Exploratory testing – Tours

Since the tour descriptions are not bound to any product type, technology or development process they even facilitate cross team communication. However, this works only if everybody fully understands each testing tour and which test approach and focus it represents. All tours are described in detail in James Whittaker's book "Exploratory Software Tours". But most teams do not have the time to go through the book so here is a cheatsheet which describes the core idea of each tour and gives a simple example. Be aware that tours are not designed for manual testing only, on the contrary, they offer great inspiration for developers who are writing automated tests of any scale, as well.

BUSINESS DISTRICT

Guidebook Tour (F1 Tour)

Idea: Use the product documentation as a guide through the application. Re-use HowTo's and tutorials. Try use cases/details described in educational documents (BC400/401, ...). After the tests the documentation should not contain any mistakes and all features should work exactly as documented.

Money Tour

Idea: The money tour tests the features which were the main reasons that the customer bought the software in the first place. Usually these are the features which salespersons show when they try to sell the product. Let a salesperson do a product demo for you in order to find out what "money" features your application has.

Landmark Tour

Idea: Pick the most important features of your application and execute them **in different orders**. The landmark tour is not meant to be just testing the main functionality. This can and should be done by automated test. What an automated test cannot do is to put many features into all possible sequence orders. The goal of the landmark tour is thereby to test that the main features of an application work together in different orders.

Intellectual Tour

Idea: The intellectual tour is used to ask the software really **hard questions**. James Whittaker gives some questions as guidance for the intellectual tour: "How do we make the software work as hard as possible? Which features will stretch it to its limits? What inputs and data will cause it to perform the most processing? Which inputs might fool its error-checking routines? Which inputs and internal data will stress its capability to produce any specific output?"

Fedex Tour

Idea: The FedEx tour is used to identify and **follow specific pieces of data** through the various part of the software which you are testing. The testers check that all parts properly interact with the data.

After-Hours Tour

Idea: Most applications perform tasks even when the **user is not interacting** with them. Backup jobs are running, data gets archived, clients might poll a backend server. The idea of the after-hours tour is to check the application for such tasks and make sure that these pieces of code which might not be directly connected to a feature are also getting tested.

Garbage Collector‘s Tour

Idea: Garbage-men go street by street, house by house. They stay only a few moments but they crisscross the neighborhood in a methodical manner. For software testing this means to **spot-check each and every capability/feature/module**/... within a defined scope.

TOURIST DISTRICT

C*ollector's Tour*

Idea: Similar to a tourist traveling across the country collecting post cards from each state/large city, the idea behind the collectors tour is that a tester tests a specific feature by **collecting all outputs possible.** The focus of this tour is **completeness.**

Lonely Businessman Tour

Idea: Businessman travel a lot. The focus of this tour is to **"travel" as far through the application as possible**. Choose long paths over short paths. Which paths through the application need the most clicks, require the most screens?

Supermodel Tour

Idea: The supermodel tour tests **only the UI and not the logic behind it.** The tester picks a feature and makes sure it follows the UI guidelines, all icons are correct, no texts contain spelling mistakes, etc. In order to make it easier to check whether features follow the UI guidelines, it is a good idea to create short checklists together with the responsible UX person.

TOGOF Test one get one free Tour

Idea: The TOGOF tour focuses on testing what happens if **two or more instances of the same application are running simultaneously**. Start multiple instances, put them through the paces by

causing each application to manipulate the memory and the disk. Come up with scenarios were two or more instances try to manipulate the same resource or let them all do the same thing at once.

Scottish Pub Tour

Idea: Large cities have a large amount of pubs. Some are busy all the time some or not. In order to find the busy, popular ones you usually need a guide to tell you where to look. Transfer this idea to software products. Large and complex products have a lot of features. However, testers usually have a hard time finding the ones which are used a lot. The idea behind the tour is that the **testers actively try to find these features by talking with single user, user groups** in order to find features which might have already not been tested.

SEEDY DISTRICT

Saboteur Tour

Idea: Test the software in the **most destructive way possible**: undermine the application when/how ever possible while focusing on the resources which the application needs, remove/restrict resources (block file access, restrict available memory/disk space), change environmental conditions (disconnect network connections,...). Focus only on needed resources.

Antisocial Tour

Idea: Always do the opposite of what the others are doing. Use **invalid inputs** instead of valid ones. Execute features in the **wrong order.** Always think about the **opposites** e.g. if valid input is a number enter a character, if valid input is a picture file use a word file and so on. Testers should expect to see the a lot of error messages when they perform this tour.

Obsessive Compulsive Tour

Idea: Repeat every step of a scenario twice or as often as you want. Especially steps which manipulate data usually contain bugs which appear after lots of repetition. Same for error messages. Error handling code usually does not get as much attention from developers than other code. If you provoked an error message try to replicate the same error again and again.

HISTORICAL DISTRICT

Bad Neighborhood Tour

Idea: All software products have **parts which contain a lot of bugs.** Of course, you do not know which parts this will be in advance. However, after testing for some time you should be able to compile a list with locations were these "bad neighborhoods" are located. Now, you can specifically focus on those areas and run tours such as the garbage collector tour to make it bug free, again.

Museum Tour

Idea: Large long running projects usually have tons of **legacy code**. The idea behind the museum tour is to have a look at these pieces of code. First find out which code **has not been changed for a long time** by running the necessary reports then use those information to test the features which execute this code. Even if your project only contains brand new code, make sure to check whether it interacts with older code via APIs etc. and make sure the interaction does not lead to bugs.

Prior Version Tour

Idea: Before a new product version gets released make sure that the all **test cases for the old version also are getting run** for the new version. Verify that no functionality got lost in the new version.

ENTERTAINMENT DISTRICT

Supporting Actor Tour

Idea: DO NOT exactly exercise the features which are in scope of the test case. Instead **test the nearest neighbor.** For example if the test case tells you to use the log in language German use English instead. If you should create a user with a German Address create one with an English address. This variation makes sure that test cases which have successfully been executed for a long time suddenly might find bugs again. It is a cheap way to introduce variations into old test cases.

Back-alley Tour

Idea: The back alley tour is run **for features which users hardly use during their daily work**. This does not mean that these features are not important, quite the contrary, for example hopefully users never have to use features like data recovery. However, if they need them they have to work properly. Other examples for rarely used features are preference dialogs or features which only appear when the user starts the application for the first time.

All-Nighter Tour

Idea: A lot of bugs only happen if the software has been **running for a long time**, for example such bugs could be memory leaks. The task of the All-nighter tour is to simulate this scenario. In order to simulate long running applications, etc it is often necessary to build some sort of test infrastructure and get computers which are not turned off overnight.

HOTEL DISTRICT

Rained-Out Tour

Idea: In this tour you **use the cancel button A LOT**! Execute every kind of long running job, process and make sure that it is possible to cancel it. Examples for such scenarios are file transfers, batch jobs, complicated searches, etc. Start such features and use the cancel button or hit the Escape key.

Coach Potato Tour

Idea: Persons who are sitting on the coach, watching TV, eating potato chips are usually called coach potatoes. The idea behind this tour is to pick a feature and try **to test it with as little work as possible.** Some examples: test a user form and only enter the data which is absolutely necessary, navigate the application by only using the mouse. One idea of this tour is to ensure that the **default cases are getting tested**.

Source: Exploratory Testing Tours Cheatsheet Posted by Thomas Alexander Ritter on Oct 18, 2010 3:52:17 PM