УДК 004.415.5
М. Горалечко, С. Метохір
Тернопільський національний технічний університет імені Івана Пулюя, Україна

СТВОРЕННЯ МНОЖИНИ АЛЬТЕРНАТИВНИХ АРХІТЕКТУР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

UDC 004.415.5
M. Goralechko, S. Metokhir
(Ternopil Ivan Puluj National Technical University, Ukraine)

DEVELOPMENT OF THE SET OF ALTERNATIVE SOFTWARE ARCHITECTURES

Для подання архітектури програмних систем була прийнята концепція, в якій функціональність програми розділена на шари. Корпорація Microsoft розробила технологію, яка базується на даній концепції [1]. Згідно з цією технологією для кожного шару розроблено набори компонентів (патернів), які реалізують функціональність даного шару. Патерни згруповані у категорії (модулі), що призначені для вирішення певних стандартизованих задач.

Кожний програмний додаток, який проектується, поділяється на логічні частини, які відповідають шарам. Визначивши категорії задач, які будуть розв'язуватись певним шаром, обирається деякий компонент з існуючого набору і, таким чином, будується каркас архітектури. Але для кожного модулю існує, як правило, декілька патернів, тому отримаємо множину альтернативних архітектур.

Для автоматизації цього процесу пропонується використати експертну технологію, де знання організовані у вигляді фрейму, зображеного на рисунку 1.

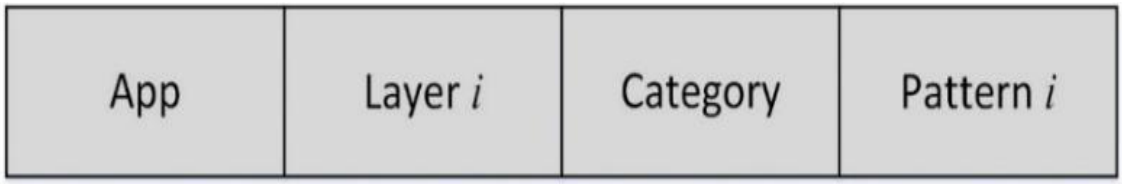

Рисунок 1. Структура фрейму бази знань експертної системи App – ім'я програми/програмного додатку; Layer – ім'я шару; Category – ім'я категорії/модулю задач; Pattern – ім'я патерну/шаблон

База даних архітектур (репозиторій патернів) використовується для формування альтернативних архітектур із стандартних патернів. Тут зберігаються правила побудови архітектури у відповідності до типу програмного додатка.

Адміністратор репозиторію патернів поділяє додатки на шари і визначає задачі, які розв'язуються на певних шарах. Архітектор заповнює фрейм-шаблон, в якому незаповненим залишається останній слот.

На основі пошуку в репозиторії патернів (шаблонів) знаходиться відповідний компонент, який поміщується в каркас архітектури, створюючи таким чином архітектуру додатку.

Основним фактором, що істотно вплинув на етапи проектування та реалізації програмного комплексу став чинник застосування системи для підтримки прийняття експертних рішень по архітектурі програмних додатків різних типів при достатньо великій кількості альтернатив для кожного типу.

Тому при проектуванні системи були враховані наступні функціональні вимоги [2]:

- зручність застосування системи для підтримки прийняття рішень на множині альтернативних архітектур експертами;

- забезпечення ефективної роботи адміністратора репозиторію патернів архітектур та архітектора програмних додатків;
- система має легко реорганізовуватись при розширенні кількості типів архітектур програмних додатків з відповідними шарами та патернами;
- доступ на модифікацію даних, які розміщені в репозиторії патернів архітектур програмних додатків, повинні мати особи, які мають відповідні повноваження.

Опишемо процес створення альтернативних архітектур. В ході проектування необхідно обирати відповідний до ТЗ тип архітектури додатку та обрати множину патернів для заповнення компонентів шарів (модулів), загалом для кожного компонента можна знайти декілька патернів, що виконують однакові задачі, але мають різну логічну, структурну чи функціональну реалізацію. Як приклад патерни шару представлення / компоненту UI :

- MVC (Model-view-controller) pattern [3].
- MVP (Model-View-Presenter) pattern [3].

За своєю функціональною метою вони реалізовують однаковий функціонал, але мають різну структурну та функціональну структуру. MVP є похідним, видозміненим патерном, відносно MVC.

Також як приклад можна розглянути патерни шару доступу до даних/компоненти доступу до даних:

- DAO (data access object) pattern [3].
- Пряма адресація [3].

За своєю функціональною метою обидва патерни реалізують доступ до даних в базі даних. Але DAO реалізовує декілька шарів абстракції за рахунок яких він є більш гнучким, захищеним та незалежним від типу бази. В свою чергу Пряма адресація реалізовую прямі запити до бази, що є швидшим методом отримання даних, але погано модернізованим та сильно залежним від типу та структури БД.

Припустимо, що обрано всі патерни архітектури однозначно, один компонент – один патерн. Таким чином комбінуючи компоненти ми отримуємо 4 альтернативні архітектури

Після створення множини альтернативних архітектур експертами проводиться попарне оцінювання сформованого масиву по різних критеріях якості. Під час цього оцінювання отримується матриця парних порівнянь по критеріях.

Було виконано порівняння альтернативних архітектур, після чого можемо зробити висновки. MVC більш широким та менш спеціалізованим патерном по відношенню до MVP. Таким чином архітектури з MVC краще модернізуються. Якщо порівняти патерни DAO та Пряму адресацію, то можна сказати, що Пряма адресація майже не піддається модернізації. В свою чергу DAO – є патерном, що розрахований на модернізацію.

Література

1. Руководство Microsoft по проектированию архитектуры приложений. 2-е издание. Microsoft, 2009. – 529 с.
2. Харченко О. Г. Експерта система проектування архітектури програмного забезпечення / О. Г. Харченко, І. О. Боднарчук, В. В. Яцишин // Комп'ютерні технології друкарства. № 29. 2013. – С. 10–26.
3. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования [Текст] / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – СПб. : Питер, 2010. – 366 с.