Інформаційні технології в технічних системах

УДК 004.5

З.В. Батюк, О.В. Тарасов

Харківський національний економічний університет імені С. Кузнеця

ВИКОРИСТАННЯ ШАБЛОНІВ ПРОЕКТУВАННЯ MVVM ПРИ РОЗРОБЦІ ЗАСТОСУВАНЬ НА ПЛАТФОРМІ MICROSOFT

В роботі розглянуті питання використання шаблону проектування MVVM при розробці застосувань на платформах Microsoft. Викладені основні принципи організації даного шаблону і його складові. Аналізуються його можливості на платформах, які використовують мову розмітки XAML, і застосування для розробки корпоративних додатків.

Ключові слова: *шаблон проектування, модель, подання, модель уявлення, MVVM, WPF, XAML, поділ коду, архітектура програми.*

Вступ

Розробка інтерфейсу користувача для професійного додатку – це досить нетривіальне завдання. Воно може включати в себе суміш даних, проектування взаємодій, візуальне проектування, безпеку, інтернаціоналізацію, можливості підключення, багатопоточність, перевірку допустимості та модульне тестування. Слід враховувати, що інтерфейс програми представляє систему і повинен задовольняти різним непередбачуваним вимогам користувачів. Застосування, які використовують однакову бізнес-логіку, можуть мати версії для різних платформ, що створює додаткові проблеми при розробці, тому виникає необхідність враховувати особливості кожної з платформ [3].

Існує кілька популярних шаблонів проектування, які дозволяють вирішити означену проблему, але завдання належного поділу безлічі питань і їх вирішення може бути важким. Чим складніше шаблони, тим імовірніше, що потім будуть використовуватися спрощення, які знецінять усі попередні спроби зробити як краще. Також розробники часто навмисно структурують код, відповідно до шаблону проектування, не дозволяючи шаблонам виникати природним шляхом.

Іноді через те, що платформа не сумісна з більш простим шаблоном проектування користувацького інтерфейсу, використовуються складні шаблони, реалізація яких вимагає великих об'ємів коду. Природно, що виникає необхідність у платформі, яка дозволить створювати інтерфейси користувача для програми за допомогою перевірених часом, простих і схвалених програмістами шаблонів проектування. На щастя, платформи Microsoft, що використовують мову розмітки XAML і їй подібні, а також шаблон проектування «модель-подання-модель подання (MVVM)» забезпечують вирішення поставленого завдання, що дає розробникам широкі можливості для створення різноманітних застосувань [1].

Метою статті є розгляд шаблону проектування MVVM і його переваг при розробці застосувань для платформ Microsoft, що використовують мову розмітки XAML.

Основна частина

MVVM – це шаблон проектування додатків для розділення коду користувацького інтерфейсу і іншого коду [4]. MVVM дозволяє декларативно визначати користувацький інтерфейс і використовувати розмітку прив'язки даних, що дає змогу зв'язати з іншими рівнями додатку, які містять логіку та дані. За допомогою прив'язки даних застосовується вільний взаємозв'язок, який синхронізує користувацький інтерфейс і пов'язані дані, а також відправляє командам вхідні дані, які ввів користувач.

Враховуючи мету зменшення працезатрат на розробку складного програмного забезпечення, припустимо, що необхідно використовувати готові уніфіковані рішення. Адже шаблонність дій полегшує комунікацію між розробниками, дозволяє посилатися на відомі конструкції, знижує кількість помилок.

Шаблон MVVM став еталонним для будь-якої програми XAML – Windows Presentation Foundation (WPF), Windows 8, Windows Phone і Silverlight. Вперше з'явившись у WPF, він поділяє обов'язки,

забезпечує тестування та ін. До переваг шаблону можна віднести і те, що його можна використовувати для будь-яких інших технологій – навіть тих, де XAML не застосовується, наприклад, ASP.NET, JavaScript тощо.

З перших кроків створення користувацьких інтерфейсів програм різні шаблони спрощували роботу розробників. Наприклад, шаблон модель-подання-презентатор (MVP) був популярний на різних платформах програмування користувацьких інтерфейсів. MVP – це вид шаблонів модель-подання-контролер, який існує вже кілька десятиліть. Суть його полягає в тому, що подання потребує презентатор для заповнення даними моделі, реакції на дії користувача, надання перевірки вводу (у тому числі за рахунок передачі цієї функції моделі) та інших подібних завдань.

Якщо говорити про відмінності MVVM і MVP, то модель подання не потребує посилання на подання. Подання прив'язується до властивостей моделі подання, яка, в свою чергу, представляє дані в об'єктах моделі та інших станах, потрібних для цього подання. Модель у даному випадку встановлюється в якості контексту подання (Data Context), тому легко створювати прив'язки між нею та поданням. Усі дані в моделі подання автоматично оновлюються через механізм прив'язки. Коли користувач натискає кнопку в поданні, для потрібної дії виконується команда в моделі подання.

Усі зміни даних моделі завжди справляє модель подання, а не подання [5].

Класи подання не знають про існування класів моделі, а модель подання і модель не знають про подання. Модель насправді взагалі не має уявлення про те, що існують модель подання та подання. Це дуже слабко пов'язана конструкція, і це дає ряд переваг.

MVVM – своєрідна загальноприйнята мова розробників WPF, тому що вона добре пристосована для платформ на XAML, а такі платформи створювалися для спрощення збірки застосувань за допомогою шаблону MVVM (та інших). У корпорації Майкрософт MVVM використовувався для внутрішніх цілей при розробці додатків WPF, наприклад, Microsoft Expression Blend, поки основна платформа WPF ще тільки створювалася. Багато частин WPF, наприклад, модель контролю без перегляду та шаблони даних, використовують значний розподіл показу від стану та поведінки, що застосовується в MVVM [3].

Одним з найважливіших моментів, який робить MVVM дуже зручним шаблоном, – це інфраструктура прив'язки даних. За рахунок механізму прив'язки властивостей подання до моделі подання виходить слабке зв'язування цих компонентів, що повністю звільняє розробника від необхідності писати в моделі подання код, який безпосередньо відповідає за оновлення подання. Дана система також підтримує перевірку допустимості введення, яка перевіряє наповнення введених даних [6].

Ще дві функції, що роблять цей шаблон таким корисним, – це шаблони даних і система ресурсів. Шаблони даних застосовують подання до об'єктів моделі подання, показаним в інтерфейсі користувача. Можна оголосити шаблони в коді XAML і дозволити системі ресурсів автоматично знаходити і застосовувати ці шаблони під час виконання.

Підтримка команд є однією з тих речей, які роблять шаблон MVVM таким універсальним. Команди дозволяють легко та зручно керувати поданням за допомогою методів, що створені у моделі подання. Команди пов’язуються із моделлю за допомогою спеціальних параметрів зв’язування, які надають практично всі мови розробки, що підтримують шаблон проектування MVVM

Крім функцій WPF і Silverlight, за рахунок яких MVVM стає ефективним способом структурування додатку, цей шаблон популярний ще й тому, що додаток, організований відповідно йому, легко піддається модульному тестуванню. Якщо логіка взаємодії додатку знаходиться в наборі класів моделі подання, стає легко написати тестуючий код. У якомусь сенсі подання і модульні тести – різні типи споживачів моделі подання. Набір тестів для моделі подання додатка забезпечує вільне і швидке регресійне тестування, яке зменшує вартість підтримки програми в майбутньому [3].

Крім зручності створення автоматичних регресійних тестів, тестування класів моделі подання може допомогти правильно проектувати інтерфейси, для яких легко робити теми оформлення. Проектуючи додаток, часто треба вирішувати що помістити в подання або в модель подання, уявивши собі, чи потрібно буде писати модульний тест, який використовує модель подання. Якщо можна написати модульні тести для моделі подання, не створюючи об'єкти користувацького інтерфейсу, то можна повністю укласти модель подання в тему оформлення, так як у неї немає залежності від певних візуальних елементів [8].

Нарешті, для розробників, які співпрацюють з проектувальниками інтерфейсів додатків, використання MVVM полегшує створення безперервного потоку робіт розробника і проектувальника. Так як подання – не більше ніж необов'язковий споживач моделі подання, то неважко замінити існуюче подання іншим, яке б відповідало даній моделі подання. Ця проста дія дозволяє швидко створювати прототипи і оцінювати користувацькі інтерфейси, зроблені проектувальниками [9].

Команда розробників може приділити основну увагу створенню стійких класів моделі подання, а

команда проектувальників – зручним для користувача поданням. Але щоб звести воєдино результати роботи обох команд, потрібно дещо більше, ніж перевірка наявності правильних прив'язок у файлі XAML [3].

Шаблон MVVM складається з трьох частин:

Model (модель відповідає даним, які бажано відобразити на екрані і маніпулювати ними),

View (подання, компонент презентаційного рівня і UI);

ViewModel (модель подання, яка використовує Model і відображає подання через зв'язування з даними, а також реагує на дії користувача).

Схема MVVM показана на рис. 1.

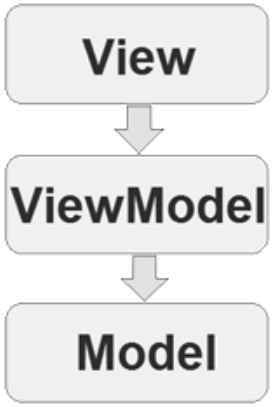

Рис. 1. Схема шаблону MVVM

Під «моделлю» (англ. Model), зазвичай розуміється частина, яка містить в собі функціональну бізнес-логіку програми.

«Модель» повинна бути повністю незалежна від інших частин продукту. «Модельний» шар нічого не повинен знати про елементи дизайну, і яким чином він буде відображатися.

Досягається результат, що дозволяє змінювати подання даних і те як вони відображаються, не чіпаючи саму «модель» [2].

«Модель» володіє такими ознаками:

– «модель» – це бізнес-логіка додатку;

– «модель» володіє знаннями про себе саму і не знає про контролери та подання;

– для деяких проектів «модель» – це просто шар даних (DAO, база даних, XML-файл);

– для інших проектів «модель» – це менеджер бази даних, набір об'єктів або просто логіка додатку.

В обов'язки «подання» (англ. View) входить відображення даних отриманих від «моделі».

Однак, «подання» не може безпосередньо впливати на «модель».

Можна говорити, що «подання» володіє доступом «тільки на читання» до даних [2].

«Подання» володіє такими ознаками:

– у «поданні» реалізується відображення даних, які надходять від «моделі» деяким способом;

– у деяких випадках «подання» може мати код, який реалізує деяку бізнес-логіку (наприклад, при використанні «подання» за технологією Razor).

«Модель подання» (англ. ViewModel) є, з одного боку, абстракцією «подання», а з іншого, надає обгортку даних з «моделі», які підлягають скріпленню. Тобто, вона містить «модель», яка перетворена до «подання», а також містить у собі команди, якими може користуватися «подання», щоб впливати на «модель» [2].

Ознаки «моделі подання»:

– двостороння комунікація з поданням.

– «модель подання» – це абстракція «подання». Зазвичай означає, що властивості уявлення збігаються з властивостями «моделі подання» / «моделі».

– «модель подання» не має посилання на інтерфейс «подання» (IView). Зміна стану «моделі подання» автоматично змінює «подання» і навпаки, оскільки використовується механізм скріплення даних (Bindings);

– одна «модель подання» пов'язана з одним поданням.

При використанні цього шаблону «подання» не реалізує відповідний інтерфейс (IView). «Подання» має містити посилання на джерело даних (DataContex), яким у даному випадку є View-модель. Елементи уявлення пов'язані з відповідними властивостями і подіями View-моделі. У свою чергу, View-модель реалізує спеціальний інтерфейс, який використовується для автоматичного оновлення елементів подання. Прикладом такого інтерфейсу в WPF може бути INotifyPropertyChanged [4].

Однією з переваг поділу коду є те, що код стає більш легким для розуміння. Це відбувається завдяки тому, що код для певних компонентів може залишатися окремим від іншого коду, що дозволяє отримувати про нього більше відомостей, а також повторно використовувати в інших проектах. Структура класів у проекті, що реалізує шаблон MVVM виглядає зазвичай так, як зазначено на рис. 2.

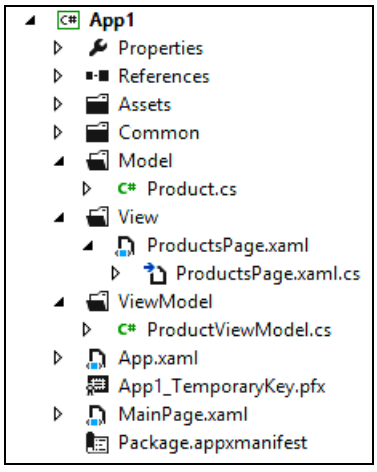

Рис. 2. Приклад структури програми на основі шаблону проектування MVVM

Ще однією важливою перевагою поділу користувацького інтерфейсу є те, що це спрощує автоматичне модульне тестування коду, не пов'язаного з інтерфейсом, порівняно з перевіркою без поділу. Microsoft Visual Studio підтримує проекти модульних тестів, які можна використовувати для перевірки конструкції коду під час розробки, а також при виявленні та діагностиці помилок [3].

В цілому надійно пов'язана архітектура ускладнює внесення змін та діагностику помилок. Основною перевагою розділеної архітектури є те, що вона ізолює вплив змін. Це дозволяє менш ризиковано експериментувати з новими можливостями, виправляти помилки і впроваджувати внесок співавторів [5].

Висновки

Шаблон MVVM – простий і ефективний набір рекомендацій для проектування та реалізації додатків. Він дозволяє розділяти дані, поведінку і подання, а значить, контролювати той хаос, який називається розробкою програмного забезпечення.

Слід зазначити, що чітко дотримуватися тільки одного шаблону проектування – не завжди найкращий вибір. Наприклад, використовувати MVVM для розробки додатків на основі Windows Forms через властивості елементів управління Bindings.

Основна мета – це відокремити подання від бізнес-логіки і логіки, яка їх пов'язує. Застосування, з одного боку, повинно легко тестуватися і підтримуватися, а з іншого бути зрозумілим для аналітиків.

Список літератури

1. Паттерни для новичків: MVC vs MVP vs MVVM [Електронний ресурс]. – Режим доступу: http://it-ua.info/news/2014/03/17/patterni-dlya-novichkv-mvc-vs-mvp-vs-mvvm.html.

2. Основы MVVM Pattern'a [Електронний ресурс]. – Режим доступу: http://svyatoslavpankratov.blogspot.com/2011/11/ mvvm-pattern-1.html.

3. Паттерны для новичков: MVC vs MVP vs MVVM [Електронний ресурс]. – Режим доступу: http:// http://habrahabr.ru/post/215605.

4. Приложения WPF с шаблоном проектирования модель-представление-модель представления [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ru-ru/magazine/dd419663.aspx.

5. Шаблон MVVM. Его реализация в bpm'online [Електронний ресурс]. – Режим доступу: http:// academy.terrasoft.ru/documents/docs/technic/SDK/7.4.0/ MVVM PatternInBPMonline.html.

6. Model-View-Controller в .Net [Електронний ресурс]. – Режим доступу: http://rsdn.ru/article/patterns/ModelViewPresenter.xml.

7. Паттерны: MVC, MVP и MVVM [Електронний ресурс]. – Режим доступу: http://outcoldman.com/ru/archive/2010/02/22/паттерны-mvc-mvp-u-mvvm.

8. Создание MVVM-приложений с помощью Xamarin и MvvmCross [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ru-ru/magazine/dn 759442.aspx.

9. Использование шаблона Model-View-ViewModel (MVVM) [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ru-ru/library/windows/apps/jj883732.aspx.

Надійшла до редколегії 2.02.2015

Рецензент: д-р техн. наук, проф. Е.Г. Петров, Харківський національний університет радіоелектроніки, Харків.

ИСПОЛЬЗОВАНИЕ ШАБЛОНА ПРОЕКТИРОВАНИЯ MVVM ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЙ НА ПЛАТФОРМАХ MICROSOFT

З.В. Батюк, А.В. Тарасов

В работе рассмотрены вопросы использования шаблона проектирования MVVM при разработке приложений на платформах Microsoft. Изложены основные принципы организации данного шаблона и его составляющие. Анализируются возможности на платформах, которые используют язык разметки XAML, и его применение для разработки корпоративных приложений.

***Ключевые слова**: шаблон проектирования, модель, представление, модель представления, MVVM, WPF, XAML, разделение кода, архитектура приложения.*

USING DESIGN PATTERN MVVM IN APPLICATION DEVELOPMENT ON THE MICROSOFT'S PLATFORMS

Z.V. Batiuk, O.V. Tarasov

In this article, we are discussing about using MVVM design pattern for application development on Microsoft's platforms, the basic principles of this pattern and its components, the possibilities on platforms that use markup language XAML, and its use for the development of enterprise applications.

***Keywords:** pattern design, model, view, view-model, MVVM, WPF, XAML, code separation, application architecture.*