Software Verification

Some Established and Experimental Techniques

Presented By: Andrew Carter

NEWFOUNDLAND & LABRADOR, CANADA
WWW.MUN.CA

MEMORIAL UNIVERSITY

Agenda

MEMORIAL UNIVERSITY

- **Introduction**
- **Overview of various verification techniques**
  - What, How, Why format
- **Recap**
- **Review**
- **Questions**

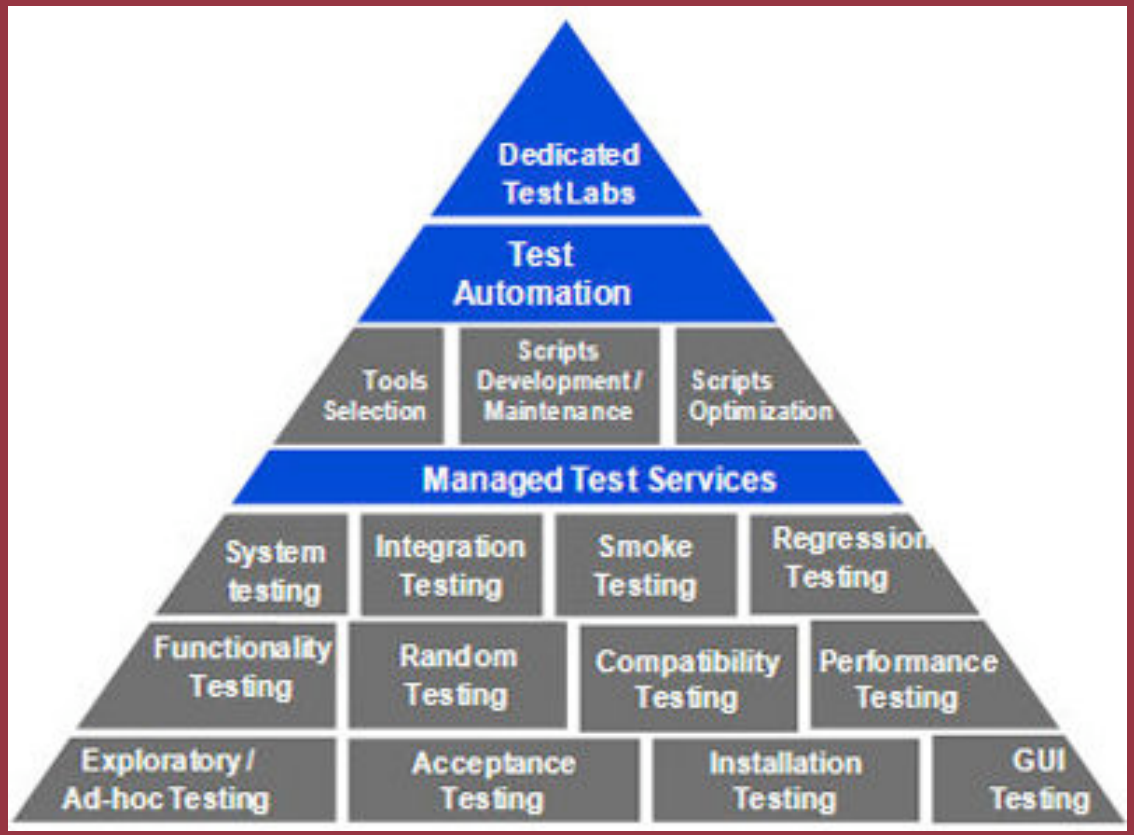
Introduction

MEMORIAL UNIVERSITY

- **What is Software Verification ?**
  - “Software verification is a broad and complex discipline of software engineering whose goal is to assure that software fully satisfies all the expected requirements.”
- **Why am I giving this Presentation ?**
  - To Provide a high level overview of a variety of software verification techniques
  - Some of these are established practices in industry others are experimental and under research

MEMORIAL UNIVERSITY

ESTABLISHED TECHNIQUES



Acceptance Testing

MEMORIAL UNIVERSITY

- **What ?**
  - Umbrella term describing a form of testing in many subfields of engineering
  - Treats system under test (SUT) as a black box upon which test cases are administered
  - A particular test case will focus on one functional area of the SUT
  - Generally no grey area when interpreting result of a test (Boolean pass or fail)
  - Passing agreed upon tests can be a contractual obligation enforced upon a development house by a customer

Acceptance Testing (cont...)

MEMORIAL UNIVERSITY

- **How?**
  - Massive amount of Acceptance Testing done in Industry, thus many approaches exist
  - Some include:
    - Manual completion of test cases by QA
    - Test case automation
    - “User story” approach seen in Extreme Programming (XP)
      - Customers work with developers to create functionality descriptions (stories)
      - Acceptance tests distilled from stories
      - XP iteration not complete distilled tests are passed

Acceptance Testing (cont...)

MEMORIAL UNIVERSITY

- **Why?**
  - Easy to ensure, to a reasonable degree, that functional areas of a program are working
  - Does so in an organized and translucent manner

- **Drawbacks**
  - Cannot uncover bugs in areas of a system which are not covered by test cases
  - Due to relative formalities present, not the most efficient way to rapidly discover bugs

Fuzz Testing

MEMORIAL UNIVERSITY

- **What?**
  - Verification technique by which random (fuzz) input is given to a software system
  - Not intended to validate functionality
  - Instead, intended to unearth “show stopping” bugs
- **How?**
  - In a simplistic implementation just need:
    - Pseudo random number generator
    - Tool to control input of events to SUT

Fuzz Testing (cont...)

MEMORIAL UNIVERSITY

- **Why?**
  - Simplistic concept and design
  - Tools required easily implementable for many systems
  - Provides increased assurance against critical failure when paired with more thorough verification
- **Disadvantages**
  - Likely provides poor code coverage on its own

Usability Testing

MEMORIAL UNIVERSITY

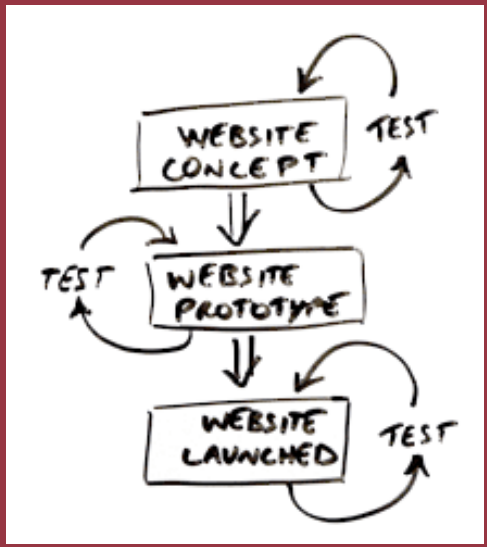
- **What?**
  - Observing typical users interaction with system to come to conclusions about its usability

- **How?**
  - Typical Approach:
  - Find a selection of subjects from the potential user base of the system
  - Have them attempt predefined tasks while members of development staff watch and take notes
  - Poll the users for their opinions such as general satisfaction level with design and creative feedback

Usability Testing (cont...)

MEMORIAL UNIVERSITY

- **Why?**
  - Many projects benefit greatly from results
  - Particularly product who's success relies on users enjoyment and ease of interaction (web apps, etc.)
  - If done in parallel with development, future iterations of system can integrate test conclusions



Regression Testing

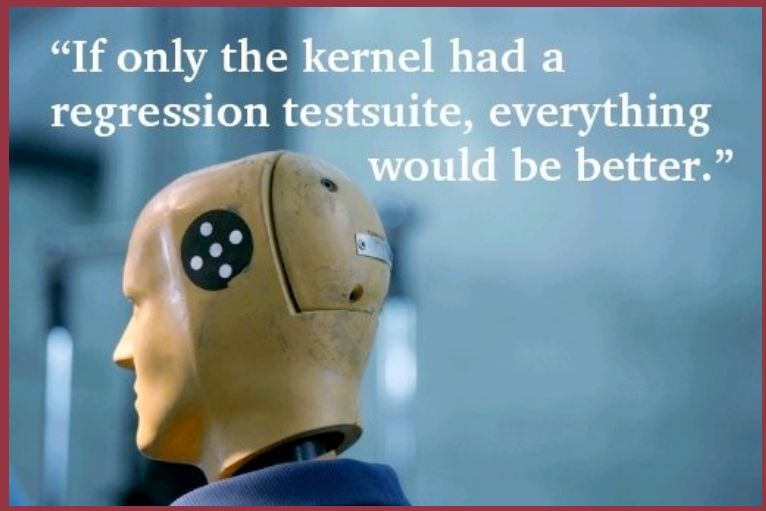
MEMORIAL UNIVERSITY

- **What?**
  - Aims to uncover issues which have emerged in previously working areas of a SUT
  - These issues have likely been caused as a side effect of new development
- **How?**
  - Create a regression test plan used to verify a system with a certain level of code coverage (ideally 100 %)
  - This test plan can involve manual regression testing but automation is ideal

Regression Testing (cont...)

MEMORIAL UNIVERSITY

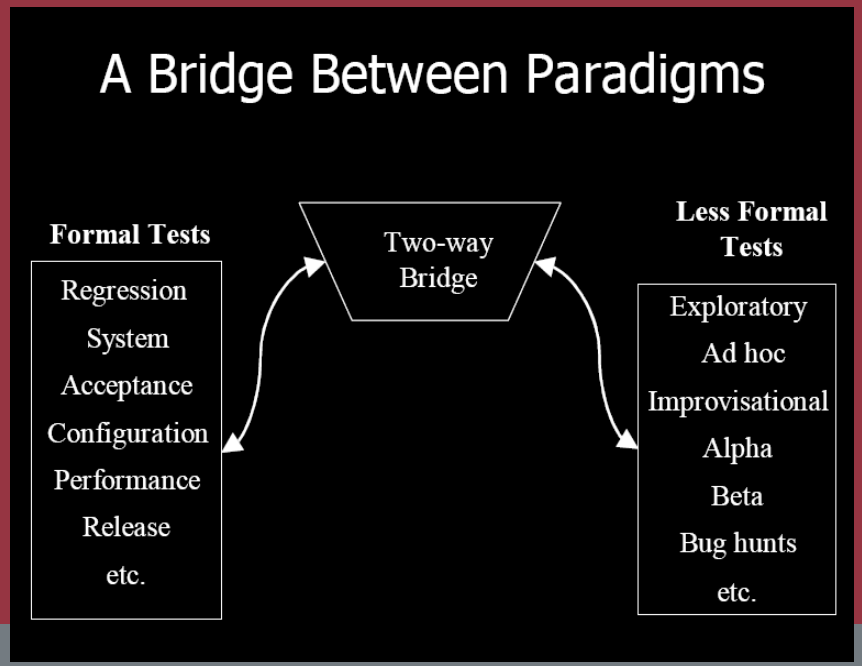
- **Why?**
  - Catch regression bugs, which can be extremely common when new development is done on a large system
  - Helps to validate the expected quality of a system
- **Drawbacks**
  - Considerable amount of overhead and maintenance involved in creating and executing a regression test plan



Exploratory Testing

MEMORIAL UNIVERSITY

- **What?**
  - Defined as “simultaneous learning, test design, and test execution”
  - Not a concrete type of testing; other testing techniques can be classed as exploratory (as we will see shortly)
  - Testing sessions lack specifically defined test cases
  - Instead, tester generates test cases on the fly while interacting with and observing the SUT



Exploratory Testing (cont...)

MEMORIAL UNIVERSITY

- **Why?**
  - Can find obscure bugs not covered by formal test cases
  - Little preparation time required, testers explore system like a typical user
  - Good for testing immature systems with little documentation/test cases
- **Disadvantages**
  - Test procedures cannot be reviewed in advance
  - Hard to know what has been verified and what has not (difficult to reproduce exact actions causing bugs)

Ad hoc Testing

MEMORIAL UNIVERSITY

- **What?**
  - Form of exploratory testing
  - Freeform and unstructured

- **How?**
  - Testers learn about the system in parallel with testing it
  - Create novel test cases on the fly
  - If a bug is found, it is recorded and test case integrated into regression test suite

Ad hoc Testing (cont...)

MEMORIAL UNIVERSITY

- **Why?**
  - Suggested as useful for verifying low level functionality
  - Testing of such functionality can be overlooked by large test cases which verify big features

- **Disadvantages**
  - Like other forms of exploratory testing, hard to guarantee level of quality
  - Therefore, best used to augment formal verification

Session-Based Testing

MEMORIAL UNIVERSITY

- **What?**
  - Exploratory testing who's effectiveness can be tracked by meaningful metrics
  - Fairly Contemporary, Originated by Jonathan and James Bach in 2000
- **How?**
  - "Charters" created prior to a testing session
  - Charters outline goals for the session and high-level details on what should be tested, but no detailed test procedures
  - During a test session (typically 1-2 hours long) tester creates test cases and executes them, recording bugs uncovered

Session-Based Testing (cont..)

MEMORIAL UNIVERSITY

- When tester is finished a session fills out a session sheet, which is parsed automatically to generate metric reports
- Finally, test manager debriefs each session to get a feel for test progress and facilitate future planning

- **Why?**
  - Reduce the amount of time spent planning and creating documentation, while still being able to judge product quality
- **Disadvantages**
  - Effectiveness reliant on skill and discipline of testers and test managers

MEMORIAL UNIVERSITY

EXPERIMENTAL TECHNIQUES

Mutation Testing

MEMORIAL UNIVERSITY

- **What?**
  - Unique in that it evaluates effectiveness of test suites (test for tests!)
  - Based on idea that making small changes (mutations) to source code will allow discovery of inadequacies in test design
- **How?**
  - Mutation operators defined by test designer (e.g. change ‘&&’ to ‘||’)
  - Source code modified autonomously based on mutation operators
  - Run “mutant” code against test suite. want to see failures

Mutation Testing (cont...)

MEMORIAL UNIVERSITY

- **Example mutant code block:**

```
bool foo(&bar, something){

    if(!bar && something)
        return true;
    else
        return false;
}

// Becomes

bool foo(&bar, something){

    if(!bar || something)
        return false;
    else
        return true;
}
```

Mutation Testing (cont...)

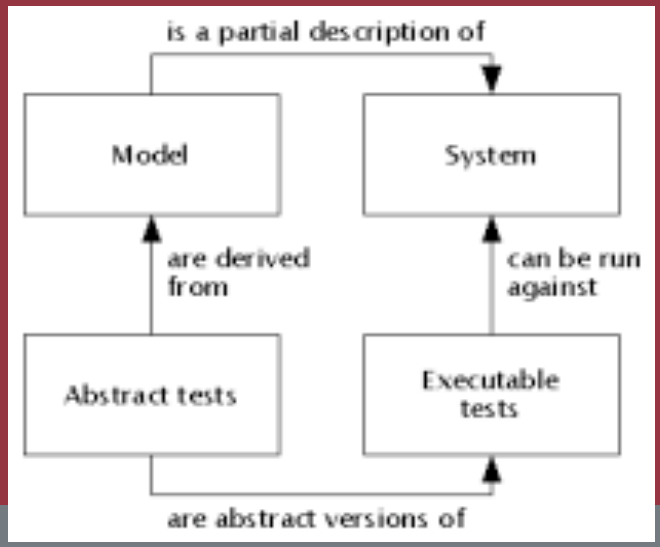
MEMORIAL UNIVERSITY

- **Why?**
  - Evaluate weaknesses in test suite
    - If test passes on mutant code could indicate that test cases are inadequate, or code is redundant and needs refactoring
- **Disadvantages**
  - When large number of mutation operators used, computationally expensive
    - If many mutation operators used, the number of source code permutations becomes prohibitive
  - Research has been done in an attempt to address this issue

Model-Based Testing (MBT)

MEMORIAL UNIVERSITY

- **What?**
  - Test beds derived from well defined modular sections of a system
  - Product of R&D in the area of Model Driven Engineering (MDE)
    - In MDE, software systems synthesized from a platform independent model (PIM) into platform specific model (PSM)
  - MBT uses models from MDE to derive corresponding tests for the system algorithmically



Model-Based Testing (cont…)

MEMORIAL UNIVERSITY

- **How?**
  - Still very much a research topic
  - Several distinct methods have been utilized to derive test cases to date
  - For instance, event-flow model can be used to create GUI's. In event-flow model, Each vertex of a graph represents an event (i.e. click Ok button)
  - GUI's can be created this way or reverse engineered to event-flow models
  - Once event-flow model obtained, test-oracles, which compare expected to actual output are applied to verify GUI functionality
  - Other techniques to generate test cases from models include: theorem proving, symbolic execution and constraint logic programming

Model-Based Testing (MBT)

MEMORIAL UNIVERSITY

- **Why?**
  - In theory, very efficient way to test.
  - Design, implementation and test case creation roughly one manual task
- **Disadvantages**
  - Immature and very much application specific
  - Requires a lot of backend R&D in MDE to go mainstream

Recap

MEMORIAL UNIVERSITY

- **In this presentation I have provided an overview of a variety of verification techniques. These include:**
  - *Acceptance Testing:* this technique verifies functional areas of a program via defined test cases.
  - *Fuzz Testing:* random (fuzz) data is input to a system in an attempt to make it crash or hang.
  - *Usability Testing:* A process in which information about product effectiveness is gathered by observing user interaction.
  - *Regression Testing:* tests are run against an existing code base to ensure new development has not broken it.

Recap (cont...)

MEMORIAL UNIVERSITY

- *Exploratory Testing:* test procedures are not defined, testers develop test cases through interaction with the system.
- *Ad hoc Testing:* A form of exploratory testing that is done without any preparation or documentation.
- *Session-based Testing:* based on the exploratory testing methodology, yet includes enough structure to provide accountability.
- *Mutation Testing:* mutant source code is generated which an existing test case is run against.
- *Model-Based Testing:* test cases are derived from the model of a software system.

Selected References

- [1] Wikipedia Software Testing Portal, http://en.wikipedia.org/wiki/Portal:Software_Testing
- [2] Wells, D. (1999). *Acceptance Tests*. Retrieved from Extreme Programming: http://www.extremeprogramming.org/rules/functionaltests.html
- [3] Offutt, J. (1995). Practical Mutation Testing. *Twelfth International Conference on Testing Computer Software*, (pp. 99-109). Washington, DC.
- [4] Bach, J. (2003). *Exploratory Testing Explained.* Retrieved from Satisfice, Inc.: http://www.satisfice.com/articles/et-article.pdf
- [5] Johnson, B., & Agruss, C. (2000). Ad Hoc Software Testing. Retrieved from Testing Craft: http://www.testingcraft.com/ad_hoc_testing.pdf
- [6] Memon, A. M. (2007). An event-flow model of GUI-based applications for testing. *Software Testing, Verification and Reliability* , 137-157.

Questions ?