WHITEPAPER

The complete Exploratory Testing Handbook

Original Software

The importance of Exploratory Testing

Exploratory Testing is a style of testing based on the knowledge and ingenuity of testers. There is no need for explicit scripted instructions, which may limit the scope or creativity.

Anybody who has worked in software for a certain amount of time will have been asked at one point to ‘have a play’ with a new application to ascertain suitability. This form of 'ad hoc' testing gave birth to exploratory testing used by teams today. Although the process has become a little more formalized, the critical requirement to allow freedom and flexibility to explore remains a central tenet of this approach.

Understanding what the test item does, trying new things, developing ideas for testing, documenting, learning, and finding potential bugs are the vital useful areas undertaken during a session.

Exploratory testing can be done by individuals or by teams. It provides a framework, helping a team to evaluate software, or a specific area of a product, in a highly creative and investigative manner but within a limited amount of time.

The first person to coin the phrase ‘exploratory testing’, Cem Kaner concisely described the approach as simultaneous learning, test design and test execution.

Scoping the exploratory experience.

Although part of the allure of exploratory testing is the ability to do this without much preparation or documentation, it is still helpful to do some setup to ensure enhanced success. So how is it best to manage the process of exploratory testing, and who will do this?

Who can do exploratory testing?

As no QA, technical, or automation knowledge, along with scripts and processes, is needed to do exploratory testing, this can be opened to a broader audience for evaluation. It is common to see people from the product side (UX/UI), development, QA, and end-users being involved. Each person will bring their own unique knowledge and skillset to bear on the process.

Exploratory testers are successful when using knowledge about the domain, the system under test, and customers. Experience is a key factor explaining the effectiveness of exploratory testing and, as such, may not be suitable for newer team members. However, sometimes it can be beneficial to get someone who does not know the development to look and get an outside perspective.

Exploratory testing is also used in critical domains, and that this approach places high demands on the person performing the testing.

When is it best to do exploratory testing?

This approach is best used when people must learn about a piece of software quickly. This can be done early on, including the prototype stage, as there is no need for test scripts or even requirements documentation. Often used in agile iterations, this technique allows a team to find issues quickly and efficiently in a rapidly changing environment.

It is also good to use this approach when the need to inspect software from an end user's perspective arises. When features are added or changed late in the software cycle, it would be valuable time to employ exploratory testing techniques.

One of the outputs of an exploratory test session is the groundwork for preparing more classic test scripts; these can then be used for regression testing during later phases.

What preparation is required?

Definition of the constraints of a test session is critical to its success. Although part of the exploratory approach's appeal is the freedom to roam and test as the will goes, it is also essential to set some guidelines to maximize efficiency and ensure the correct areas are looked at. This will mean pre-defining features, functions, specific processes, etc. to align expectations.

Define how much time each session participant shall spend for the test session. This will ensure focus on what is needed whilst still allowing a more playful approach. Additionally, this will ensure that the participants can plan time and remain uninterrupted by other people whilst the effort is ongoing.

Include a wide range of resources with different backgrounds, knowledge and roles, examples of this can be found in the who section above. This should include end-users and any stakeholders for the software, including those external to your own organization. Sometimes it can be beneficial to invite people to participate who haven't been involved in the project at all to get an outside perspective and different viewpoint.

A definition of the approach to feedback, issues or any other type is required. What format will be used to collate test reports, is there any need for proof of testing for auditing or defect reproductions? How will the information be centralized and shared for collaboration with other interested parties from the team?

Executing exploratory testing.

With disparate teams spread across departments and companies, and countries, it is less likely that people can be brought together in one room to commit to a conference room pilot. Therefore, how long it takes and who is involved are essential questions that need to be answered before the execution phase. This technique is significantly teachable and manageable and is therefore suitable for a wide range of participants both within IT and, more importantly, within the business.

What is required to execute a test?

A mission statement for the 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 made during this period 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.

How is it best to evaluate a testing session?

To continue improving and getting the best out of exploratory testing, it is necessary to evaluate what happened. After the session, the team will have a debriefing. In the debriefing, the team and the organizer discuss the results, find agreement on what is a bug and what is not, and decide about the next steps. Critical elements for the discussion are as follows:

- What test cases can be created from the session
- Current product behavior pros/cons
- What defect reports are needed?
- New requirements (UX/UI)
- Risks identified and recorded
- Questions (may help drive into the user training documentation)

Benefits of exploratory testing.

Explore the unknown to avoid the unexpected.

There are considerable and multiple benefits from taking an exploratory testing approach. One of the main advantages is that the testing can begin early in the development cycle as it doesn’t need a lot of preparation or test assets. Experiments have shown that while scripted and exploratory testing results in similar defect detection effectiveness (the total number of defects found), exploratory results in higher efficiency (the number of defects per time unit) as no effort is spent pre-designing the test cases.

With a mindset of investigating the application under test, removing the limitations of scripted testing, the testers are more intellectually stimulated. This leads to a happier team that is more willing to help and give input into software development. Using each person's deductive reasoning, a different viewpoint of the software is provided by each user; this is very useful to get a bigger picture view of what is going on. The feedback given then extends beyond the standard 'bug' reports and allows inputs of all kinds, further enhancing user buy-in to the application.

At the same time as testing, the team gains insights into the system behaviors and outputs through adaptation and reasoning. This allows the process to evolve naturally within the given mission statement guidelines and has a more beneficial output. It works well in the system areas that have not yet been specified or where documentation and requirements exist. Accelerated bug detection is also a key consideration, especially within agile teams, as all the main processes can be tested first in a ‘real world’ setting.

What are the cons of exploratory testing?

Disadvantages are that tests invented and performed on the fly can't be reviewed in advance (and by that prevent errors in code and test cases) and that it can be difficult to show exactly which tests have been run. Therefore, it is important to scope the sessions so that coverage can more accurately be deduced.

Freestyle exploratory test ideas, when revisited, are unlikely to be performed in precisely the same manner, which can be an advantage if it is crucial to find new errors; or a disadvantage if it is more important to repeat specific details of the earlier tests.

Benefits of exploratory testing Software Tools

Exploratory testing itself is beneficial, but when dovetailed with other testing processes, it becomes a powerful way to understand the application better, build better functional tests and finally enhance the quality of the application.

To enable the most significant benefits to the exploratory testing session, it would be good to get help from an application supporting both this style and the standard scripting style of testing.

Key requirements for help would be:

- Easy collaboration & resource management
- Standardized documentation and feedback
- Process review & triage capability for all feedback types
- Efficient co-ordination
- Simplicity in use
- Flexibility to enable different test paradigms
- Centralized & easy to deploy

Conclusion

If done correctly, exploratory testing can be a powerful tool in the software validation arsenal. Simple to set up, it is the quickest approach to testing in the first instance, the many outputs described can also be used in the software development lifecycle moving forwards. Not only will it ensure software quality, but it involves people in a deeper way, giving them training on new products at the same time as allowing a feeling of being part of the solution.

Ultimately this will lead to not only happy customers, through a more business centric approach to solutions, but happier testers that get involved as they get a profound buy in to the various applications in scope.

Original Software