**Лекція 2. Основи якості програмного забезпечення**

***Причини виникнення технології програмної інженер***ії

 В кінці 60-х – на початку 70-х років з’явилися перші ознаки кризи в області програмування – колосальні успіхи в галузі розвитку засобів обчислювальної техніки прийшли в протиріччя з низькою продуктивністю праці програмістів. У зв’язку з ускладненням програмних систем стало очевидним, що їх важко проектувати, кодувати, тестувати і особливо важко розуміти, коли виникає необхідність їх модифікації в процесі супроводу. З’явилася життєва потреба в створенні технології розробки програмних засобів і інженерних методів їх проектування для істотного поліпшення продуктивності праці розробників. Кардинальні зміни в галузі створення програмного забезпечення (ПЗ) були обумовлені і швидким зростанням ринкового програмного продукту – тієї частини розроблених програм, яка отримувалася користувачем у вигляді готових до експлуатації пакетів програм різного призначення. Не дивлячись на те, що значна частина створюваного програмного забезпечення не доводиться до комерційного використання, тобто не виходить за межі фірми-розробника, вона представляє велику цінність для подальших розробок і для накопичення досвіду і знань.

Вже до початку 80-х років тільки в США було створено ПЗ на сотні мільярдів доларів. Впровадження комп’ютерних технологій в різноманітні сфери людської діяльності привело до виникнення і бурхливого розвитку нової галузі суспільного виробництва – промисловості обробки даних, сумарний обсяг продажів продукції в якій швидко залишив позаду всі традиційні галузі промисловості. Перерозподіл кількості працюючих у сфері матеріального виробництва привів до того, що в найрозвиненіших країнах більше половини працівників виявилися зайнятою обробкою інформації. В США в 90-ті роки цей показник досяг 80%. Основа даної галузі в першу чергу є технічне і програмне забезпечення систем обробки даних. При цьому самою наукомісткою залишається програмна продукція.

Природно, що і в наукових дослідженнях, і в практичній діяльності постійно робилися спроби перевести виготовлення програмної продукції на інженерну основу. Так, в 70-х роках виникла нова інженерна дисципліна – програмотехніка, або інженерія ПЗ (Software Engineering). Становленню програмотехніки сприяло розширення ринку ПЗ, поява могутніх фірм, зайнятих виробництвом виключно цієї продукції, і зростання кількості користувачів програмних виробів. Програмна інженерія вивчає теорії, методи й засоби професійної розробки ПЗ.

**Аспекти визначення якості та її атрибути.**

Якість ПЗ — набір характеристик продукту або сервісу, які характеризують його здатність задовольнити встановленим або передбачуваним потребам замовника. Поняття якості має різні інтерпретації в залежності від конкретної системи і вимог до програмного продукту. Крім того, в різних джерелах моделі якості відрізняються. Кожна модель має різне число рівнів і загальне число характеристик якості.

Область знань «Якість ПЗ (Software Quality)» складається з наступних розділів:

* концепція якості ПЗ (Software Quality Concepts);
* визначення та планування якості (Definition & Planning for Quality);
* діяльності і техніки гарантії якості і V & V (Activities and Techniques for Software Quality • Assurance, Validation-V & Verification - V);
* вимірювання в аналізі якості ПЗ (Measurement in Software Quality Analysis).

Під час вивчення даної галузі знань детально розглядаються проблеми якості ПЗ та шляхів його досягнення в процесі проектної діяльності груп розробників.

Концепція якості ПЗ включає зовнішні і внутрішні характеристики якості, їх метрики, а також моделі якості, визначені на множині зовнішніх і внутрішніх характеристик, які визначені у стандартах якості — це шість характеристик і для кожного з них 4—5 атрибутів.

*До характеристикам якості відносяться:*

Функціональність.

Надійність.

Зручності використання.

Ефективність.

Супроводжуємість

Переносимість.

Базова модель якості включає ці характеристики і відноситься до будь-якого типу програмних продуктів. Під час розробки вимог замовник формулює ті вимоги до якості, які найбільше підходять для програмного продукту, що замовляється.

Визначення і планування якості ПЗ ґрунтується на положеннях стандартів у цій галузі, складанні планів графіків робіт та процедури перевірки та ін. План забезпечення якості включає набір дій для перевірки процесів забезпечення якості (верифікація, валідація тощо) і формування документа щодо управління якістю. Управління якістю застосовується до процесів, продуктами і ресурсами, а також включає вимоги до процесів та їх результатів.

*Планування якості включає:*

* Визначення продукту в термінах заданих характеристик якості.
* Планування процесів для отримання необхідної якості.
* Вибір методів оцінки планованих характеристик та встановленню відповідності продукту сформульованим вимогам.

Щоб зрозуміти широту визначення якості програмного забезпечення, потрібно відповісти на питання, що часто виникає: що таке якість? Можна визначити, що існують два основних напрями, визначення якості програмного забезпечення:

1. Відповідність специфікації: якість, що визначається як властивості продуктів і послуг, на основі вимірювальних характеристик, що задовольняють фіксованій специфікації — тобто відповідність в заздалегідь визначеній специфікації.

2. Задоволення потреб: якість, що визначається незалежно від будь-яких вимірних характеристик. Тобто, якість визначається як продукти чи послуги, що дають можливість задовольнити очікування клієнтів — явні чи ні.

Протягом багатьох років окремі автори й цілі організації визначали термін "якість" по—різному.

Уотс Хемпфрі (Watts Hamphrey, автор концепції моделі оцінки зрілості CMM, а також PSP і TSP — People Software Process і Team Software Process) описує якість як "досягнення відмінного рівня придатності до використання".

Уолтер Едвардс Демінг в "Out of the crisis: quality, productivity and competitive position ", говорить: Труднощі у визначенні якості є переклад майбутніх потреб користувачів у вимірні характеристики, так що продукт може бути розроблений і розповсюджений, щоб дати задоволення з приводу ціни, яку користувач буде платити. Це не легко, і як тільки один стає досить успішним у цих зусиллях, виявляється, що потреби споживачів змінилися, з’явилися конкуренти в т.д.

Одна із сильних сторін Демінга в тому, що якість має бути визначено з точки зору задоволеності клієнтів — що набагато ширше поняття, ніж "відповідність специфікації" визначення якості (наприклад, з точки зору "задоволення потреб клієнтів"). Демінг пропонує, що якість має бути визначено тільки з точки зору агента — судді якості.

Філософія якості Демінга наголошує, що задовольнити і перевищити вимоги замовників є завданням, що всі в організації повинні виконати. Крім того, система управління якості працює з тим, що кожен несе відповідальність за якість його продукції з його внутрішніми клієнтами.

*Компанія IBM, у свою чергу, ввела в обіг фразу* "якість, керована ринковими потребами " ("market-driven quality ").

Критерій Белдріджа (Baldrige) для організаційного якості (NIST — National Institute of Standards and Technology, "Baldrige National Quality Program") використовує схожу фразу — "якість, що задається споживачем" ("customer—driven quality").

Найчастіше, поняття якості використовується відповідно з визначенням системи менеджменту якості ISO 9001 як "ступінь відповідності властивих характеристик вимогам".

Фейгенбаум визначає, що ім'я та термін контролю якості, практично є синонімом через його глибокий вплив на концепцію тотального контролю якості (а також у зв'язку з його авторством концепції). У "Total quality control" Арманд Фейгенбаум Валле пояснює свій погляд на якість наступним текстом:

Якість визначається клієнтом, а не інженером, це не визначення маркетингу, ні загального визначення управління. В його основу покладено на фактичному досвіді клієнта з продуктом або послугою, вимірюваний проти його вимоги, — очевидні або неявні, свідомо чи просто відчув, технічно або повністю суб'єктивно — і завжди представляє рухомі мішені в умовах конкурентного ринку.

Якість товарів і послуг може бути визначена як: загальна складова продукту і експлуатаційні характеристики маркетингу, інжинірингу, виробництва та обслуговування, продуктів і послуг у використанні, що буде відповідати очікуванням замовника.

У визначенні Фейгенбаума якість як "задоволення потреб клієнтів" не викликає сумнівів. Насправді, він йде дуже широко у своєму визначенні якості, підкресливши важливість задоволення клієнтів у реальних і очікуваних потребах. Ясно, що у визначенні якості Фейгенбаум охоплює не тільки управління продуктами і послугами, а й сподівань клієнтів.

В " Jurans's Quality Control Handbook" Джозеф М. Джуран передбачає два значення для якості:

Слово якість має кілька значень. Два цих значення домінують у використанні цього слова: 1) Якість складається з тих властивостей продукту, які відповідають потребам клієнтів і тим самим забезпечують успіх продукту. 2) Якість складається через відсутність недоліків. Тим не менш, виходячи з цього найзручніше дати коротке визначення поняття якості, як "придатність для використання".

Таким чином, прийнятна якість може розглядатися як кількісно виражений компроміс між замовником і виконавцем щодо характеристик продукту, створюваного виконавцем в інтересах вирішення завдань замовника з урахуванням інших обмежень проекту (зокрема, вартістю).

Якість ПЗ є відносним поняттям, яке має сенс тільки під час врахування реальних умов його застосування, тому вимоги, що пред'являються до якості, ставляться відповідно до умов і конкретної області їх застосування.

Якість ПЗ характеризується трьома головними аспектами:

•         якість програмного продукту;

•         якість процесів ЖЦ;

•         якість супроводу або впровадження (Рис. 1).

Аспект, пов'язаний з процесами ЖЦ, характеризується ступенем формалізації, достовірністю і якістю самих процесів ведення розробки ПЗ, а також верифікацією та валідацією отриманих проміжних результатів на процесах ЖЦ, що зменшують кількість помилок у ПЗ і тим самим сприяють підвищення якості готового продукту.

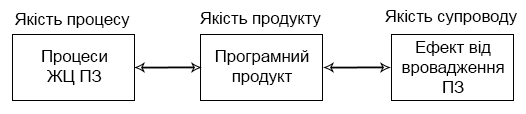


Рисунок 1 –Основні аспекти якості ПЗ

Якість продукту цілком і повністю визначається процесами ЖЦ. Ефект від впровадження отриманого програмного продукту в значній мірі залежить від якості супроводу і знань обслуговуючого персоналу.

**Кодекс етики програмної інженерії**

Своєю появою інженерія ПЗ зобов'язана діяльності потужних професійних об'єднань — The Assotiation for Computer Machmery (ACM) і Institute of Electrical and Electrorncs Engmeers Computer Sodety (IEEE Computer Sotiety). Спільними зусиллями цих двох об'єднань розроблений кодекс етики програмної інженерії. Він фокусує мораль, правила і норми поведінки професіоналів, їх зобов'язання і відповідальність по відношенню до суспільства і один до іншого.

Етика інженерної діяльності в програмуванні відрізняється від етики прикладних досліджень, де дослідники працюють у прикладній науці, спрямовують свої зусилля на реалізацію можливостей і відповідають певним вимогам. Інженерна діяльність у програмну інженерію, ще включає:

* технічні вміння;
* відповідальність перед користувачами;
* вміння керувати і приводити до вдалого завершення великі програмні проекти.

Інакше кажучи, інженери повинні добре знати, що є ризик не зробити реалізацію швидко або високоякісно. Кодекс складається з преамбули і восьми принципів, яких повинні дотримуватися професіонали з інженерії ПЗ.

У преамбулі професіонал визначається як фахівець, який приймає безпосередньо участь у діяльності з аналізу, специфікації, проектування, розробки, сертифікації, супроводження та тестування програмних систем. Сформульовані принципи забезпечують здоров'я, безпеку та добробут суспільства як головний фактор, який необхідно брати до уваги під час прийняття рішень у професійній діяльності інженерії ПЗ.

*У кодексі задекларовано вісім принципів, які стосуються відповідно:*

1) узгодження професійної діяльності з інтересами суспільства;

2) взаємовідносини між клієнтом, роботодавцем і виконавцем розробки;

3) досягнення відповідності якості продукту кращим професійним стандартам;

4) дотримання чесності і незалежності при професійних оцінках;

5) дотримання етичних норм у менеджменті і в супроводі розробок;

6) підтримка становлення професії у відповідності з кодексом етики;

7) дотримання етичних норм у взаєминах між колегами;

8) удосконалення кваліфікації розробників.

Кожен з наведених принципів має детальні пояснення щодо різних спектрів його дотримання.

**Значення і вартість якості**

Поняття "якість", насправді, не настільки очевидно і просто, як це може здатися на перший погляд. Для будь-якого інженерного продукту існує безліч інтерпретацій якості, залежно від конкретної «системи координат». Безліч цих точок зору необхідно обговорити і визначити на етапі вироблення вимог до програмного продукту. Характеристики якості можуть вимагатися в тому або іншому ступені, можуть бути відсутніми або можуть ставити певні вимоги, все це може бути результатом певного компромісу. Вартість якості (cost of quality) може бути диференційована на:

* вартість попередження <дефектів> (prevention cost);
* вартість оцінки (appraisal cost);
* вартість внутрішніх збоїв (internal failure cost);
* вартість зовнішніх збоїв (external failure cost).

Рушійною силою програмних проектів є бажання створити програмне забезпечення, що володіє певною цінністю. Цінність програмного забезпечення може виражатися у формі вартості, а може і ні. Замовник, звичайно, має своє уявлення про максимальні вартісні вкладення, повернення яких очікується в разі досягнення основних цілей створення програмного забезпечення. Замовник може, також, мати певні очікування щодо якості ПЗ.

Іноді, замовники не замислюються про питання якості і пов'язаної з ними вартості. Характеристики якості є предметом обговорення ступеня залучення замовника в процес прийняття рішень і повного розуміння замовником вартості та вигоди, пов'язаної з досягненням того чи іншого рівня якості. В ідеальному випадку, більшість такого роду рішень повинно прийматися процесі роботи з вимогами, однак ці питання можуть підніматися протягом усього життєвого циклу програмного забезпечення. Не існує якихось "стандартних" правил того, як саме необхідно приймати такі рішення. Однак, інженери повинні бути здатні представити різні альтернативи (у досягненні різного рівня якості) і їх вартість.

**Філософія якості Демінга**

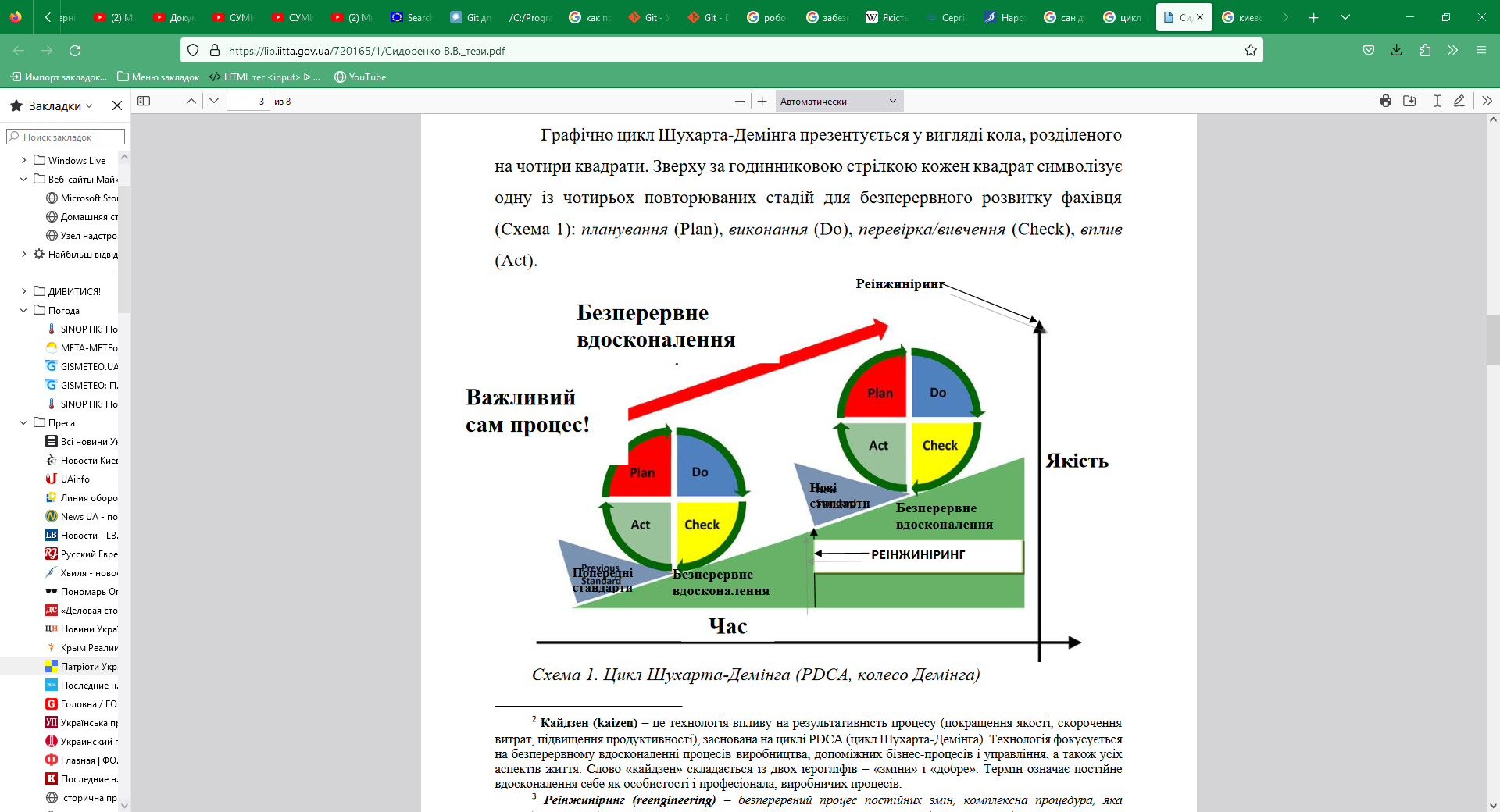
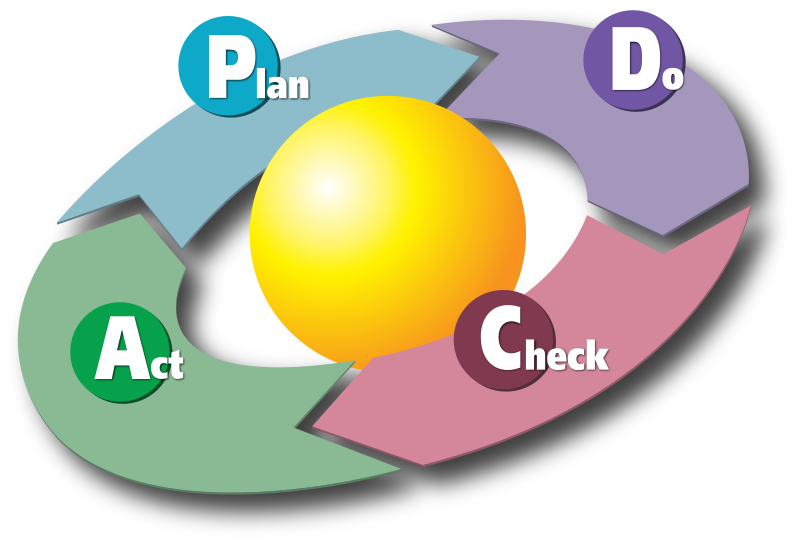


Рисунок 2 – Цикл Шухарта-Демінга (PDCA, колесо Демінга)

**Ци́кл Шу́харта-Де́мінга** (**Цикл PDCA**, часто використовується назва **Цикл Демінга**) — модель безперервного поліпшення процесів, цикл PDCA — плануй (**Plan**), роби (**Do**), перевіряй (**Check**), впливай (**Act**). При її застосуванні в різноманітних областях діяльності, зокрема, управління якістю, дозволяє ефективно керувати цією діяльністю на системній основі.

Бажаний результат досягається ефективніше, коли професійною діяльністю і пов’язаними з нею ресурсами фахівець керує як безперервним процесом. Незмінними залишаються постійність, циклічність щодо вдосконалення процесів, прийняття рішень. Оскільки коло не має кінця, то цикл повторюється знову, сприяючи постійному вдосконаленню певних процесів.

Діяльність здійснюється за схемою циклічного оновлення (рішення) проблеми, у кожному з квадратів кола треба намалювати внутрішнє коло того ж циклу.

Цикл починається із планування.

Планування (**Plan**): установлення цілей і процесів, необхідних для досягнення цілей, планування робіт по досягненню цілей, планування виділення і розподілу необхідних ресурсів. На цьому етапі надаються відповіді на запитання : **Що робити? Як робити**?

Виконання (**Do**) – виконання запланованого плану, певних дій, робіт. На цьому етапі вирішується питання :**Зробити те, що заплановано**.

Перевірка/вивчення (**Check**) – збір інформації і контроль результату на основі ключових показників ефективності, виявлення і аналіз відхилень, установлення причин відхилень, перевірка ступеня реалізації мети. На цьому етапі надаються відповіді на запитання : **Відповідність цілям досягнуто**?

Вплив (**Act**), управління, корекція: прийняття заходів щодо усунення причин відхилень від запланованого результату, коригування, зміни в плануванні та розподілі ресурсів, удосконалення процесу завдяки постійному контролю. На цьому етапі надаються відповіді на запитання : **Як і що покращити в подальшому**?

Серед переваг циклу Шухарта-Демінга виділяємо такі:

людиноцентризм, орієнтація на людей; простота засвоєння і використання;

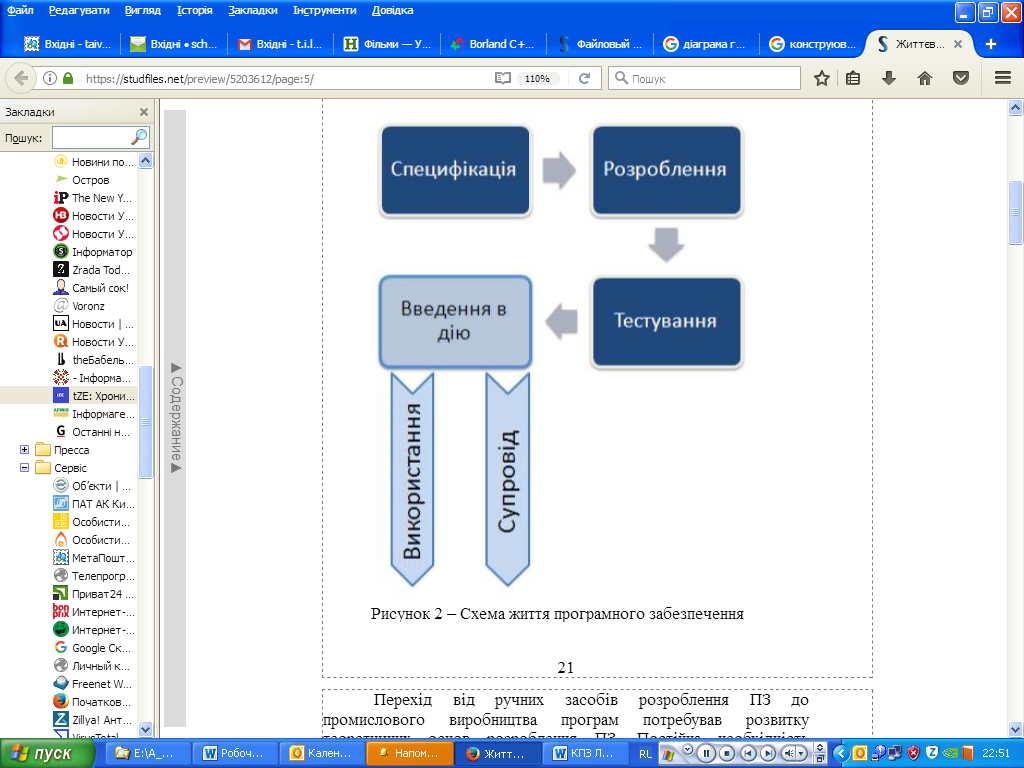
універсальність; забезпечення конкурентних переваг.

Недоліками циклу Шухарта-Демінга є:

потребує значних управлінських навичок щодо змін, упровадження інновацій; потребує дисципліни.

На думку японців, цикл Демінга передусім спрямований на боротьбу із трьома головними ворогами: втрати (усі види дій, які споживають ресурси, не створюючи цінності); невідповідність (будь-яке відхилення від процесу: як хороше, так і погане); нераціональні дії (перевантаження, робота з напругою).

**Моделі життєвого циклу програмного забезпечення (ЖЦ ПЗ)**

У процесі створення ПЗ можна виділити 4 базових етапи/стадії (рис.3):

**Специфікація** – визначення основних вимог.

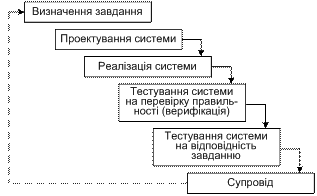
**Розроблення** – створення ПЗ відповідно до специфікацій.

**Тестування** – перевірка ПЗ на відповідність вимогам клієнта.

**Супровід/Модернізація** – розвиток ПЗ відповідно до змін потреб замовника.

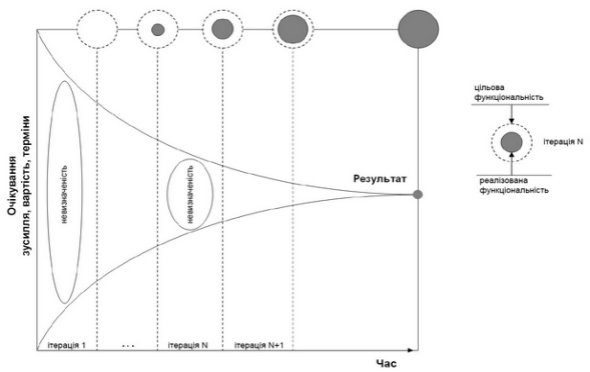
**Модель життєвого циклу** - це структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до

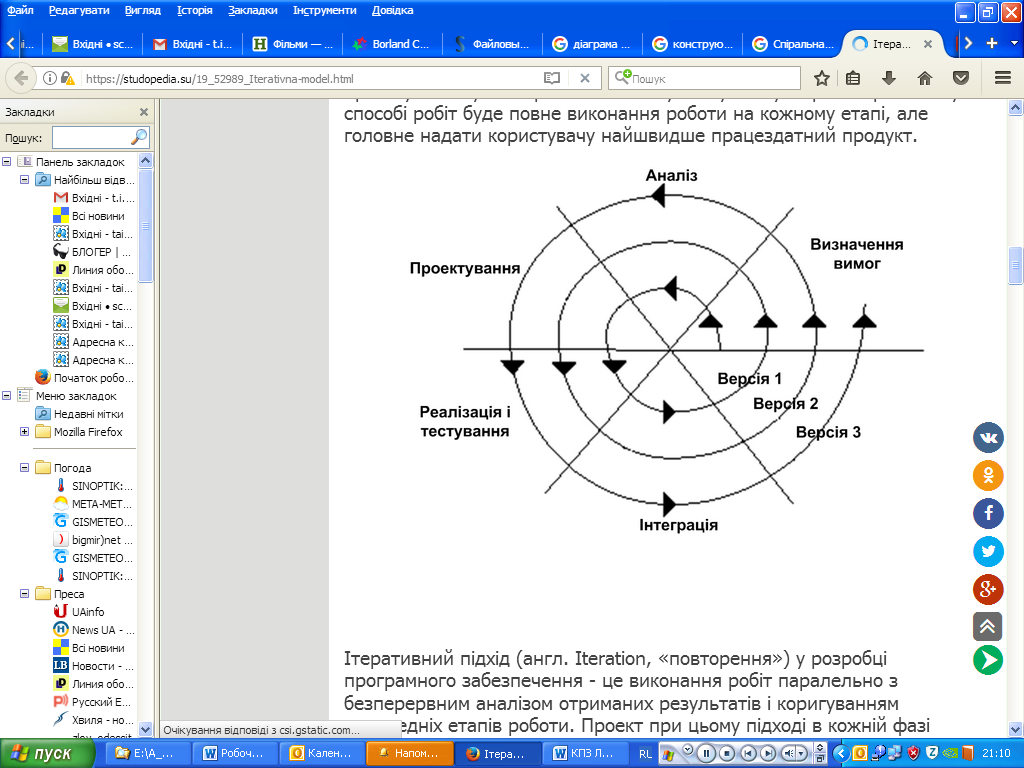
**Рисунок 3 – Схема ЖЦ ПЗ** припинення її використання. На сьогодні найбільшого розповсюдження набули дві: каскадна модель та спіральна модель, яка реалізується через адаптивні моделі Agile.

[](http://fitm.nusta.edu.ua/mediawiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Pii03.gif)Пригадаємо основні моделі.

**Каскадна модель**. Всі вимоги повністю з‘ясовуються перед проектуванням. Кожен етап роботи виконується лише раз і настільки ретельно, щоб потреби повертатись до попереднього етапу не виникало. Результат виконання кожного етапу, перед передачею в наступний, піддається верифікації.

**Рисунок 4 – Каскадна модель**

**Ітеративна та інкрементна модель (**(Iterative and incremental development - IID).Модель IID передбачає розбиття ЖЦ проекту на послідовність ітерацій, кожна з яких нагадує "міні-проект", включаючи всі процеси розробки в застосуванні до створення менших фрагментів функціональності, порівняно з проектом в цілому. Всі вимоги повністю з‘ясовуються перед першою ітерацією. Мета кожної ітерації — отримання працюючої версії програмної системи, що включає функціональність, визначену інтегрованим змістом всіх попередніх та поточної ітерації. Результат фінальної ітерації містить всю необхідну функціональність продукту.

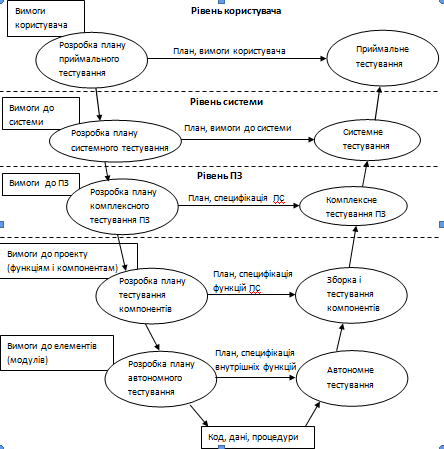
**Спіральна модель** передбачаєнеповне завершення робіт на кожному етапі, що дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При такому підході відсутню роботу можна буде виконати на наступній ітерації. Головне ж завдання - щонайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог. Дана модель ЖЦ

**Рисунок 5 – Спіральна модель** допускає аналіз продукту на витку розробки, його перевірку, оцінку правильності та прийняття рішення про перехід на наступний виток або повернення на попередній виток для доопрацювання на ньому проміжного продукту. Вимоги можуть багаторазово уточнюватися протягом розробки ПС і навіть після завершення та випробовування, і часом може з'ясуватися, що замовник «хотів зовсім інше».

### Еволюційна модель. У разі еволюційної моделі система послідовно розробляється з блоків конструкцій. На відміну від інкрементної моделі в еволюційній моделі вимоги встановлюються частково і уточнюються в кожному наступному проміжному блоці структури системи. Модель використовується для розробки нескладних і некритичних систем, де головною вимогою є реалізація функцій системи. При цьому вимоги не можуть бути визначені відразу і повністю. Тому розробка системи здійснюється

### Рисунок 6 – Еволюційна модель ітераційним шляхом її еволюційного розвитку з отриманням деякого варіанта системи–прототипу, на якому перевіряється реалізація вимог.

Розвитком цієї моделі є **модель еволюційного прототипування (макетування)** в рамках усього ЖЦ розробки ПС. У літературі вона часто називається моделлю швидкої розробки програм RAD (Rapid Application Development). Основною ідеєю цієї моделі є моделювання окремих функцій системи в прототипі і поступове еволюційне його доопрацювання до виконання всіх заданих функціональних вимог.

**V-подібна модель.** В цій моделі тестування розглядається як неперервний процес, інтегрований в процес розробки ПС. Він включає два взаємопов’язаних підпроцеси: 1) планування тестування в рамках процесів розробки системи (ліва гілка) ; 2) проведення тестування відповідних об’єктів (права гілка)

Характеристики V-подібної модел**і**: 1) Перевірка і оцінка тестопридатності вимог на ранніх стадіях розробки (з допомогою аналізу, який виконується під час тестування); 2) Наявність документованих тестових вимог

Переваги V-подібної моделі:

1) Забезпечує зворотний зв’язок з

**Рисунок 7 – V-подібна модель** користувачем на ранніх стадіях ЖЦ; 2) Покращує планування і розподіл затрат на тестування; 3) Чіткі документовані цілі тестування.

***Адаптивні моделі*** *( Adaptive Software Development* ASD) Адаптивна розробка замінює водоспадну модель повторюваними серіями обдумування, співробітництва та навчання. Цей динамічний цикл передбачає постійне навчання та адаптацію до виникаючих станів проекту. Характеристиками життєвого циклу АРП є зосередженість уваги, ітеративність, обмеженість за часом, управління ризиком, терпимість до змін.

**Моделі розробки ЖЦ ПЗ та контроль якості**.

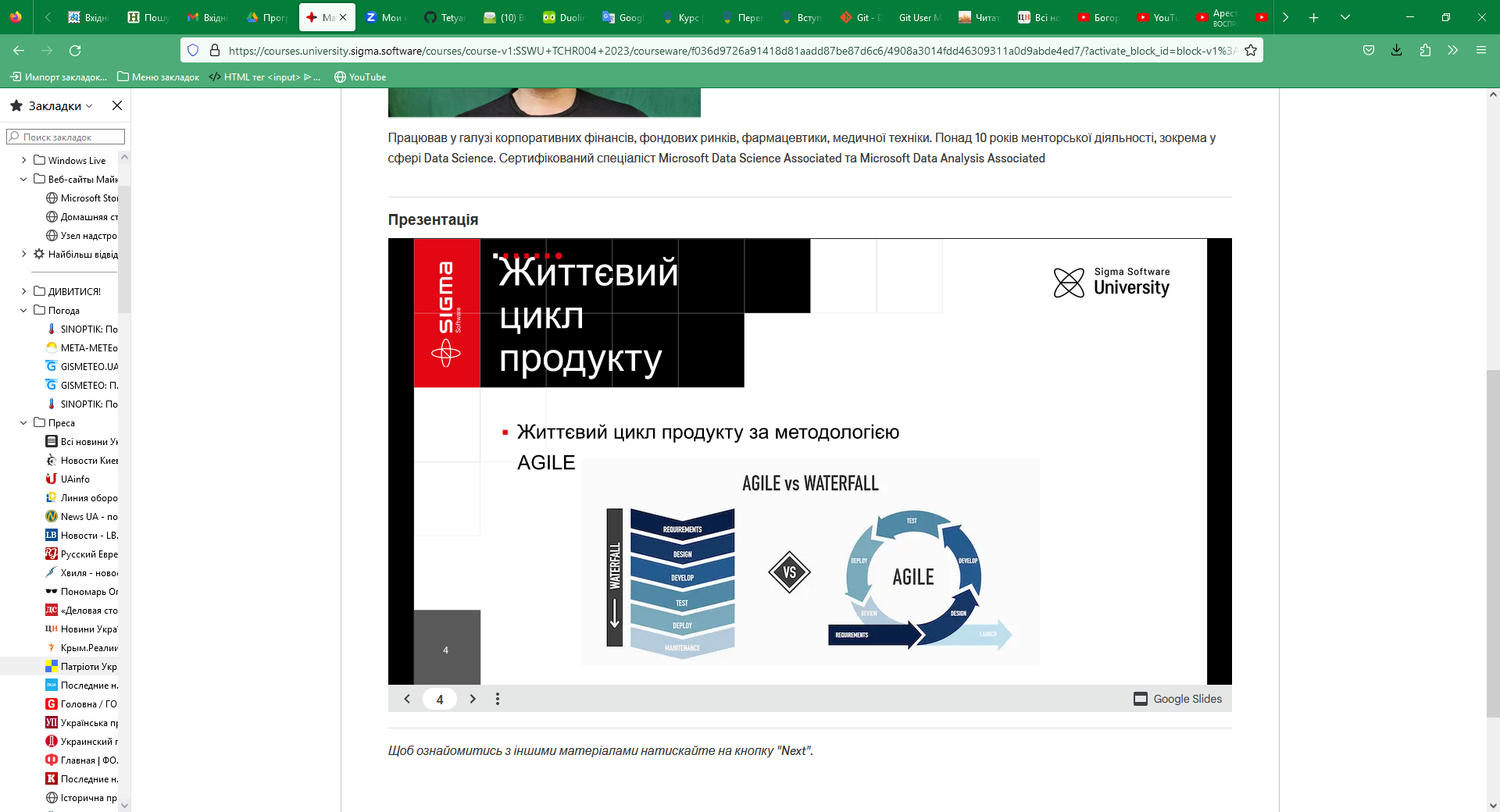


Рисунок 8 – Моделі Waterfall та Agile

Agile дозволяє використовувати CI/CD —**систему, яка може поєднувати нові рядки коду зі старою системою, перевіряючи помилки або можливі помилки**. Це спрощує випуск нових оновлень із меншою кількістю помилок. Це також дозволяє постійно випускати оновлення, що було неможливо раніше. Безперервна інтеграція (Continuous Integration, CI) і безперервне постачання (Continuous Delivery, CD) є культурою, набором принципів і практик, які дозволяють розробникам частіше і надійніше розгортати зміни програмного забезпечення. Обидва терміни дуже тісно пов’язані з професією DevOps і являються основними термінами їх еволюції.

Термін DevOps поєднує у собі розробку (DEVelopment) та операційну діяльність (OPerationS). Це наголошує на необхідності інтеграційної роботи всіх команд для ефективної доставки працюючого ПЗ. Для створення коду, готового до розгортання у виробничому середовищі, розробники повинні мати можливість відстежувати всі проміжні кроки від розробки до релізу. Необхідно подолати роз’єднаність та організувати спільну роботу з командами, що відповідають за QA , безпеку та інфраструктуру з урахуванням їхнього вкладу в загальний процес.

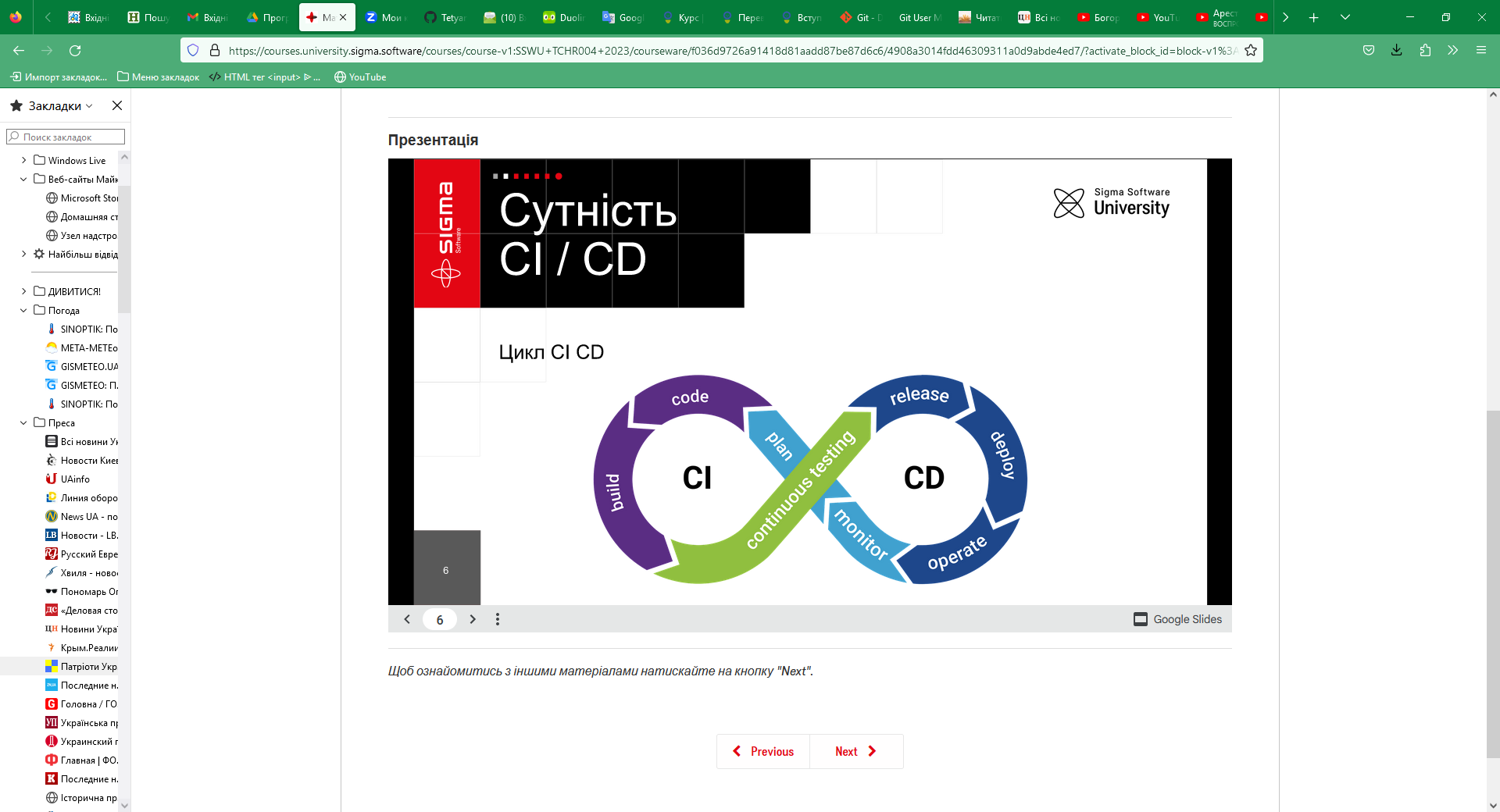


Рисунок 9 – Реалізація циклу CI/CD

Основні принципи CI/CD:

* сегрегація відповідальності зацікавлених сторін; зниження ризику;
* короткий цикл зворотного зв’язку;
* реалізація середовища.

Перший пункт списку вирішує проблему комунікації та розставляє чіткі зони відповідальності між учасниками. Розробники і дизайнери проектують бізнес-логіку, а також забезпечують позитивний досвід взаємодії з готовою системою. Інженери за якістю вводять наскрізні функції та приймальні тести, DevOps- інженери організують логістику коду, а користувачі дають зворотний зв’язок за результатами використання системи.

Принцип зниження ризику потребує щоб кожна група учасників розробки мінімізувала всі можливі ризики при проходженні продукту через стадії життєвого циклу (контроль цілісності бізнес-логіки, користувальницького досвіду, оптимізація зберігання та обробки даних, міграції та ін.).

Принцип циклу зворотного зв’язку дає змогу вирішити інциденти при виникненні помилок, чи проблем з новим функціоналом. Щоб додавати в продукт новий функціонал швидше за конкурентів, необхідно прагнути до автоматизації складання та тестування коду. Однак, у ситуаціях, коли для вирішення потрібна участь людини, автоматизація може лише нашкодити. Для таких ситуацій рекомендується скорочувати кількість інформаційних посередників, забезпечуючи короткий цикл зворотного зв’язку.

Якість продукту підвищується за допомогою паралельного тестування функціональних блоків майбутньої системи. Вузькі місця та критичні моменти фіксуються та відпрацьовуються ще на ранніх ста-діях циклу. Проте керівники проектів помилково приймають методологію як панацею і прагнуть запровадити її у всі свої розробки. Нестача досвіду призводить до ускладнення робіт з IT-продуктами компанії. Отже, CI/CD можна назвати кращою методикою розробки, яка відповідає задачам сьогодення і поступово стає звичним терміном в усіх IT компаніях. Розглядаючи весь процес розробки та доставки ПЗ в цілому та оптимізуючи кожен його етап, можна швидше створювати продукт і та отримувати зворотний зв’язок. Тим самим забезпечується розвиток та покращення продукту.

***Використана література***

1. Лавріщева К.М. Програмна інженерія. Електронний підручник: http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf
2. Лавріщева К.М. Програмна інженерія. – Підручник.–К.:Академперіодика, 2008.–415с.
3. И. Соммервиль. Инженерия программного обеспечения,  
    6 изд. – И.д. "Вильямс", 2002.

***Для самостійного вивчення***: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

***Питання для самоконтролю*:**

**1.** Описати область знань «якість програмного забезпечення», виділити її основні проблеми.

**2.** Надати визначення поняттю якості та її атрибутам.

**3.** Описати основні принципи кодексу етики в інженерії програмного забезпечення.

**4.** Парадигма «вбудови» якості в інженерії програмного забезпечення (QFD).

**5.** Сертифікація програмного забезпечення в Україні.

**6.** Значення і вартість якості.