**Лекція 8. Методи доведення правильності програм, верифікація та валідація.**

В архітектурі процесів ЖЦ визначені 4 процеси підтримки які направлені на підвищення якості ПЗ: Верифікація, Валідація, Сумісний перегляд і Аудит.

Виходячи з положення стандарту ISO/IEC 12207 призначення верифікації складається в підтвердженні того, що робочий продукт ПЗ відповідає вимогам в результаті успішної реалізації процесу:

* Розроблена і впроваджена стратегія верифікації;
* Встановлені критерії верифікації всіх необхідних робочих продуктів ПЗ;
* Виконуються належні дії з верифікації;
* Виконується пошук дефектів у робочих продуктах та їх усунення;
* Забезпечується доступ замовника та інших зацікавлених сторін  
  до результатів верифікаційної діяльності.

***Основні поняття: верифікація та валідація програмного забезпечення***.

Ці два поняття тісно пов’язані з процесами тестування і забезпечення якості. На жаль, їх часто плутають, хоча відмінності між ними досить істотні.

**Верифікація (Verification)** – це статична практика перевірки документів, дизайну, архітектури, коду, тощо

* **Верифікація** – це процес який включає в себе перевірку планів (Plans), технічні умови (Requirement Specifications), Специфікація проекту (Design Specifications), код (Code), тестові випадки – тест кейси (Test Cases), список перевірок - чек-ліст (Chek-Lists) тощо.
* **Верифікація** завжди проходить без запуску коду.
* **Верифікація** використовує два основних види динамічних методів –моніторинг, в рамках якого йде тільки спостереження, запис і оцінка результатів роботи **ПЗ** при його звичайному використанні, і тестування, при якому **ПЗ** виконується в рамках заздалегідь підготовлених сценаріїв. Статичні методи – аналіз та експертіза.
* **Верифікація** відповідає на питання “Чи робимо ми продукт правильно?”
* **Верифікація** допоможе визначити, чи є програмне забезпечення **високої якості**, але воно не гарантує, що система **корисна**. Перевірка пов’язана з тим, що система добре спроектована і безпомилкова.
* **Верифікація** відбувається до **Валідації**.

Вона містить всі активності які дозволяють досягти високої якості програмного забезпечення:

1. Інспекція (**Inspection**) в розробці програмного забезпечення стосується експертної перевірки будь-якого робочого продукту навченими особами, які шукають дефекти за допомогою чітко визначеного процесу. (інспекція за Фаганом).
2. Покрокові інструкції (**Walkthroughs**) з розробки програмного забезпечення, покрокове керівництво — це форма експертної оцінки програмного забезпечення, під час якої дизайнер або програміст керує членами групи розробників та іншими зацікавленими сторонами, які переглядають програмний продукт, а учасники ставлять запитання та коментують можливі помилки, порушення стандартів розробки та інші проблеми.
3. Рецензування – огляди (**Reviews**) в розробці програмного забезпечення, рецензування – це тип перевірки програмного забезпечення, під час якого робочий продукт (документ, код або інше) перевіряється його автором та одним або кількома колегами з метою оцінки його технічного змісту та якості .

Верифікація програмного забезпечення – більш загальне поняття, ніж тестування. Метою верифікації є досягнення гарантії того, що верифікований об'єкт (вимоги або програмний код) відповідає вимогам, реалізований без непередбачених функцій і задовольняє проектним специфікаціям і стандартам. Процес верифікаціі включає в себе інспекції, тестування коду, аналіз результатів тестування, формування та аналіз звітів про проблеми. Таким чином, прийнято вважати, що процес тестування є складовою частиною процесу верифікації.

**Валідація (validation)** – це процес оцінки кінцевого продукту, необхідно перевірити, чи відповідає програмне забезпечення очікуванням і вимогам клієнта. Це динамічний механізм перевірки та тестування фактичного продукту.

* **Валідація** завжди включає в себе запуск коду програми.
* **Валідація** використовує методи, такі як тестування Black Box, тестування White Box і нефункціональне тестування.
* **Валідіція** відповідає на питання “Чи робимо ми правильний продукт?”
* **Валідація** перевіряє, чи відповідає програмне забезпечення вимогам і очікуванням клієнта.
* **Валідація** може знайти помилки, які процес **Верифікації** не може зловити.
* **Валідація** відбувається після **Верифікації**.

Процес валідації ( *validation process* ) передбачає визначення повноти відповідності заданих вимог і створеної системи або програмного продукту їх конкретному функціональному призначенню. Під *атестацією* зазвичай розуміються підтвердження і оцінка достовірності проведеного тестування ПЗ. Валідація повинна гарантувати повну відповідність ПС специфікаціям, вимогам і документації, а також можливість його безпечного і надійного застосування користувачем. Валідацію рекомендується виконувати шляхом тестування у всіх можливих ситуаціях і використовувати при цьому незалежних фахівців. Валідація може проводитися на початкових стадіях життєвого циклу програмних засобів або як частина роботи з приймання ПЗ.

Валідація, так само як верифікація, може здійснюватися з різними ступенями незалежності. Якщо процес валідації виконується організацією, що не залежить від постачальника, розробника, оператора або служби супроводу, то він називається *незалежної валідацією.*

На практиці, відмінності верифікації та валідації мають велике значення: замовника цікавить більшою мірою валідація (задоволення власних вимог); виконавця, в свою чергу, хвилює не тільки дотримання всіх норм якості (верифікація) при реалізації продукту, а й відповідність всіх особливостей продукту бажанням замовника.

***Вхідні дані V&V***

* Опис проекту ПЗ
* Потреби користувачів
* Стандарти проектування
* Звіти про критичність задач
* Звіти про аналіз загроз
* Вихідний код
* Плани та графіки розробки
* Процедури тестування

***Цілі V&V***

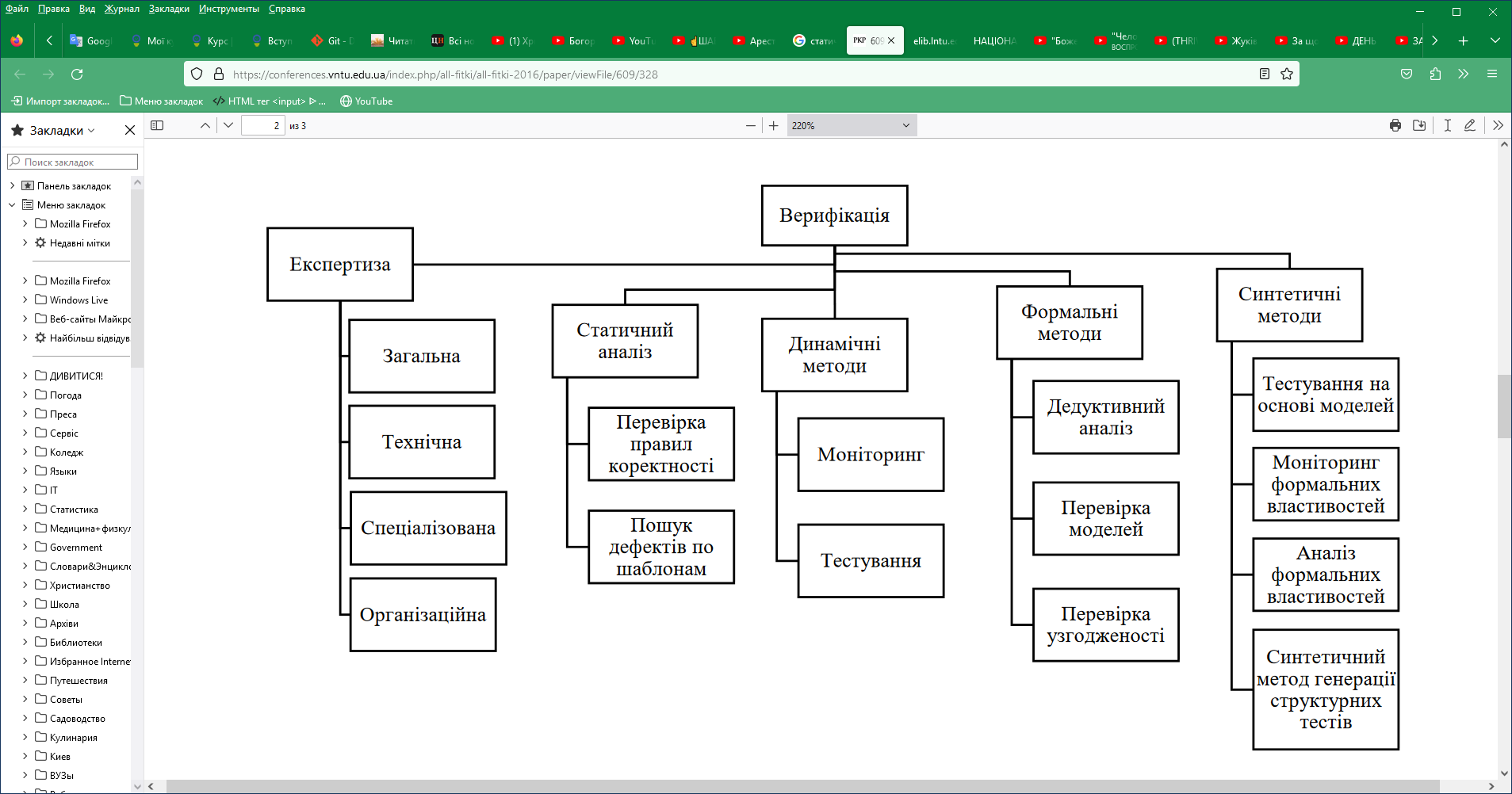
* Аналіз критичності
* Аналіз розподілу системних потреб
* Аналіз відсліжуваності
* Аналіз загроз
* Аналіз ризику
* Аналіз інтерфейсів
* Оцінка документу опису концепції
* Оцінка вимог до ПЗ
* Оцінка стану керування налаштуваннями
* Створення/верифікація тестування
* Аудит інсталяційного налаштування
* Пробний запуск
* Створення кінцевого звіту

***Результати V&V***

* Звіти про задачі
* Звіти про аномалії
* Процедури тестування для V&V
* Заключний звіт про V&V

Основним стандартом, що регулює планування і проведення верифікації ПЗ, є стандарт IEEE 1012 на процеси верифікації та валідації [24]. Цей стандарт містить опис наборів окремих завдань верифікації, що відповідають різним видам діяльності, рекомендований шаблон плану проведення перевірки та затвердження, визначення 4-х рівнів критичності ПЗ (від високої до мінімальної).

Загальноприйнятий поділ методів верифікації можна представити в вигляді діаграми.



**Експертизою ПЗ** називають всі методи верифікації, в яких оцінка артефактів життєвого циклу ПЗ виконується людьми, що безпосередньо аналізують ці артефакти. Перевагою даного методу є те, що при його використанні виявляються в середньому 50-90% помилок.

Цей метод має такі недоліки. Пошук помилок, оцінка і аналіз властивостей ПЗ людиною (зазвичай групою 2-5 осіб). Потрібні справжні експерти, програмісти з досвідом роботи не менше 10 років.

**Статичний аналіз** – аналіз без виконання програми. Методи статистичного аналізу можна розділити на два види: контроль того, що всі формалізовані правила коректності побудови цих артефактів виконані, та пошук типових помилок і дефектів в них на основі деяких шаблонів. Часто інструменти статичного аналізу використовують обидва типи перевірок. Статичний аналіз можна вважати найбільш широко застосовуваним методом верифікації. Перевірені на практиці правила коректності коду або шаблони типових помилок переносяться в середовища розробки.

Переваги статичного аналізу:

* Автоматичний аналіз багатьох шляхів виконання одночасно.
* Виявлення помилок, що проявляються лише на одиничних шляхах виконання або на незвичайних вхідних даних.
* Можливість аналізу на неповному наборі вхідних файлів.
* Відсутність накладних витрат під час виконання програми.

Недоліки даного методу:

* Велика кількість помилкових спрацювань.
* Необхідна ручна перевірка результатів роботи, що вимагає значних часових, людських та матеріальних ресурсів

Динамічні методи верифікації використовують результати реальної роботи програмної , і системи або її прототипів, щоб перевіряти відповідність цих результатів вимогам і проектним рішенням.

Існує два основних види динамічних методів верифікації: моніторинг, в рамках якого йде тільки спостереження, запис і оцінка результатів роботи ПЗ при його звичайному використанні тестування, при якому ПЗ виконується в рамках заздалегідь підготовлених сценаріїв. Перевагою даного методу є висока точність виявлення помилок. А недоліками – необхідно мати набір вхідних даних та середовище виконання, а також високі вимоги до ресурсів.

Формальні методи верифікації. Їх відмітною особливістю є можливість проведення пошуку помилок на математичній моделі, без звернення до фізичної реалізації, що в деяких випадках досить зручно і економічно. Для проведення аналізу формальних моделей застосовуються специфічні техніки, такі як дедуктивний аналіз, перевірка моделей, перевірка узгодженості. Для побудови таких моделей завжди необхідно виходити так само з коректності та адекватності моделі ПЗ. Лише після правильної побудови цієї моделі можна автоматично проаналізувати деякі з її властивостей. Проте, в більшості випадків для ефективного аналізу від фахівців будуть потрібні глибокі знання математичної логіки і алгебри і деякого набору навичок роботи з цим апаратом.

**Забезпечення якості: Інспекції**

***Інспекція ПЗ*** - аналіз та перевірка різних робочих продуктів ПЗ (специфікацій, архітектурних схем, діаграм, вихідного коду та ін.) і виконується на всіх етапах ЖЦ розробки ПЗ.

***Метою інспекцій*** є виявлення різних аномальних станів ПЗ незалежними фахівцями та з залученням авторів проміжного або кінцевого продукту.

* Наявність обов’язкових фаз: вимоги / проектування / кодування / тестування / і т. д.
* Неформальні огляди:
  + Самостійні огляди.
  + Незалежні огляди.
  + Бажана незалежність думок.
* **Формальні інспекції:**
  + Інспекції та варіації Фагана.
  + Процесу і структури.
  + Індивідуальні та групові перевірки.
  + Що / як перевірити: методики
* Для тестування необхідна виконувана програма, а для інспекції – ні, тому інспекція дозволяє:
  + Виявлення дефектів на ранніх стадіях життєвого циклу ПЗ
  + Виявлення дефектів у таких робочих продуктах як плани тестування, посібники користувача, графіки проектів і т.д.
* Існує велика кількість прийомів інспектування ПЗ
* Існують різні рівні формальності інспекцій

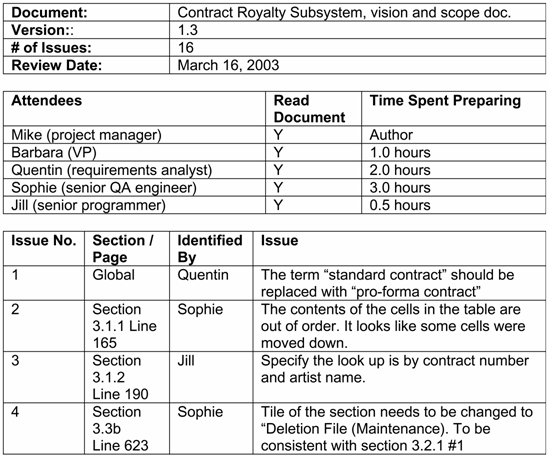
На початковому етапі проектування інспектування передбачає перевірку повноти, цілісності, однозначності, несуперечності та сумісності робочих продуктів з вихідними вимогами до програмної системі.

На етапі реалізації системи під інспекцією розуміється аналіз текстів програм на дотримання вимог стандартів і прийнятих керівних документів технології програмування. Ефективність такої перевірки полягає в тому, що залучаються експерти намагаються поглянути на проблему "з боку" і піддають її всебічному критичному аналізу

**Узагальнений процес інспектування**



Приклад журналу інспекцій



**Планування та приготування**

Необхідно дати відповідь на такі питання:

* Яка мета проведення інспекції
* Які об'єкти інспекції
* Хто виконуватиме інспекцію
* Хто ще повинен бути залучений, їх ролі та обов'язки
* Якими буде процес інспектування та діяльність після інспекції

**Інспектування**

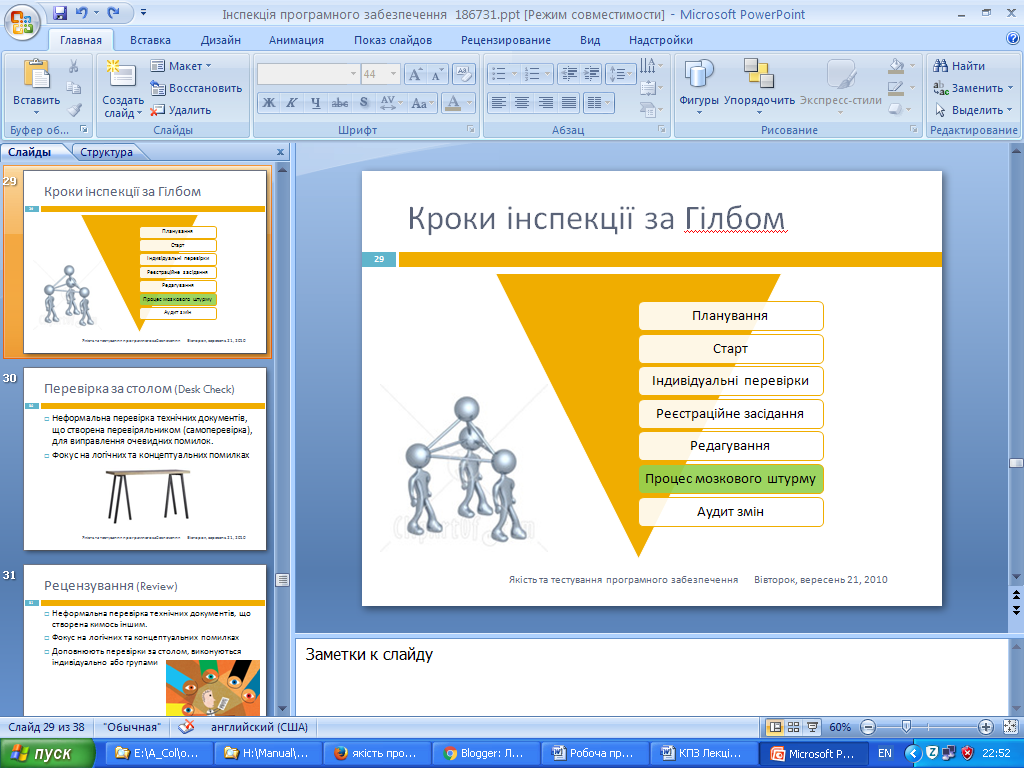
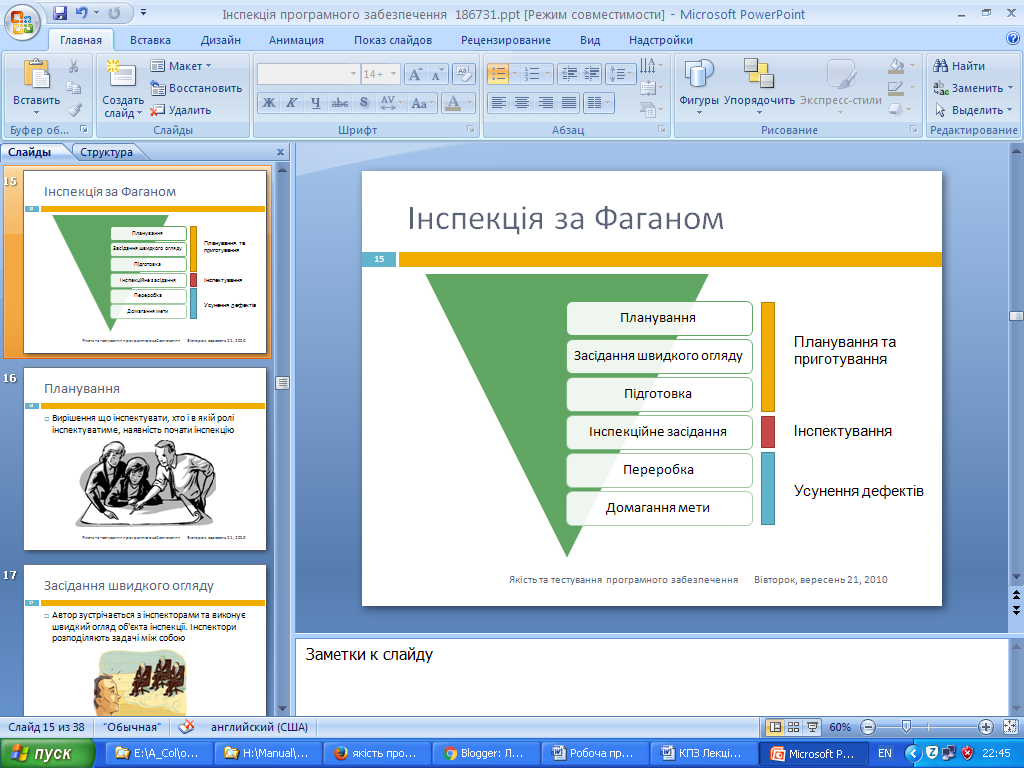
* Виконання власне інспекції обраних робочих продуктів
* Проведення інспекторських зустрічей
* Запис результатів інспекції

**Усунення дефектів**

* Виявлені дефекти мають бути усунені людьми, які відповідають за конкретні робочі продукти
* Проведені усунення дефектів мають бути підтверджені

**Інспекція за Фаганом (Fagan)**

Процесс, запропонований Фаганом, управляючим розробкою ПЗ у IBM в 1976 р. Найбільш відома робота з інспекції ПЗ. Використовується у різних галузях та для різних робочих продуктів Майже усі інші процеси інспекцій вважаються похідними від процесу, запропонованого Фаганом.

****

Планування. Вирішення що інспектувати, хто і в якій ролі інспектуватиме, наявність почати інспекцію

Засідання швидкого огляду. Автор зустрічається з інспекторами та виконує швидкий огляд об'єкта інспекції. Інспектори розподіляють задачі між собою

Підготовка. Індивідуальна інспекція проводиться кожним інспектором, увага приділяється можливим дефектам та сумнівним частинам

Інспекційне засідання. Збирають та консолідують окремі результати інспекції. Визначення дефекту відбувається шляхом досягнення консенсусу учасників

Переробка. Автор виконує переробку робочого продукту для усунення дефекту або забезпечує інший відгук

Домагання мети. Закриття процесу інспекції проведенням остаточної перевірки

**Інспектори за Фаганом**

* Інспекцію проводять приблизно чотири інспектори
* Інспекторів визначають на етапі планування серед досвідчених інженерів, які знайомі з об'єктами інспекції але самі над ними не працюють (бажано мати людей різних спеціалізації, ролей та досвіду)
* При розподілі завдань необхідно визначитись із загальним покриттям інспекції та зонами фокусування уваги

**Висновки застосування інспекції за Фаганом**

**Інспекція вдвох (two-person inspection)**

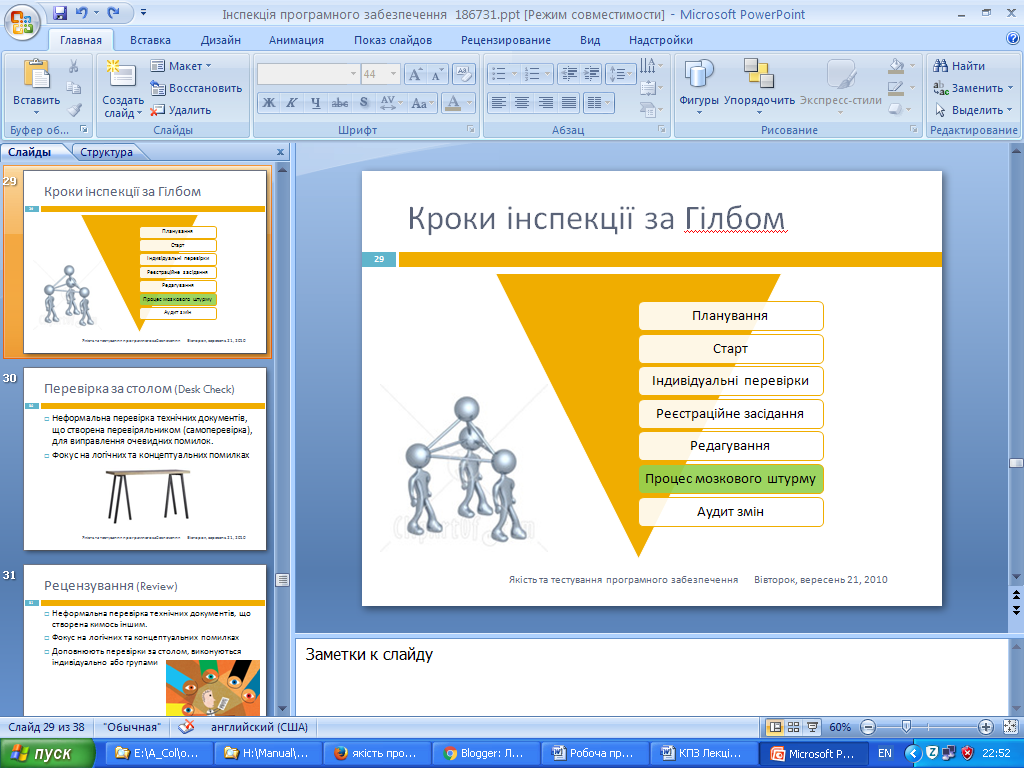
Запропонована для спрощення інспекції за Фаганом, проте слідує її основним етапам. Широко застосовується у ітеративних процесах розробки ПЗ. Типова реалізація – оборотна автор-інспектор пара. Простіше управляти – оскільки користь отримують як автор так і інспектор (за Фаганом – лише автор)

**Інспекція за Гілбом (Gilb)**

* На інспекційному засіданні інспектори прагнуть окрім виявлення дефектів з'ясувати їх причини та запропонувати виправлення
* Гілб вирішує проблему шляхом додавання до процесу інспекції крок “Процес мозкового штурму”.
* У центрі уваги цього кроку є аналіз причин, спрямований на вироблення профілактичних заходів для зниження ін'єкцій дефектів у ПЗ

**Особливості інспекції за Гілбом**

* Входом процесу інспекції є документи правила, контрольні переліки. Наголос що будь-який технічний документ може бути інспектований
* Виходом є виправлені вхідні документи, та пропозиції по вдосконаленню процесу
* Процес інспектування формує цикл із зворотнім зв'язком

**Кроки інспекції за Гілбом**

**Перевірка за столом (Desk Check)**

* Неформальна перевірка технічних документів, що створена перевіряльником (самоперевірка), для виправлення очевидних помилок.
* Фокус на логічних та концептуальних помилках
  + - Рецензування (Review)
* Неформальна перевірка технічних документів, що створена кимось іншим.
* Фокус на логічних та концептуальних помилках
* Доповнюють перевірки за столом, виконуються індивідуально або групами

**Проходження (Walkthrough)**

* Спеціальна, більш організована форма рецензування для програмних коду та моделей
* Використовуються засідання де головує автор
* Імітування виконання програми (перевірка чи підходять алгоритми для вирішення завдань)

**Формальне інспектування – прийоми читання коду**

* Читання з покроковим абстрагуванням
* Декомпозиція дозволяє фокусуватися на частинах програми, потім абстрагуватись від них та фокусуватись на частинах більш високого рівня

**Прийоми виявлення дефектів**

* Виявлення дефектів спонтанно (Виявлення дефектів не застосовуючи жодних прийомів)
* Виявлення дефектів на основі контрольних списків
* Використання різних контрольних списків для гарантії покриття важливих частин документів
  + Контрольні списки по робочим продуктам – перевірка основних функцій, структур даних, визначень даних компонентів
  + Контрольні списки по властивостям – перевірка стилів коду, відповідність стандартам, зв'язаності та залежностей модулів
* Виявлення дефектів на основі сценаріїв. Сценарії використання системи застосовуються для управління пошуком дефектів, що об'єднує декілька компонентів ПЗ

***Спільний перегляд***

Призначення процесу спільного перегляду, відповідно до ISO / IEC 12207, полягає в тому, щоб «підтримувати відповідний рівень розуміння замовником того, як проект просувається до цілей договору. Спільні перегляди проводяться в результаті успішного здійснення процесу:

* перегляди будуть проводитися в заздалегідь визначені терміни;
* стан і продукти діяльності процесу будуть оцінюватися в ході виконання спільного перегляду за участю замовників, постачальників та інших відповідальних осіб і користувачів;
* результати перегляду доводитимуться до всіх сторін, яких вони стосуються;
* заходи, рішення про виконання яких прийнято при переглядах, відстежуватимуться до їх завершення;
* проблеми будуть ідентифікуватися і вирішуватися

Мета періодичних спільних переглядів на рівні управління проектом - оцінити стан проекту по відношенню до затверджених планів, графіків, стандартам і посібникам, встановити, чи немає необхідності в зміні планів, чи правильно розподілені ресурси, чи виконується управління ризиками.

Мета періодичних спільних переглядів на технічному рівні – оцінити стан робочих продуктів ПЗ по відношенню до вимог замовника і критеріям приймання, зафіксованим в договорі.

Мета внутрішніх періодичних спільних переглядів процесу – оцінити його спроможність і придатність для проекту.

***Аудит***

Призначення процесу аудиту, згідно з ISO / IEC 12207, полягає в незалежному встановленні відповідності вибраних продуктів і процесів вимогам, планам та договором. В результаті успішного здійснення процесу:

* буде розроблена і впроваджена стратегія аудиту;
* аудити будуть проводитися;
* узгодженість обраних робочих продуктів ПЗ і / або послуг або процесів до вимог, планами та договором буде встановлюватися відповідно до стратегії аудиту;
* проведення аудитів придатною для цього незалежною стороною буде організовано;
* проблеми, виявлені під час аудиту, будуть ідентифіковані, доведені до тих, хто несе відповідальність за заходи з коригування, і вирішені.

Стандарт визначає робочі продукти і результати діяльності з процесу ЖЦ, що підлягають аудиту:

* Відповідність *коду* програмного продукту проектної документації;
* Адекватність *вимог* до приймальних переглядами і випробувань, зафіксованих в документації, *приймання* програмного продукту;
* Відповідність *тестових даних* специфікації;
* Коректність *звітів про випробування;*
* *Усунення розбіжностей* між фактичними і очікуваними результатами;
* Відповідність *документації користувача* встановленим стандартам;
* Відповідність виконаних *дій* застосовним до них вимогам, планам і договором;
* Відповідність *витрат і термінів* плановим.

**Рівні цілісності програмного забезпечення**

Рівні цілісності ПЗ системи зв'язуються з діапазоном значень критичних характеристик ПЗ (надійності, безпеки функціонування, захисту інформації, продуктивності, складності та ін), які дозволяють утримувати ризики в прийнятних межах. Критичне вискоцілісне ПЗ вимагає V & V, більше за обсягом і суворістю, ніж не критичне.

|  |  |  |
| --- | --- | --- |
| **Критичність** | **Опис** | **Рівні** |
| Виска | Обрана функція служить критичним фактором для ефективної роботи системи | 4 |
| Велика | Обрана функція потрібна для ефективної роботи системи | 3 |
| Помірна | Обрана функція впливає на роботу системи, але можуть бути реалізовані стратегії обходу для компенсації втрати ефективності | 2 |
| Низька | Обрана функція вносить істотний внесок в роботу системи. Якщо вона не буде виконуватися належним чином, користувач відчує незручності | 1 |

**Контрольні питання й завдання**

1. Визначте поняття якість ПС і рівні моделі якості ПС.

2. Визначте характеристики якості ПС і їхнє призначення.

3. Які методи визначають показники якості?

4. Визначте метрики програмного продукту і їхні складові.

5. Які існують стандарти з якості ПС?

6. Назвіть основні цілі й завдання системи керування якістю.

7. Визначте типи моделей надійності і їхній базис.

8. У чому різниця марковських і пуассонівських моделей надійності?

9. Сформулюйте параметри й припущення моделі Джелинського.

10. Визначте процеси досягнення надійності на ЖЦ.

11. Що таке сертифікація програмного продукту?

**Використана література**

1. *ISO/IEC* 9126. Infofmation Technology. – Software Quality Characteristics andmetrics. – 1997.

2. *ДСТУ* 2844–1994. Программные средства ЭВМ. Обеспечение качества. Термины и определения.

3. *ДСТУ* 2850–1994. Программные средства ЭВМ. Обеспечение качества. Показатели и методы оценки качества программного обеспечения.

4. *ДСТУ* 3230–1995. Управление качеством и обеспечение качества. Терминыи определения.

5. *Барлоу Р., Прошан Ф.* Математическая теория надежности. Пер. с англ. М.: 1969. – 483 с.

6. *Липаев В.В.* Надежность программного обеспечения АСУ. – М.: Сов.радио, 1977. – 400 с.

7. *Лаврищева Е.М.* Методы программирования. Теория, инженерия, практика.–Киев: Наук. думка, 2006.–451с.

8. *Гласс Г.* Руководство по надежному программированию. – М.: Финансы и Статистика, 1982. – 256 с.

9. *Тейер Т., Липов Р., Нельсон Э.* Надежность программного обеспечения. – М.: Мир, 1981. – 325 с.

10. *Мороз Г.Б., Лаврищева Е.М.* Модели роста надежности програмного обеспечения. – Киев: ИК АН УССР.–Препр. 92–38, 1992. – 23 с.

11. *Meyer B*. The role of Object–Oriented Metrics.–Computer, 1998.–№11.–P. 23–125.

12. *Кулаков А.Ю*. Оценка качества программ ЭВМ. – Киев: Технiка. – 1984. – 167 с.

13. *Липаев В.В.* Методы обеспечения качества крупномасштабных программных систем. – М.: СИНТЕГ. – 2003. – 510 с.

14. *Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю.* Основы качества программных систем. – Киев: Академпериодика.– Второе изд.– 2007. – 669 с.

15. *NASA* –STD–2201 / Software Assurance Standart, 1993.

16. *John D. Musa,* Anthony Iannino, and Kazuhira Okumoto. Software Reliability: Measurement, Prediction, Application. Whippany, NJ: McGraw–Hill, 1987.

17. *Musa J.D. Okumoto K.A*. Logarithmic Poisson Time Model for Software Reliability Measurement // Proc. Sevent International Conference on Software Engineering. – Orlando, Florida. – 1984. – P. 230–238.

18. *Goel A.L.* Software reliability models& Assumptions, Limitations and Applicability// IEEE Trans. – N2. – P. 1411–1423.

19. *Sukert A.N., Goel A.L*. A guidebook for software reliability assessment / Proc. Annual Reliability and Maintainability Symp. – Tokio (Japan). – 1980. – P. 186–190.

20. *Jelinski Z., Moranda P.* Software reliability research /Statistical computer performance evaluation W.–Freiberger, Ed. Academic Press. – 1972. – Р. 465– 484.

21. *Shick G.J., Wolverton R.W.* An analysis of computing software reliability models / IEEE Tras. Software Eng. – V.– SE–4. – № 2. – 1978. – P. 104–120.

22. *Yamada S., Ohba M., Osaki S.* S–shaped software reliability grows modeling for software error detection // IEEE Trans. Reliability. – 1983. – R– 32. – № 5. – P. 475–478.

23. *Schneidewind N.F.* Software Reliability Model with Optimal Selection of Failure Data // IEEE Trans. on Software Eng. – 1993. – № 11. – P. 1095–1104

24. IEEE 1012-2004 Standard for Software Verification and Validation. IEEE, 2005.