ПРИЧИНИ ПОЯВИ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

Т.В. Тюріна

Україна, Запорізька обл., м. Мелітополь, Мелітопольська загальноосвітня школа І-ІІІ ступенів №7

вчитель інформатики І категорії

Об'єктно-орієнтоване програмування стало надзвичайно популярно в останні роки. Виробники програмного забезпечення стрімголов кидаються створювати об'єктно-орієнтовані версії своїх продуктів. З'явилася незліченна кількість книг і спеціальних журналів, присвячених цьому предмету. Всі прагнуть до запису "компетентний в об'єктно-орієнтованому програмуванні" в своїх характеристиках.

Коли програмісти запитують один одного: "Чим же, зрештою, є об'єктно-орієнтоване програмування?", - відповідь найчастіше акцентується на синтаксичних властивостях таких мов, як C++ або Object Pascal, порівняно з їх попередніми, не об'єктно-орієнтованими версіями, тобто C або Pascal. Але при цьому нехтують найбільш важливим моментом в об'єктно-орієнтованому програмуванні, що не має нічого спільного із питаннями синтаксису. Використання об'єктно-орієнтованої мови не є ні необхідною, ні достатньою умовою для того, щоб займатися об'єктно-орієнтованим програмуванням. Найбільш важливий аспект в ООП - техніка проектування, заснована на виділенні й розподілі обов'язків між компонентами системи.

Об'єктно-орієнтоване програмування - це не просто кілька нових властивостей, доданих до вже існуючих мов. Швидше це новий крок в осмисленні процесів декомпозиції задач і розробки програмного забезпечення. Для того, щоб ефективно використовувати ООП, потрібно дивитися на світ інакше. Власне - застосування об'єктно-орієнтованої мови програмування (такої як C++) не примушує ставати об'єктно-орієнтованим програмістом.

Головні причини (на думку Бадда [1]) *величезної популярності об'єктно-орієнтованого програмування протягом останніх десятиліть полягають в:*

1. Надії, що ООП може просто й швидко спричинити зростання продуктивності й поліпшення надійності програм, допомагаючи тим самим розв'язати кризу в програмному забезпеченні;

2. Можливості створювати великі програмні компоненти, придатні для повторного використання.

Багаторазове використання програмного забезпечення – мета, якої постійно прагнуть, проте зрідка досягають. Основна причина цього – значна взаємозалежність більшої частини програмного забезпечення, створеного традиційними способами. Проблематично витягти із проекту фрагменти програмного забезпечення, що було би легко використовувати в новому програмному продукті (кожна частина коду зазвичай пов'язана з іншими фрагментами). Як уже відзначено, ООП частково вирішує цю проблему.

Сучасна програмна індустрія розробляє системи для розв'язання найрізноманітніших задач. Визначальна риса таких систем – рівень складності: один розроблювач практично не в змозі охопити всі аспекти такої системи. Складність промислових програм перевищує можливості людського інтелекту...

На світанку інформатики більшість програм створювалися на асемблері. Вони не відповідають сьогоднішнім стандартам. По мірі того як програми ставали все складнішими, розроблювачі виявили, що вони не в змозі пам'ятати всю інформацію, необхідну для налагодження та удосконалювання їхнього програмного забезпечення. Які значення перебувають у регістрах? Чи вступає новий ідентифікатор у конфлікт із визначеними раніше? Які змінні необхідно ініціалізувати перед тим, як передати керування наступному коду?

Поява таких мов програмування високого рівня, як Fortran, Cobol і Algol, розв'язало деякі проблеми (було введено автоматичне керування локальними змінними та неявне присвоювання значень). Одночасно зросла віра користувачів у можливості комп'ютера. По мірі того, як з'являлися спроби розв'язати все більш складні проблеми з його використанням, виникали ситуації, коли навіть кращі програмісти не могли утримати все в своїй пам'яті. Звичними стали команди програмістів, які працюють разом.

У той час, як програмні проекти ставали все складнішими, було підмічено цікаве явище. Завдання, для розв'язання якого одному програмістові було потрібно два місяці, не розв'язувалася двома програмістами за один місяць.

Причиною такого нелінійного поводження є *складність.* Зокрема, взаємозв'язки між програмними компонентами стали складніше, і розроблювачі змушені були постійно обмінюватися між собою значними об'ємами інформації.

Складність породжує не просто великий обсяг розглянутих задач, а унікальна властивість

програмних систем, розроблених з використанням традиційних підходів, — велика кількість перехресних посилань між компонентами (саме це робить їх одними з найбільш складних людських витворів). Перехресні посилання в цьому випадку наочно демонструють залежність одного фрагмента коду від іншого.

До появи об'єктно-орієнтованої технології програмування вважалося, що можливості програмістів обмежені наступними об'ємами програмного коду [2] (див. табл. 1.1.):

З появою об'єктно-орієнтованої технології програмування можна сміливо стверджувати, що можливості програмістів збільшилися на порядок! Практично це означає, що один програміст в змозі створювати програмні продукти розміром до 100 000 рядків тексту. Звичайно, ОО технологія створювалася саме для використання великими колективами програмістів, але той факт, що можливості людини суттєво розширено, поза всяким сумнівом надихає і кожного окремого фахівця.

Однак, незважаючи на те, що об'єктно-орієнтоване програмування дійсно допомагає при створенні складних програмних систем, важливо пам'ятати, що ООП не є "срібною кулею" (термін набув популярності завдяки Фреду Бруксу), що легко справляється із чудовиськом. Програмування, як і раніше, є однією з найбільш важких задач, що звалюють на свої плечі люди. Для того, щоби стати професіоналом у програмуванні, необхідні талант, здатність до творчості, інтелект, знання, логіка, вміння будувати й використовувати абстракції й, найголовніше, досвід - навіть у тому випадку, коли використовуються вдосконалені засоби розробки.

Література

1. Бадд Т. Оъектно-ориентованное программирование. – СПб.: Питер, 1997. – 464с.

2. Єремєєв В.С., Тюрін О.Г., Тюріна Т.В. Об’єктно-орієнтоване програмування. Навчальний посібник. – Київ: Фітосоціоцентр, 2006. – 150с.