Куліковська Оксана,
магістрант фізико-математичного факультету;
Вакалюк Тетяна Анатоліївна,
кандидат педагогічних наук, доцент
(Житомирський державний університет імені Івана Франка)

ПЕРЕВАГИ ЗАСТОСУВАННЯ ІСНУЮЧИХ МОДЕЛЕЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У РЕАЛЬНОМУ ПРОЕКТІ

Бурхливий розвиток науково-технічного прогресу і комп'ютерної техніки зумовлюють потребу розробляти нові підходи створення програмного забезпечення. У, процесі узагальнення та систематизації знань, а також відбору оптимальних способів програмування з метою одержання програмної продукції із заданими властивостями все більшого розвитку набувають нові технології програмування. Вони, в свою чергу, вимагають знань, правил, умінь, навичок та засобів, що дозволяють одержувати гарантований якісний результат. '

Сучасні технології проектування програмного забезпечення (ПЗ) в першу чергу, спрямовані на часткову автоматизацію етапів і на суміщення їх у часі задля скорочення термінів виконання проектів. Процес створення та використання програмної системи називається життєвим циклом ПЗ і включає декілька етапів:

1 Етап специфікації вимог - етап, у якому готується повне і чітке визначення задачі, документи з вимогами до задачі користувачам і аналітикам для утвердження.

2. Етап аналізу відповідає за вивчення задачі, оцінку альтернативних методів розв'язання, вибір оптимального методу.

3. Етап проектування - визначає структуру програмної системи, її проектування та розбиття на окремі компоненти, виділення ключових елементів структури даних.

4. Реалізація - етап, що визначає розробку алгоритмів і кодів для окремих модулів, використовуючи вибрану мову програмування, створення вихідного коду програми та його налагодження.

5. Тестування і верифікація - етап, на якому тестується вихідний текст за участю користувачів і тестерів.

6. Шостий етап - експлуатація і супроводження - визначає використання готової програмної системи, оцінку її ефективності, усунення помилок, що були знайдені у процесі експлуатації, корегування програмної системи за необхідністю, перевірка правильності внесених змін [І].

Етапи дають уявлення про розробку ПЗ, проте в силу деяких обставин не кожен проект і його розробка може повністю їм відповідати. Наприклад, на етапі проектування можуть виникнути проблеми, що будуть вимагати змін специфікації вимог, у процесі реалізації може виникати необхідність переглянути результати, що були отримані під час розробки, етап тестування може виявити помилки тощо. Тому етапи можуть застосовуватися повторно для покращення якості розроблюваного програмного продукту.

Залежно від вимог, що ставляться до програмного продукту, інженери-програмісти створили моделі життєвого циклу програмного продукту. Найбільш відомими є каскадна модель, спіральна модель та поетапна модель з проміжним контролем.

Каскадна модель (інколи водоспадна) базується на послідовному використанні різних видів діяльності, включаючи кожен етап, як початковий - вироблення вимог, так і завершальний етап супроводу, де чітко визначені межі

між етапами, на яких набір документів, створений на попередній стадії, передається в якості вхідних даних для наступної. При цьому на певному етапі життєвого циклу виконується відповідний вид діяльності. У "класичній" каскадній моделі передбачається лише рух вперед за даною схемою: все необхідне для проведення чергової діяльності має готуватися в ході попередніх робіт.

Спіральна модель вимагає на кожному витку спіралі випускати чергову версію продукту, уточнювати вимоги до проекту, забезпечувати якість і розробляти план роботи наступного витка Особлива увага приділяється початковим етапам розробки - аналізу та проектуванню, де ведеться перевірка і обгрунтування реалізації тих чи інших технічних рішень за допомогою створення прототипів [2].

Поетапна модель із проміжним контролем передбачає розробку програмного продукту з веденням ітерацій з циклами зворотного зв'язку між етапами. Міжетапні корегування існують для врахування реально існуючих взаємовпливів на результати розробки різних етапів; час життя кожного з етапів триває впродовж всього періоду розробки.

Серед поданих моделей каскадний вимагає на кожному етапі формувати закінчену проектну документацію відповідно до критеріїв повноти та узгодженості, що надає моделі велику перевагу. На перших кроках є можливість формулювання усіх вимог до системи якомога точніше і в повному обсязі. Щоправда, тут є недолік - схема жорстка, що майже не дає змоги вкластися в реальному процесі, постійно виникає потреба в поверненні до попередніх етапів і уточнення або перегляд раніше прийнятих рішень. Таким чином, процес виявляється аналогічним до поетапної моделі з проміжним контролем. Однак ця схема оперативно не враховує уточнення вимог до розробки та зміни, що виникають. Лише після завершення кожного етапу роботи формується точка та узгоджуються результати розробок з користувачами, а загальні вимоги залишаються зафіксованими у вигляді технічного завдання на весь час її створення [3]. Отже, реальні потреби користувачів часто не відповідають отриманій системі. Проте найбільш коректною варто вважати спіральну.

Спіральна модель життєвого циклу, що була запропонована для подолання проблем попередніх методів, на етапах аналізу і проектування перевіряє реалізацію технічних рішень і ступінь задоволення потреб замовника шляхом створення прототипів. Кожен виток спіралі забезпечує працездатність створеного фрагмента або версії програми. Це дозволяє уточнити вимоги, цілі й характеристики проекту, визначити якість розробки, спланувати роботу наступного витка спіралі.

Таким чином поглиблюються та послідовно конкретизуються деталі проекту і в результаті вибирається обгрунтований варіант, який відповідає дійсним вимогам замовника та доводиться до реалізації.

Ітеративна розробка відбиває об'єктивно існуючий спіральний цикл мтворення складних систем. Вона дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному, і вирішити головне завдання - якомога швидше показати користувачам системи працездатний продукт. тим самим активізуючи процес уточнення та доповнення вимог [1].

Основна проблема спірального циклу —визначення моменту переходу на наступний етап. Для її вирішення вводяться тимчасові обмеження на кожен з етапів життєвого циклу, і перехід здійснюється відповідно до плану, навіть якщо не вся запланована робота закінчена. Планування виробляється на основі статистичних даних, отриманих у попередніх проектах, та особистого досвіду розробників.

Застосування моделі життєвого циклу допомагає створити програмний продукт зі збереженням функціональності, надійності, легкості застосування та ефективності. Ці критерії визначають якість програмного продукту, тому варто якомога доцільніше вибирати модель.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загальносистемні принципи та етапи створення програм. Життєвий цикл програмного виробу [Електронний ресурс]. - Режим доступу: http://lib.rndpu.org.ua2e-book/vstup/L7.htm

2. Управління життєвим циклом [Електронний ресурс]. — Режим доступу: http://uudoc.zavantag.com/text/l 8584/index-1.html

3. Основні моделі життєвого циклу ПЗ [Електронний ресурс]. - Режим доступу: http://5fan.ru/wievjob.php?id=21297