УДК 621.3

Я.М. Грицишин, С.П. Ткаченко
Національний університет “Львівська політехніка”,
кафедра систем автоматизованого проектування

ШАБЛОНИ ПРОЕКТУВАННЯ У ВЕБ-ОРІЄНТОВАНІЙ СИСТЕМІ РОЗКРОЮ

Наведено спрощену функціональну схему веб-орієнтованої системи розкрою. Описано та обґрунтовано використання шаблонів для проектування її підсистем.

Reductive functional schema of web-oriented cutting system is shown in this article. Using of design patterns in this system is described and grounded below.

Вступ. Задача розкрою матеріалів є надзвичайно поширена у багатьох галузях промисловості, зокрема в легкій, машинобудівній, кораблебудівній та ін. При цьому використовуються різні форми листів, які підлягають розкрою на деталі. Ефективним вважається таке розташування об’єктів, за якого відходи є мінімальними. Існує велика кількість програмних рішень задачі розкрою матеріалів. Основним недоліком таких систем є велике співвідношення вартості до обсягу задач, які ці системи можуть розв’язувати. Одним з варіантів усунення такого недоліку є розширення функціональності системи розкрою, але це одразу ж робить систему громіздкою та дорогою. Ще одним недоліком існуючих систем є складнощі з їх поновленням і розвитком. При виході нових версій користувач має самостійно здійснювати міграцію на новий програмний продукт, що може викликати великі втрати часу та фінансів [1, 2, 4, 5].

Проблеми при проектуванні веб-орієнтованої системи розкрою та підходи для їх вирішення. Враховуючи вищесказане, можна зробити висновок, що необхідно створити програмно-апаратний термінал (сервер), який би забезпечував розв’язання широкого спектра задач розкрою матеріалів і надавати користувачам тільки інтерфейс до такого терміналу. Отже, користувачам не потрібно встановлювати в себе громіздкі програми і дороге апаратне забезпечення достатньо буде використати недорогий комп’ютер із виходом в Інтернет [5].

Спрощену функціональну схему такої системи показано на рис. 1.

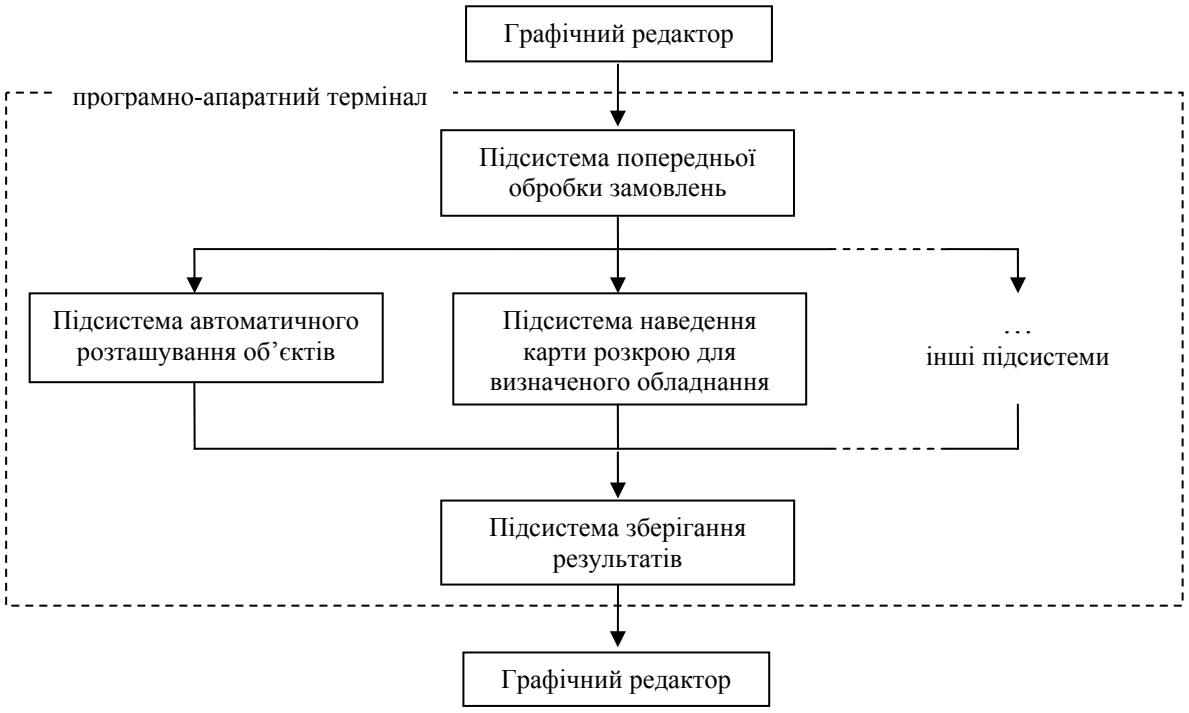

Рис. 1. Спрощена функціональна схема Веб-орієнтованої системи розкрою матеріалів

Під час проектування такої системи виникають проблеми, вирішення яких безпосередньо вплине на відкритість, гнучкість та ефективність системи, зокрема:

- форма організації структури запиту до програмно-апаратного терміналу;
- розширюваність системи функціональними підсистемами;
- підготовка та збереження результатів обробки запиту.

Запит до програмно-апаратного терміналу – організована деяким способом множина параметрів, які можуть бути текстовими, числовими та графічними. Всі ці параметри містять вичерпну інформацію про зміст запиту. Вони організовані у фізичні підструктури структури запиту, які описують автора запиту, проблеми, які його цікавлять та дані для їхнього вирішення. Ці підструктури, своєю чергою, можуть складатися з дрібніших підструктур. Тобто ми зокрема зможемо робити результати розв'язання однієї задачі вхідними даними іншої задачі. Ця структура повинна надавати можливість збільшувати кількість типів таких підструктур без впливу на вже існуючі типи. Це зробить систему гнучкішою. Такий запит містить ієрархічно структуровану інформацію, для представлення якої часто використовується метод рекурсивної композиції [3], який дає змогу будувати все більш складні елементи з простих. Рекурсивна композиція, наприклад, дає можливість будувати графічні об'єкти з точок, карти розкрою з графічних об'єктів тощо. Структуру запиту, організованого таким чином, зображено на рис. 2.

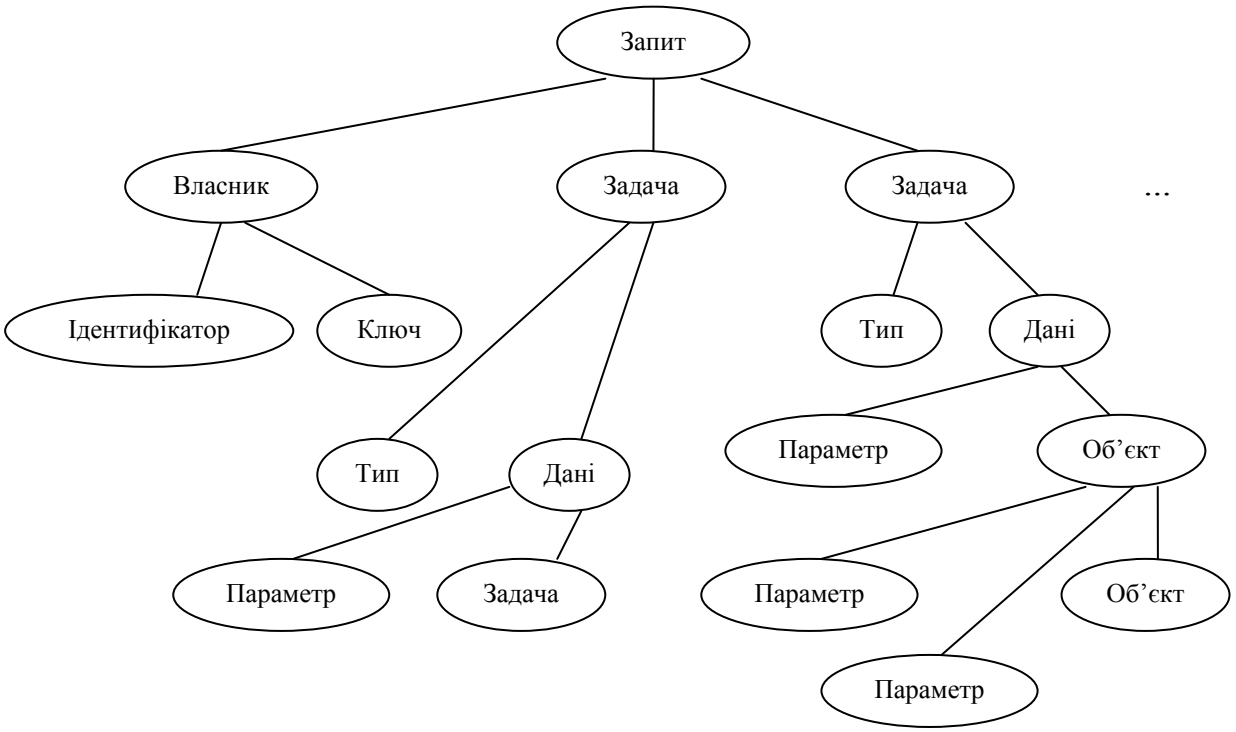

Рис. 2. Структура запиту для рекурсивної композиції задач та даних

Описана структура має декілька наслідків. Кожен вузол має бути зрозумілий тільки для підсистеми, яка його опрацьовуватиме. Відповідно кожна підсистема має ігнорувати гілки, які для неї не призначені. Кожна підсистема синтезує об'єкт необхідний для рішення поставленої задачі з відповідних даних. Всі гілки задач мають мати однаковий інтерфейс розпізнавання. Для розв'язання такої задачі проектування призначений шаблон Компонувальник (Composite). Цей шаблон компонує об'єкти у деревоподібні структури для представлення ієрархій [3]. Для представлення запиту пропонується використати формат XML (Extensible Markup Language) [6]. Він дає змогу чітко розділити гілки запиту і передавати функціональним підсистемам тільки необхідні для них гілки. Для представлення графічних зображень пропонується використати формат SVG (Scalable Vector Graphics), побудований на базі XML, для представлення векторної графіки [6].

На вході в програмно-апаратний термінал запит потрапляє в сервіс Консьєрж, на рис. 1 він фігурує як “Підсистема попередньої обробки замовлень”. До його обов’язків входить отримання запиту, його перевірка і передача необхідних гілок функціональним підсистемам залежно від типів поставлених задач.

Для отримання запиту Консьєрж звертається до підсистеми, до якої запити надходять від Графічного редактора. Оскільки способи надходження запитів можуть бути різні, необхідно привести їх інтерфейс до єдиного зручного для Консьєржа. У такому випадку Консьєрж працюватиме з одним інтерфейсом і його не цікавитиме, який спосіб надходження запитів використовується. Цей шаблон проектування називається Фасад (Facade) [3]. Він визначає інтерфейс вищого рівня, який спрощує використання підсистеми. Методів отримання запитів може бути декілька. Наприклад, запити надходять через мінівеб-сервіс за допомогою HTTP пакетів. Є ряд бібліотек, які забезпечують роботу з такими мінівеб-сервісами. Інтерфейси цих бібліотек можна привести до інтерфейсу, який Консьєрж використовує для отримання запитів. Інший спосіб отримання запитів – через електронну пошту. Знову ж таки є ряд бібліотек, які автоматизують роботу з поштовими клієнтами, і можна привести інтерфейс цих бібліотек до інтерфейсу, який зручний Консьєржу. Шаблон, який передбачає приведення інтерфейсу одного об’єкта до інтерфейсу іншого, називається Адаптер (Adapter) [3]. Він забезпечує роботу класів з несумісними інтерфейсами, яка без нього була б неможлива. Отже, ми можемо використовувати різні способи доставки запитів, і це не впливатиме на решту системи.

Після отримання запиту Консьєрж його аналізує. Консьєрж перевіряє правильність структури запиту, а також права власника запиту на користування функціональними підсистемами, необхідними для розв’язання задач, вказаних у запиті. При аналізі XML структури запиту Консьєрж використовує уніфікований інтерфейс до існуючих бібліотек функцій роботи з XML форматом. Цього досягають також за допомогою шаблонів Фасад та Адаптер [3]. Права власника запиту можна отримати з бази даних, використовуючи її адаптований інтерфейс. Після цього підсистема попередньої обробки замовлення передає дані про власника та відповідні гілки задач функціональним підсистемам.

Функціональними підсистемами ми називаємо підсистеми, призначені для розв’язання задач, описаних у запиті. Оскільки система розкрою передбачає розширення функціональності, тобто зростання кількості типів задач, які вона може розв’язувати, то необхідно забезпечити однаковий інтерфейс цих підсистем (шаблон Фасад) [3]. Тоді буде можна додавати нові функціональні системи, не впливаючи на інші підсистеми.

Однією з функціональних підсистем є підсистема автоматичного розташування об’єктів, призначена для формування карт розкрою, виходячи з площин для розташування і розташовуваних об’єктів. Сьогодні існує велика кількість різних типів задач розкрою. Вони відрізняються за формами об’єктів та площин, за критеріями ефективності та різноманітними обмеженнями, а також алгоритмами для розв’язання задач розкрою. Постійно з’являються нові ефективніші алгоритми, за якими можна розв’язувати нові задачі розкрою [2, 4, 5]. Підсистема автоматичного розташування аналізує гілку задач і вибирає необхідний алгоритм для реалізації розташування. Ці алгоритми мають однаковий інтерфейс для отримання вхідних даних (шаблон Фасад) [3]. Вони готують карту розкрою, яку підсистема автоматичного розташування передає до підсистеми зберігання результатів.

До обов’язків підсистеми зберігання результатів належить тривале зберігання результатів роботи функціональних підсистем у доступному для автора запиту місці, та повідомлення власника запиту про його виконання. Для тривалого збереження результатів можна використати файлову систему. Оскільки як вид результатів роботи функціональних систем може відрізнятися, то і спосіб збереження цих результатів може бути різним. Для того, щоб прив’язати формат збереження результату до функціональної підсистеми, можна використати шаблон Стратегія (Strategy) [3], за яким змінювати алгоритми незалежно від клієнтів, які ними користуються. Для під’єднання підсистеми повідомлення власника про результат можна використати шаблони Фасад та Адаптер.

Тобто наша система буде достатньо гнучкою та розширюваною, а також дасть змогу використовувати вже існуючі бібліотеки реалізацій підсистем. Але водночас ускладнилось створення системи та налагодження підсистем. Зокрема, існують однотипні класи, під час роботи з якими клієнтові важливий тільки інтерфейс (тип) і неважлива реалізація: наприклад, класи, які представляють алгоритми розташування, функціональні підсистеми, підсистеми, сховані за фасадами. Для створення таких класів використовують шаблон проектування Будівник (Builder) [3], щоб відділити конструювання складного об'єкта від його представлення, так що в результаті одного і того ж процесу конструювання можна отримати різні представлення.

У нашій системі існують об'єкти, які є підсистемами, деякі з них, наприклад, адаптер бази даних або підсистема ведення логів, тільки один екземпляр. Створення і спосіб роботи з такими об'єктами описує шаблон Одинак (Singleton) [3]. Цей шаблон гарантує, в класі є один екземпляр і пропонує глобальну точку доступу до нього. Тобто доступ до адаптера бази даних здійснюватиметься за допомогою статичного методу, а створення цього адаптера здійснюватиметься під час першого звертання до цього статичного методу.

Реалізація певних класів системи розкрою матеріалів залежить також від операційної системи та її компонентів. Якщо потрібно забезпечити незалежність системи від платформи, точніше, спростити процес адаптації системи розкрою до різних операційних систем, є сенс об'єднати всі створення платформо-залежних компонентів у окремі породжувальні класи, фабрики. Такий шаблон проектування називається Абстрактна Фабрика (Abstract Factory) [3]. Цей шаблон пропонує інтерфейс для створення наборів взаємозалежних та взаємозв'язаних об'єктів, не специфікуючи їхніх конкретних класів.

Під час проектування такої системи можна застосовувати й інші шаблони проектування залежно від деталізації підсистем. У випадках використання вже існуючих бібліотек, які містять частини функціональності деяких підсистем, не має необхідності заглиблюватися в їхню реалізацію та деталізацію.

Висновок. Внаслідок процесу проектування ми отримаємо достатньо гнучку та легко розширювану архітектуру системи розкрою матеріалів, легко розширювати її функціональність шляхом додавання нових підсистем та замінювати реалізацію вже існуючих. Також це спростить процес супроводу та налаштування системи.

1. Грицишин Я., Лобур М., Ткаченко С., Чура И. Особенности разработки САПР раскроя материалов // IEEE AIS'02 CAD-2002. – Sept. 2002. – P. 404–409. 2. Ткаченко С., Чура І., Грицишин Я. Оптимізація розміщення плоских об'єктів на площині довільної форми // Вісн. Нац. ун-ту "Львівська політехніка". – 2002. – № 444. – P. 134–136. 3. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns. Elements of Reusable Object-Oriented Software // Addison Wesley Longman, Inc, 1995. 4. Tkachenko S., Chura I., Hrytsyshyn Y., Karkulyovsky V. The placement of optional form objects on the optional platform // CADSM'2003, pp.417-420, Feb. 2003. 5. Hrytsyshyn Y., Tkachenko S. Web-oriented CAD system for material cutting // CADSM'05. – Feb.2005. – P. 456–457. 6. www.w3.org, World Wide Web Consortium.