Security testing is essential for software that processes confidential data to prevent system intrusion by hackers.

Internationalization and localization

The general ability of software to be internationalized and localized can be automatically tested without actual tra
still works, even after it has been translated into a new language or adapted for a new culture (such as different curr

Actual translation to human languages must be tested, too. Possible localization failures include:

- Software is often localized by translating a list of strings out of context, and the translator may choose the wron
- If several people translate strings, technical terminology may become inconsistent.
- Literal word-for-word translations may sound inappropriate, artificial or too technical in the target language.
- Untranslated messages in the original language may be left hard coded in the source code.
- Some messages may be created automatically in run time and the resulting string may be ungrammatical, functi
- Software may use a keyboard shortcut which has no function on the source language's keyboard layout, but is us
- Software may lack support for the character encoding of the target language.
- Fonts and font sizes which are appropriate in the source language, may be inappropriate in the target language; small.
- A string in the target language may be longer than the software can handle. This may make the string partly in
- Software may lack proper support for reading or writing bi-directional text.
- Software may display images with text that wasn't localized.
- Localized operating systems may have differently-named system configuration files and environment variables an

To avoid these and other localization problems, a tester who knows the target language must run the program w
readable, translated correctly in context and don't cause failures.

Destructive testing

Destructive testing attempts to cause the software or a sub-system to fail, in order to test its robustness.

The testing process

Traditional CMMI or waterfall development model

A common practice of software testing is that testing is performed by an independent group of testers after the fu
practice often results in the testing phase being used as a project buffer to compensate for project delays, thereby co

Another practice is to start software testing at the same moment the project starts and it is a continuous process un

Agile or Extreme development model

In counterpoint, some emerging software disciplines such as extreme programming and the agile software developme
this process, unit tests are written first, by the software engineers (often with pair programming in the extreme pro
expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites ar
discovered, and they are integrated with any regression tests that are developed. Unit tests are maintained along wi
build process (with inherently interactive tests being relegated to a partially manual build acceptance process). Th
where software updates can be published to the public frequently. [33] [34]

A sample testing cycle

Although variations exist between organizations, there is a typical cycle for testing.[35] The sample below is commo

- **Requirements analysis**: Testing should begin in the requirements phase of the software development life cy
  determining what aspects of a design are testable and with what parameters those tests work.
- **Test planning**: Test strategy, test plan, testbed creation. Since many activities will be carried out during test
- **Test development**: Test procedures, test scenarios, test cases, test datasets, test scripts to use in testing soft
- **Test execution**: Testers execute the software based on the plans and test documents then report any errors fo
- **Test reporting**: Once testing is completed, testers generate metrics and make final reports on their test effort
- **Test result analysis**: Or Defect Analysis, is done by the development team usually along with the client, in
  software working properly) or deferred to be dealt with later.
- **Defect Retesting**: Once a defect has been dealt with by the development team, it is retested by the testing t
- **Regression testing**: It is common to have a small test program built of a subset of tests, for each integration
  delivery has not ruined anything, and that the software product as a whole is still working correctly.
- **Test Closure**: Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons le

used as a reference for future projects.

Automated testing

Many programming groups are relying more and more on automated testing, especially groups that use test-dr
continuous integration software will run tests automatically every time code is checked into a version control system

While automation cannot reproduce everything that a human can do (and all the ways they think of doing it), it c
developed test suite of testing scripts in order to be truly useful.

Testing tools

Program testing and fault detection can be aided significantly by testing tools and debuggers. Testing/debug tools i

- Program monitors, permitting full or partial monitoring of program code including:
  - Instruction set simulator, permitting complete instruction level monitoring and trace facilities
  - Program animation, permitting step-by-step execution and conditional breakpoint at source level or in mach
  - Code coverage reports
- Formatted dump or symbolic debugging, tools allowing inspection of program variables on error or at chosen poi
- Automated functional GUI testing tools are used to repeat system-level tests through the GUI
- Benchmarks, allowing run-time performance comparisons to be made
- Performance analysis (or profiling tools) that can help to highlight hot spots and resource usage

Some of these features may be incorporated into an Integrated Development Environment (IDE).

- A regression testing technique is to have a standard set of tests, which cover existing functionality that result in
  data, where there should not be differences, using a tool like diffkit. Differences detected indicate unexpected fun

Measurement in software testing

Usually, quality is constrained to such topics as correctness, completeness, security,[*citation needed*] but can also in
ISO/IEC 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability.

There are a number of frequently-used software measures, often called *metrics*, which are used to assist in determini

Testing artifacts

Software testing process can produce several artifacts.

Test plan

A test specification is called a test plan. The developers are well aware what test plans will be executed and this
idea is to make them more cautious when developing their code or making additional changes. Some companies

Traceability matrix

A traceability matrix is a table that correlates requirements or design documents to test documents. It is used t
the test results are correct.

Test case

A test case normally consists of a unique identifier, requirement references from a design specification, precondit
output, expected result, and actual result. Clinically defined a test case is an input and an expected result.[36] T
whereas other test cases described in more detail the input scenario and what results might be expected. It can
separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one
ID, test step, or order of execution number, related requirement(s), depth, test category, author, and check boxe
test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place
document, spreadsheet, database, or other common repository. In a database system, you may also be able to se
configuration was used to generate those results. These past results would usually be stored in a separate table.

Test script

The test script is the combination of a test case, test procedure, and test data. Initially the term was derived fro
Today, test scripts can be manual, automated, or a combination of both.

Test suite

The most common term for a collection of test cases is a test suite. The test suite often also contains more detai
contains a section where the tester identifies the system configuration used during testing. A group of test cases
following tests.

Test data

In most cases, multiple sets of values or data are used to test the same functionality of a particular feature. All t
in separate files and stored as test data. It is also useful to provide this data to the client and with the product o

Test harness

The software, tools, samples of data input and output, and configurations are all referred to collectively as a test

Certifications

Several certification programs exist to support the professional aspirations of software testers and quality assura
applicant to demonstrate the ability to test software. No certification is based on a widely accepted body of knowl
certification.[37] Certification itself cannot measure an individual's productivity, their skill, or practical knowle
tester.[38]

Software testing certification types

- *Exam-based*: Formalized exams, which need to be passed; can also be learned by self-study [e.g., for ISTQB o
- *Education-based*: Instructor-led sessions, where each course has to be passed [e.g., International Institute for

Testing certifications

- Certified Associate in Software Testing (CAST) offered by the Quality Assurance Institute (QAI)[40]
- CATe offered by the *International Institute for Software Testing*[41]
- Certified Manager in Software Testing (CMST) offered by the Quality Assurance Institute (QAI)[40]
- Certified Software Tester (CSTE) offered by the Quality Assurance Institute (QAI)[40]
- Certified Software Test Professional (CSTP) offered by the *International Institute for Software Testing*[41]
- CSTP (TM) (Australian Version) offered by *K. J. Ross & Associates*[42]
- ISEB offered by the Information Systems Examinations Board
- ISTQB Certified Tester, Foundation Level (CTFL) offered by the International Software Testing Qualificati
- ISTQB Certified Tester, Advanced Level (CTAL) offered by the International Software Testing Qualificatio
- TMPF TMap Next Foundation offered by the *Examination Institute for Information Science*[45]
- TMPA TMap Next Advanced offered by the *Examination Institute for Information Science*[45]

Quality assurance certifications

- CMSQ offered by the *Quality Assurance Institute* (QAI).[40]
- CSQA offered by the *Quality Assurance Institute* (QAI)[40]
- CSQE offered by the American Society for Quality (ASQ)[46]
- CQIA offered by the American Society for Quality (ASQ)[46]

Controversy

Some of the major software testing controversies include:

What constitutes responsible software testing?

Members of the "context-driven" school of testing[47] believe that there are no "best practices" of testing, but rat
testing practices to suit each unique situation.[48]

Agile vs. traditional

Should testers learn to work under conditions of uncertainty and constant change or should they aim at process
popularity since 2006 mainly in commercial circles,[49][50] whereas government and military[51] software provid
the Waterfall model).[*citation needed*]

Exploratory test vs. scripted[52]

Should tests be designed at the same time as they are executed or should they be designed beforehand?

Manual testing vs. automated

Some writers believe that test automation is so expensive relative to its value that it should be used sparingly.[5
should write unit-tests of the XUnit type before coding the functionality. The tests then can be considered as a

Software design vs. software implementation

Should testing be carried out only at the end or throughout the whole process?

Who watches the watchmen?

The idea is that any form of observation is also an interaction—the act of testing can also affect that which is b

References

1. Exploratory Testing (http://www.kaner.com/pdfs/ETatQAI.pdf), Cem Kaner, Florida Institute of Technology, *Conference*, Orlando, FL, November 2006
2. Leitner, A., Ciupa, I., Oriol, M., Meyer, B., Fiva, A., "Contract Driven Development = Test Driven Developmer ns/cdd_leitner_esec_fse_2007.pdf), Proceedings of ESEC/FSE'07: European Software Engineering Conference a Engineering 2007, (Dubrovnik, Croatia), September 2007
3. Software errors cost U.S. economy $59.5 billion annually (http://www.abeacha.com/NIST_press_release_bugs_
4. Myers, Glenford J. (1979). *The Art of Software Testing*. John Wiley and Sons. ISBN 0-471-04328-1.
5. Company, People's Computer (1987). "Dr. Dobb's journal of software tools for the professional programmer". *Dr* (M&T Pub) **12** (1-6): 116. http://books.google.com/?id=7RoIAAAAIAAJ.
6. Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". *CACM* **31** (6). ISSN 0001-0782.
7. *until 1956 it was the debugging oriented period, when testing was often associated to debugging: there was no cle* (1988). "The Growth of Software Testing". *CACM* **31** (6). ISSN 0001-0782.
8. *From 1957–1978 there was the demonstration oriented period where debugging and testing was distinguished nov* Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". *CACM* **31** (6). ISSN 0001-0782.
9. *The time between 1979–1982 is announced as the destruction oriented period, where the goal was to find errors.* *CACM* **31** (6). ISSN 0001-0782.
10. *1983–1987 is classified as the evaluation oriented period: intention here is that during the software lifecycle a pr* Hetzel (1988). "The Growth of Software Testing". *CACM* **31** (6). ISSN 0001-0782.
11. *From 1988 on it was seen as prevention oriented period where tests were to demonstrate that software satisfies* Hetzel (1988). "The Growth of Software Testing". *CACM* **31** (6). ISSN 0001-0782.
12. Kaner, Cem; Falk, Jack and Nguyen, Hung Quoc (1999). *Testing Computer Software, 2nd Ed.*. New York, et al: **Invalid <ref> tag; name "Kaner1" defined multiple times with different content Invalid < different content**
13. Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Managemen* http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470042125.html.
14. Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Managemen* http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470042125.html.
15. Section 1.1.2, Certified Tester Foundation Level Syllabus (http://www.istqb.org/downloads/syllabi/SyllabusFou
16. Kaner, Cem; James Bach, Bret Pettichord (2001). *Lessons Learned in Software Testing: A Context-Driven Appr*
17. McConnell, Steve (2004). *Code Complete* (2nd ed.). Microsoft Press. pp. 960. ISBN 0-7356-1967-0.
18. Principle 2, Section 1.3, Certified Tester Foundation Level Syllabus (http://www.bcs.org/upload/pdf/istqbsyll.p
19. Tran, Eushiuan (1999). "Verification/Validation/Certification". in Koopman, P.. *Topics in Dependable Embedde* http://www.ece.cmu.edu/~koopman/des_s99/verification/index.html. Retrieved 2008-01-13.
20. see D. Gelperin and W.C. Hetzel
21. Introduction (http://www.bullseye.com/coverage.html#intro), Code Coverage Analysis, Steve Cornett
22. Laycock, G. T. (1993) (PostScript). *The Theory and Practice of Specification Based Software Testing*. Dept of C http://www.mcs.le.ac.uk/people/gtl1/thesis.ps.gz. Retrieved 2008-02-13.
23. Bach, James (June 1999). "Risk and Requirements-Based Testing" (PDF). *Computer* **32** (6): 113–114. http://wv 2008-08-19.
24. Savenkov, Roman (2008). *How to Become a Software Tester*. Roman Savenkov Consulting. p. 159. ISBN 978-0-(
25. Binder, Robert V. (1999). *Testing Object-Oriented Systems: Objects, Patterns, and Tools*. Addison-Wesley Profe
26. Beizer, Boris (1990). *Software Testing Techniques* (Second ed.). New York: Van Nostrand Reinhold. pp. 21,430.
27. IEEE (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. Nev

28. van Veenendaal, Erik. "Standard glossary of terms used in Software Testing". http://www.astqb.org/educational
29. Globalization Step-by-Step: The World-Ready Approach to Testing. Microsoft Developer Network (http://msdn
30. e)Testing Phase in Software Testing:- (http://www.etestinghub.com/testing_lifecycles.php#2)
31. Myers, Glenford J. (1979). *The Art of Software Testing*. John Wiley and Sons. pp. 145–146. ISBN 0-471-04328-1
32. Dustin, Elfriede (2002). *Effective Software Testing*. Addison Wesley. p. 3. ISBN 0-20179-429-2.
33. Marchenko, Artem (November 16, 2007). "XP Practice: Continuous Integration". http://agilesoftwaredevelopmer
34. Gurses, Levent (February 19, 2007). "Agile 101: What is Continuous Integration?". http://www.jacoozi.com/blo
35. Pan, Jiantao (Spring 1999). "Software Testing (18-849b Dependable Embedded Systems)". *Topics in Dependable* Carnegie Mellon University. http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/.
36. IEEE (1998). *IEEE standard for software test documentation*. New York: IEEE. ISBN 0-7381-1443-X.
37. Kaner, Cem (2001). "NSF grant proposal to "lay a foundation for significant improvements in the quality of acac http://www.testingeducation.org/general/nsf_grant.pdf.
38. Kaner, Cem (2003). "Measuring the Effectiveness of Software Testers" (pdf). http://www.testingeducation.org/a
39. Black, Rex (December 2008). *Advanced Software Testing- Vol. 2: Guide to the ISTQB Advanced Certification a* ISBN 1933952369.
40. Quality Assurance Institute (http://www.qaiglobalinstitute.com/)
41. International Institute for Software Testing (http://www.testinginstitute.com/)
42. K. J. Ross & Associates (http://www.kjross.com.au/cstp/)
43. "ISTQB". http://www.istqb.org/.
44. "ISTQB in the U.S.". http://www.astqb.org/.
45. EXIN: Examination Institute for Information Science (http://www.exin-exams.com)
46. American Society for Quality (http://www.asq.org/)
47. context-driven-testing.com (http://www.context-driven-testing.com)
48. Article on taking agile traits without the agile method. (http://www.technicat.com/writing/process.html)
49. “We’re all part of the story” (http://stpcollaborative.com/knowledge/272-were-all-part-of-the-story) by David S
50. IEEE article about differences in adoption of agile trends between experienced managers vs. young students of tl n.jsp?url=/iel5/10705/33795/01609838.pdf?temp=x). See also Agile adoption study from 2007 (http://www.aml
51. Willison, John S. (April 2004). "Agile Software Development for an Agile Force". *CrossTalk* (STSC) (April 2004) http://web.archive.org/web/20051029135922/http://www.stsc.hill.af.mil/crosstalk/2004/04/0404willison.html.
52. IEEE article on Exploratory vs. Non Exploratory testing (http://ieeexplore.ieee.org/iel5/10351/32923/01541817.
53. An example is Mark Fewster, Dorothy Graham: *Software Test Automation*. Addison Wesley, 1999, ISBN 0-201-3
54. Microsoft Development Network Discussion on exactly this topic (http://channel9.msdn.com/forums/Coffeehous

External links

- Software testing tools and products (http://www.dmoz.org/Computers/Programming/Software_Testing/Produc
- "Software that makes Software better" Economist.com (http://www.economist.com/science/tq/displaystory.cfm?
- Automated software testing metrics including manual testing metrics (http://www.innovativedefense.com/img/U

Unit Tests

In computer programming, **unit testing** is a method by which individual units of source code are tested to dete application. In procedural programming a unit may be an individual function or procedure. Unit tests are created by

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects,[1] fakes and test are typically written and run by software developers to ensure that code meets its design and behaves as intended. to being formalized as part of build automation.

Benefits

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.[2] A satisfy. As a result, it affords several benefits. Unit tests find problems early in the development cycle.

Facilitates change

Unit testing allows the programmer to refactor code at a later date, and make sure the module still works correctly functions and methods so that whenever a change causes a fault, it can be quickly identified and fixed.

Readily-available unit tests make it easy for the programmer to check whether a piece of code is still working proper

In continuous unit testing environments, through the inherent practice of sustained maintenance, unit tests will co in the face of any change. Depending upon established development practices and unit test coverage, up-to-the-seco separately.

Simplifies integration

Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach parts, integration testing becomes much easier.

An elaborate hierarchy of unit tests does not equal integration testing. Integration with peripheral units should Integration testing typically still relies heavily on humans testing manually; high-level or global-scope testing can and cheaper.[*citation needed*]

Documentation

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is a basic understanding of the unit's API.

Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicat that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, alth code to document the product in development.

By contrast, ordinary narrative documentation is more susceptible to drifting from the implementation of the prog relaxed practices in keeping documents up-to-date).

Design

When software is developed using a test-driven approach, the unit test may take the place of formal design. Each u observable behaviour. The following Java example will help illustrate this point.

Here is a test class that specifies a number of elements of the implementation. First, that there must be an int constructor called AdderImpl. It goes on to assert that the Adder interface should have a method called add, with the behaviour of this method for a small range of values.

```
public class TestAdder {
   public void testSum() {
      Adder adder = new AdderImpl();
      assert(adder.add(1, 1) == 2);
      assert(adder.add(1, 2) == 3);
      assert(adder.add(2, 2) == 4);
      assert(adder.add(0, 0) == 0);
      assert(adder.add(-1, -2) == -3);
      assert(adder.add(-1, 1) == 0);
      assert(adder.add(1234, 988) == 2222);
   }
}
```

In this case the unit test, having been written first, acts as a design document specifying the form and behaviour of the programmer. Following the "do the simplest thing that could possibly work" practice, the easiest solution that w

```
interface Adder {
   int add(int a, int b);
}
class AdderImpl implements Adder {
   int add(int a, int b) {
      return a + b;
   }
}
```

Unlike other diagram-based design methods, using a unit-test as a design has one significant advantage. The implementation adheres to the design. With the unit-test design method, the tests will never pass if the developer d

It is true that unit testing lacks some of the accessibility of a diagram, but UML diagrams are now easily generated to IDEs). Free tools, like those based on the xUnit framework, outsource to another system the graphical rendering