**Лекція 12.** Якість програмного забезпечення. Забезпечення якості

**Забезпечення якості: Формальний метод**

* **Причини використання**
  + Наявність дефектів:
    - розкривається через тестування / інспекції / і т.д..
  + Дефекти відсутні: формально перевірити.  
    (Формальні методи ► помилки відсутні)

Термін *«формальні методи»* має на увазі ряд операцій, до складу яких входять створення формальної специфікації системи, аналіз і доказ специфікації, реалізація системи на основі перетворення формальної специфікації до програм і верифікація програм. У формальній мові системної специфікації закладені математичні концепції. При цьому використовується область дискретної математики, заснованої на алгебрі, теорії множин та алгебри логіки.

У 1980-х рр. багато дослідників вважали, що формальні специфікації і формальні методи є найбільш ефективним шляхом підвищення якості ПЗ. Вони привели вагомі аргументи на користь того, що строгий і детальний аналіз, який є невід'ємною частиною формальних методів, призведе до створення програм з малою кількістю помилок. Експерти передбачали, що до XXI сторіччя більшість програмного забезпечення буде розроблятися з використанням формальних методів.

Однак програмна інженерія пішла іншим шляхом. Минуло більше 25 років досліджень з використання математичних методів в процесі створення програмного забезпечення. Так звані формальні методи розробки програмних систем не отримали широкого визнання, незважаючи на те, що при їх використанні може бути досягнуто підвищення якості програм шляхом докази їх правильності. Багато компаній, що розробляють програмне забезпечення, не вважають економічно вигідним застосування цих методів в процесі розробки.

Розвиток структурних методів, управління конфігурацією і та інших дозволило підвищити якість програм при більш низьких витратах в порівнянні з вартістю розробки формалізованими методами. Крім того, з формальної специфікації погано узгоджуються методи швидкої розробки програмного забезпечення. Це вступає в протиріччя з тим, що в даний час головним критерієм програмної індустрії для деяких класів систем є не якість, а час поставки їх на ринок.

Формальні математичні специфікації не містять деталей реалізації системи, але повинні представляти її повну математичну модель. Існує два основні підходи до розробки формальної специфікації:

* алгебраїчний підхід, при якому система описується в термінах операцій і їх відносин;
* підхід, орієнтований на моделювання, при якому модель системи будується з використанням математичних конструкцій, таких, як безлічі і послідовності, а системні операції визначаються тим, як вони змінюють стан системи.

Формальні методи виявилися рентабельні в обмеженій області застосування: це розробка критичних систем, де важливі такі властивості, як безпека, безвідмовність і захищеність. Прикладами критичних систем, при розробці яких успішно застосовувалися формальні методи, є інформаційні системи управління повітряним транспортом, системи сигналізації на залізниці, бортові системи космічних кораблів і медичні системи управління

Узагальнемо:

* **Основні ідеї**
  + Поведінка офіційно зазначена:
    - перед / пост умови, або
    - як математичні функції.
  + Перевірити « правильність »:
    - проміжних станів / кроків,
    - аксіом і композиційних правил.
  + Підходи: аксіоматичний, функціональний

**Забезпечення якості: Інспекції**

* Наявність обов’язкових фаз: вимоги / дизайн / 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

***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.

* 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

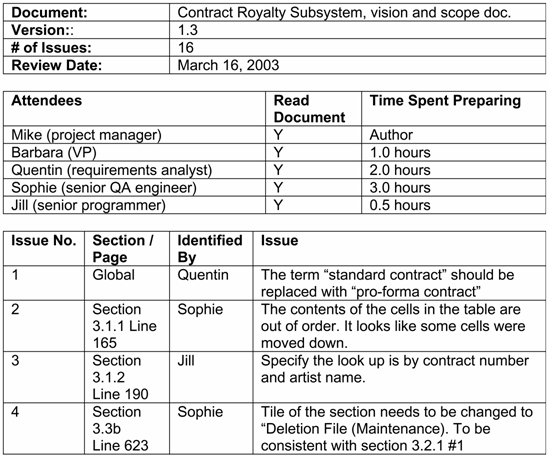
На початковому етапі проектування інспектування передбачає перевірку повноти, цілісності, однозначності, несуперечності та сумісності робочих продуктів з вихідними вимогами до програмної системі.

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**

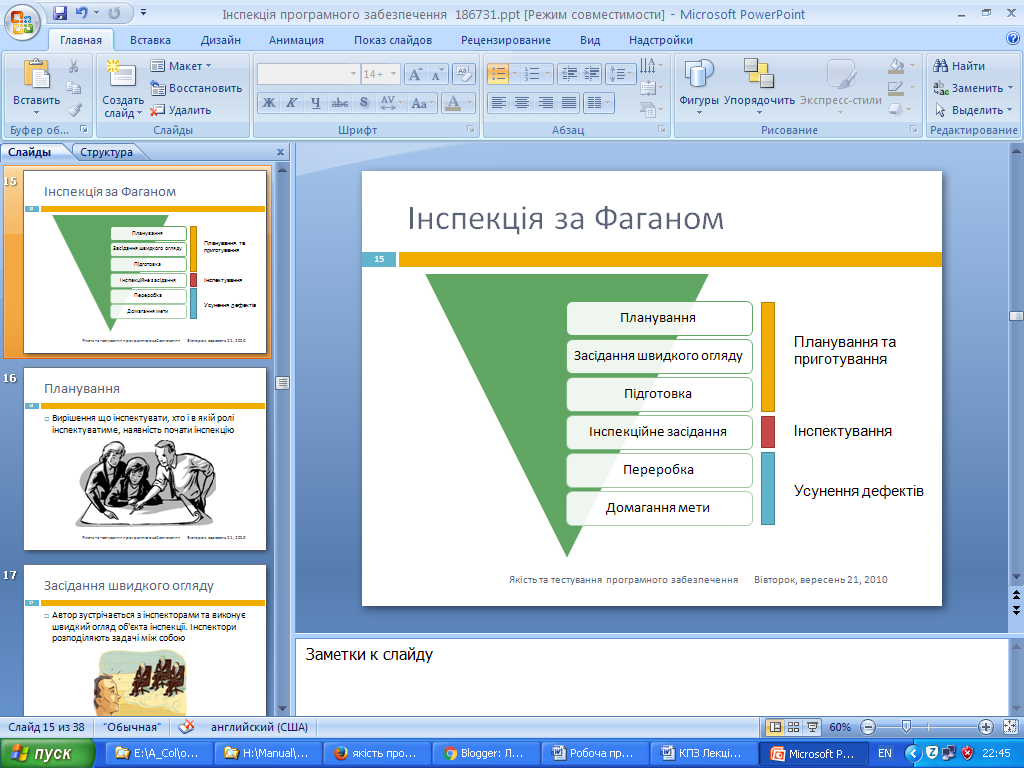
* Виконання власне інспекції обраних робочих продуктів
* Проведення інспекторських зустрічей
* Запис результатів інспекції

**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)

**Парне програмування**

Інспекція без засідань.

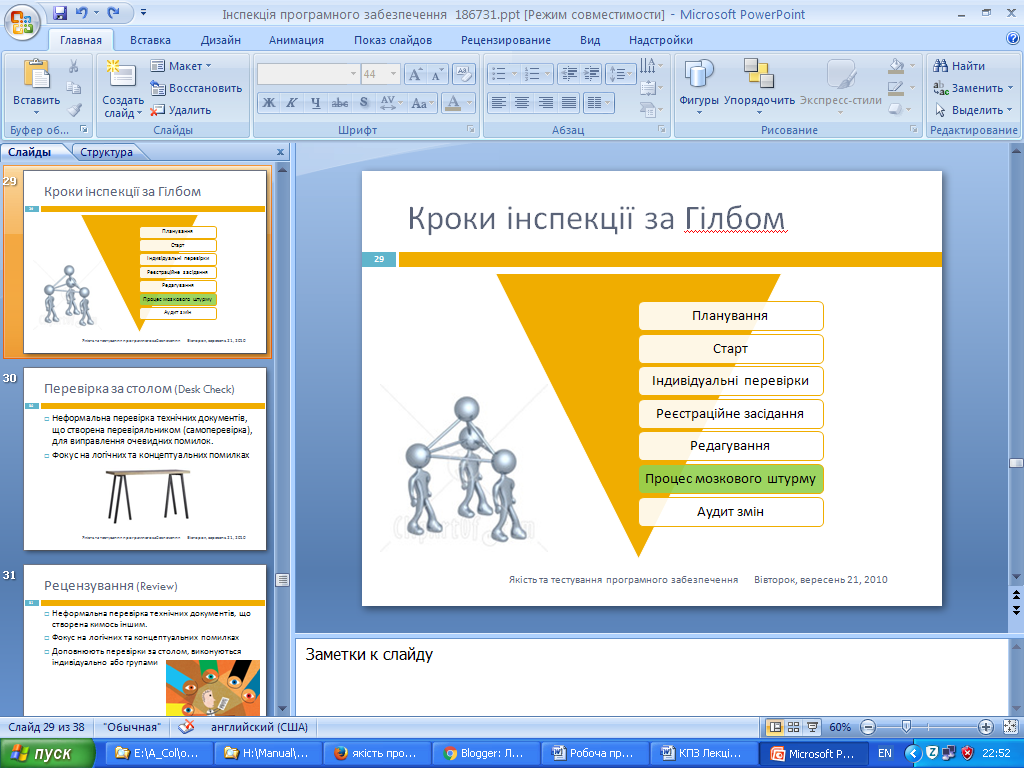
* На засіданнях виявляють лише 5-30% дефектів
* Засідання забирають багато ресурсів
* Інспекція без засідань зменшує витрати не набагато погіршуючи ефективність
* Збільшує кількість помилкових тривог

**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

**Забезпечення якості: Тестування**

* **Характеристики Продукту / Процесу:**
  + Об'єкт: тип продукту, мова і т.д.
  + Масштаб / впорядкованість: елементи, компоненти, системи
  + Хто: автономний, незалежний, третя сторона
* **Як перевірити:**
  + Верифікація і валідація
  + Зовнішні специфікації (чорний ящик)
  + Внутрішня реалізація (білий / прозорий ящик)
* Критерії: коли потрібно зупинитися?
  + висвітлення специфікації / структур.
  + надійність ►на основі використання тестування

**Забезпечення якості: Відмово-стійкість (FT -** Fault Tolerance**)**

* **Причини використання**
  + Присутня несправність, але видалення неможливе / недоцільне
  + відмово-стійкість ► наявність дефектів
* **Методи FT**: розрив зв’язку “дефект - відмова”
  + Відновлення: відкат і повтор
  + NVP: програмування N-версії несправність блокована

**Відмовостійкість** або **поступо́ва деґрада́ція** (*graceful degradation*), або **поступо́ве скоро́чення можли́востей**, або **поступо́вий ви́хід із робо́ти**, або **пла́вне зни́жування ефекти́вності** це властивість комп'ютерної системи, що дозволяє їй продовжувати правильно діяти у випадку помилки або декількох помилок в деяких її частинах. Якщо при цьому падає якість експлуатації, то це відбувається пропорційно до серйозності помилки. Відмовостійкість особливо важлива у високо доступних та життєво критичних системах. Наприклад, TCP розроблений для уможливлення надійного двобічного зв'язку в мережах з комутацією пакетів, навіть за ненайкращого і перевантаженого стану ліній зв'язку. Він робить це через вимогу до кінцевих точок комунікацій *очікувати* втрату пакетів, повторення, невірний порядок і пошкодження, таким чином ці умови не шкодять цілістності даних, а лише зменшують пропускну спроможність на пропорційну величину.

Відмовостійка система має:

1. Мати більш ніж одну точку відновлення (наприклад, перехоплення керування при відмові може бути виконано одним з двох серверів)
2. Локалізація пошкоджень у пошкодженому компоненті
3. Обмеження розповсюдження несправності
4. Доступність режимів реверсії

Додатково, відмовостійка система характеризується в термінах планових і позапланових перерв в роботі служби. Зазвичай ця величина вимірюється саме на застосунковому рівні, а не на рівні апаратного забезпечення. Ця характеристика зветься доступністю і виражається у відсотках. Наприклад, система п'яти дев'яток статистично забезпечує доступність на рівні 99.999%. Відмовостійкі системи зазвичай базуються на концепції надлишковості.

N-версійне програмне забезпечення. Відомо, що в наш час найбільшу загрозу для сучасних обчислювальних систем представляють несправності проектування. Різке зростання складності програмного забезпечення призвело до того, що стало практично неможливим виявлення та видалення всіх несправностей проектування ПЗ до вводу системи в фазу використання. Наприклад, була виявлена несправність проектування в мікропроцесорі Pentium, з яким уже були виготовлені та розповсюджені тисячі комп’ютерів. Ця несправність коштувала фірмі-виробнику $475 мільйонів. Таких прикладів можна привести безліч. Фундаментальним рішенням проблеми несправностей проектування в ПЗ є принцип різноманітності проектів. Різноманітність проектів виникла як аналог дублювання апаратних засобів. Але в апаратурі застосовується просте копіювання каналів обчислення, бо воно є ефективним засобом маскування випадкових фізичних несправностей апаратних засобів. Проте копіювання блока ПЗ буде копіювати і невиявлені в ньому несправності проектування. Тому кожен блок ПЗ, що дублює виконання однієї і тієї ж функції, повинен бути розроблений незалежно від інших блоків. У цьому і полягає концепція різноманітності проектування програмного забезпечення або багатоверсійних обчислень.

**Забезпечення якості: Забезпечення безпеки**

* Розширення ідеї відмово-стійкості для забезпечення безпеки:
  + Від “допустимості” дефектів до “допустимості” відмов
* Поняття, пов'язані з безпекою:
  + Безпека: безаварійність
  + Аварія: відмова, тяжкі наслідки
  + Небезпека: передумова аварії
* Забезпечення безпеки:
  + Аналіз небезпек
  + Усунення / скорочення / контроль небезпеки
  + Ремонтно-відновлювальні роботи

**Забезпечення якості потребує додаткової підтримки:**

* + Планування та постановка цілей
  + Управління:
    - Коли потрібно зупинитися?
    - Коригування і вдосконалення, і т.д.
    - Засноване на оцінках / прогнози

Як об'єкт забезпечення технологічної і експлуатаційної безпеки ПЗ розглядається вся сукупність його компонентів в рамках конкретної системи. Як домінуюча повинна використовуватися стратегія наскрізного тотального контролю технологічного і експлуатаційного етапів життєвого циклу компонентів ПЗ. Сукупність заходів щодо забезпечення технологічної і експлуатаційної безпеки компонентів ПЗ повинна носити конфіденційний характер. Необхідно забезпечити постійний, комплексний і дієвий контроль за діяльністю розробників і користувачів компонентів. Окрім загальних принципів, звичайно необхідно конкретизувати принципи забезпечення безпеки ПЗ на кожному етапі його життєвого циклу. Далі приводяться один з варіантів розробки таких принципів.

*Принципи забезпечення технологічної безпеки при обгрунтуванні, плануванні робіт і проектному аналізі П*З

Принципи забезпечення безпеки ПЗ на даному етапі включають наступні принципи:

*Комплексності забезпечення безпеки П*З, що передбачає розгляд проблеми безпеки інформаційно-обчислювальних процесів з урахуванням усіх структур системи, можливих каналів просочування інформації і несанкціонованого доступу до неї, часу і умов їх виникнення, комплексного застосування організаційних і технічних заходів.

*Планування застосування засобів безпеки програм*, що передбачає перенесення акценту на сумісне системне проектування ПЗ і засобів його безпеки, планування їх використання в передбачуваних умовах експлуатації.

*Обгрунтованості засобів забезпечення безпеки П*З, що полягає в глибокому науково-обгрунтованому підході до ухвалення проектних рішень за оцінкою ступеня безпеки, прогнозуванні загроз безпеці, всебічній апріорній оцінці показників засобів захисту.

*Достатності безпеки програм*, що відображає необхідність пошуку найбільш ефективних і надійних заходів безпеки при одночасній мінімізації їх вартості.

*Гнучкості управління захистом програм*, що вимагає від системи контролю і управління забезпеченням інформаційної безпеки ПЗ здібності до діагностування, випереджаючої нейтралізації, оперативного і ефективного усунення виникаючих загроз в умовах різких змін обстановки інформаційної боротьби.

*Завчасності розробки засобів забезпечення безпеки і контролю виробництва П*З, що полягає в попереджувальному характері заходів забезпечення технологічної безпеки робіт на користь недопущення зниження ефективності системи безпеки процесу створення ПЗ.

*Можливості документування технології створення програм*, що має на увазі розробку пакету нормативно-технічних документів з контролю програмних засобів на наявність навмисних дефектів.

*Принципи досягнення технологічної безпеки ПЗ в процесі його розробки*

Принципи забезпечення безпеки ПЗ на даному етапі включають наступні принципи:

*Регламентації технологічних етапів розробки П*З, що включає впорядковані фази проміжного контролю, специфікацію програмних модулів і стандартизацію функцій і формату представлення даних.

*Автоматизації засобів контролю керуючих та обчислювальних програм* на наявність дефектів, створення типової загальної інформаційної бази алгоритмів, вихідних текстів і програмних засобів, що дозволяють виявляти навмисні програмні дефекти.

*Послідовної багаторівневої фільтрації програмних модулів* в процесі їх створення із застосуванням функціонального дублювання розробок і поетапного контролю.

*Типізації алгоритмів*, програм і засобів інформаційної безпеки, що забезпечує інформаційну, технологічну і програмну сумісність, на основі максимальної їх уніфікації по всіх компонентах і інтерфейсах.

*Централізованого управління базами даних проектів ПЗ* і адміністрування технології їх розробки з жорстким розмежуванням функцій, обмеженням доступу відповідно до засобів діагностики, контролю і захисту.

*Блокування несанкціонованого доступу* співвиконавців і абонентів державних мереж зв'язку, підключених до стендів для розробки програм.

*Статистичного обліку і ведення системних журналів* про всі процеси розробки ПО з метою контролю технологічної безпеки.

*Використання тільки сертифікованих і вибраних в якості єдиних інструментальних засобів розробки програм* для нових технологій обробки інформації і перспективної архітектури обчислювальних систем.

*Принципи забезпечення технологічної безпеки на етапах стендових і приймально-здавальних випробувань*

Принципи забезпечення безпеки ПЗ на даному етапі включають наступні принципи:

*Тестування ПЗ* на основі розробки комплексів тестів, що параметризуються на конкретні класи програм з можливістю функціонального і статистичного контролю в широкому діапазоні зміни вхідних і вихідних даних.

*Проведення натурних випробувань програм* при екстремальних навантаженнях з імітацією дії активних дефектів.

*Здійснення "фільтрації"* програмних комплексів з метою виявлення можливих навмисних дефектів певного призначення на базі створення моделей загроз і відповідних скануючих програмних засобів.

*Розробки і експериментального відпрацювання засобів верифікації* програмних виробів.

*Проведення стендових випробувань ПЗ* для визначення ненавмисних програмних помилок проектування і помилок розробника, що призводять до невиконання цільових функцій програм, а також виявлення потенційно "вузьких" місць в програмних засобах для руйнівної дії.

*Відпрацювання засобів захисту від несанкціонованого впливу* порушників на ПЗ.

*Сертифікації програмних виробів АСУ на вимоги безпеки* з випуском сертифікату відповідності цього виробу вимогам технічного завдання.

*Принципи забезпечення безпеки при експлуатації програмного забезпечення*

Принципи забезпечення безпеки ПЗ на даному етапі включають наступні принципи:

*Збереження і обмеження доступу* до еталонів програмних засобів, недопущення внесення змін в них.

*Профілактичного вибіркового тестування і повного сканування* програмних засобів на наявність навмисних дефектів.

*Ідентифікації ПЗ* на момент введення його в експлуатацію відповідно до передбачуваних загроз безпеці ПЗ і його контроль.

*Забезпечення модифікації програмних виробів* під час їх експлуатації шляхом заміни окремих модулів без зміни загальної структури і зв'язків з іншими модулями.

*Строгого обліку і каталогізації* всіх програмних засобів, що супроводжуються, а також інформації що збирається, оброблюється та зберігається.

*Статистичного аналізу інформації* про всі процеси, робочі операції, відхилення від режимів штатного функціонування ПЗ.

*Гнучкого застосування додаткових засобів захисту ПЗ* у разі виявлення нових, непрогнозованих загроз інформаційній безпеці.

**Оцінка якості / надійності:**

* + Потреба збору даних
  + Аналіз та моделювання

Ці питання будуть розглядатися детально в лекції ОПІ..

**Забезпечення зворотного зв'язку для управління**

****

**Постановка цілей якості**

* Визначити якість позицій / атрибутів
  + Клієнт / очікування користувачів,
  + Стан ринку,
  + Тип продукту і т.д.
* Вибрати прямі вимірювання якості
  + Прямі: надійність
  + Вимірювання дефектів
  + Інші вимірювання
* Очікувана оцінка якості в порівнянні вартістю
  + Дослідження вартості якості/дефектів
  + Економічні моделі: COCOMO т.д.

***Контрольні запитання*.**

1. Де доцільно використовувати формальні методи забезпечення якості?
2. В чому полягає інспекція ПЗ?
3. Які види інспекції ПЗ ви знаєте?.
4. В чому полягає процес, запропонований Фаганом?
5. Як визначається відмовостійкість?
6. Які властивості повинна мати відмовостійка система?
7. Як можна забезпечити безпеку ПЗ?

***Література.***

1. Лавріщева К.М. Програмна інженерія. Електронний підручник. URL: <http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf>.
2. Соммервил И. Инженерия программного обеспечения / И.Соммервил.–М. : Издательский дом «Вильямс», 2002. –623 с.
3. ДСТУ ІS0/ІЕС 9126-1:2013 (ІS0/ІЕС 9126-1:2001, IDТ) Національний стандарт України. Програмна інженерія. Якість продукту. Частина 1. Модель якості.
4. ДСТУ ISO 9001:2015 (ISO 9001:2015, IDT) Національний стандарт України. Системи управління якістю.
5. *Голицина О.Л., Партыка Т.Л. «Програмное обеспечение», 2-ое издание, Москва, 2008*
6. *Гальдштейн Б.С. «Системы коммутации».- «БХВ – Санкт – Петербург», 2003. Стор. 242-243.*

***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.