УДК 004.4
В.М. Юзьвак
Тернопільський національний технічний університет імені Івана Пулюя, Україна

АНТИПАТЕРНИ ПРИ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

V.M. Iuzvak

SOFTWARE DEVELOPMENT ANTIPATTERNS

У літературі більшість уваги дослідників присвячено саме патернам проектування, тобто, типовим способам вирішення поширених проблем при проектуванні програмного забезпечення. Водночас, не менш важливе значення має дослідження і антипатернів проектування – типових прийомів, які викликають негативні наслідки для підтримки життєздатності, внесення змін та доповнення програмного забезпечення. Антипатерни в процесі створення інформаційних систем можна категоризувати на три групи в залежності від об'єкта діяльності на який спрямована їх дія: 1) антипатерни в розробці програмного забезпечення (процес усунення їх недоліків називається рефакторингом програмного коду, мають локальні наслідки для окремих компонентів системи), 2) антипатерни архітектури інформаційних систем (знаходяться на більш вищому рівні – їх наслідки охоплюють усю систему), 3) антипатерни організаційного процесу розробки інформаційної системи – пов'язані з комунікацією між людьми в процесі створення інформаційних систем. Зосередимо свою увагу на першій групі антипатернів.

До найбільш поширених різновидів антипатернів програмування слід віднести: 1) клас Бога або Блоб (в честь однойменного кіномонстра, який постійно збільшувався у розмірах та пожирав усе на своєму шляху) – невиправдане розростання певного класу програми, що призводить до дизбалансу розподілу функціональностей між класами, коли виділяється один "ключовий" клас, а інші носять явно виражений допоміжний або штучний характер, 2) безперервне застарівання – пов'язане з жорстким використанням залежностей від технологій, які швидко розвиваються замість використання інтерфейсів, які дають гнучкість у заміні залежностей, 3) мертвий код, також відомий як потік лави (lava flow) – явище, коли незавершений (як правило, неналежним чином документований) на етапі розробки програмний компонент використовується у готовому продукті та як правило пов'язаний зі змінами у команді розробників, внаслідок чого функціонування і структура певних частин системи недокінця зрозумілі для інших розробників, 4) множинність точок зору при поділі на моделі, що призводить до неможливості фундаментального відділення інтерфейсів від реалізації деталей, 5) функціональна декомпозиція (вважається антипатерном в ООП), 6) розростання класів (також відомий як полтергейст чи циганський антипатерн) характеризується наявністю класів з дуже короткими життєвим циклом і функціоналом, який зводиться до ініціалізації інших класів чи передачі їм параметрів, 7) п'яте колесо (Boat Anchor) – ситуація, за якої частина коду або обладнання не приносить значної корисної дії для системи, хоча часто зумовлює значні ресурсні витрати, 8) золотий молоток – антипатерн, який полягає у поширенні певного стилю чи рішення на усі ситуації базуючись насамперед на уподобаннях розробника, а не особливостях ситуації, 9) глухий кут – виникає коли використовується компонент, який більше не підтримується виробником, 10) код спагеті – виникає при скупості сруктури коду, внаслідок чого складно розширити або внести зміну у програму, 11) неопрацювання відхилень від очікуваних вхідних даних, 12) мінне поле – запуск програми у виробничу стадію без належного тестування, 13) програмування вставками (Cut-and-Paste Programming) –

підхід за якого готові рішення копіюються повністю без належної адаптації, 14) сліпа розробка – підхід за якого виключається контакт представника з групи розробки програмного забезпечення з кінцевим користувачем системи, і як наслідок неврахування потреб останнього при розробці інформаційних систем [1]. Існують і інші антипатерни розробки програмного забезпечення як: 1) жорсткий код – вміщення припущень про налаштування середовища системи у її реалізації, 2) м'ягкий код – розміщення логіки системи всередині конфігураційних файлів, 3) код лазаньї – антипатерн за якого структура програми невиправдано складна, 4) магічні числа – використання незадокументованих та некоментованих констант, 5) послідовність циклів-розгалужень, 6) повторення ділянок коду та інші [2].

Слід відзначити, що до виявлення антипатернів необхідно проявити гнучкий і системний підхід, так як в одних ситуаціях певні прийоми будуть розглядатися як патерни, а в інших як антипатерни. Так, функціональна декомпозиція є патерном і хорошою практикою у функціональному програмуванні та антипатерном в об'єктно-орієнтованому програмуванні. Патерн «одинак» є дуже поширеним у програмуванні і дозволяє отримати глобальну точку доступу до класу, що гарантовано має лише один екземпляр. Водночас, оскільки вищенаведений клас крім виконання своїх безпосередніх обов'язків контролює ще й кількість своїх екземплярів, то такий клас порушує принцип єдиної відповідальності класів (single responsibility principle), що є складовою SOLID. Даний патерн також піддається критиці зі сторони противників глобальних станів у програмному коді (так як усі об'єкти використовують один екземпляр класу спроектованого за даним патерном, то важко прогнозувати некритичний стан такого екземпляра на момент його використання). Вищенаведене зумовлює те, що деякі програмісти відносять «одинака» до антипатернів. Розмежування патернів і антипатернів слід проводити в залежності від доцільності їх застосування в обраній парадигмі програмування, наслідків у витратах на обслуговування проекту та його модифікацію.

Для профілактики утворення антипатернів при програмуванні інформаційних систем необхідні дії як зі сторони розробників так і зі сторони організаторів процесу розробки. До перших відносять дотримання обраної концепції програмування (як об'єктно–орієнтований так і функціональний підхід можуть вирішити завдання та створити підтримуваний і життєздатний в тривалій перспективі продукт, але не їхня суміш), дотримання встановлених для обраної концепції програмування принципів, наприклад, для ООП – DRY, DIE, KISS, SOLID, YAGNI[3], постійний розвиток розробників і уникнення замкнутості на улюблених технологіях та підходах, належне документування та тестування програмного продукту. Стосовно організаційних заходів, то необхідно забезпечити умови при яких розробники будуть максимально довго супроводжувати проект і після їхнього виходу з команди не залишиться недокументованого функціоналу розробленого ними, налагодити донесення цінностей кінцевого споживача до розробників.

Література

1. William J. Brown, Raphael C. Malveau, Hays W. McCormick III, and Thomas J. Mowbray. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons, Inc., NY, USA, 1998: 41-78 - ISBN:0−471−19713−0.

2. Антипатерни [Електронний ресурс] / – Режим доступу: https://uk.wikipedia.org/ wiki/Антипатерн – Назва з екрана.

3. Что такое DRY, DIE, KISS, SOLID, YAGNI в программировании [Електронний ресурс] / – Режим доступу: https://www.stijit.com/web-tips/dry-kiss-solid-yagni – Назва з екрана.