UATA

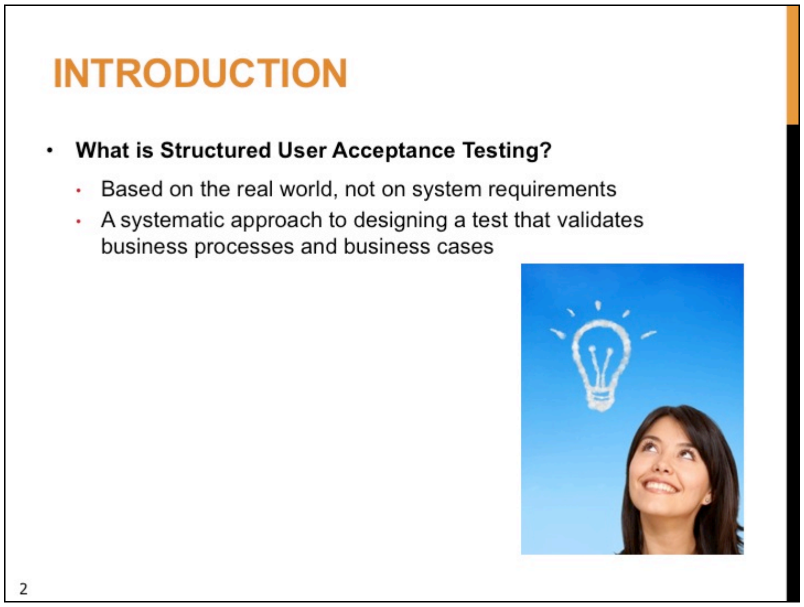
Introduction to User Acceptance Testing

Objectives

- Understand what user acceptance testing is
- Learn about the 1:10:100 rule
- Learn how to make the most of your testing investment
- Learn about life-cycle testing

Synopsis

Learn what user acceptance testing is all about.



INTRODUCTION

- **What is Structured User Acceptance Testing?**
  - Based on the real world, not on system requirements
  - A systematic approach to designing a test that validates business processes and business cases

2

Introduction

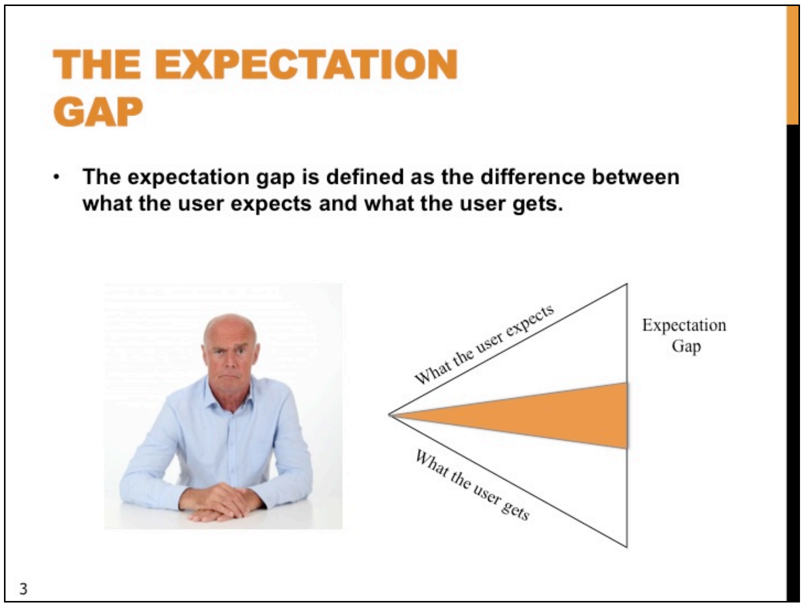
The goal of structured user acceptance testing is to test new or modified software from a business or user perspective

Approach

Structured approaches have given development teams a reliable and systematic method to test software.

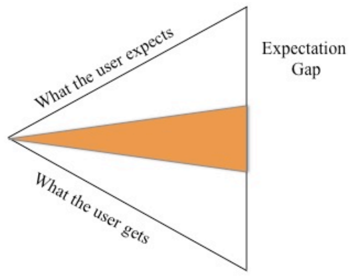
Users can also benefit from this methodology by pre-defining test results, using minimal test conditions, and formalizing the validation process.

In structured user acceptance testing, users design a test that is based on real-world scenarios, regardless of any formal or defined requirements. The reason that user acceptance testing is based on real-world scenarios is that requirements are often misinterpreted. It's as if developers and users see the same world from two different worlds. The developer sees the world from a systems point of view, building the system as he or she interprets the requirements. The user sees the world from a business or operational point of view. The user's main concern is whether or not the system will work in the business or operational environment.



THE EXPECTATION GAP

- **The expectation gap is defined as the difference between what the user expects and what the user gets.**



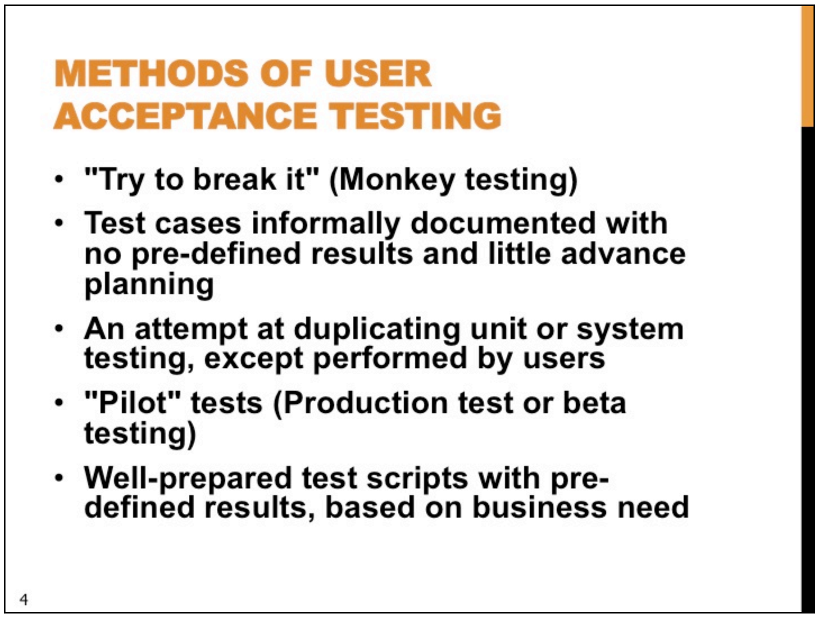
3

The Expectation Gap

In traditional methods of system development, the user approaches the Information Technology (IT) department with a need. The IT people determine what the user says they want. Then, the designer's design and the programmer's code. Eventually, a system is delivered to the user - sometimes tested, often times not.

In this scenario, the user is often disappointed because what they described in the original request is different from what they have been delivered. This is called "The Expectation Gap."

User acceptance testing is not the answer to avoiding the expectation gap. Continuous involvement throughout the development process is the only way to close the expectation gap. Prototyping, Joint Application Development and checkpoint reviews are effective ways for the user to be involved in the systems development process.



METHODS OF USER ACCEPTANCE TESTING

- "Try to break it" (Monkey testing)
- Test cases informally documented with no pre-defined results and little advance planning
- An attempt at duplicating unit or system testing, except performed by users
- "Pilot" tests (Production test or beta testing)
- Well-prepared test scripts with pre-defined results, based on business need

4

Methods Of User Acceptance Testing

- **"Try to break it" (Monkey testing)**

In this approach, a group of people are assembled in a room and told to try to break the system for an hour or so. The theory is that random testing is a good way to find defects. The fact is that you can find the easy defects, but the tougher defects will remain hidden.

- **Test cases informally documented with no pre-defined results and little advance planning**

A step up from random testing, there is thought given to what will be tested, but the expected results are not defined. This can lead to doubts and disputes as to what the outcome should actually be.

- **An attempt at duplicating unit or system testing, except performed by users**

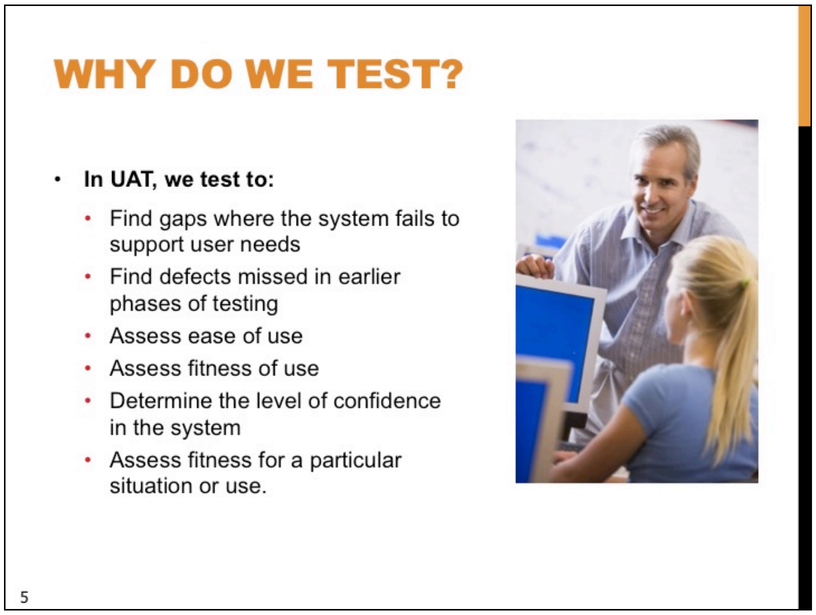
In this method, the users follow the same trail as the developers and might find a few defects, but will miss testing to validate that the system will work in the real world.

- **"Pilot" tests (Production test or beta testing)**

There is a saying that "Every system is eventually tested - by the developers or by the customer." In this method, there is no user-oriented test before production. Defects are found in the real world where the costs to fix them are the highest.

- **Well-prepared test scripts with pre-defined results, based on business need**

This is the essence of structured user acceptance testing. You know what will be tested and you have a high degree of confidence that the business processes and cases will be supported in the real world by the system. This approach takes more time and costs more, but it balances the risk of a system or project failure.



WHY DO WE TEST?

- **In UAT, we test to:**
  - Find gaps where the system fails to support user needs
  - Find defects missed in earlier phases of testing
  - Assess ease of use
  - Assess fitness of use
  - Determine the level of confidence in the system
  - Assess fitness for a particular situation or use.

5

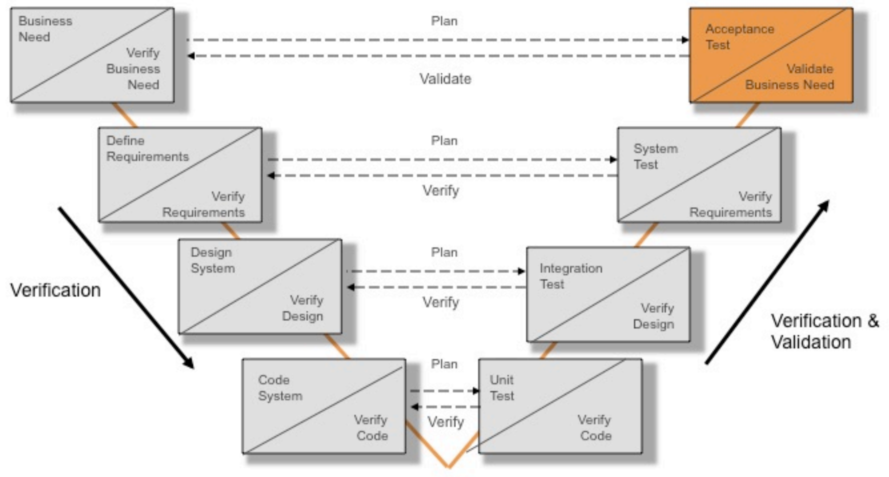
Why Do We Test?

The basic reason we test software and systems is that people are fallible and we can prevent loss and mitigate risks before the software is released to business users and/or customers.

UAT has some unique distinctions. In UAT, we test to:

- Find gaps where the system fails to support user needs
- Find defects missed in earlier phases of testing
- Assess ease of use
- Assess fitness of use
- Determine the level of confidence in the system
- Assess fitness for a particular situation or use.

WHEN USER ACCEPTANCE TESTING OCCURS



Source: Surviving the Top Ten Challenges of Software Testing by William E. Perry and Randall W. Rice

6

The "V" Diagram

In this diagram, the major phases of development are shown along with the corresponding phases of testing. The order of execution is also indicated from the upper left to the bottom of the "V" and back up to the upper right.

Each box has either a verification step (a test performed by inspecting something) or a validation step (a test performed by executing software on the computer).

The "V" diagram can be used to depict testing in any methodology, including agile, which in essesnce has it's own "mini-V".

It is important to note that user acceptance testing is at the top of the "V" and validates business or operational need, not that the system was built according to requirements. The reason this distinction is important is because if the requirements contain errors (and most requirements, if the exist, do contain errors), requirments-based tests will not reveal them.

THE CHALLENGES

- Complex systems
- Poor documentation
- Inadequate structural testing
- Lack of training
- Mission critical applications
- Short time frames

7

Challenges Of User Acceptance Testing

There are many factors that distinguish user acceptance testing from other phases of software testing. The introduction of the end-user to the process of software acceptance can have unpredictable effects on the outcome of the entire project.

These challenges are:

- **More complex systems due to:**
  - Extensive rewrite projects
  - New technologies: Computer Aided Software Engineering (CASE), Object-Oriented (O-O), Relational Databases, Client/Server, and Data Warehouse
  - Many application domains are complex due to their nature of use. For example, medical, aerospace, and other complex applications.
- **Poor Documentation**
  - Often the user acceptance test is the final test before production. This is also when the documentation is usually written, if at all.
  - Without clear documentation and training, the user will be frustrated when using the software. This also contributes to misunderstandings and false reporting of defects. The phrase often heard is, "You mean it's *supposed* to work this way?!"
  - The user must understand the software to test it or use it.
- **Inadequate Structural Testing**
  - To meet deadlines, the developers are under constant pressure to "cut corners". One of the most popular corners to cut is testing.
  - Developers might fail to test key scenarios. This results in shifting the focus from usability to debugging.

- **Users often lack technical or testing knowledge. This leads to:**
    - Haphazard testing
    - Incomplete testing (not knowing how to use the system)
    - Informal defect reporting.
- **Demand for Defect-free Software**
    - Critical Applications
    - High management expectations
    - High customer expectations
- **Short User Acceptance Testing Timeframes**
    - If any extra time is needed in the development phases, acceptance testing is one area that traditionally gets shortened.

The Solution:

Users need a way to test software in a well-defined and structured manner, while providing maximum efficiency and flexibility.

The solution must focus on making sure the right system was built (or purchased).

MANAGEMENT'S ROLE IN TESTING

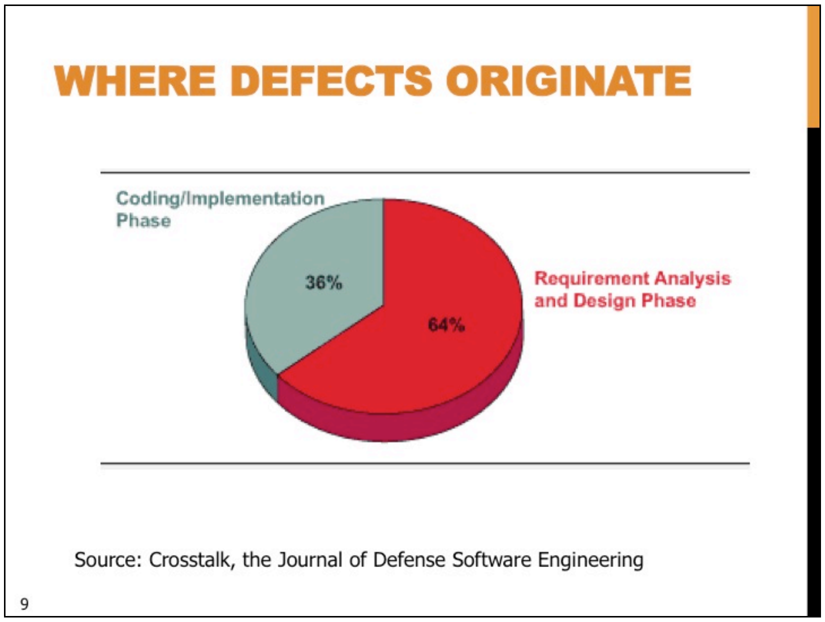
- Management's commitment to quality must be backed up by actions consistent with the organization's purpose and vision for improving quality.

8

Management’s Role in Testing

One of the most commonly identified testing problems is the lack of *congruent* management support. That is, that if management says they want quality software, they will support the efforts to achieve it even if it means missing a deadline.

Management’s actions must back up the quality message. Otherwise, the test will be an academic exercise to find defects, but do nothing about them.



WHERE DEFECTS ORIGINATE

Source: Crosstalk, the Journal of Defense Software Engineering

9

Where Defects Originate

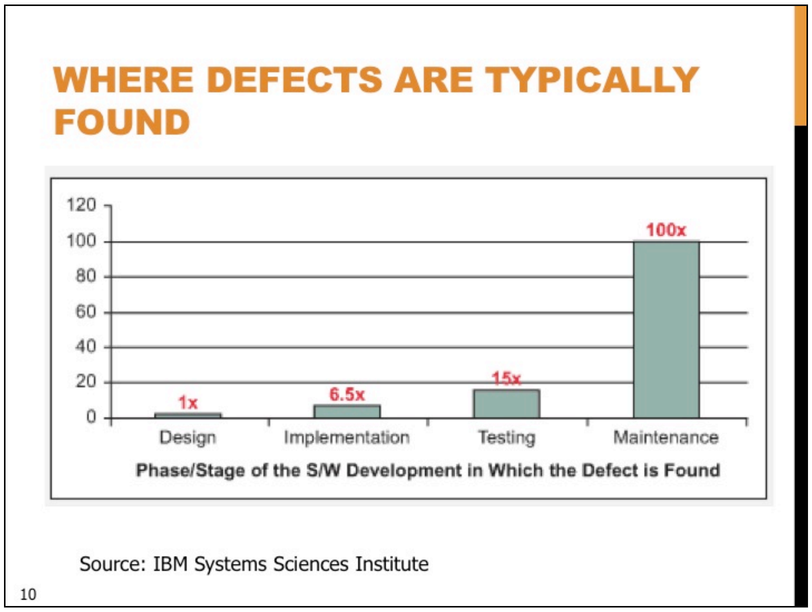
In numerous studies performed since the 1970's, it has been shown that most defects originate in the requirements definition phase of a project. In many cases, these defects are not found until the later stages of a project.

Requirements defects can be classified as:

- Missing - System functionality is not described for some or all functions.
- Ambiguous or unclear - Requirements may be confusing or in conflict with other requirements.
- Wrong - Requirements may simply be incorrect.

The system should be tested to ensure it is being built according to the requirements at several checkpoints during the project.

However, it is important to understand that just because the system meets the requirements does not mean that the system will work in the real world. Requirements are very difficult to define completely and accurately.



WHERE DEFECTS ARE TYPICALLY FOUND

Source: IBM Systems Sciences Institute

10

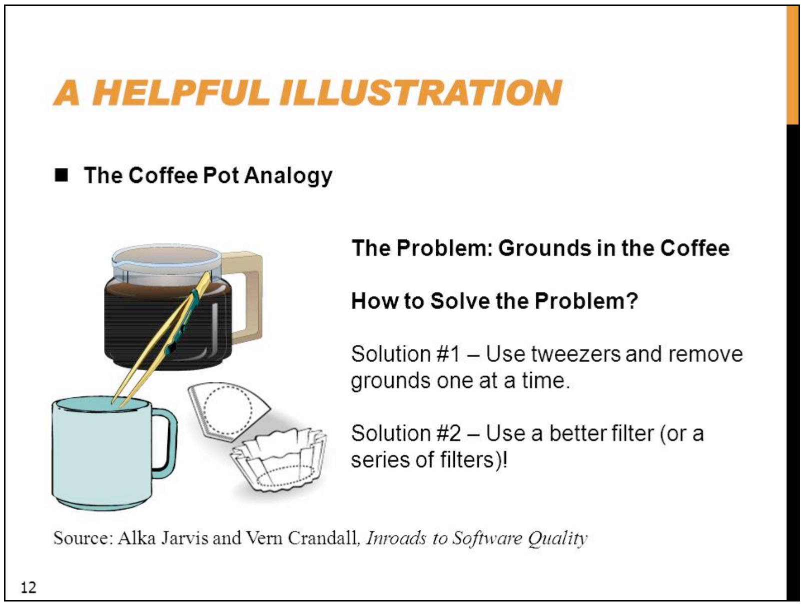
Where Defects are Typically Found

Testing is most commonly performed at the end of a project. This is sometimes called "The Big Bang" approach to testing, since most of the testing effort is located in one big phase.

The main problem with this approach is that it is difficult and costly to fix defects found at the end of a project. Fixing defects can cause other changes or other defects.

Testing should begin on the first day of the project by verifying requirements. In fact, testing can even begin during the feasibility or cost/benefit analysis. The best defect you can find is the system that shouldn't be built.

Another finding from years of research is, that defects fixed in production cost much more to fix than during requirements or design.



A HELPFUL ILLUSTRATION

- **The Coffee Pot Analogy**

The Problem: Grounds in the Coffee

How to Solve the Problem?

Solution #1 – Use tweezers and remove grounds one at a time.

Solution #2 – Use a better filter (or a series of filters)!

Source: Alka Jarvis and Vern Crandall, *Inroads to Software Quality*

12

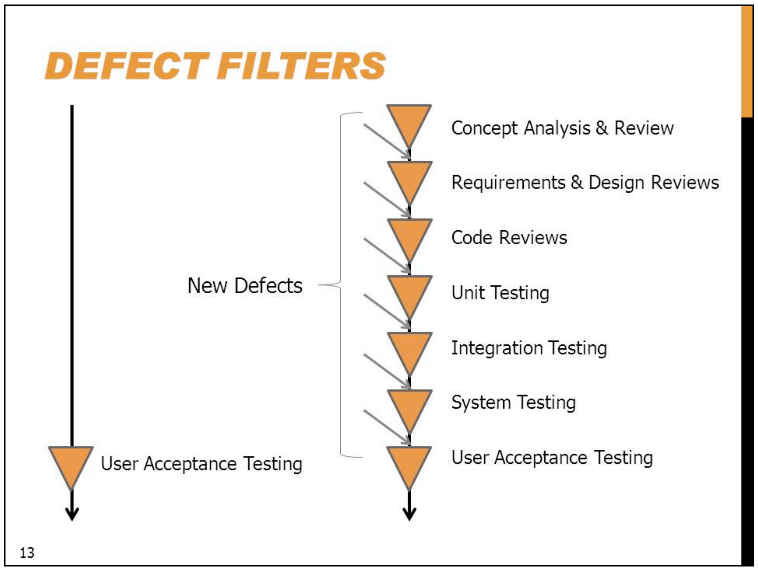
The Coffee Pot Analogy

Let's look at a coffee making analogy to explain the difference between common testing approaches and an approach needed for a life cycle approach to testing. Let's assume that we have brewed a pot of coffee and it is filled with coffee grounds. These coffee grounds are errors or defects in the coffee.

In testing we need to remove these defects. Traditional approaches might use tweezers and try to pick out the coffee grounds one at a time. With enough time and effort all of the coffee grounds would be removed, but at that time the coffee may be cold and undesirable. Testing by using the life cycle testing approach proposed in this course would pour the coffee through a filter. The filter would catch all of the coffee grounds and would have defect-free hot coffee quickly.

Other solutions include using instant coffee instead, but there is a tradeoff with taste. Also, the coffee could be ground so fine there are no grounds, just sediment. Both of these solutions are examples of preventing the grounds (defects) in the first place. Life cycle testing incorporates the experiences of leading test organizations over long periods of time. It incorporates good testing practices together with knowledge of why systems fail.

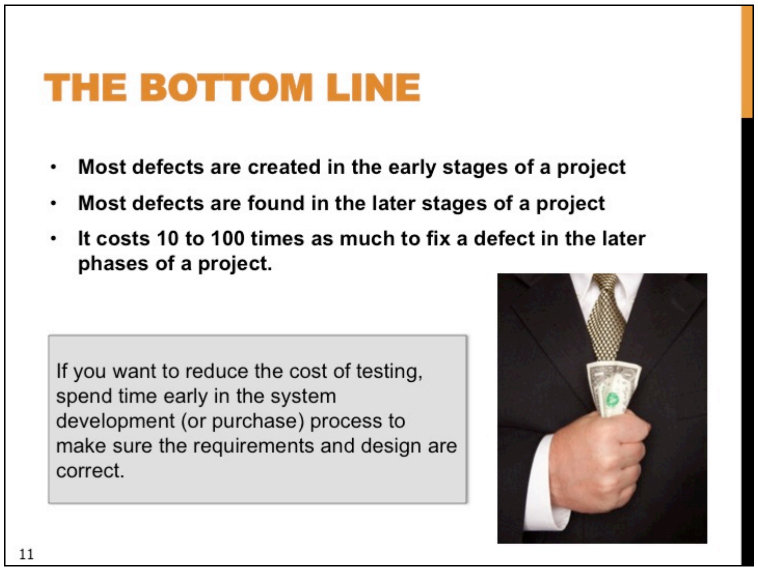
It's basically incorporating the combined experiences of many testers into an effective test approach that attempts to filter defects from ever getting into the final product.



DEFECT FILTERS

Defect Filters

In this slide, we see some examples of defect filters. On the left is a single filter, UAT. On the right, you can see a variety of filters which will have a greater chance of finding more defects throughout a project. Keep in mind that since work is continually performed, new defects will be injected into the system and related work products.



THE BOTTOM LINE

- **Most defects are created in the early stages of a project**
- **Most defects are found in the later stages of a project**
- **It costs 10 to 100 times as much to fix a defect in the later phases of a project.**

If you want to reduce the cost of testing, spend time early in the system development (or purchase) process to make sure the requirements and design are correct.

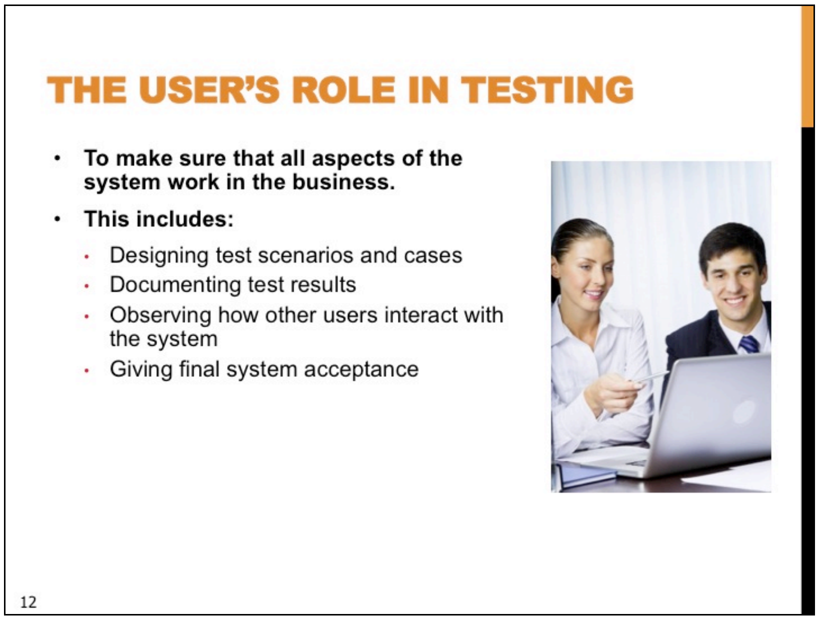
11

The Bottom Line

The conclusion from these findings indicate that many organizations are focusing on their testing efforts at the wrong place - at the end of the project.

Does this mean that user acceptance testing should be performed earlier? No, because the system must be completed and system tested before it is ready for acceptance testing.

What this does mean is that users should be involved from day one throughout the project to review project deliverables such as requirements, design, prototypes, etc.



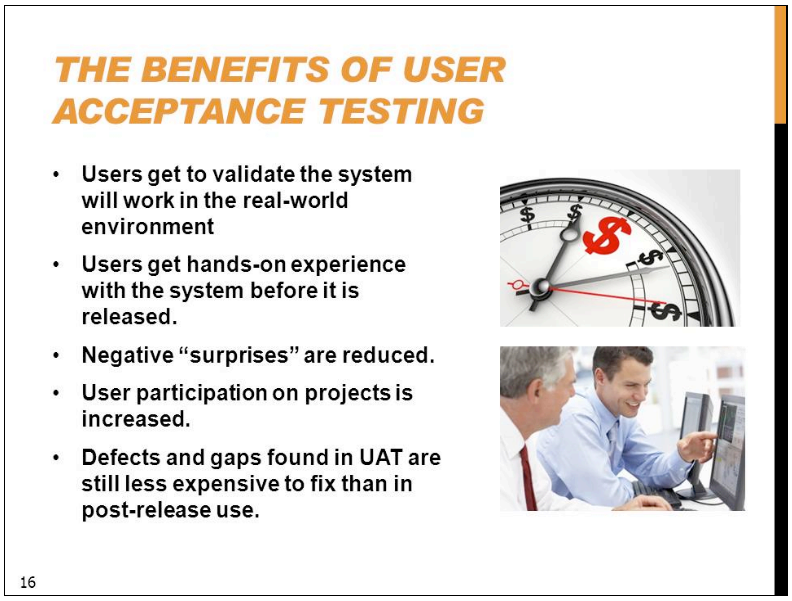
THE USER'S ROLE IN TESTING

- **To make sure that all aspects of the system work in the business.**
- **This includes:**
  - Designing test scenarios and cases
  - Documenting test results
  - Observing how other users interact with the system
  - Giving final system acceptance

12

The User's Role in Testing

The main value that the user brings to the testing effort is the ability to understand the business or day-to-day operation. Other people can test details such as edits or software structure. The primary focus of the user should be to ensure the system will support the business in the real world.



THE BENEFITS OF USER ACCEPTANCE TESTING

- Users get to validate the system will work in the real-world environment
- Users get hands-on experience with the system before it is released.
- Negative "surprises" are reduced.
- User participation on projects is increased.
- Defects and gaps found in UAT are still less expensive to fix than in post-release use.

16

The Benefits of User Acceptance Testing

Users get to validate the system will work in the real-world environment

Testing the system based on paper specifications is one thing. Testing in a simulated real-world environment is another.

Users get hands-on experience with the system before it is released.

UAT can be some of the best training for users.

Negative "surprises" are reduced.

This especially true when users can be involved throughout the project.

User participation on projects is increased.

UAT is a great opportunity for users to participate in the project.

Defects and gaps found in UAT are still less expensive to fix than in post-release use.

Even though UAT is often performed late in a project, it is still better to find problems in UAT than in actual system use.