УДК 004.04

Л. А. Власова
О. М. Міхайлуца

ЗАСТОСУВАННЯ ПАТЕРНІВ ПРОЕКТУВАННЯ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО КОМПЛЕКСУ З ВИЗНАЧЕННЯ ПОТРЕБИ У ЗАПАСНИХ ЧАСТИНАХ

Інженерний навчально науковий інститут ім Ю. М. Потебні Запорізького національного університету

Анотація
Робота присвячена використання патернів проектування на прикладі програмного комплексу для визначення необхідної кількості запасних частин. Розглянуто способи використання різноманітних патернів проектування для покращення програмного продукту.
Ключові слова: патерни проектування, програмний комплекс, запасні частини

Abstract
The work is devoted to the use of design patterns on the example of a software package to determine the required number of spare parts. The ways of using various design patterns to improve the software product are considered.
Key words: design patterns, software package, spare parts

В Україні функціонує безліч підприємств різних галузей, які роблять свій внесок у розвиток країни. Безперервна робота цих підприємств забезпечує надходження коштів до державного бюджету. Для цього вкрай важливо своєчасно проводити ремонт обладнання. Проте, навіть це не гарантує 100% надійності, тому на складах необхідно мати певний запас запасних частин.

Однак для раціонального використання коштів на закупівлю цих запчастин, важливо чітко визначити їх необхідну кількість. Ці розрахунки можуть бути досить складними, тому для полегшення цього процесу можна розробити програмний комплекс, який автоматизує розрахунок необхідної кількості запасних частин, що дозволить оптимізувати витрати та мінімізувати ризики простою виробництва.

Для розробки ефективних програмних комплексів можна використовувати патерни проектування, застосування яких сприяє створенню гнучкого, модульного та масштабованого коду, який легко модифікувати та розширювати. Розглянемо кілька ключових патернів, які можуть бути корисними при створенні комплексу для визначення потреби у запасних частинах [1]:

Патерн «Абстрактна фабрика» дозволяє створювати сімейства пов'язаних об'єктів без прив'язки до конкретних класів. Надає можливість розширення системи без внесення змін до існуючого коду, а також більш просте та логічне управління класами розрахунків. Можна використати для створення різних алгоритмів розрахунку потреби у запасних частинах для різних типів обладнання.

Патерн «Заступник» (Proxy) – це структурний патерн проектування, який дозволяє створити об'єкт-заміну для іншого об'єкта. Цей замінник, або проксі, контролює доступ до основного об'єкта, даючи можливість виконати додаткові дії до або після того, як запит дійде до нього. Використання даного шаблону допоможе кешувати результати розрахунку, щоб уникнути додаткових обчислень при повторних зверненнях з однаковими даними; фільтрувати запити, щоб не допускати доступу до оригінального об'єкта з несанкціонованих джерел; вести журнал запитів до оригінального об'єкта, що може бути корисно для аналізу та налагодження системи.

Патерн «Міст» (Bridge Pattern) – це структурний патерн проектування, який розділяє один або кілька класів на дві окремі ієрархії: абстракцію та реалізацію. Дозволяє незалежно змінювати абстракцію та реалізацію класу, не впливаючи на інші компоненти системи, легко додавати нові реалізації класу без необхідності змінювати існуючий код. У контексті програми для визначення потреби у запасних частинах рівень абстракції може описувати загальний інтерфейс для роботи з обладнанням, незалежно від його типу чи виробника, а рівень реалізації – описувати конкретні типи обладнання, такі як металугійне, машинобудівне або будь яке виробниче, тощо.

Патерн «Спостерігач» (Observer Pattern): Автоматизоване оновлення інформації про потреби у закупках. Це поведінковий патерн проєктування, який створює механізм підписки, дозволяючи одним

об'єктам (спостерігачам) відстежувати та реагувати на події, що відбуваються в інших об'єктах (суб'єктах). У контексті програми для визначення потреби у запасних частинах суб'єктом є модуль, який відстежує використання обладнання та генерує події при його зносі, спостерігачем – модуль, який отримує сповіщення про знос обладнання та формує запити на закупівлю нових запчастин.

Патерн «Компоновщик» (Composite Pattern): Структурування та управління даними про обладнання. Це структурний патерн проектування, який дозволяє групувати багато об'єктів у деревоподібну структуру, яка може розглядатися як єдиний об'єкт, що спрощує роботу з нею. Даний патерн дозволяє чітко структурувати дані про обладнання, представляючи його у вигляді ієрархії, де кожен рівень відповідає певному рівню деталізації; легко додавати, видаляти та змінювати елементи у структурі без впливу на інші компоненти; може використовуватися для виконання різних задач, таких як обхід дерева, пошук інформації, виконання операцій над елементами тощо.

Патерн «Ітератор» (Iterator Pattern): Послідовний доступ до елементів складених об'єктів. Це поведінковий патерн проектування, який дозволяє послідовно обходити елементи складених об'єктів, не розкриваючи їх внутрішнє представлення. В контексті додатку об'єктом є модуль, який представляє собою ієрархію обладнання, ітератором – об'єкт, який дозволяє послідовно обходити машини, механізми та змінні елементи в ієрархії. Для управління машинами, та пошуку потрібних елементів в обхід ієрархії. Для реалізації такої можливості користувач напрацьовує власну базу елементів, наприклад присвоюючи ідентифікатор за номером креслення та позицією специфікації.

Патерн «Декоратор» (Decorator Pattern): Динамічне додавання функціональності. Структурний шаблон проектування, який дозволяє динамічно додавати об'єктам нову функціональність, загортаючи їх у корисні «обгортки». Приклад застосування патерну «Декоратор» в контексті програми для визначення потреби у запасних частинах: базовим об'єктом є модуль, який виконує базовий розрахунок потреби у запасних частинах; декоратором є об'єкт, який додає до базового розрахунку додаткові функції, такі як фільтрація результатів, сортування за різними параметрами або збереження даних у звіт.

Загалом, використання патернів проектування робить програмне забезпечення більш логічним, функціональним, читабельним та гнучким. Це веде до кращої якості, надійності та керованості програмного комплексу, а також полегшує його розробку, модифікацію та подальшу підтримку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Head First. Патерни проєктування / Е. Фрімен та ін. ; пер. з Англ. А. Якубовська. Київ : Фабула, 2020. 672 с.
2. . Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Boston : Addison-Wesley Professional, 2018. 191 p.

Власова Лілія Андріївна *здобувач освіти групи 6.1211-пзс Інженерного навчально наукового інституту ім. Ю. М. Потебні Запорізького національного університету м. Запоріжжя liliya2003vlasova@gmail.com*

Міхайлуца Олена Миколаївна *кандидат технічних наук, доцент, доцент кафедри електроніки, інформаційних систем та програмного забезпечення Інженерного навчально наукового інституту ім. Ю. М. Потебні Запорізького національного університету м. Запоріжжя elenamikhaylutsa7@gmail.com*

Vlasova Liliia Andriivna*, a student of the Engineering Educational and Scientific Institute named Yu. M. Potebnya, Zaporizhzhya National University, Zaporizhzhia, liliya2003vlasova@gmail.com*

Mikhaylutsa Olena Mykolayivna *Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Electronics, Information Systems and Software of the Engineering Educational and Scientific Institute named Yu. M. Potebnya, Zaporizhzhya National University, Zaporizhzhia, elenamikhaylutsa7@gmail.com*