academy

Довідник з QA Manual

Урок 3. SDLC і STLC. Методології розробки ПЗ

Beetroot Academy

Яку теорію опануємо:

1. Життєвий цикл програмного забезпечення. Етапи розробки ПЗ
2. Активності та етапи процесу тестування
3. Послідовні та ітеративні методології розробки ПЗ
4. Scrum та Kanban

1. Життєвий цикл програмного забезпечення. Етапи розробки ПЗ

Життєвий цикл розробки програмного забезпечення (Software Development Life Cycle) — це структура, яка визначає кроки в розробці програмного забезпечення на кожному етапі й охоплює детальний план побудови, розгортання та обслуговування програмного забезпечення. SDLC визначає повний цикл розробки, тобто всі завдання, пов'язані з плануванням, створенням, тестуванням та розгортанням програмного продукту. [1]

Фази SDLC:

1. **Збір та аналіз вимог (Requirement gathering and analysis)** — це процедура проведення всебічного аналізу висунутих замовником вимог до створюваного ПЗ, щоб визначити ключові цілі та завдання кінцевого продукту. У рамках цієї стадії відбувається максимально ефективна взаємодія клієнта і співробітників компанії-розробника. Результатом проведеного аналізу є формування основного регламенту, на який спиратиметься виконавець у своїй роботі — специфікації (Software Requirement Specification). SRS має повністю описувати поставлені перед розробником завдання та описати кінцеву мету проєкту у розумінні замовника.

2. **Дизайн (Design)** — на цьому етапі проєкт системи та програмного забезпечення готується на основі специфікацій вимог, які вивчалися попередньо. Проєктування системи допомагає визначити вимоги до обладнання та системи, а також її загальну архітектуру.

3. **Впровадження або кодування (Implementation or coding)** — після отримання проєктної документації системи, робота поділяється на модулі/блоки й починається фактичне кодування. Розробники зосереджуються на написанні коду всіх функцій програми. Це найдовша фаза життєвого циклу розробки програмного забезпечення.

4. **Тестування (Testing)** — після написання коду для функціонала, його перевіряють на відповідність вимогам. Мета цього етапу — переконатися, що продукт дійсно вирішує потреби, які розглядаються та збираються на етапі збору вимог. Цей етап передбачає проведення всіх видів функціонального тестування, як-от модульного, інтеграційного, системного, приймального, а також нефункціональних видів тестування.

5. **Розгортання (Deployment)** — після успішного тестування продукт доставляється/розгортається замовнику для використання.

6. **Технічне обслуговування (Maintenance)** — це процес, коли команда розробки технічно обслуговує розроблений продукт, оновлює його та виправляє проблеми, якщо такі виникають. [3]

7. **Закриття проєкту** — цей етап не завжди трапляється під час розробки, але іноді коли проєкт перестає існувати, команді необхідно подбати про його коректне відключення від апаратних складових та інтеграцій. Також можлива ситуація передачі проєкту в іншу компанію або команду — у такому випадку необхідно підготувати всю документацію та іншу необхідну інформацію для подальшої роботи над ПЗ.

2. Активності та етапи процесу тестування

Життєвий цикл тестування програмного забезпечення (STLC, Software Testing Life Cycle) — це послідовність конкретних дій, що проводяться під час процесу тестування для забезпечення досягнення цілей якості програмного забезпечення. STLC включає як верифікацію, так і валідацію. Він складається з низки заходів, які проводяться методично, щоб допомогти сертифікувати програмний продукт. [4]

Процес тестування складається з таких активностей:

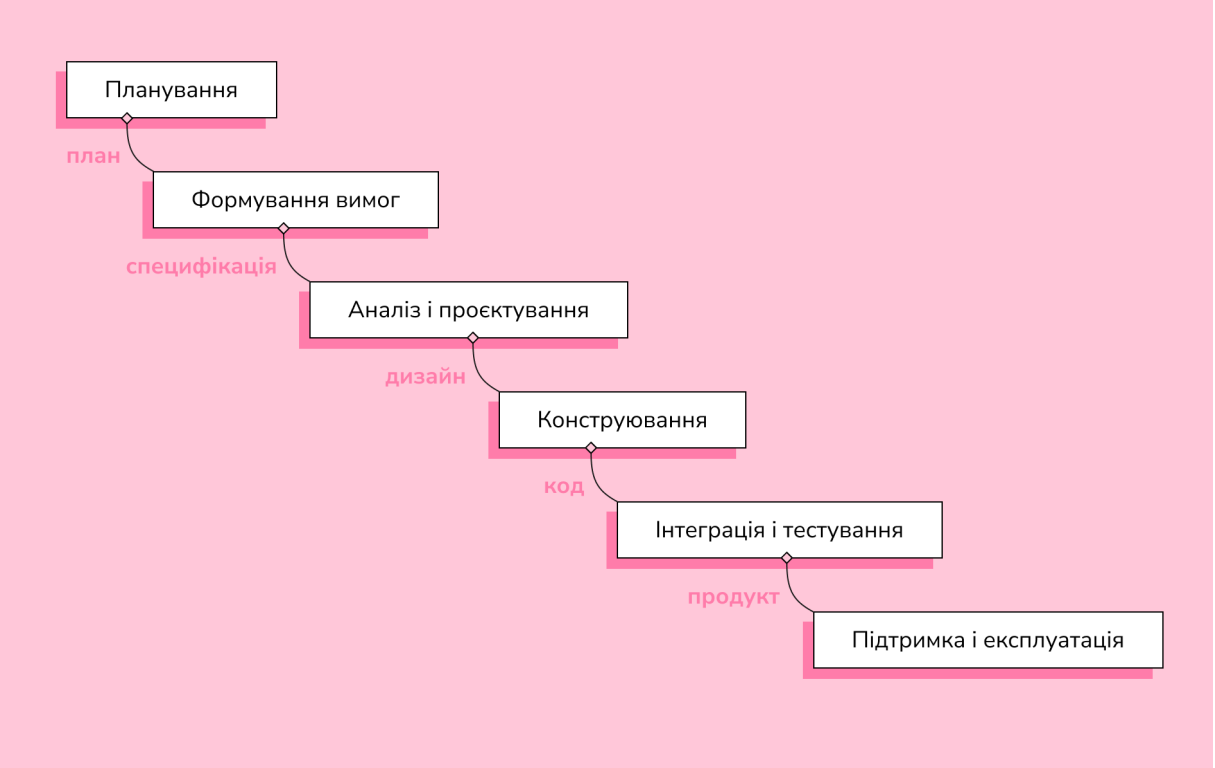
1. **Планування тестування (Test planning)** передбачає активності, які визначають цілі тестування та підходи для досягнення цих цілей (визначення відповідних методів та підходів тестування, а також його графіку). Тест-план (документ, що формується в результаті цього етапу) може бути переглянутий та змінений під час моніторингу та контролю.
2. **Аналіз (Test Analysis)** ПЗ та вимог до нього потрібен для визначення функцій, які необхідно буде протестувати, та встановлення відповідних тестових умов. Загалом тест-аналіз вирішує «що тестувати?».
3. **Тест-дизайн (Test Design)** — це процес формування тестових випадків (кейсів) на основі вимог з етапу тест-аналізу, які будуть тестуватись. Тест-дизайн відповідає на запитання «як тестувати?»
4. **Реалізація (Test Implementation)** містить такі заходи: розробка тестових випадків та розставлення їх пріоритетності, створення автоматизованих сценаріїв тестування, налаштування тестового середовища та підготовка необхідних тестових даних.
5. **Виконання тестування (Test execution)** — це запуск та виконання тестових сценаріїв, заведення на аналіз знайдених дефектів, а також формування звітності за результатами виконаних робіт. [5]
6. **Завершення тестування (Test closure)** — це формування звітності про завершення тестування, щоб визначити стратегії для реалізації в майбутньому та усунути проблеми процесу для майбутніх циклів тестування. [4]

Моніторинг та контроль тестування (Test monitoring and control) — це процес спостереження за всіма показниками, потрібний для гарантії, що проєкт працює добре, за графіком і не виходить за рамки бюджету. Моніторинг — це процес збору, реєстрації та надання інформації про діяльність проєкту, яку необхідно знати менеджеру проєкту та замовнику. Контроль передбачає виконання дій, необхідних для досягнення цілей плану тестування, на основі даних моніторингу. Моніторинг та контроль відбуваються під час усіх етапів тестування.

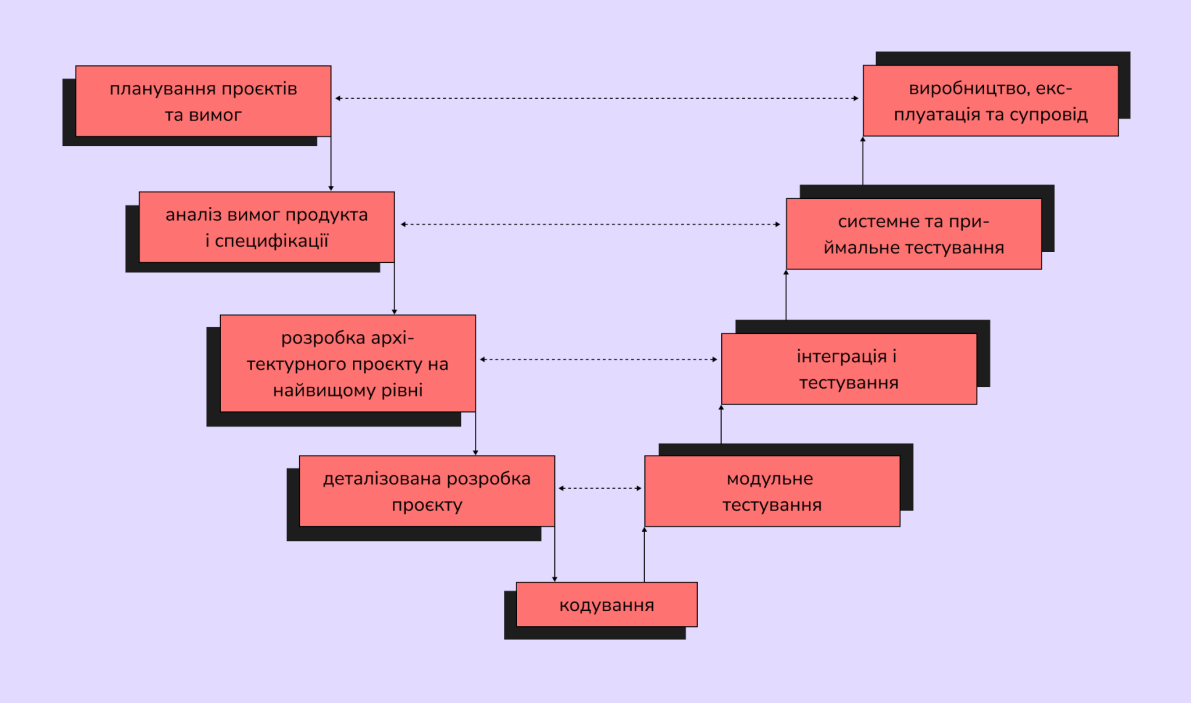
3. Послідовні та ітеративні методології розробки ПЗ

Розгляньмо основні види методологій розробки програмного забезпечення.

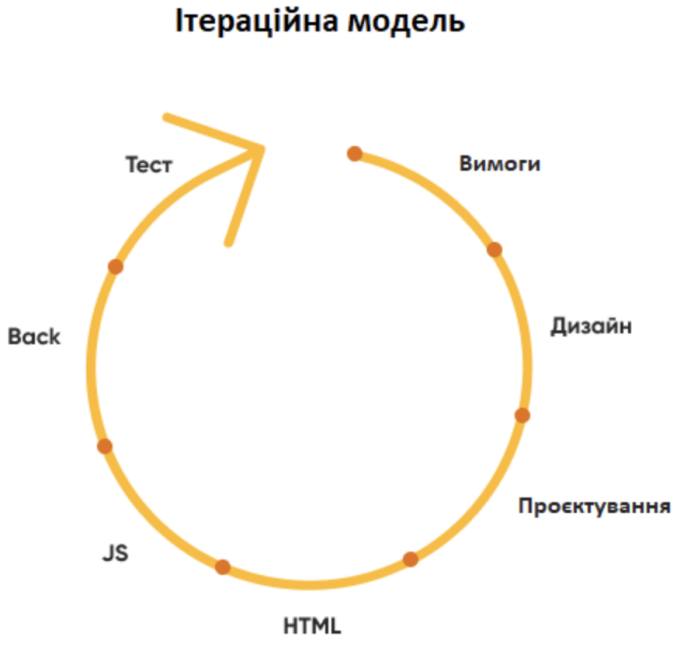
Waterfall (Водоспад, каскадна модель) — методика управління проєктами, яка передбачає послідовний перехід з одного етапу на інший (фази аналізу, проєктування, реалізації, тестування, інтеграції та підтримки) без пропусків і повернень на попередні стадії. [6]



V-модель — це покращена версія класичної каскадної моделі. Тут на кожному етапі відбувається контроль чинного процесу, щоб переконатись у можливості переходу на наступний рівень. У цій моделі тестування починається ще на стадії написання вимог. До того ж для кожного наступного етапу передбачений свій рівень тестового покриття. [9]

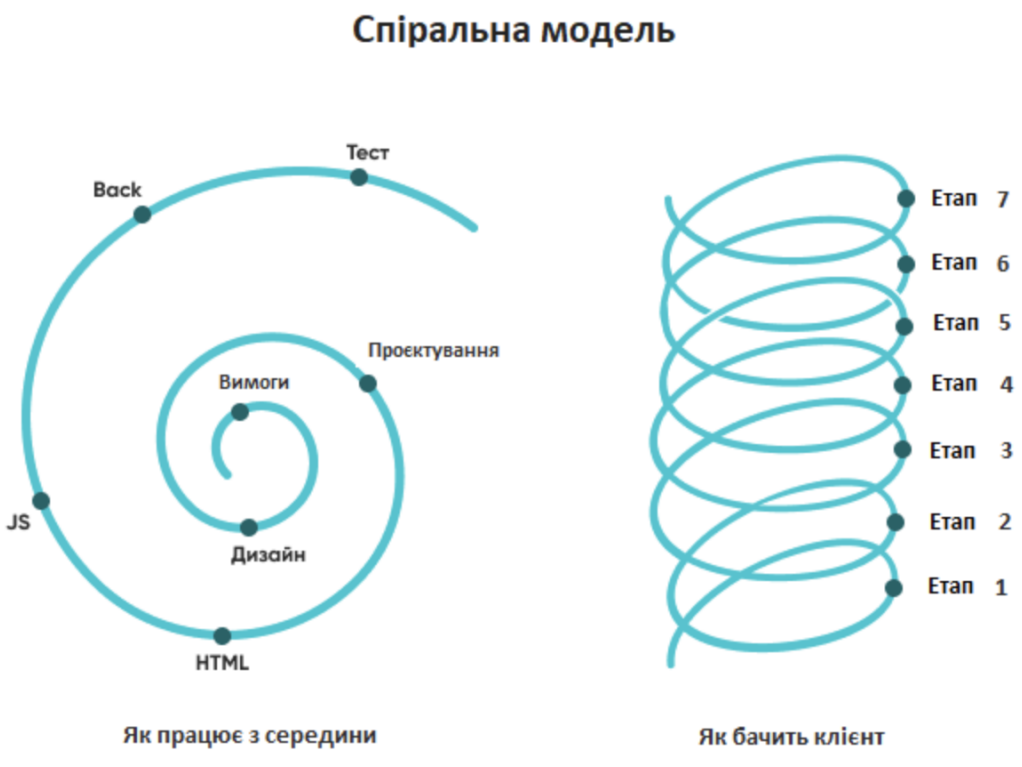


Ітераційна модель передбачає розбиття проєкту на частини (етапи, ітерації) і проходження етапів життєвого циклу на кожному з них. Кожен етап є самостійно довершеним, а сукупність етапів формує кінцевий результат.



Інкрементна модель має в основі принцип, що передбачає розширення можливостей, вдосконалення модулів і функцій програми. Буквальний переклад слова інкремент — «збільшення на один». Це «збільшення на один» застосовується також для позначення версій продукту. Якщо у каскадній моделі є два стани продукту: «нічого» і «готовий продукт», то з появою ітераційних та інкрементних моделей з'являється версіювання продукту. Кожна ітерація позначається цифрою: 1,2,3 — і відповідно продукт після кожної ітерації має нову версію (інкремент) з відповідним номером: v.1, v.2, v.3.

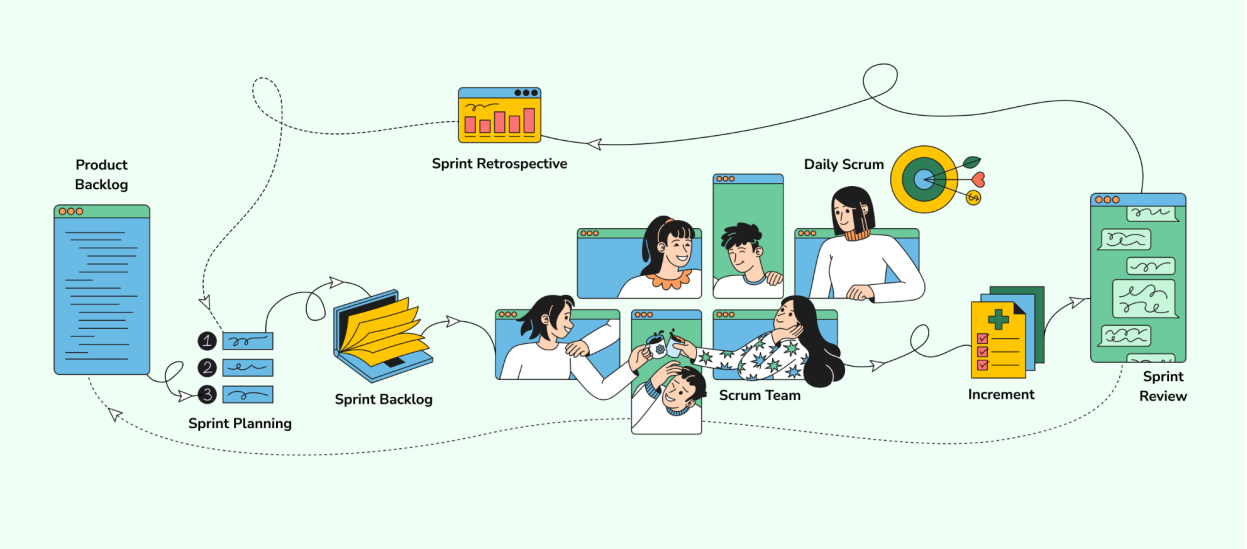
У **спіральній моделі** розробка програми має вигляд серії послідовних ітерацій. На перших етапах уточнюються специфікації продукту, на наступних — додаються нові можливості та функції. Мета цієї моделі — провести оновлену оцінку ризиків продовження робіт після кожної ітерації.
На кожному витку спіралі створюється чергова версія продукту, уточнюються вимоги проєкту, визначається його якість і плануються роботи наступного витка. Особлива увага приділяється початковим етапам розробки — аналізу і проєктуванню, де реалізованість тих чи інших технічних рішень перевіряється й обґрунтовується за допомогою створення прототипів (макетування). Завдяки ітеративній природі спіральна модель допускає коригування в процесі роботи, що сприяє поліпшенню продукту.



4. Scrum та Kanban

Agile — система ідей і принципів «гнучкого» управління проєктами, на основі якої розроблені популярні методи Scrum, Kanban та інші. Ключовий принцип Agile — це розробка шляхом коротких ітерацій (циклів), в кінці кожного з яких замовник (користувач) отримує робочий код або продукт.

Scrum — це методологія управління проєктами, побудована на принципах тайм-менеджменту. Основною її особливістю є залучення у процес розробки всіх учасників команди, і в кожного є певна роль. [7]



Scrum має низку термінів і назв, які допомагають імплементувати цю методологію на проєкті та швидко розуміти одне одного в команді. Ось найпоширеніші терміни та ролі у Scrum:

Власник продукту (Product owner) — людина, яка безпосередньо зацікавлена у якісному кінцевому продукті та розуміє, як цей продукт повинен виглядати/працювати. Product owner зазвичай працює на стороні замовника/клієнта та розставляє пріоритети для задач.

Scrum-майстер — це людина, яка допомагає команді слідувати скрам-процесам та втілює цінності Agile у повсякденному житті команди.

Scrum-команда — це команда, яка приймає всі принципи Scrum і готова з ними працювати.

Спринт — відрізок часу, відведений на виконання визначеного (обмеженого) списку завдань. Зазвичай це період від 1 до 4 тижнів.

Беклог (backlog) — це список усіх завдань проєкту.

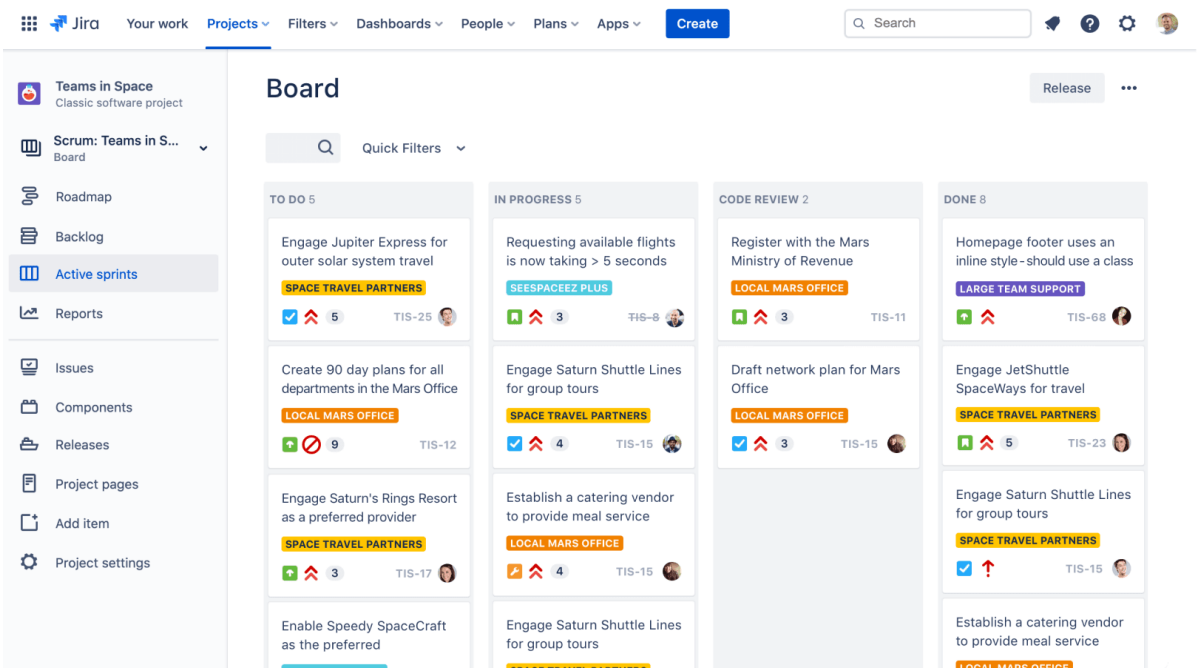
Спринт-беклог (sprint backlog) — це список завдань, який команда визначила та узгодила з власником продукту на найближчий звітний період (спринт).

Планування спринту (sprint planning) — це зустріч, де вся скрам-команда (команда розробників та QA, Scrum-майстер, Product owner) визначають пріоритетні задачі, оцінюють час та ресурси, необхідні для виконання спринту. З цього формується список робіт, які команда має виконати протягом наступної ітерації (спринту).[7]

Daily Scrum Meeting (Daily standup/daily meeting) — це щоденна коротка зустріч, де кожен член команди коротко звітує про свою роботу у форматі відповіді на запитання: Що було зроблено вчора? Що буде зроблено сьогодні? Які проблемами виникли? Це допомагає команді самоорганізовуватись і розуміти, що відбувається із проєктом.

Наприкінці кожного спринту скрам-команда збирається на **Ретроспективу.** Мета ретроспективи — переглянути якість чинних процесів, взаємовідносини членів команди та застосовані інструменти. Команда визначає, що пройшло добре, а що не дуже, а також виявляє потенційні можливості для покращення. [7]

Канбан (Kanban) — це методологія для оптимізації процесів розробки ПЗ, яка є частиною Agile-філософії. Kanban починається з візуалізації, щоб команда зосередилася на процесах. Для цього використовують спеціальну дошку і картки чи наліпки. Дошка — це обов'язковий елемент і у Scrum, і в Kanban. Кожен член команди має до неї доступ у будь-який час і бачить, на якому етапі знаходиться завдання. Канбан-модель відрізняється від скраму відсутністю або частковою відсутністю часових рамок, обов'язкових мітингів, ролей тощо. Обов’язковою є пріоритизація та постійний безперервний випуск нових версій продукту.



Джерела:

[1] https://uk.myservername.com/sdlc-phases

[2] http://tryqa.com/what-are-the-software-development-life-cycle-sdlc-phases/

[3] https://www.guru99.com/software-testing-life-cycle.html

[4] ISTQB 1.4.2

[5] https://worksection.com/ua/blog/waterfall-vs-agile.html

[6] https://studfile.net/preview/5203612/page:6/

[7] https://evergreens.com.ua/ua/articles/software-development-metodologies.html

[9] https://qalight.ua/baza-znaniy/v-model-v-model/