УДК 519.685

Верес М. М., Глибовець А. М., Кумейко Н. О.

СИСТЕМА АВТОМАТИЗОВАНОГО ГЕНЕРУВАННЯ ПАТЕРНІВ

Патерни проектування спрощують повторне використання вдалих проектних й архітектурних рішень. Можливість генерувати код автоматично, лише обравши мову, – частина автоматизації, що дає змогу не витрачати дорогоцінний час і запобігає виникненню помилок через людський фактор. Запропоновану систему автоматизованого генерування патернів можна рекомендувати для використання в освітньому процесі, для навчання мислення складними категоріями проектування (патернами) із зручним візуальним інтерфейсом оптимізації програмного коду.

Ключові слова: патерни програмування, проектування, повторне використання, автоматизація, генерування патернів.

Вступ

Об'єктне й об'єктно-орієнтоване програмування (ООП) виникло внаслідок розвитку ідеології процедурного програмування, де дані й програми (процедури, функції) їхньої обробки формально не зв'язані. У сучасному ООП часто велике значення мають поняття події (подійно-орієнтоване програмування) і компонента (компонентне програмування). Наразі, воно залишається визнаним лідером у галузі прикладного програмування [1]. Парадигма ООП створила нові потужні «інструменти», але залишилось відкритим питання ефективного та безпечного використання цих «інструментів» [4].

Саме за таких обставин з'явилися і почали розвиватися патерни проектування, які акумулювали досвід проектування складних програмних систем. Можна згадати Еріха Гамма, як одного з першопрохідців у дослідженні, систематизуванні та каталогізації шаблонів проектування. Його дослідження значно вплинули на розвиток підходу виділення та вивчення патернів як структурних одиниць проектування [2], що дають змогу ліпше розуміти систему в цілому та перспективи її розвитку, підтримки тощо.

Ефективність та масштаби використання проектування на основі патернів робить цей напрямок досліджень дуже важливим та актуальним.

Патерн проектування – *Design Patterns* (шаблон проектування), це опис взаємодії об'єктів і класів, адаптованих для розв'язання загального завдання проектування в конкретному контексті. У загальному випадку патерн складається з чотирьох основних елементів: ім'я, завдання, рішення, результат [6].

Патерни проектування спрощують повторне використання вдалих проектних й архітектурних рішень. Подання перевірених часом методик у вигляді патернів проектування полегшує доступ до них розробникам нових систем [3]. Патерни проектування дають розробнику можливість швидше знайти «правильний» шлях побудови складної програмної системи. Вони надають проектувальникам єдину термінологію, якою можна користуватися для спілкування, документування й вивчення можливих альтернатив. Система стає менш складною, оскільки про неї стає можливим говорити на вищому рівні абстракції, ніж нотації мови проектування чи програмування [5].

Добре продумані об'єктно-орієнтовані системи зовні схожі на зібрання численних патернів, але зовсім не тому, що їхні проектувальники мислили саме такими категоріями. Композиція на рівні патернів, а не класів або об'єктів, дає змогу домогтися тієї ж синергії, але з меншими зусиллями.

Зрозуміло, що найважливішою частиною розробки програмного продукту є проектування, що з самого початку встановлює чи обмежує перспективи розвитку, повторного використання, масштабовності, гнучкості до змін. Дуже важливо навчитись по-творчому підходити до цієї частини, випробовуючи різні підходи та аналізуючи отримані результати (як позитивні так і негативні), що посприяє розумінню сутності процесу проектування об'єктно-орієнтованого програмного продукту. А механічне використання патернів «за довідником» гальмує процес розвитку майбутнього професіонала і робить неефективним сам процес проектування, не даючи розуміння причин і наслідків.

Розроблене програмне середовище (репозиторій) ми представили у вигляді *Pattern System*, який можна знайти за адресою – http://glybovets.com.ua/patterns/pattern_system.zip.

Pattern System

Програма написана на мові Java, використовуючи бібліотеки графічної оболонки SWT, що забезпечує багатоплатформність застосування і швидкий GUI.

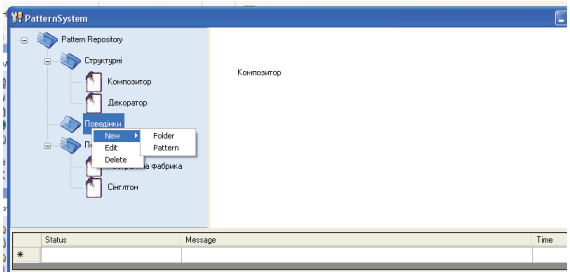

Рис. 1. Головне вікно програми

На рис. 1. зображено головне вікно інтерфейсу програми. Його зміст досить простий: з лівого боку є ієрархічний контейнер патернів, з правого боку – ділянка для візуалізації діаграми (чи діаграм) класів, внизу – поле для виводу історії операцій.

У контейнері патернів можна продивитися дійсні, додавати нові, видаляти та виконувати інші дії маніпулювання, що одразу дає змогу побачити, який саме цей шаблон та які основні варіанти його реалізації.

Обраний патерн промальовується в правій частині (відображаються всі наявні діаграми класів обраного патерну). Кожна відображена діаграма класів надає наступні можливі дії: редагувати, реалізувати (тобто, корегуючи певні параметри, згенерувати код на вибраній мові програмування) та видалити. «Клацнувши» на порожньому місці через контекстне меню можна додати нову діаграму. Особливо цікавими є можливість редагувати та реалізовувати діаграму класів.

Тож розглянемо їх по черзі детальніше.

Перша вкладка редактора містить візуальний редактор діаграми класів. Доступні основні можливості: створення, видалення, переміщення класів, інтерфейсів, усіх основних типів зв'язків між ними. Можна створити і *клієнта* – муляж класу (об'єкта), який виконує взаємодію з класами (об'єктами) патерну. Відразу після зміни системи класів/інтерфейсів зміни відображаються в списку вкладок (з'являються нові, чи зникають видалені класи/інтерфейси за зростанням рівня в ієрархії класів).

На рис. 2. відображено вкладку редагування класу (чи інтерфейсу, редактор працює з ними одночасно) надає можливість змінити ім'я, конкретизувати тип класу/інтерфейсу (абстрактний, конкретний), дозволити зміну на етапі реалізації. Також може бути встановлена відмітка, що фіксує стан класу (тобто в реалізаторі не матимемо змогу додавати, видаляти чи змінювати мето-

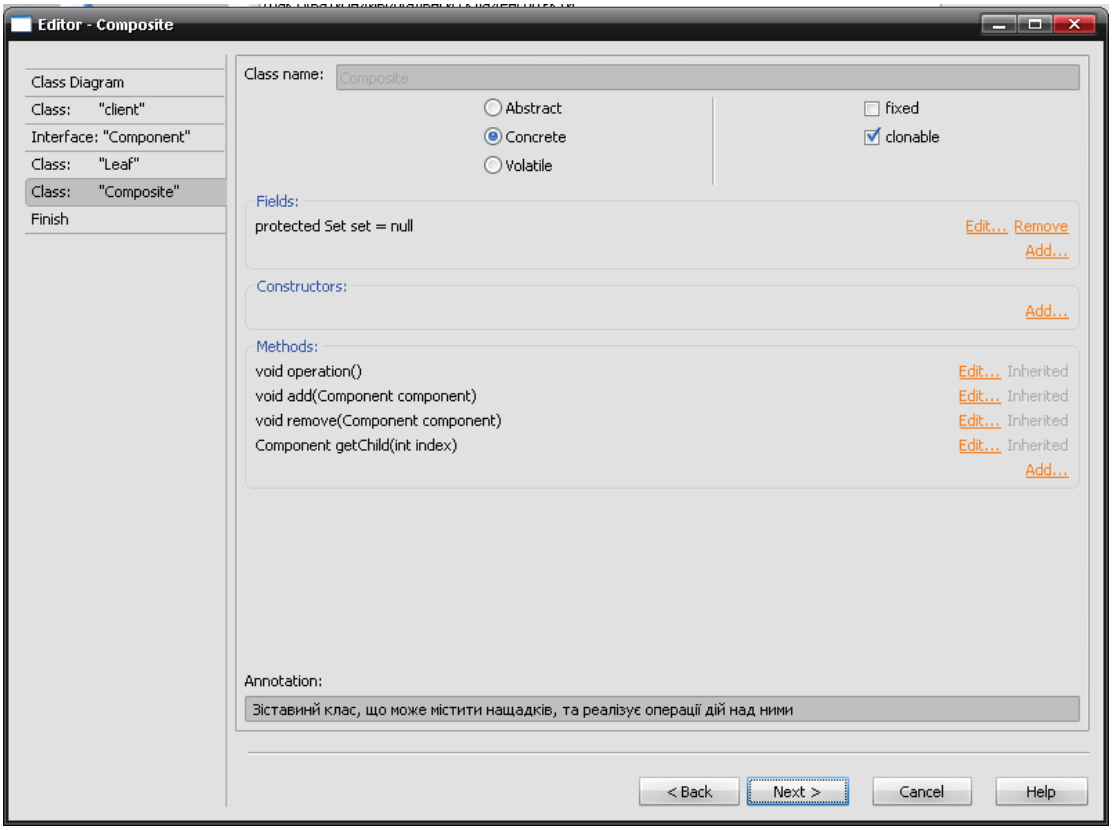

Рис 2. Вкладка редагування класу в редакторі

ди, поля, конструктори), і відмітка про можливість клонування гілки ієрархії класів на етапі реалізації, що має коренем клас/інтерфейс, який можемо редагувати.

Редагування методів, конструкторів та полів доступне через невеличкі конструктори, що дають змогу виконати всі необхідні налаштування, а також (для конструкторів та методів) задати реалізацію.

Рис. 3. Редактор полів

Рис. 4. Редактор методів

Вказаними засобами можна створити та налаштувати конкретну діаграму класів і, додавши опис патерну та його варіанту реалізації, зберегти.

Тепер розглянемо реалізатор (рис. 5) та його особливості. Суть реалізатора в тому, що ми можемо обраний варіант патерну корегувати, змінити імена класів, де це потрібно, додати чи змінити методи (можна додати/реалізувати) і т.д., після чого обраним генератором коду створити код реалізації патерну на вибраній мові програмування.

Вкладка редагування класу відрізняється від однойменної в редакторі тим, що в ній треба однозначно вказувати тип класу, якщо це не фіксовано на етапі редагування. Також з'явились дві нові кнопки: клонувати та видалити клоновану гілку, які дають змогу виконувати ці операції.

Важливо згадати про генератори коду. В цьому проекті вони реалізовані як окремі модулі, що підключаються. Достатньо дописати в папку генераторів ім'я файлу нового генератору як його «підхоплює» система – він стає доступним для використання. Це створено для можливості розвивати проект, додаючи нові мови реалізації (патерни не залежать від мови, а від концепцій). Від мови програмування залежить лише складність реалізації патерну.

Висновок

Розроблений програмний продукт буде цікавий як початківцям, так і професіоналам. Першим він буде корисним для навчання мислити складними категоріями проектування (патернами), другим – зручним візуальним інтерфейсом

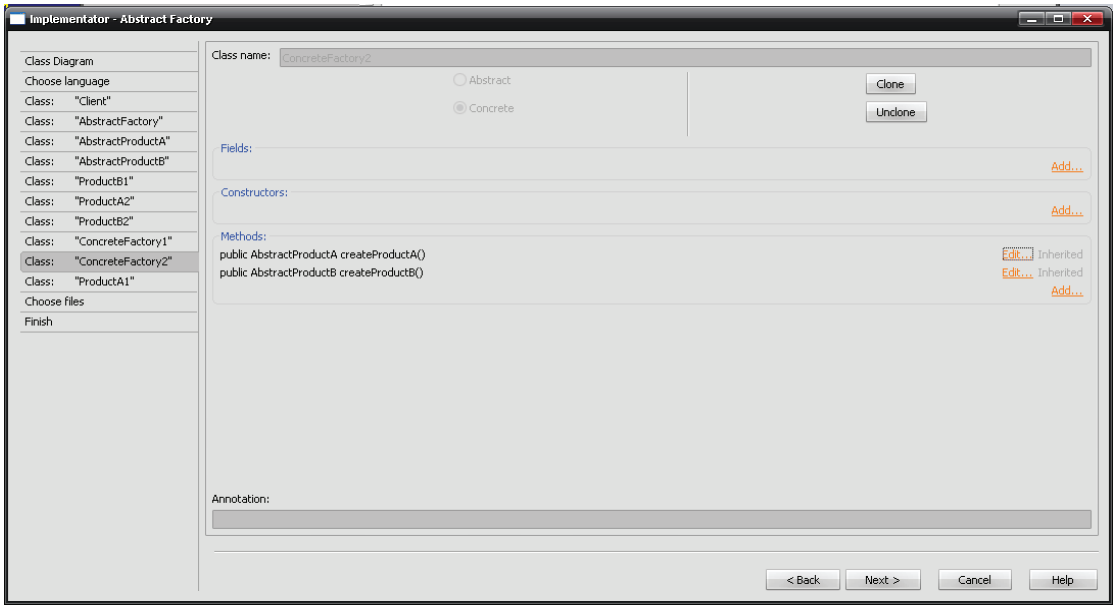

Рис. 5. Вкладка редагування класу в реалізаторі

оптимізації програмного коду. Його можна застосувати і в електронному навчанні.

Можливість генерувати код автоматично, лише обравши мову – частина автоматизації, що дає змогу не витрачати дорогоцінний час і запобігає виникненню помилок. Програмування повинно бути творчим процесом, тому механічні дії бажано робити автоматизованими, що й забезпечує наша розробка.

На наступному етапі розвитку програмного продукту, плануємо додати можливість створювати нові проекти, компонуючи вже наявні патерни. Важливою перспективою дослідження є і реалізація завдання оберненого проектування: побудови діаграми класів з програмного коду, а також виділення на діаграмі класів патернів з готової бібліотеки шаблонних рішень.

Література

1. Буч Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон. – М. : ДМК-пресс, 2007. – 257 с.
2. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб. : Питер, 2007. – 366 с.
3. Глибовец Н. Н. Упрощенная инфраструктура для трансформации XML-моделей / Н. Н. Глибовец, В. М. Федорченко // Кибернетика и системный анализ. – 2010. – № 1. – С. 105–111.
4. Фаулер М. UML. Основы. Краткое руководство по унифицированному языку моделирования / М. Фаулер, С. Кендалл. – СПб. : Символ-Плюс, 2002 г. – 192 с.
5. http://ru.wikipedia.org/wiki/Шаблоны_проектирования. – Назва з екрана.
6. http://www.dofactory.com/Patterns/Patterns.aspx. – Назва з екрана.

M. Veres, A. Glybovets, N. Kumeyko

AUTOMATED SYSTEM OF PATTERNS GENERATION

Design patterns facilitate reuse of successful design and architectural decisions. Ability to generate code automatically, just choose your language - are part of automation that cannot spend precious time and prevents errors by reason of the human factor. The proposed system for automated generation patterns can be recommended for use in education for teaching complex thinking design patterns with convenient visual interface optimization software.

Keywords: patterns of programming, design, reuse, patterns generating, patterns development.

Матеріал надійшов 11 травня 2011 р.