик якості програмного коду

Назва метрики якості	Значення для об'єктно-орієнтованої реалізації	Значення для реалізації із використанням паттерна
LOC	328	285
NOC	9	14
довжина програми N	307	273
словник програми n	95	86
складність програми D	38,77	35,48
обсяг програми V	2016,99	1754,3
трудомісткість E	78198,7	62242,56

Окрім визначених за допомогою (1)-(5) кількісних характеристик якості програмного коду, значення яких вказує на ефективність застосування паттерна Abstract Factory для створення об'єктів сімейств класів, необхідно перелічити ознакові характеристики якості коду, які досягаються шляхом застосування наведеного архітектурного рішення:

- ізоляція конкретних класів, що дозволяє підвищити гнучкість системи шляхом інкапсулювання відповідальності за створення класів і сам процес їх створення та ізоляції класів-клієнтів від деталей реалізації класів;

- спрощення заміни сімейств продуктів, при якій класи конкретних фабрик виникають один раз при інстанціруванні, що запобігає крихкості системи шляхом полегшення заміни конкретної фабрики, яка використовується у додатку, який може змінити конфігурацію продуктів просто підставивши нову конкретну фабрику. Оскільки абстрактна фабрика створює все сімейство продуктів, то і замінює відразу все сімейство;

- використання шаблонів проектування апріорі позитивно впливає на структурованість програмного коду та відтворює механізм уніфікованої архітектури, зрозумілої для широкого загалу розробників ПЗ.

Ще одним прикладом ефективного застосування породжувальних паттернів при розробці ПЗ веборієнтованих ІС є використання паттерна Singleton (Одинак). Позитивний вплив використання цього паттерну вбачається в можливості забезпечення функціональності при реалізації підсистеми авторизації та аутентифікації користувача. Для доступу до створення замовлення користувачу необхідно зареєструватися та пройти аутентифікацію, вказавши своє ім'я та пароль. При цьому в будь-який момент часу інформаційній системі повинні бути відомі персональні дані всіх користувачів, а саме: ім'я, пароль, e-mail, контрольне питання та відповідь на нього. Тобто необхідно мати програмний об'єкт, який є доступним із будь-якої точки системи та може існувати лише в єдиному екземплярі. Найбільш

вдале рішення цього завдання полягає у наділенні класу можливостей:

- контролювання того, що у нього існує лише один екземпляр;

- реалізування механізмів заборони створення додаткових екземплярів та перехвату запитів на створення нових об'єктів [6]. Саме ця ідея лежить в основі паттерну Singleton – породжувальний паттерн, який створює об'єкт класу, надає доступ до нього і гарантує, що він буде єдиним [7]. В момент аутентифікації користувача відповідний Контролер стає класом-клієнтом до класу Account, сигнатура якого наведена на рис. 2. Це запобігає події, під час якої в системі реєструються два або більше користувачів із однаковими реєстраційними даними.

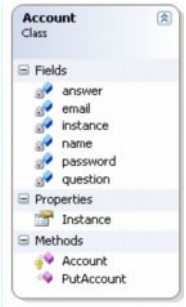

Рис. 2. Сигнатура класу Account, що реалізує паттерн Singleton

У рамках паттерну Singleton необхідно створити клас Singleton (Account), який несе відповідальність за створення власного унікального екземпляру, забороняючи класам-клієнтам власними силами створювати екземпляри, шляхом закриття доступу до конструктора (модифікатор доступу визначається як private або protected). Програмне вбудовування мікро-архітектури паттерну Singleton до ПЗ ІС складається з наступних кроків:

- створення класу для реалізації системи аутентифікації користувача, для заборони створення екземплярів класу зовнішніми класами-клієнтами, його конструктор оголошується як захищений – protected;

- члени класу (окрім змінної, в якій зберігається посилання на єдиний об'єкт і методу надання доступу до нього) повинні бути нестатичними, що дозволить, при необхідності, редагувати поведінку класу, створивши класи-нащадки.

- створення статичного екземпляру класу Account (C#):

private readonly static Account instance = new Account(name, password, email, question, answer);

- реалізація методу-властивості Instance, для забезпчення доступу до єдиного екземпляру класу.

Оскільки передбачається використання розробленої ІС в багатопоточному середовищі, то виникає необхідність враховувати той факт, що подія створення єдиного екземпляру може виконуватися одночасно паралельними потоками і може призвести до порушення цілісності даних системи. Для виходу з цієї ситуації код, відповідальний за створення об'єкту, було розміщено у критичну секцію lock, забезпечивши тим самим доступ до нього лише одному потоку. Що стосується оцінки результатів ефективності використання паттерну Singleton при реалізації ПЗ ІС, то необхідно зауважити, що ефект від його впровадження слабо відображається метриками якості коду, оскільки кількісно він несуттєво впливає на програмний код системи. Але, що стосується ознакових характеристик якості, то тут можна навести наступне:

- застосування паттерну Singleton дозволяє запобігти використанню глобальних змінних, тобто запобігає засміченню простору імен для зберігання унікальних екземплярів класів;

- для забезпечення гнучкості системи застосовані механізми можливого породження підкласів, а для класів-клієнтів залишається можливість працювати з вже розширеним екземпляром, не вносячи ніяких змін у свій код.

Висновки

В роботі представлено спосіб вбудування мікроархітктектури породжувальних паттернів проектування AbstractFactory та Singleton до програмного коду веборієнтованої інформаційної системи. З метою визначення ефективності використання паттерну проектування AbstractFactory представлено розрахунки метрик якості коду для традиційної об'єктно-орієнтованої реалізації та реалізації із застосуванням паттерну. Визначено та описано переваги та недоліки використання шаблонів. Визначено, що представлений підхід використання породжувальних паттернів позитивно впливає на гнучкість та прозорість системи, здатність до повторного використання, зменшення інформаційної складності програмного забезпечення, що в цілому покращує надійність та якість програмного забезпечення. Обґрунтовано ефективність застосування породжувальних паттернів проектування при проведені рефакторингу програмного коду інформаційної системи.

ЛІТЕРАТУРА

1. Говорущенко Т. О. Визначення ефективності метрик якості на етапі проектування програмного забезпечення / Т. О. Говорущенко, Є. В. Питлик // Вісник Хмельницького національного університету. 2012. — №2. — С. 149—155.
2. Михеева Т. И. Паттерновое проектирование интеллектуальных транспортных систем / Т. И. Михеева, О. К. Головнин, А. А. Федосеев // Современные проблемы науки и образования. 2012. — № 6. Режим доступу: http://www.science-education.ru/106-7967.
3. Колдовский В. В. Актуальні проблеми оцінки якості в управлінні програмними проектами з автоматизації економіки / В. В. Колдовский // Ефективна економіка. 2011. — № 5. Режим доступу: http://www.economy.nayka.com.ua/index.php?operation =1&iid=547.
4. Design Patterns: Elements of Reusable Object-Oriented Software / Gamma, Erich; Richard Helm, Ralph Johnson, John Vlissides. Publisher: Addison-Wesley. 1995. — 395. ISBN 0-201-63361-2.
5. Никоненко А. А. Использовние паттернов проектирования в компьютерной лингвистике. Порождающие паттерны // Українська лінгвістична лабораторія. 2010. Режим доступу: http://lingvoworks.org.ua/index.php?option=com_

ontent&view=article&id=62:2010-10-14-13-06-06&catid=2:misc&Itemid=3.
6. James W. Cooper. Introduction to design patterns in C#. IBM TJ Watson Research Center. 2002. — 424 p.
7. Design Patterns in C# / Jean Paul V.A. Publisher: Addison-Wesley. 2012. — 95 p.

пост.01.04.14