

[2:28:13 PM] Pradeep Soundararajan: WTA-24 - Notes of Pradeep Soundararajan

Mission:

- \* In today's mission we would like you to frame your exploration on your own.
- \* We are about to give you a 'black box machine'.
- \* You will have 60 minutes.
- \* At the end of the timebox present your notes (no more than page long) to the group chat.
- \* What to report and in what format - we leave it up to your decision.
- \* We hope everybody will bring a unique bit of experience we all can learn from.

Thoughts:

I opened the link provided to find it was James Lyndsay's one of the puzzles. As I have worked on such, I decide not to do it the "typical" way of trying to understand

how the system works. Instead, I am going to attempt to make a checklist of heuristics that may be needed to crack such puzzles. I could also call it the meta info on

how to learn black box machines. So, here I go.

Checklist of learning black box machines

- \* Try OFAT (One Factor at a time) and learn one bit at a time.
- \* You are fine to make inferences and conjectures but do tests that refute it.
- \* Don't be carried away by what is shown on the UI or the animation.
- \* Reading code is a faster way to understand the blackbox.
- \* Identify parts of the black box you are already familiar with.
- \* Make a list of tests you have run.
- \* In free style exploratory (testing)learning mode, if you don't have a track of what you are doing, you may end up repeating tests that is probably not necessary.
- \* The best way to fool yourself is to tell, "Oh yeah, now I know how it works".
- \* Record a video/screencast of what you did and how the product responded to it.
- \* A specific behavior can be represented in many different ways.
- \* Patterns can be "obvious" and "dancing in front of you" or "hidden" and "shying away from you"

- \* You may need a combo of Logical, Analytical, Lateral and Critical thinking to make good progress.
- \* While you are clicking on something, observe what happens around the entire screen.
- \* Clues maybe hiding in places where it is not so obvious for people to click or initiate an action.
- \* Make a note of colors & icons used and relate it to what you already know of those colors.
- \* If you are stuck it is because you have ran out of ideas. Try reading some stuff that can help you generate ideas.
- \* When there is no specific information available to you, make as many assumptions as you want to but also list them.
- \* It is a good idea to pair up but don't let that distract you from achieving the goal.
- \* It could be a good idea to first come out with our own investigation before pairing up / sharing notes with others.
- \* Keep your tests under control.
- \* An object with a lot of action could be intentionally put in there to distract you from not seeing something important.
- \* Use tools for improving your productivity. For example, you could use excel to maintain a tabular column of possible test combinations and its result as your

experiment.

- \* When there is no unit mentioned, feel free to come out with your own (or the standard unit you intend to follow) and explain it up front.
- \* Do you know when to stop? Is it only when the time runs out?
- \* Instead of spending too much time trying to figure out what it means, check if the stakeholders are available for a chat and shoot your questions.
- \* Explain your conjectures to somebody and ask them to check if they find any faults with it.
- \* Use tools to disassemble code of the product from the application itself
- \* Be conscious of your thought process and make notes.
- \* Do you vary your experiments? Think equivalence classes and ask how different was your previous test to the current one?
- \* When the context is abstract, quote the oracle you used to support the claim of having found a problem.
- \* Model/Draw the product as you test and learn about it.
- \* Some products come with a warning to not be re-engineered. Check if your product contains any of it directly or indirectly.
- \* Prove and disprove your theories
- \* When you don't have a mechanism to identify if you are learning has been good, it is likely that you may be confused.
- \* Making a list of what you could learn from it (although it doesn't appear to teach you directly) could be wise

- \* Look at the meta info than just the data that dances on the GUI
- \* Any tests done without understanding the purpose of why the product was built makes you go far away from solving the problem.
- \* There are quick and unplanned tests, planned and carefully timed tests and there are tests that require some setup and planning.
- \* Create filters to your report to see patterns
- \* <and more>

\_ End of time \_