УДК 004.42

ПОРІВНЯЛЬНИЙ АНАЛІЗ РОЗРОБКИ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ З КОМПОНЕНТНОЮ АРХІТЕКТУРОЮ І НАТИВНИМИ МОВАМИ ПРОГРАМУВАННЯ

А.І. Шаренко, магістрант
Київський національний університет технологій та дизайну
О.З. Колиско, кандитат технічних наук., доцент
Київський національний університет технологій та дизайну

Ключові слова: інтерфейс користувача, компонентна архітектура, нативна розробка, порівняльний аналіз.

Переважна більшість програмного забезпечення (далі – ПЗ), що розробляється на сьогоднішній день, має розвинутий інтерфейс користувача (англ. User Interface - UI), який дозволяє не тільки задавати налаштування роботи цього ПЗ, а й спілкуватися з ним у реальному часі при використанні за прямим призначенням. Зважаючи на велику конкуренцію майже у всіх сегментах сучасної галузі розробки ПЗ, нерідко більш ергономічний інтерфейс користувача стає вирішальним фактором, що спричинює перевагу на ринку того, чи іншого програмного продукту. Також слід відмітити, що конкурентоспроможність на рівні проектів також у великій мірі обумовлена мінімізацією витрат на різні етапи розробки ПЗ, причому одним із таких етапів є прототипування UI та його подальша реалізація у програмних кодах. Очевидно, що дизайнер, який виконує розробку зовнішнього вигляду програми, повинен орієнтуватися на технічні можливості по реалізації того, чи іншого елементу. Таким чином, із самого початку зовнішній вигляд залежить від підходу до розробки UI, який є прийнятим для певного проекту. Відповідно, можна сказати, що вибір між підходом до зведення інтерфейсу користувача є надзвичайно важливим (оскільки впливає і на популярність, і на економічні показники і т.д.) і тому має бути обґрунтований за певними критеріями. У даній роботі наводиться аналіз подібних критеріїв та особливостей двох основних підходів до зведення UI – компонентного та нативної розробки.

При компонентному проектуванні розробник спирається на певну сукупність стандартних елементів інтерфейсу, оформлених у вигляді порівняно незалежних (інкапсульованих) сутностей – компонентів. Нативна розробка навпаки передбачає використання низькорівневих функцій (можливо, об'єднаних у класи – при використанні об'єктно-орієнтованих фреймворків та бібліотек). Із цих особливостей витікають усі переваги й недоліки кожного з цих двох основних підходів до зведення інтерфейсу користувача, які розглянемо докладніше.

Найпершим показником є швидкість розробки інтерфейсу, яка є на порядок кращою у компонентного підходу, адже усі елементи управління уже є готовими і їх додавання до вікна програми часто зводиться до простого перетягування мишею з подальшим налаштуванням властивостей

у графічному режимі. Таким чином, більш дешевим є використання бібліотек компонентів, у порівнянні з нативною розробкою, що в середньому потребує значно більшої кількості людино-годин для розробки аналогічного за виглядом інтерфейсу.

Однак, наявність сукупності стандартних компонентів спричинює порівняно низьку гнучкість компонентного підходу у побудові UI. Дійсно, з використанням нативної розробки (наприклад, при створенні інтерфейсу на основі API-функцій ОС Windows, що є прикладом нативного програмування для платформи комп'ютерів архітектури x86-64) використовуються самі низькорівневі можливості, що дозволяють реалізувати будь-який елемент управління, якого немає у доступних бібліотеках компонентів. При цьому специфіка вигляду та поведінки даного елемента управління може налаштовуватися як завгодно докладно.

Наступним критерієм для порівняння може бути кросплатформенність ПЗ, побудованого на базі двох описуваних підходів. Тут очевидно, що нативні програмні продукти є призначеними для певної конкретної платформи і не можуть бути адаптовані (без суттєвої переробки) до іншої (наприклад, програма, написана на API-функціях Windows, не може бути адаптована для Linux). Що ж стосується компонентної програми, то можливість її швидкої адаптації залежить від зусиль розробників бібліотеки компонентів: якщо вони реалізували її версію під якусь платформу, то і програма, написана на основі даної бібліотеки, легко портуватиметься на усі такі платформи.

Не зважаючи на низьку кросплатформенність, нативні програми мають суттєву перевагу у швидкості своєї роботи, адже компонентні програми використовують проміжний програмний шар, а тому принципово будуть працювати повільніше у порівнянні з нативним ПЗ.

Таким чином, можна стверджувати, що у більшості випадків розробки більш-менш стандартного програмного забезпечення компонентне програмування буде переважним підходом для зведення інтерфейсу користувача (через його значно меншу трудомісткість). В той же час, якщо програма потребує нестандартних елементів управління, або існують особливі вимоги до її швидкості виконання, то доцільно переходити до нативної розробки її інтерфейсу.

Список використаних джерел

1. Lane, T. A Design Space and Design Rules for User Interface Software Architecture // Technical Report CMU/SEI-90-TR-022. [Електронний ресурс]. Доступно: https://insights.sei.cmu.edu/library/a-design-space-and-design-rules-for-user-interface-software-architecture/ (дата звернення: 23.10.2024).

2. Richard N. Taylor, Nenad Medvidovic, Kenneth M. Anderson, E. James Whitehead, and Jason E. Robbins. A component- and message-based architectural style for GUI software // In Proceedings of the 17th international conference on Software engineering. Association for Computing Machinery, New York, NY, USA, 295–304. https://doi.org/10.1145/225014.225042