Lessons Learned In Software Testing A Context Driven Approach

Smoke testing (software)

2016 Cem Kaner, James Bach, Bret Pettichord, Lessons learned in software testing: a context-driven approach. Wiley, 2001 McConnell, Steve. "Rapid Development"

In computer programming and software testing, smoke testing (also confidence testing, sanity testing, build verification test (BVT) and build acceptance test) is preliminary testing or sanity testing to reveal simple failures severe enough to, for example, reject a prospective software release. Smoke tests are a subset of test cases that cover the most important functionality of a component or system, used to aid assessment of whether main functions of the software appear to work correctly. When used to determine if a computer program should be subjected to further, more fine-grained testing, a smoke test may be called a pretest or an intake test. Alternatively, it is a set of tests run on each new build of a product to verify that the build is testable before the build is released into the...

Software testing

Cem; Bach, James; Pettichord, Bret (2001). Lessons Learned in Software Testing: A Context-Driven Approach. Wiley. pp. 31–43. ISBN 978-0-471-08112-8. Kolawa

Software testing is the act of checking whether software satisfies expectations.

Software testing can provide objective, independent information about the quality of software and the risk of its failure to a user or sponsor.

Software testing can determine the correctness of software for specific scenarios but cannot determine correctness for all scenarios. It cannot find all bugs.

Based on the criteria for measuring correctness from an oracle, software testing employs principles and mechanisms that might recognize a problem. Examples of oracles include specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, and applicable laws.

Software testing is often dynamic in nature...

Installation testing

ISBN 9780471469124. Kaner, C; Bach, J; Pettichord, B (2001). Lessons Learned in Software Testing: A Context-Driven Approach. Wiley. p. 41. ISBN 9780471081128.

Most software systems have installation procedures that are needed before they can be used for their main purpose. Testing these procedures to achieve an installed software system that may be used is known as installation testing. These procedures may involve full or partial upgrades, and install/uninstall processes.

Installation testing may look for errors that occur in the installation process that affect the user's perception and capability to use the installed software. There are many events that may affect the software installation and installation testing may test for proper installation whilst checking for a number of associated activities and events. Some examples include the following:

A user must select a variety of options.

Dependent files and libraries must be allocated, loaded...

Cem Kaner

California Technical Publications Competition.) Lessons Learned in Software Testing: A Context-driven Approach. New York: Wiley. 15 December 2001. ISBN 0-471-08112-4

Cem Kaner is a professor of software engineering at Florida Institute of Technology, and the Director of Florida Tech's Center for Software Testing Education & Research (CSTER) since 2004. He is perhaps best known outside academia as an advocate of software usability and software testing.

Prior to his professorship, Kaner worked in the software industry beginning in 1983 in Silicon Valley "as a tester, programmer, tech writer, software development manager, product development director, and independent software development consultant." In 1988, he and his co-authors Jack Falk and Hung Quoc Nguyen published what became, at the time, "the best selling book on software testing," Testing Computer Software. He has also worked as a user interface designer.

In 2004 he cofounded the non-profit Association...

Exploratory testing

Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution. Cem Kaner

Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution. Cem Kaner, who coined the term in 1984, defines exploratory testing as "a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project."

While the software is being tested, the tester learns things that together with experience and creativity generates new good tests to run. Exploratory testing is often thought of as a black box testing technique. Instead, those who have studied it consider...

Unit testing

Unit testing, a.k.a. component or module testing, is a form of software testing by which isolated source code is tested to validate expected behavior.

Unit testing, a.k.a. component or module testing, is a form of software testing by which isolated source code is tested to validate expected behavior.

Unit testing describes tests that are run at the unit-level to contrast testing at the integration or system level.

Agile software development

Agile software development is an umbrella term for approaches to developing software that reflect the values and principles agreed upon by The Agile Alliance

Agile software development is an umbrella term for approaches to developing software that reflect the values and principles agreed upon by The Agile Alliance, a group of 17 software practitioners, in 2001. As documented in their Manifesto for Agile Software Development the practitioners value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

The practitioners cite inspiration from new practices at the time including extreme programming, scrum, dynamic systems development method, adaptive software development, and being

sympathetic to the need for an alternative to documentation-driven, heavyweight software development processes.

Many software development...

Arcadia (engineering)

Integrated Approach) is a system and software architecture engineering method based on architecture-centric and model-driven engineering activities. In the development

ARCADIA (Architecture Analysis & Design Integrated Approach) is a system and software architecture engineering method based on architecture-centric and model-driven engineering activities.

Mockup

Orion Mock-Up for Testing". News.softpedia.com. Mock-ups. Interaction-design.org. 16 February 2010. Cline, Todd, "Lessons Learned From Product Manager

In manufacturing and design, a mockup, or mock-up, is a scale or full-size model of a design or device, used for teaching, demonstration, design evaluation, promotion, and other purposes. A mockup may be a prototype if it provides at least part of the functionality of a system and enables testing of a design.

Mock-ups are used by designers mainly to acquire feedback from users. Mock-ups address the idea captured in a popular engineering one-liner: "You can fix it now on the drafting board with an eraser or you can fix it later on the construction site with a sledge hammer".

Mockups are used as design tools virtually everywhere a new product is designed.

Mockups are used in the automotive device industry as part of the product development process, where dimensions, overall impression, and...

Minimum viable product

market-tested expansion models such as the real options model. A simple method of testing the financial viability of an idea would be discovery-driven planning

A minimum viable product (MVP) is a version of a product with just enough features to be usable by early customers who can then provide feedback for future product development.

A focus on releasing an MVP means that developers potentially avoid lengthy and (possibly) unnecessary work. Instead, they iterate on working versions and respond to feedback, challenging and validating assumptions about a product's requirements. The term was coined and defined in 2001 by Frank Robinson and then popularized by Steve Blank and Eric Ries. It may also involve carrying out market analysis beforehand. The MVP is analogous to experimentation in the scientific method applied in the context of validating business hypotheses. It is utilized so that prospective entrepreneurs would know whether a given business...