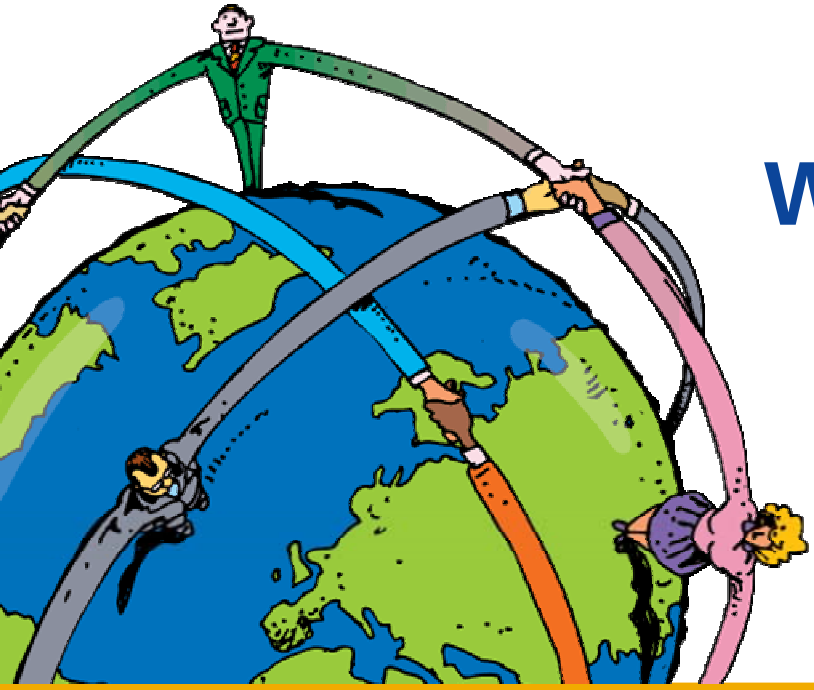


System Testing
Why, What, and How

SQGNE

October 10, 2007

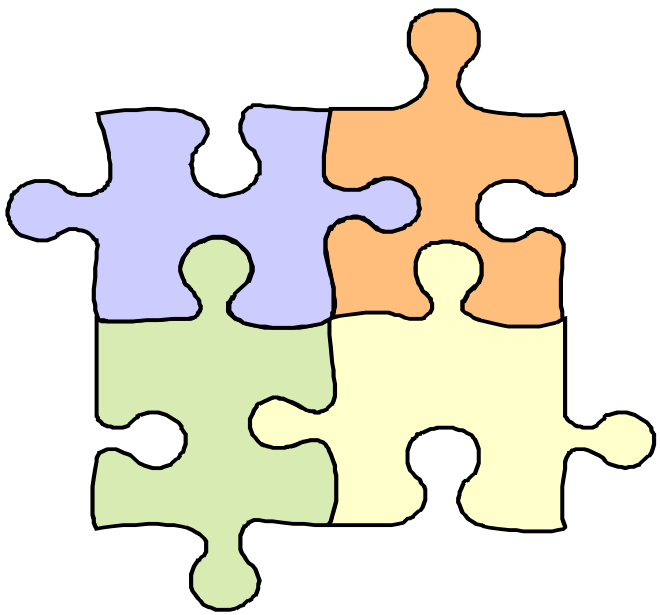
Carol Perletz

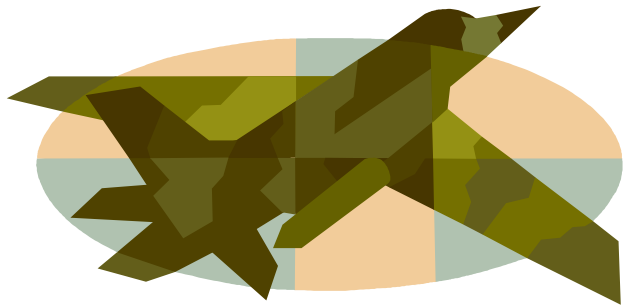
Agenda

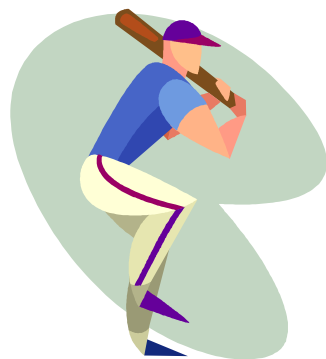
Systems

System Definition

A group of interacting, interrelated, or interdependent elements forming a complex whole.







Systems Engineering

Definition of Systems Engineering

An interdisciplinary field of engineering that focuses on the development and organization of complex systems.

Software Systems

Definition of Software Systems

A complex system consisting of hardware and software elements which work together to deliver functionality and features providing end-users with the ability to perform the series of actions that make their job.

Why System Testing ?

History of Software System Testing

- System testing has been conducted in companies producing computer operating systems on hardware since the 1960s
- System testing has been mentioned in software engineering text books and lectures since the 1980s
- In the 1980s, software testing became a discipline performed by specialized test engineers
- Standards developed by IEEE, DoD, and ISO-9000 incorporated elements of software testing in the ’90s

References to System Testing

- **40,600,000 references in Google**
  - System testing services
  - System testing tools
  - Specialized system testing
    - **Highly available clustered systems**
    - **Embedded systems**
    - **Agile testing**
    - **UNIX system testing**
    - **Testing in specialized vertical markets**
    - **Web applications**
    - **IT applications**

Why do System Testing?

- Unit testing verifies and validates each unit in the system
- Functional testing verifies and validates each feature and functionality
- Integration testing verifies that the units have been successfully integrated

End-users Want to Know …

- Does the software meet the customer's overall needs in their environment?
- Can the end-user perform their daily work?
- Is the software compatible with different versions of hardware and other software found in the customer environment?
- Will the software interoperate with other hardware and software in the customer environment that it interfaces with?
- Will the software scale to the needs of the customer?

End-users Want to Know …(continued)

- What happens to the software when load is put on the system?
- What happens when the software is stressed?
- Is the software reliable; will it run without error as long as the customer expects?
- Does the user interface meet the customer needs; is it consistent, intuitive, easy-to-use?
- Are errors handled to the end-user's satisfaction?
- Will the Online Help assist the customer to do what they need to do?
- Does the software recover from destructive acts?

End-users Want to Know …(continued)

- Does the software provide the necessary throughput; does it have the needed capacity?
- Can the software be both installed and upgraded in all customer environments?
- Does the software meet industry security standards as well as all of the end-user's security needs?
- Does this version of the software continue to provide all of the features and functionality of the previous version?

What is System Testing?

System Testing Defined

- Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements
- It is generally black box testing and often referred to as non-functional testing
- It follows integration testing
- The focus is to find errors
- Test the behavior of the system including the believed expectations of the customer
- Test up to and beyond the bounds defined in both the hardware and software requirements specifications

Types of System Testing

- **User interface testing** - testing of any interface provided to the customer. These interfaces can be graphical (GUI) or command line interfaces (CLI)

- **Usability testing (use cases, user scenarios)** - testing the individual functionality of the system exposed to end-users. System scenarios or use cases should cover the normal and regular functions performed by the administrator, end-user, and customer care representatives

- **Performance, capacity, throughput testing** – determining the actual performance of the system against the performance objectives under peak and normal conditions (transaction rates and response times)

Types of System Testing (continued)

- **Compatibility and interoperability testing** – the compatibility of the software with different versions of other system elements; testing the interoperation of all of the interfaces in the system
- **Error handling testing** – the ability of the system to flag errors to the user through a user interface; logging and tracing capabilities
- **Load and stress testing** – identifying the peak load conditions at which the system fails
- **Volume testing** – identifying the level of continuous heavy load at which the system will fail

Types of System Testing (continued)

- **Security testing** – look for breaches to the security provisions of the software and system, denial of service (hacker mentality)
- **Scalability testing** – the ability to increase the capacity of the system, by adding more processors, computers, and/or memory; dimensioning the system
- **Sanity, smoke, acceptance testing** – verifying the viability of an entire product build; verifying user acceptance of the product
- **Exploratory and ad-hoc testing** – intentional misuse of the system, unplanned testing by unsophisticted users, exploration to ”see what the software can do”
- **Reliability, longevity, availability testing** – verifying over long periods of time that the system remains available and does not degrade in performance

Types of System Testing (continued)

- **Regression testing** – verifying that code changes, bug fixes, and the introduction of new features to the previously integrated code do not break functionality that ”used to work”
- **Recovery or destructive testing** – determining the behavior of the system after the introduction of an error or abnormal events such as disconnection from the network
- **Installation testing** – testing the installation of the product in the system in all realistic user environment scenarios
- **Upgrade testing** – ability to upgrade the product from all supported previous versions to the current version within the full system environment

How to Perform System Testing

Making System Testing Successful

- **The program must have testable, measurable, end-user requirements**
  - Use cases
  - Deployment scenarios
  - System scenarios
  - End-user profiles
- **System Test Plan**
  - Establish test objectives
  - Define system test strategy and methodologies to be used
  - Enumerate each type of system testing to be performed and whether a specific test plan is required

System Test Planning (continued)

- Indicate which individual test plans are required for the different types of system testing
  - **System compatibility and interoperability**
  - **Performance and scalability**
  - **Reliability, longevity, and availability**
  - **Security, denial of service**
  - **Regression**
  - **Installation and upgrading**
  - **Acceptance, sanity, smoke testing**
  - **User interface and usability**
  - **Recovery and destructive testing**

System Test Planning (continued)

- Establish schedules and responsibilities for each test activity
- Determine the availability of tools, software, hardware, devices
- Define what test automation will be used
- Create all of the test environments needed to perform the system testing (include all supported platforms)
- Establish the procedures and standards to be used for planning, executing, and reporting results of the tests
- Set the criteria for test completion as well as for the success of each test

Making System Testing Successful (continued)

- **Design the "right" test cases**
  - Full test coverage is impossible – propose those test cases in combinations of system elements that are most likely to yield the highest number of errors
  - Create the most test cases for the most complex areas of the system
  - Include the following in each test case:
    - **Required setup**
    - **Descriptive test name**
    - **Manual or automated test**
    - **Positive or negative test**
    - **Test procedure (steps)**
    - **Expected results**

Making System Testing Successful (continued)

- **Test execution**
  - Treat testing with dogged determination – pursue and record all strange and unexpected events (note and investigate all anomalous behavior)
  - Take notes during testing
  - Store all test plans and test cases in a repository
  - Store all test results and test reports in a repository
  - Be systematic and meticulous about setting up the proper environment before a test and cleaning up the environment after the test

Making System Testing Successful (continued)

- **Test reporting**
  - After the conclusion of each type of system testing, a test report should be produced
  - Information to be included in the test report is driven by the specific test plan
  - General components of the test report:
    - **Names of the program, test plan, test cases**
    - **Identify the testers**
    - **Identify the build(s) and release versions for all elements**
    - **Provide a diagram of the test environment(s)**
    - **Include the test results, measurements, number of runs, bugs found**

Making System Testing Successful (continued)

- **Testing analysis**
  - Look for specific functionality or features that contain a majority of reported problems
  - Look for the root cause of bugs (check logs, use sniffer tools)
  - Go back to design documents in problematic areas
  - Look for functionality or features that have been modified by several different developers
  - Look for classes of errors such as dependency or interface problems
  - Gather data on the test case yield – # of bugs found, # of test cases, length of test case runs

Making System Testing Successful (continued)

- **Metrics program**
  - During early stages of the program planning, design a metrics program
  - Regularly provide metrics reports consisting of metrics in the following areas:
    - **Bugs**
    - **Test execution**
    - **Performance measurements**
    - **Work efficiency, effort (testers, developers)**
    - **Root cause analysis of bugs (in what stage of the development process were bugs introduced?)**
    - **Scheduling performance (critical path tasks and resources); reaching milestones**
    - **Tracking the program budget**

Summary

- **Software systems** – hardware and software elements working together to deliver functionality and features providing end-users the ability to perform their job
- **System testing** – conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements
- **Why do it?** – are customer needs met in their environment; is the product installable and upgradable; is usability sufficient? Is the software reliable, available; does it scale, interoperate with other elements in the environment, recover from destructive events? What load causes the system to fail? Is performance adequate? Is error handling sufficient? Are security standards met?
- **Types of system testing** – usability, performance, interoperability, load, volume, security, scalability, reliability, recovery, installation, upgrading, regression, acceptance, ad-hoc, error handling

Summary (continued)

- **Successful system testing requires:**
  - Testable, measurable, end-user requirements
  - System Test Plan(s)
  - The “right” test cases (most often scenarios)
  - Test execution with a “dogged determination”
  - Complete test reporting
  - Analysis of test results
  - A metrics program

References

- Beizer, Boris. Software System Testing and Quality Assurance, Van Nostrand Reinhold Electrical / Computer Science and Engineering Series, 1984.
- Black, Rex. Managing the Testing Process, Second Edition, Hoboken: John Wiley & Sons, Inc., 2002.
- Chernak, Yuri. “Understanding the Logic of System Testing”, March 30, 2006.
- Hetzel, Bill. The Complete Guide to Software Testing, Second Edition, Wellesley: John Wiley & Sons, Inc, 1988.
- Humphrey, Watts S. Managing the Software Process, Boston: Addison-Wesley Co.,1990.

References (continued)

- IEEE 829-1998 IEEE Standard for Software Test Documentation.
- Kit, Edward. Software Testing in the Real World: Improving the Process, Boston: Addison-Wesley Publishing Co., 1995.
- Myers, Glenford J., Sandler, Corey, Badgett Tom, and Thomas Todd M. The Art of Software Testing, Second Edition, Hoboken: John Wiley & Sons, Inc., 2004
- Perry, William E. Effective Methods for Software Testing, Second Edition, Hoboken: John Wiley & Sons, Inc., 2000.
- Pressman, Roger. Software Engineering: A Practioner's Approach, Fifth Edition, Boston: Addison-Wesley Publishing Co., 2000.

Q & A

Carol Perletz
cperletz@juniper.net