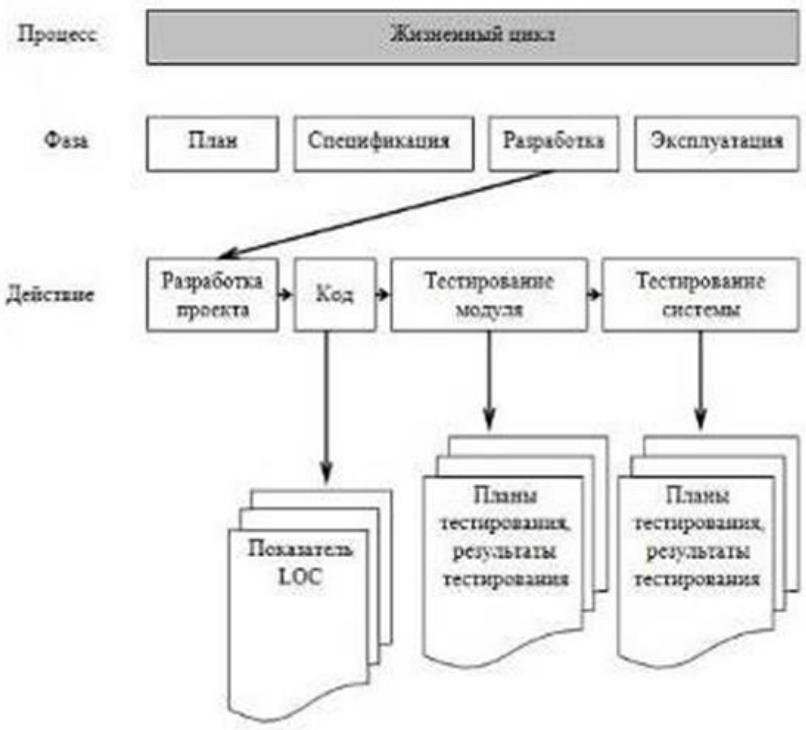
Модель життєвого циклу програмного проекту. Каскадна модель

Одним з ключових понять управління програмними проектами є життєвий цикл (ЖЦ) проекту.

Модель життєвого циклу проекту (Software life cycle model, SLCM) - це структура, що визначає послідовність виконання та взаємозв'язку процесів, дій та задач на протязі виконання проекту. Модель життєвого циклу залежить від специфіки умов, в яких даний продукт створюється та функціонує.

Стандарт ISO/IEC 12207 не пропонує конкретну модель ЖЦ та методи розробки програмного забезпечення(ПЗ), він описує структуру процесів ЖЦ ПЗ, але не конкретизує в деталях як реалізовувати чи виконувати дій чи задачі, що включені в ці процеси.

Тобто модель ЖЦ розробки ПЗ схематично пояснює яким чином будуть виконуватися дії з розробки програмного продукту, шляхом опису "послідовності" цих дій. Така послідовність може бути або не бути лінійною, оскільки фази можуть слідкувати одна за одною, повторюватися або відбуватися послідовно.



Існує два основних типи моделей життєвого циклу:

1. прогнозовані моделі життєвого циклу - в основи цих моделей лежить чітке планування усіх стадій процесу розробки програмного забезпечення
2. адаптивні моделі життєвого циклу (так звані гнучкі технології) - особливістю цих моделей є сприйняття та адаптація процеси розробки під потреба замовника та ринку.

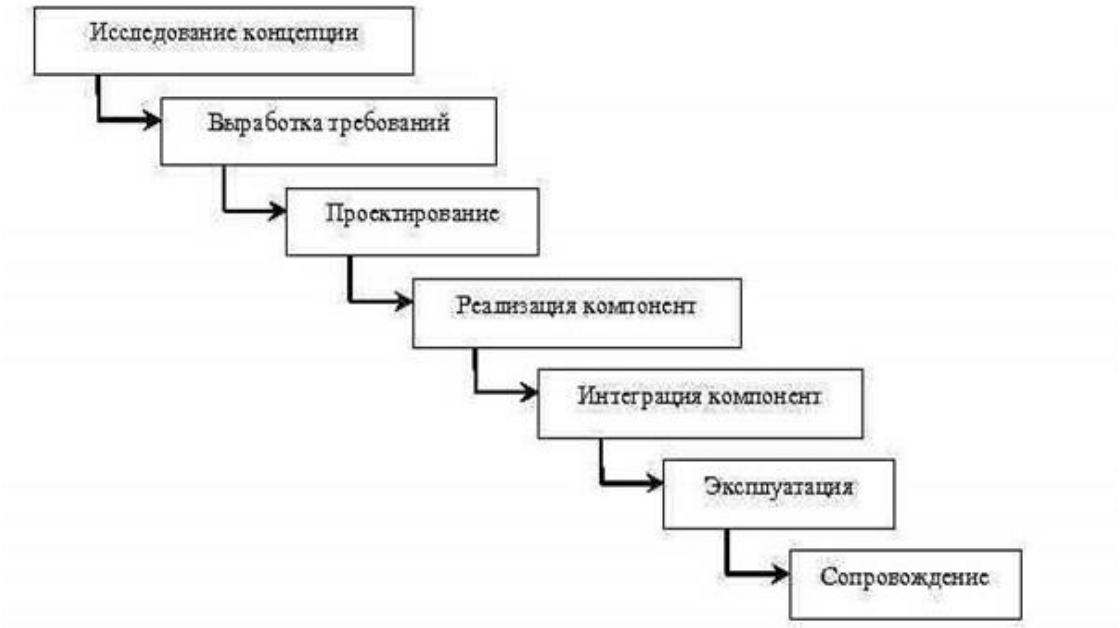
На сьогоднішній момент найбільшого поширення серед прогнозованих моделей набули такі:

1. каскадна модель;
2. V-подібна модель;
3. модель еволюційного прототипування;
4. модель швидкої розробки;
5. інкриментна модель;
6. спіральна модель;

Адаптивні моделі життєвого циклу з’явилися досить недавно, але здобули вже досить широке поширення особливо серед західних компаній.Найвідомішими є

1. Scrum
2. Extreme Programming, XP
3. Adaptive Software development, ASD
4. Dynamic System Development Model, DSDM
5. Feature Driven Development, FDD

Адаптивні моделі життєвого циклу включають не тільки опис структури фаз, та їх пояснення, але і рекомендації, підходи з їх ефективного використання.



1.2 Особливості каскадної моделі розробки програмного забезпечення

Каскадна модель життєвого циклу була розроблена Вінстоном Ройсом в 1970 році та вперше представлена в його книжці “Управління розробкою великих програмних систем”.

Характерною рисою каскадної моделі можна назвати те, що вона

представляє собою формальний метод, різновид розробки “зверху-вниз”, вона складається з незалежних фаз, що виконуються послідовно.

Продовження процесу виконання реалізується за допомогою впорядковано послідовності кроків. В моделі передбачено, що кожна наступна фаза починається лише тоді коли повністю завершено виконання попередньої фази. Кожна фаза має певні критерії входу та виходу: вхідні та вихідні дані. В результаті виконання генеруються внутрішні та зовнішні дані проекту, включаючи документацію та ПЗ. Документи з аналізу вимог згодом передаються системним спеціалістам, які в свою чергу передають їх розробникам систем більш високого рівня.

Розглянемо детальніше фази каскадної моделі:

1. **дослідження концепції** - відбувається дослідження вимог на системному рівні ціллю визначення можливості реалізації концепції;

2. **процес системного розподілення** - може бути пропущений для систем, що стосуються виключно розробки програмних продуктів. Для систем, в яких необхідна розробка як апаратного, так і програмного забезпечення, необхідні функції застосовуються до програмного продукту і обладнання у відповідності з загальною архітектурою системи;
3. **процес визначення вимог** - визначаються програмні вимоги до інформаційної предметної області системи, призначення, лінії поведінки, продуктивності та інтерфейси. В разі необхідності в процес також включено функціональний розподіл системних вимог щодо апаратного і програмного забезпечення
4. **процес розробки проекту** - розробляється і формулюється логічно послідовна технічна характеристика програмної системи, включаючи структури даних, архітектуру програмного забезпечення, інтерфейси представлення та алгоритмічну деталізацію
5. **процес реалізації** - в результаті виконання цієї фази ескізний опис ПЗ перетворюється в повноцінний програмний продукт. При цьому створюється вод, баз даних та документація, що лежать в основі фізичного перетворення проекту. Якщо програмний продукт являє собою придбаний пакет прикладних програм, то основними діями

щодо його реалізації є установка та тестування пакету програм. Якщо програмний продукт розробляється на замовлення, то основними діями є програмування та тестування

6. **процес установки** - включає в себе установку ПЗ, його перевірку та офіційне затвердження замовником.
7. **процес експлуатації та підтримки** - дана фаза включає в себе запуск користувачем системи та поточну підтримку, що включає в себе надання технічної допомоги, обговорення виниклих у користувача питань, реєстрація запитів користувача щодо модернізації та внесення змін, а також коректування та усунення помилок
8. **процес супроводу** - дана фаза пов’язана з усуненням помилок, недоліків, з ітерації розробки і передбачає зворотний зв’язок для надання інформації щодо встановлених дефектів.
9. **процес виводу з експлуатації** - виведення існуючої системи з її активного використання шляхом припинення її роботи, або завдяки її заміни новою системою або модернізованою версією вже існуючою системи;
10. **інтегральні задачі** - включають початок роботи над проектом, моніторінг проекту та його управління, управління якістю, верифікацію та атестацію, менеджмент конфігурації, розробку документації та професійну підготовку на протязі всього життєвого циклу.

Перехід від однієї фази до іншої здійснюється за допомогою формального огляду. Таким чином, клієнт отримує загальне представлення про процес розробки, крім того проходить перевірка якості програмного продукту. Як правило проходження стадії огляду вказує на домовленість між командою розробників і клієнтом про те, що фаза завершена і можна перейти до виконання наступних фази.Завершення фази зручно сприймати за стадію в процесі виконня проекту.

В результаті завершення певних фаз формується базова лінія, яка в даній точці “заморожує” продукти розробки. Якщо виникає

потреба в їх зміні, тоді для внесення змін використовується формальний процес змін.

1.2.1 Аналіз переваг та недоліків застосування каскадної моделі

Каскадна модель набула дуже широкого поширення і використовується на практиці вже більш ніж 30 років. За всі ці роки використання дана модель здобула багато прихильників і багато противників.

Проаналізуємо дану модель та виділимо її основні слабкі та сильні сторони.

Основні переваги даної систем:

1. дана модель є простою та зрозумілою, вимогою для успішного завершення программного продукту є виконання запланованих дій; фази моделі гарно визначені;
2. каскадна модель дозволяє учасникам проекту, що закінчили свої фази, брати участь в інших проектах;
3. зручна для використання менеджером проекту: сприяє здійсненню строго контролю над перебігом проекту, полегшує роботу над розробкою проектного плану;
4. забезпечує гарний контроль над якістю проекту. Усі закінчені фази та отримані дані підлягають огляду. Ця процедура забезпечує визначення якості програмної системи.

Розглянемо тепер основні недоліки даної моделі:

1. значне збільшення затрат та значні зміни в плануванні при необхідності виправити помилку в одній з попередніх фаз. Оскільки дана модель має лінійну структуру, то при бажанні повернутися на якусь з попередніх фаз. Оскільки дана модель має лінійну структуру, то при бажанні повернутися на якусь з попередніх фаз для виправлення знайдених помилок, необхідно буде пройти всі ці фази ще раз;

1.2.2 Область застосування каскадної моделі

Через недоліки каскадної моделі її застосування необхідно обмежувати ситуаціями, в

яких вимоги та їх реалізація максимально чітко визначені та зрозумілі.

Каскадна модель гарно функціонує при її застосуванні в циклах розробки програмного

продукту, в яких використовується незмінне визначення продукту та зрозумілі технічні

методики.

Якщо компанія має досвід побудови визначеного виду систем - автоматизованого

бухгалтерського обліку, нарахування зарплати тощо - тоді в проекті, орієнтованому на

побудову ще одного продукту того ж типу, можливо навіть з використанням вже існуючих

розробок, то в цьому випадку можна ефективно використовувати каскадну модель,

Іншим прикладом застосування каскадної моделі може бути створення та випуск нової

версії вже існуючого продукту, якщо внесені зміни визначені та керовані. Перенесення вже

існуючого проекту на нову платформу часто приводять в якості ідеального приклада

використання каскадної моделі в проекті.

Незважаючи на справедливість критики цієї моделі все ж варто визнати, що

модифікована версія каскадної моделі є в значній степені менш жорсткою, ніж її первинна

форма. Тут включаються ітерації між фазами, паралельні фази та менеджмент змін. Не

дивлячись на те, що модифікована каскадна модель є значно гнучкішою, ніж класична

модель, вона все ж не є найкращим вибором для проектів з пришвидшеною розробкою,

Каскадні моделі на протязі всього часу їх існування використовуються для виконання

великих проектів, в яких задіяна декілька великих команд розробників