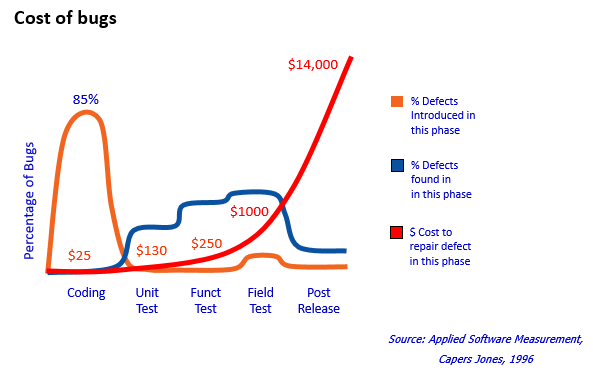
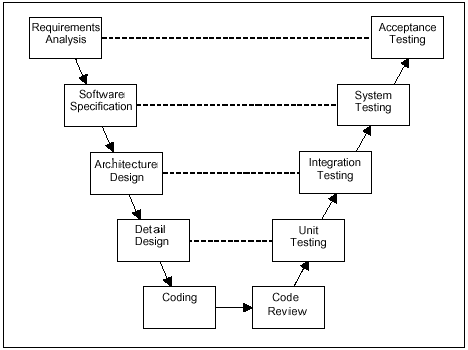
**NHẬP MÔN KỸ THUẬT PHẦN MỀM**

**Bài 12 - Lý thuyết 07: Testing Basic**

1. **Nội dung:**
   * Unit Test Fundamentals
   * Manual Unit Testing
   * Automated Unit Testing
2. **Unit Test Fundamentals**
   1. **Unit Test Fundamentals**
   * Nguyên tắc cơ bản của Unit Test là phải trả lời các câu hỏi:
   * WHAT and WHO
   * WHY
   * WHEN
   * HOW
   * Unit Test Technique: Introduce approaches and techniques to do unit test
   1. **Unit Test Fundamentals – WHAT and WHO ?**
   * Unit Testing Actions:
   * Xác nhận rằng các unit của phần mềm đang làm việc đúng.
   * Unit là các phần nhỏ nhất trong phần mềm (nó có thể là function, procedure,... Trong lập trình hướng đối tượng thì Unit luôn luôn là method)
   * Units khác với modules. Module được tạo thành từ các units
   * Unit Testing Deliverables:
   * Tested software units
   * Tài liệu liên quan (Unit Test case, Unit Test Report)
   * Unit Testing Conductor: Development Team
   1. **Unit Test Fundamentals – WHY?**
   * Đảm bảo chất lượng của các đơn vị phần mềm
   * Phát hiện lỗi và các vấn đề sớm
   * Nâng cao chất lượng phần mềm & giảm chi phí sữa chữa



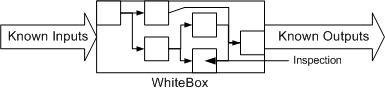
* 1. **Unit Test Fundamentals – WHEN?**
  + Sau khi Coding
  + Trước khi kiểm thử tích hợp (Integration Test)



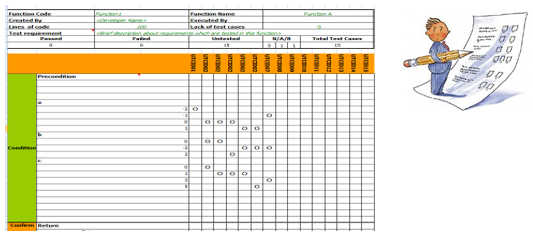
* 1. **Unit Test Fundamentals – HOW?**
  + ***Black-box testing***
  + Functional testing: chắc chắn mỗi unit hoạt động đúng theo thiết kế.
  + Business testing: chắc chắn software program hoạt động đúng theo yêu cầu requirement.



* + ***White-box testing***
  + Kiểm tra cú pháp của Code bởi trình biên dịch để tránh các lỗi cú pháp.
  + Chạy Code trong chế độ gỡ lỗi, từng dòng, thông qua tất cả các đường dẫn độc lập của chương trình để đảm bảo rằng tất cả các code đã được thực hiện ít nhất một lần
  + Kiểm tra cấu trúc dữ liệu local để đảm bảo rằng dữ liệu được lưu trữ tạm thời duy trì tính toàn vẹn của nó trong tất cả các bước của mã thực thi
  + Kiểm tra điều kiện biên để đảm bảo rằng mã sẽ chạy đúng như là yêu cầu
  + Xem lại tất cả các xử lý lỗi

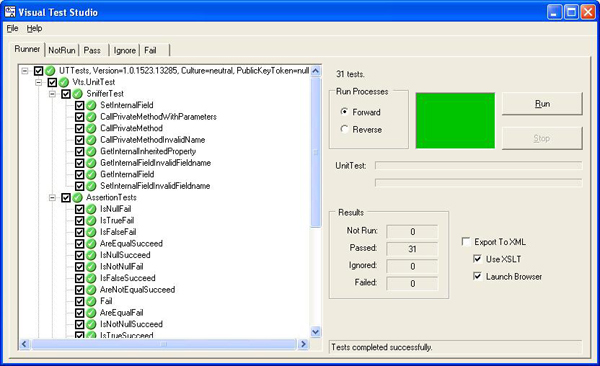


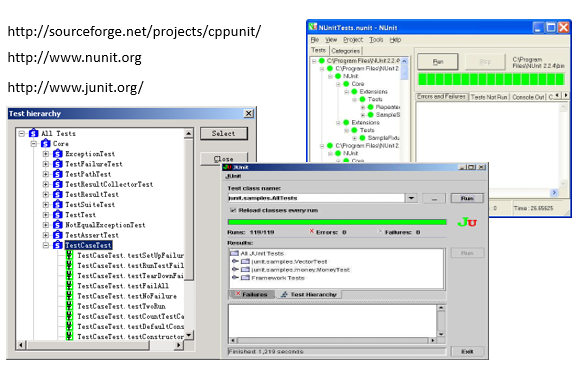
1. **Manual Unit Testing**
2. **Manual Unit Testing**
3. Viết code
4. Uploading the code
5. Build
6. Running the code manually (sử dụng nhiều case từng bước để test)
7. Kiểm tra Log files, Database, External Services, giá trị các biến, Output trên màn hình,…
8. Nếu nó không chạy, lập lại các bước trên
9. **Manual Unit Testing – Hạn chế**
   * Developer nhớ được trường hợp nào thì test trường hợp đó
   * Đến cuối dự án số lượng test case càng lúc càng nhiều, khả năng cover của lập trình viên giảm xuống!
   * Nhiều test case bị trùng lặp
   * Nhiều test case bị lack
   * Team lead không thể review hết được
   * Kết quả dự án chỉ trông chờ vào tester
   * Rất nhiều lỗi phát sinh sau khi system test, đa phần các lỗi xuất phát do Dev test không kỹ từ lúc Unit Test
   * Để giải quyết vấn đề trên, mỗi khi developer test xong phải viết tài liệu mô tả test case trên word hoặc excel
   * Điều này giúp team rất dễ dàng review. Tuy nhiên, cách làm này vẫn có thể có hạn chế



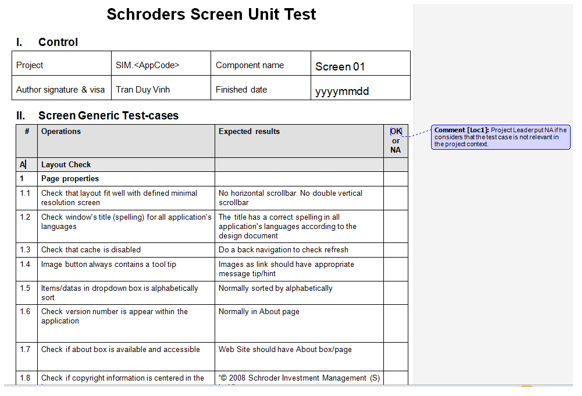
* + ***So, what is the solution?***
  + Các dự án lớn thì số lượng tài liệu test case thường cũng rất lớn!
  + Các dự án lớn thì requirement thường hay thay đổi
  + Mỗi khi requirement thay đổi => Phải sửa code => phải cập nhật lại tài liệu testcase => và lại manual retest , rất tốn sức
  + Càng đến cuối dự án, lượng việc sinh ra càng nhiều , viết test case document trở thành “địa ngục “ thực sự! => dev không còn đủ sức update test case document, tài liệu nhanh chóng bị lạc hậu, hoặc việc update chỉ là đối phó!
  + Một số trường hợp không thể dùng Excel Unit TestCase

1. **Manual Unit Testing - Thảo luận**
   * So, what is the solution?
2. **Automated Unit Testing**
3. **Automated Unit Testing**
   * Coding Process with Automated Unit Tests
   * Viết code
   * Viết một hoặc nhiều test cases script
   * Auto-compile and run (Biên dịch tự động và chạy)
   * Nếu tests fail -> fix lỗi.
   * Nếu tests pass -> lặp lại với method khác
   * Công cụ sử dụng:
   * Java: JUnit, J2MEUnit
   * C/C++: cppUnit
   * Python: pyUnit
   * Perl: PerlUnit
   * Visual Basic: vbUnit
   * C# .NET: Nunit,csUnit
   * Tham khảo:
   * <http://www.testingfaqs.org/t-unit.html>
   * [www.junit.org](http://www.junit.org/)
   * <http://www.codeproject.com/gen/design/autp5.asp>
   * <http://sourceforge.net/projects/cppunit/>
   * <http://www.nunit.org>
   * <http://www.junit.org/>





1. **Unit Testing Document**



1. **THẢO LUẬN MỞ**
   * Nhóm sinh viên cùng làm Unit Test Document theo yêu cầu Giảng viên
2. **Chuẩn bị Workshop 5**
   * Nhóm sinh viên chuẩn bị Unit Test Document cho đề tài đã chọn