

# CS631G Software Verification

## EXPLORATORY TESTING (W12)

Class instructor: Yuri Chernak, PhD

# Outline

2

- What is Exploratory Testing?
- History of Exploratory Testing
- Best Practices for Exploratory Testing
- Exploratory Testing Process
- Types of Exploratory Testing
- Executing Exploratory Testing
- Using ChatGPT for Exploratory Testing
- Advantages of Exploratory Testing
- Disadvantages of Exploratory Testing
- Challenges of Exploratory Testing
- Differences between Exploratory and Scripted Testing
- Exploratory Testing and Agile Development
- Exploratory Testing FAQs

# Class Objectives

3

The objective of this class is to introduce the Exploratory Testing methodology.

At the end of this class you will:

- Understand what Exploratory Testing is and its history;
- Become familiar with Exploratory Testing process and best practices;
- Become familiar with the types of Exploratory Testing and how ChatGPT can assist to perform testing;
- Understand the difference between Exploratory Testing and scripted testing.

# What is Exploratory Testing?

4

**Exploratory Testing** is an approach to software testing that emphasizes simultaneous learning, test design, and test execution.

Unlike scripted testing, where tests are pre-planned and documented in detail before execution, exploratory testing involves testers exploring the software system in an ad-hoc and unscripted manner. Testers explore the application's features, functionalities, and user interfaces dynamically, creating and executing test cases on the fly based on their observations and insights.

Key characteristics of Exploratory Testing include:

- Simultaneous Learning and Testing
- Ad-Hoc (on-the-fly) Test Design
- Flexibility and Creativity
- Focus on Realistic User Scenarios
- Risk-Based Testing
- Continuous Feedback Loop
- Exploration of Edge Cases and Boundaries
- Complement to Scripted Testing

# History of Exploratory Testing

5

- **Exploratory testing** has always been performed by skilled testers. In the early 1990s, ad hoc testing was too often synonymous with sloppy and careless work.
- As a result, a group of test methodologists (now calling themselves the Context-Driven School) began using the term "exploratory" seeking to emphasize the dominant thought process involved in unscripted testing, and to begin to develop the practice into a teachable discipline.
- This new terminology was first published by Cem Kaner in his book *Testing Computer Software* and expanded upon in another book *Lessons Learned in Software Testing*.
- Exploratory testing can be as disciplined as any other intellectual activity.

C. Kaner's exploratory testing tutorial is available on the course portal (Classes), in the Supplementary Materials module.

# Best Practices for Exploratory Testing

**Understand the customer:** For effective exploratory testing, it is important to understand the customer's viewpoint and expectations properly. End users browse the same software in different ways based on age, gender preferences, and other factors. Testers must be able to approach the software from all those user perspectives.

**The aim of testing should be clear:** For effective exploratory testing, the testers need to have a clear mindset and have clarity on the mission of testing. Testers should maintain clear notes on what needs to be tested, and why it needs to be tested.

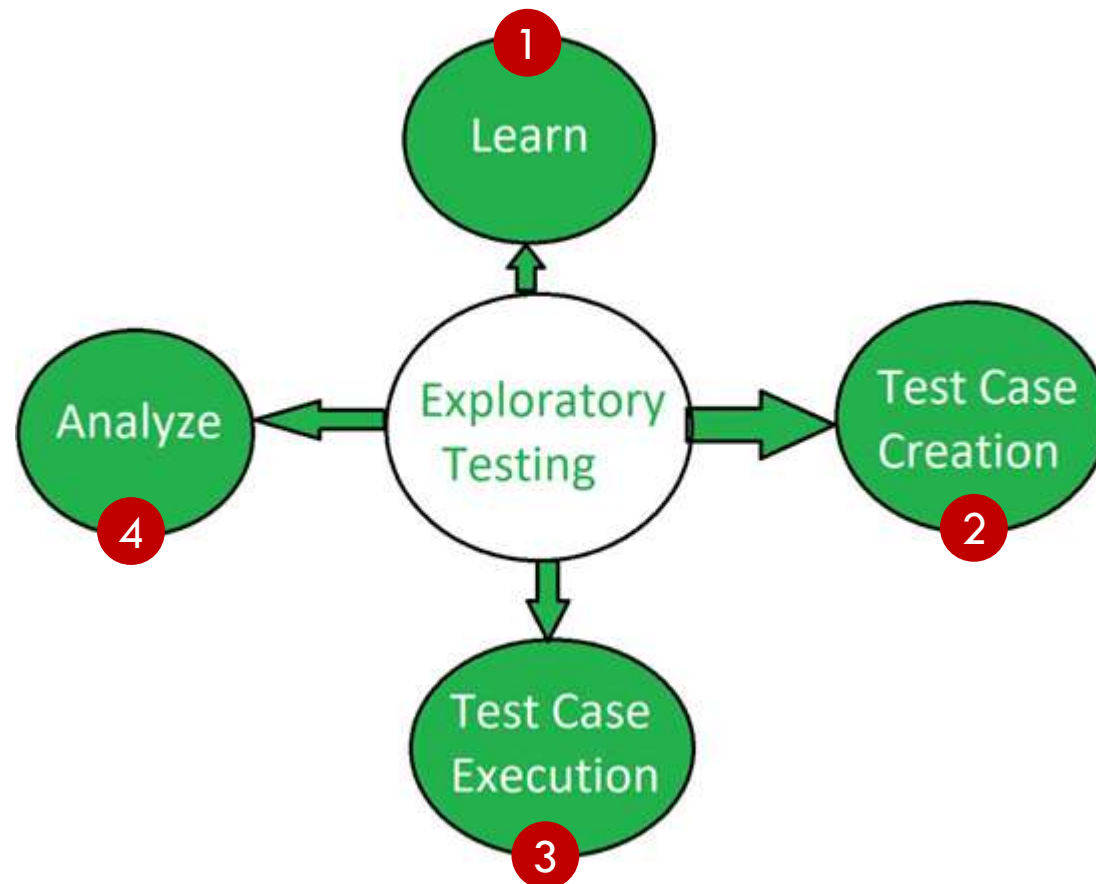
**Proper documentation:** It is important to make proper notes and take a document and monitor test coverage, risk, Test Execution log, issues, and queries.

**Tracking of issues:** The tester should maintain a proper record of questions and issues raised during testing.

# Exploratory Testing Process

7

The following 4 steps are involved in the exploratory testing process:



# ET Process Steps Explained

- 1 **Learn:** This is the first phase of exploratory testing in which the tester learns about the faults or issues that occur in the software. The tester uses his/her knowledge, skill, and experience to observe and find what kind of problem the software is suffering from. This is the initial phase of exploratory testing. It also involves different new learning for the tester.
- 2 **Test Case Creation:** When the fault is identified, i.e., a tester comes to know what kind of problem the software is suffering from then the tester creates test cases according to defects to test the software. Test cases are designed by keeping in mind the problems end users can face.
- 3 **Test Case Execution:** After the creation of test cases according to end user problems, the tester executes the test cases. Execution of test cases is a prominent phase of any testing process. This includes the computational and operational tasks performed by the software to get the desired output.
- 4 **Analysis:** After the execution of the test cases, the result is analyzed and observed whether the software is working properly or not. If defects are found, they are reported and then fixed and the above three steps are performed again.



# Types of Exploratory Testing

9



## **Free-style exploratory testing:**

This type of testing follows a free-style approach to testing and does not include any rules, approaches, or specifications. Simply testers go through the application quickly, mainly to identify defects, verify the work of other testers, or even perform a quick smoke test.

## **Scenario-based exploratory testing:**

This form of testing is based on real-user scenarios wherein testers take each scenario and then test the software from all angles to match the scenario. The main aim of this testing type is to explore the software for maximum possible scenarios and ensure to increase the test coverage.

## **Strategy-based exploratory testing:**

This type of testing is performed by an experienced tester who has complete knowledge of the software under test. It involves various testing techniques such as boundary value analysis, equivalence, risk-based, error guessing, etc., to identify more challenging bugs.

# Executing Exploratory Testing

## What is required to execute a test?

A **mission** statement for the exploratory session is defined, this describes the common overall purpose for the session. It should tell the participants why the session is done and what we hope to achieve on an elevated level. This allows the session to be measurable and reportable on a larger scale.

To ensure a focused approach a **timebox** is required for the session, this will also allow the team to synchronize and collaborate. A typical timespan would be 45 mins to 1.5 hours, all effort is done during this time span only. Opportunities to extend or cut by specific amounts can be given e.g. 30 mins.

If necessary individual objectives, test ideas, or agendas can be described in **test charters**. Some or all charters may be needed to be carried out to achieve the mission. These charters should suggest what to test, how it could be tested and what may need to be looked at. Resources may have different skills so test charters can be assigned to specific individuals if required.

During the session, testers do their Exploratory Testing, use charters, and document what they do or find.

# Using ChatGPT for Exploratory Testing

11

ChatGPT can be a valuable tool in exploratory testing by providing real-time assistance, generating test ideas, simulating user interactions, and offering insights into the software under test.

By leveraging ChatGPT's capabilities, testers can enhance their exploratory testing efforts, generate innovative test ideas, simulate user interactions, and gain insights into the software under test. ChatGPT's real-time assistance and collaborative features can contribute to more effective and efficient exploratory testing processes.

Here are some ideas how ChatGPT can assist in Exploratory Testing:

- Generating Test Ideas
- Assisting with Test Case Creation
- Providing Contextual Information about the software under test
- Exploring Edge Cases and Boundaries for individual features
- Offering Suggestions and Insights, etc.

# Example: Using ChatGPT for Exploratory Testing

12

Using ChatGPT to generate test ideas includes the following steps:

1. Train the AI model, explain your system context.
2. Describe a feature under test.
3. Ask for test ideas.

**Step 1.** Explain your system context.

Xero Software is a New Zealand–based technology company that provides cloud-based accounting software for small businesses. The company has offices in New Zealand, Australia, the United Kingdom and the United States. Xero's products are based on the software-as-a-service model and sold by subscription, based on the type and number of entities managed by the subscriber.

**Step 2.** Explain the feature under test “Create New Invoice”.

As a user, I want create a new invoice by selecting the client, adding line items for the services provided, and specifying payment terms.

# Example: Using ChatGPT for Exploratory Testing

13

**Step 3.** can you suggest test ideas for testing “Create New Invoice” feature?

## Chat GPT response:

Certainly! Here are some test ideas for testing the “Create New Invoice” feature:

### Basic Invoice Creation:

- Verify that users can create a new invoice with all required fields filled out.
- Test that the invoice number is automatically generated or manually entered and is unique.
- Ensure that users can add multiple line items to the invoice with correct details such as item name, quantity, unit price, and total amount.

### Validation Checks:

- Test validation checks for mandatory fields such as customer name, invoice date, and due date.
- Verify that appropriate error messages are displayed when mandatory fields are left blank or invalid data is entered.
- Test that users cannot submit the invoice creation form until all mandatory fields are filled out correctly.

### Discounts and Taxes:

- Verify that users can apply discounts to the invoice and that the total amount is recalculated correctly.
- Test applying various discount percentages (e.g., 0%, 5%, 10%) to ensure accurate calculation of the discounted amount.
- Verify that taxes can be applied to the invoice and that the total amount reflects the tax amount correctly.

### Integration with Customer Data:

- Test that customer details are populated correctly in the invoice creation form when selecting a customer from the database.
- Verify that changes to customer details (e.g., address, email) are reflected accurately on the invoice.

### Sending Invoices:

- Test sending the invoice to the customer's email address and verify that it is delivered successfully.
- Verify that the invoice status is updated to "Sent" after sending.
- Test sending invoices to multiple recipients and verify that all recipients receive the invoice.

# Advantages of Exploratory Testing

14

**Less preparation is required:** It takes no preparation as it is an unscripted testing technique.

**Finds critical defects:** Exploratory testing involves an investigation process that helps to find critical defects very quickly.

**Improves productivity:** In exploratory testing, testers use their knowledge, skills, and experience to test the software. It helps to expand the imagination of the testers by executing more test cases, thus enhancing the overall quality of the software.

**Generation of new ideas:** Exploratory testing encourages creativity and intuition, thus the generation of new ideas during test execution.

**Catch defects missed in formal testing:** Exploratory testing helps uncover bugs that are normally missed by other testing techniques.

# Disadvantages of Exploratory Testing

15

**Tests cannot be reviewed in advance:** In exploratory testing, Testing is performed randomly so once testing is performed it cannot be reviewed.

**Dependent on the tester's knowledge:** In exploratory testing, the testing is dependent on the tester's knowledge, experience, and skill. Thus, it is limited by the tester's domain knowledge.

**Difficult to keep track of tests:** In Exploratory testing, as testing is done in an ad-hoc manner, keeping track of tests performed is difficult.

**Not possible to repeat test scenarios:** Due to the ad-hoc nature of testing in exploratory testing, tests are done randomly and thus it is not suitable for longer execution time, and it is not possible to repeat the same test scenarios.

# Challenges of Exploratory Testing

Exploratory testing, while highly effective, comes with its own set of challenges. These challenges can impact the efficiency, effectiveness, and reliability of the testing process. Addressing these challenges requires careful planning, effective communication, continuous learning, and collaboration within the testing team.

Here are some common challenges associated with exploratory testing:

- **Test Coverage:** One of the main challenges of exploratory testing is ensuring adequate test coverage.
- **Testing Effectiveness:** Evaluating the effectiveness of exploratory testing can be challenging.
- **Repeatability and Consistency:** Since exploratory testing is spontaneous and unscripted, there's a lack of repeatability and consistency in the testing process. The same test scenario may yield different results when executed by different testers or at different times. This variability makes it challenging to reproduce and verify reported issues reliably.



# Differences between Exploratory and Scripted Testing

17

- Both approaches have their strengths and weaknesses.
- When appropriate, testers should use the Exploratory Testing to complement the formal Scripted Testing approach.

Scripted Testing	Exploratory Testing
Testers follow the sequence and steps of test cases which are predefined	There are no predefined steps or sequence of test cases. Testers test the software on the fly.
Feedback is slower	Enables rapid feedback
Test scripts can be traced back to original requirements to check the test coverage	There is no such traceability for test coverage in this testing type
Testers test the application based on the specifications	Testers test the application based on their understanding, creativity, expectations.
Tests can be reproduced in this testing type	Tests may not be reproduced in this testing type

# Exploratory Testing and Agile Development

18

- Exploratory testing can play a crucial role in Agile development by aligning with Agile principles and practices, facilitating rapid feedback, supporting iterative development, and promoting collaboration among team members.
- By integrating exploratory testing into their Agile practices, teams can deliver high-quality software that meets customer needs and expectations iteratively and incrementally.

Here's how exploratory testing can help Agile development:

- Early and Continuous Feedback
- Support for Iterative Development
- Rapid Bug Detection and Fixing
- Dynamic Test Case Generation
- Risk-Based Testing
- Continuous Learning and Improvement

# Exploratory Testing FAQs

## 1. Is Exploratory Testing white box testing?

No, Exploratory Testing is black box testing, not white box testing. White box testing is a testing technique where the tester has knowledge of the internal structure, design, and implementation of the software being tested.

Exploratory Testing does not rely on knowledge of the system's internal workings.

## 2. Can exploratory testing be automated?

Exploratory Testing is primarily a manual testing technique. Some aspects of exploratory testing can be supported by automated tools, but the core essence of exploratory testing relies on human exploration and judgment. Test scripts for bugs found by exploratory testing should still be created for regression test runs to verify that they have been fixed.

## 3. Is exploratory testing functional testing?

Since it is just an approach, exploratory Testing can include **both functional testing and non-functional testing**.

## 4. Why exploratory testing is required for Agile projects?

Exploratory Testing is well-suited for Agile projects due to its flexible and adaptive nature. Agile projects often require fast feedback and continuous testing to accommodate frequent changes and iterations, and exploratory testing is perfect for that.

## 5. What are some exploratory testing best practices?

- Defining clear objectives and charters for each exploratory testing session.
- Adapting the testing approach based on the software's context and risk areas.
- Combining domain knowledge, user perspective, and intuition to uncover issues.
- Capturing detailed notes, screenshots, and videos to document findings.
- Collaborating with developers, business analysts, and other stakeholders during testing.
- Balancing exploration with structured testing techniques to achieve comprehensive coverage.
- Sharing insights and lessons learned from exploratory testing with the team.