

Testing Information Systems

(Version 6.0)

Duration: 2 days

Overview

This course presents topics, tools and techniques on how to plan, structure and execute testing efficiently and effectively in the real world. It does not try to sell you on testing. You already recognize the critical importance of testing. You just want to know how to do it better.

Elements of Effectiveness

Effective testing is planned and budgeted at the beginning of your project. Test plans are based on a test strategy that you establish to manage business risks. Each test plan encompasses a set of test cases that are documented in test scripts. Test strategies, test plans, test cases and test scripts are components of your testing methodology. You need a consistent approach for developing each component individually and for maintaining it as an integral part of your test environment.

Techniques of Efficiency

Efficient testing is engineered for optimal performance throughout the system life cycle. A minimum number of test cases ensures that the system works in accordance with expectations. You need to identify, define and prioritize test cases from data flow diagrams, data models, event/response diagrams, program specifications, user documentation or plain business requirements. You need to engineer your test data to compare test cases from any source, reduce redundancy and optimize your testing performance.

Approach

All of the tools and techniques of this 2-day seminar are based on two simple philosophies:

an information system should be tested at the appropriate level of detail to reduce the risk of exposure caused by a potential failure.

testing doesn't follow development, it is an integral part of the creative process.

The seminar takes a pragmatic “tell me, show me, let me” approach to testing. Because hands-on experience is the best teacher, practical exercises are the foundation for the seminar.

Audience

The target audience includes project leaders, analysts, designers, programmers, testing professionals, test process managers and end-users who are interested in reducing the effort required to deliver usable information technology with a minimum of time investment.

Customization

The modular structure of this class allows you to mix and match the subjects that matter most to your organization. The selected modules can be presented as a conventional seminar to a single group or a different mix of participants can attend each module. This approach lets participants optimize their time investment by targeting specific techniques and maximizes the company's return on its training dollars.

Developed and presented by:

Hathaway & Associates, Inc.
16057 Tampa Palms Blvd. W., Suite 197
Tampa, FL 33647

"Effective Business Use of Information and Systems"

Telephone: (813) 973-3046
Fax: (813) 864-0131
Email: training@thehathaway.com
Website: www.thehathaway.com
www.businessanalysisbooks.com

Outline

GI General Introduction

Duration: .5 - 1 hour

The major challenge of testing is to find a balance between the necessity for quality and the cost of delivering the information technology that the business community needs. To meet this challenge, you need to understand the risk of system failure and the mechanisms that can reduce that risk.

1. What is the true goal of information system testing?
2. How do you define the major testing activities and deliverables?
3. What relationship does testing have to your system development methodology?
4. What do you understand under unit, integration, system and acceptance testing?

BT Developing Behavioral Tests

Duration: 1 - 3 hours

Behavioral testing (a.k.a. “black box” testing) is the major technique for testing information systems. It does not require knowledge of the inner workings. You need to know how to activate the system and how to interpret the reaction. Because of this, professional testers, analysts, designers and end-users perform behavioral testing throughout the lifetime of the system. The challenge is not how to do “black box” testing, but how to do it well.

1. What can you use to identify behavioral tests?
1. How can you figure out what to test?
2. What level of documentation should you create for test cases?
2. Which tests are most likely to find errors?

ST Testing Code Structures

Duration: 1 - 3 hours

Structural (a.k.a. “white” or “clear” box) testing is a powerful technique. It requires that you can either interpret the source code of the object under test or express the logic of the object in a structural manner. This testing approach gives us mathematical models which can calculate the number of tests needed to achieve pre-defined coverage levels.

1. What do models show you that code does not?
2. What do coverage levels prove and what do they neglect?
3. How can you calculate the number of test cases needed for path coverage?
4. What techniques identify specific structural tests?

DE Engineering Test Data

Duration: 1- 3 hours

Careful selection of the value of each field that you can manipulate will drastically reduce the number of test cases that you need to test a solution thoroughly. Engineered data can give you a much higher confidence in the reliability of the system without identifying every unique situation that the system might eventually face.

1. Where do you find out which data you can manipulate?
2. How can you identify the optimal content of each data element?
3. What are realistic expectations for reducing the number of tests needed?
4. What is the payback for engineering the data?

Outline (*continued*)**TP** **Creating Test Plans***Duration: 1- 3 hours*

The test plan is where all of the pieces have to fit together. You have to prioritize, sequence, resource, schedule and manage your defined test cases. A good test plan lets you identify problems before they occur and allows for proactive adjustments.

1. When can you start testing and how long should it take?
2. What does a good test plan encompass?
3. Which techniques minimize the time needed without sacrificing quality?
4. How do you acquire the resources needed to complete your testing?
5. How do you know when you've tested enough?

TS **Establishing a Test Strategy***Duration: 1- 3 hours*

A sound test strategy is the foundation upon which successful implementations build. The strategy has to identify the tools and techniques that will work for your organization. Suitable testing practices and methods allow sufficient flexibility to adapt to specific corporate needs.

1. Which tools and techniques fit your testing approach?
2. Which manual techniques are the most effective?
3. What categories of automated testing tools are available?
4. How should you adapt your approach based on organizational influences?

CT **Testing Without Code***Duration: 1- 3 hours*

Do not defer testing until you have developed the solution. In terms of “bang for the buck”, testing the concept saves far more money than testing the solution. There is a difference between building the *right system* and building the *system right*.

1. How do you test an evolving information system?
2. How can customer requirements be effectively "tested"?
3. What proves a good design?
4. What makes good non-execution techniques work?

FN **Making a Difference***Duration: .5 - 1 hour*

Developing or changing the testing process within an existing organization is a daunting task. It requires a defined project with all the implied controls, decisions, support, etc. What can you do without redesigning the known universe?

1. How do you prioritize the aspects that are the most important for you and your organization?
2. What are you going to do about it?

Objectives

GI	General Introduction	<ul style="list-style-type: none"> ➤ Evaluate a formal testing methodology for applicability ➤ Determine the impact of testing on a system development life cycle ➤ Define common testing terms consistently ➤ Recognize unit, thread, system, integration and acceptance testing
TS	Establishing a Test Strategy	<ul style="list-style-type: none"> ➤ List 7 categories of automated testing tools and describe their use ➤ Recognize organizational impacts on testing ➤ Evaluate usability, performance, release, and configuration testing ➤ Improve manual testing techniques ➤ Distinguish black box from clear box tests
TP	Creating Test Plans	<ul style="list-style-type: none"> ➤ Plan testing activities and deliverables based on business risks ➤ Leverage a team's thinking styles to improve testing performance ➤ Assemble the 17 key elements of effective test plans ➤ Plan testing based on 10 different software error categories
ST	Testing Code Structures	<ul style="list-style-type: none"> ➤ Develop test cases from different views of a component's logic ➤ Use 3 coverage levels as a tool for determining system reliability ➤ Compute how many tests are needed to achieve a selected coverage ➤ Evaluate drivers and stubs for testing an evolving system or release
BT	Developing Behavioral Tests	<ul style="list-style-type: none"> ➤ Recognize test cases on data flow and/or event/response diagrams ➤ Define business test cases based on requirement decomposition ➤ Extract behavioral test cases from end-user documentation ➤ Identify event based test cases using exception analysis techniques
DE	Engineering Test Data	<ul style="list-style-type: none"> ➤ Document test actions and expected results in test scripts ➤ Engineer test data using equivalence and boundary value analysis ➤ Determine data dependencies from Entity/Relationship Diagrams ➤ Minimize the number of test cases needed with content optimization
CT	Testing Without Code	<ul style="list-style-type: none"> ➤ Prepare and conduct quality assurance walk-throughs ➤ Validate the completeness of a defined set of test cases ➤ Test whether the requirements solve the business problems ➤ Cross-reference test cases to system requirements
FN	Making a Difference	<ul style="list-style-type: none"> ➤ Evaluate the usefulness of accelerated sessions in testing ➤ Adapt a testing approach to your organization's needs

