

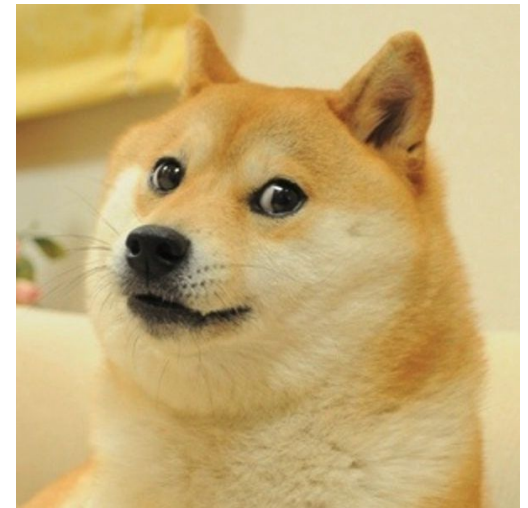
# Exploratory testing benefits

---

Diana Pinchuk  
QA @ [GetSocial.im](#)



- Tested a lot of stuff
- [GDG Lviv](#) & [GDG DevFest Ukraine](#)  
co-organizer
- ISTQB Advanced holder /(. . )
- Big fan of tech podcasts and blogs
- Love memes



# Agenda

- What is exploratory testing (ET)
- Why do we need ET
- Helpful tools
- Example of usage
- Q&A

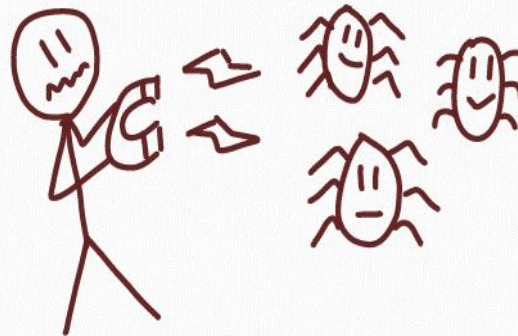
# Exploratory testing

---

Я нажимаю на  
кнопочки и ссылки

Я просто тыкаю куда  
попало и пытаюсь  
сломать приложение

Зачем что-то  
документировать?



Баги - это моя цель

Я магнит для багов!

Давайте быстренько все  
проверим перед релизом



# Exploratory testing is NOT

*Exploratory testing can be done in an unskilled, slapdash, silly way. Just as an unskilled driver behind the wheel of a car is still a driver who is driving a car, a poor tester can still be doing ET– albeit probably not very well.*

James Bach

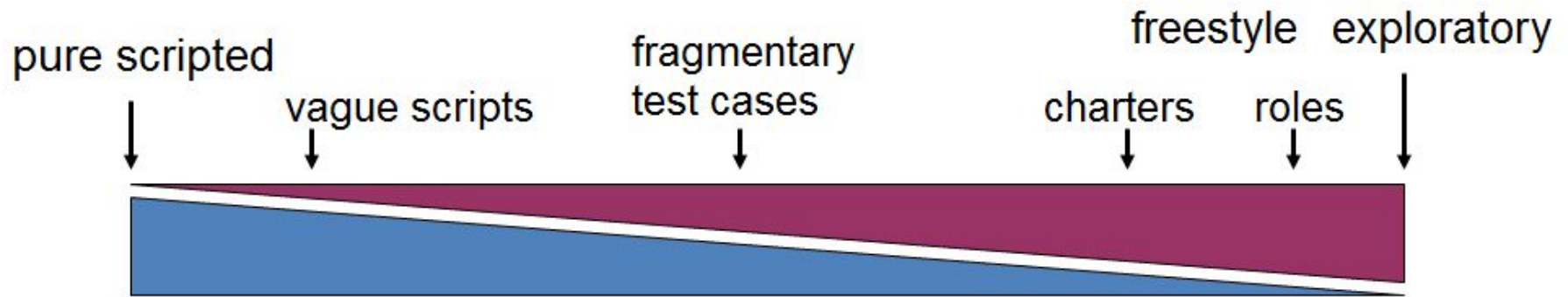




# Exploratory testing is...

ET is an **experience**-based testing technique and approach

A simple way to think of ET is **concurrent test design and test execution**.



# Exploratory testing 3.0 (2011-nowadays)

We now define all testing as exploratory

Any testing is exploratory to the extent that **the tester actively controls** the design of the tests as those tests are performed and uses information gained while testing to design new and better tests.

James Bach's [ET 3.0](#) & [translation](#)





Exploratory testing => constant learning

---

# Why do we need ET?

- Provide rapid feedback on a new product or feature
- Find important problems quickly
- Hit upon the right tests at the right time
- Immediately incorporate new ideas into the tests
- Useful in complex testing situations, when little is known about the product, or as part of preparing a set of scripted tests

What to start from?

---

# Краткое пособие для начинающих: Начните



# What to start from

- Use any software product during a short test session
- Note your observations
- Change the perspective
- Start the cycle again



# Helpful stuff

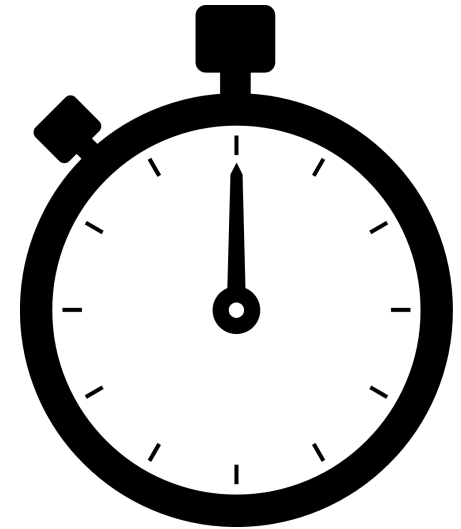
- SBTM
- Heuristics
- Mnemonics
- Mind maps ([example](#))
- Testing tours ([example](#))
- Weekend Testing Sessions





# Session-based test management (SBTM)

- Session lasts 30-120 mins
- Use of charters and reports in the end
- List of bugs found during the session
- Any interesting stuff is noted
- Each session is debriefed



SBTM <http://www.satisfice.com/articles/sbtm.pdf>

And a great mind maps about SBTM <http://bit.ly/2GELOdU> and <http://bit.ly/2qdWCp1>

# Test charters

Describe mission statement and areas to be tested

<b>Charter</b>	Analyze ordering functionality in online taxi platform
<b>Areas</b>	
OS	iOS 10.x, iOS 11.x; Android 6, Android 7, Android 8
Menu	Ordering view
Strategy	Functional testing

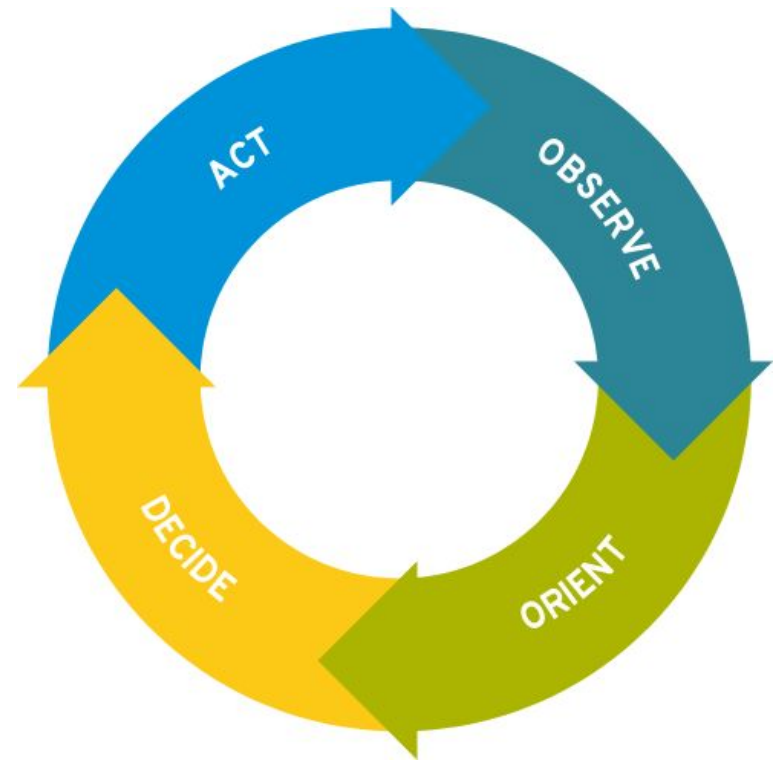
One more nice article about SBTM and Lean testing <http://bit.ly/2Eu4CGU>

# Heuristics

Heuristics testing is the testing of algorithms, code modules or other kinds of projects where **testing strategies rely on past data about probabilities**

For example, OODA loop:

- Observe
- Orient
- Decide
- Act



[Heuristic Test Strategy Model](#)

# Use checklists and cheat sheets

You are not done yet

<http://www.thebraidyttester.com/downloads/YouAreNotDoneYet.pdf>

Checklist base

<https://checkvist.com/checklists/476089-checklists-base>

OWASP cheat sheets

[https://www.owasp.org/index.php/OWASP\\_Cheat\\_Sheet\\_Series](https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series)

Test Heuristics Cheat Sheet

<http://testobsessed.com/wp-content/uploads/2011/04/testtheuristicscheatsheetv1.pdf>

# Mnemonics - TAP IT UP

**T:** Tasks the user wants to complete with our app

**A:** Application composition and features

**P:** Platforms we will need to test

**I:** Inputs and Outputs, such as gestures, files, data feeds

**T:** Time and Timing and its effect on testing

**U:** Usage and typical user interaction

**P:** Places where the app is used

### Why

You've discovered a problem (you might call it a bug). People need to know about it to discuss if it's important enough to fix and stop users from discovering it.

### How

Capture a one line summary. Describe how you discovered it. Include screenshots, videos, logs, scribbles, highlights and more!

## PROBLEMS



### Why

You'd like to clarify something you're unsure about.

### How

Ask a question. Avoid overwhelming recipients – keep it to one question per topic.

### Example

"I'm a little unsure how to force an error.  
How might I do that?"

## QUESTIONS



## PQIP

## Capture and Collaborate

[source](#)

## IDEAS



### Why

Your exploration has sparked an idea. There's value in sharing! Feel good contributing to your product.

### How

Share your idea in a way that invites people to comment.

### Example

"I like the save feature. Perhaps we can use a "save icon" instead of the word "Save". This feels consistent with other parts of the app. What do you think?"

## PRAISE



### Why

Let others know that you value their work.  
You'll both feel great!

### How

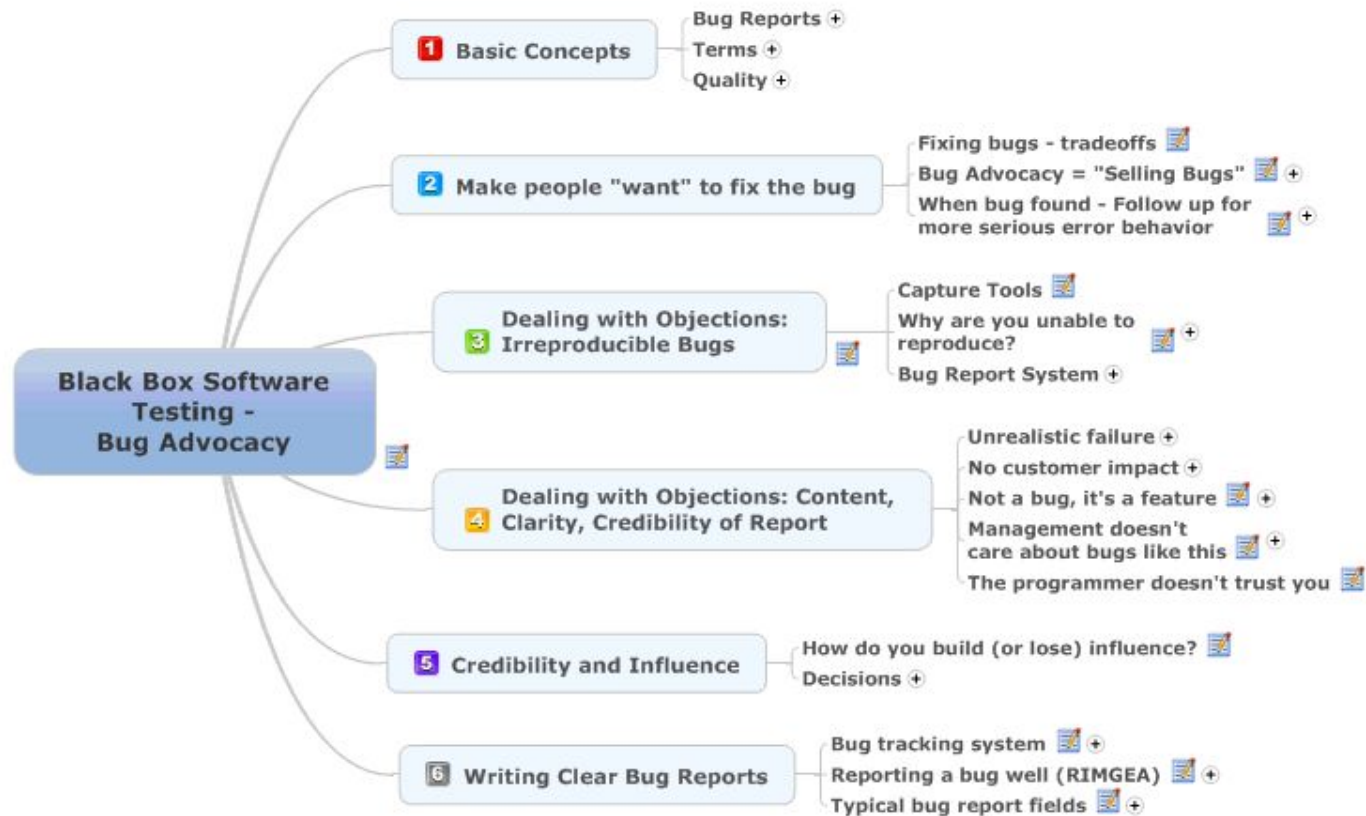
Describe what you've enjoyed exploring.

### Example

"This new feature is awesome! I really like how you've paid attention to email validation. :-)"



# Mind maps



A [great collection](#) of the ET mind maps

# Testing tours

---

# The tourist metaphor

You're visiting your App for the first time

- You can go it alone
- You can buy a guidebook
- You can hire a guide
- You can take the bus tour



The main point - **FOCUS**

James A. Whittaker "Exploratory Software Testing" <http://bit.ly/2qcMjle>

# The Guidebook tour

- Follow the possibilities to a 'best of' list
- Use User manual, follow its advice
- Online help (the F1 tour)
- Competing products (the competitor's tour)



# The FedEx Tour

- Concentrate on data moving through the software.
- Try to identify inputs that are stored and “follow” them around the software.
- Try to find every feature that touches the data so that, just as FedEx handles their packages, you are involved in every stage of the data’s life cycle.



# The All-Nighter Tour (Clubbing Tour)

- Software should stay out all night
- Keep it working, never shut it down
- Keep files open without saving or closing them

This tests timeout functionality, finds memory problems and leaks that are masked by shutdown-restart





# The Sabotage tour

- Start doing some process
- Define resources required for the process
- Delete/limit system access for the resources
- Repeat scenario



# Tools

---

# Helpful tools

- ET extension for Chrome <http://bit.ly/2qdk0M>
- Test & feedback extension for VS <http://bit.ly/2uZmZnK>
- Timers and recorders
- Block any distractors (e.g. via [GFW](#) extension)
- Google Sheets



Example of usage

---

# ET session of a mobile app

## Test breakdown

Test design and execution, %	90
Bug investigation and reporting, %	10
Session setup, %	0
Charter vs opportunity, %	80/20

## Testing notes:

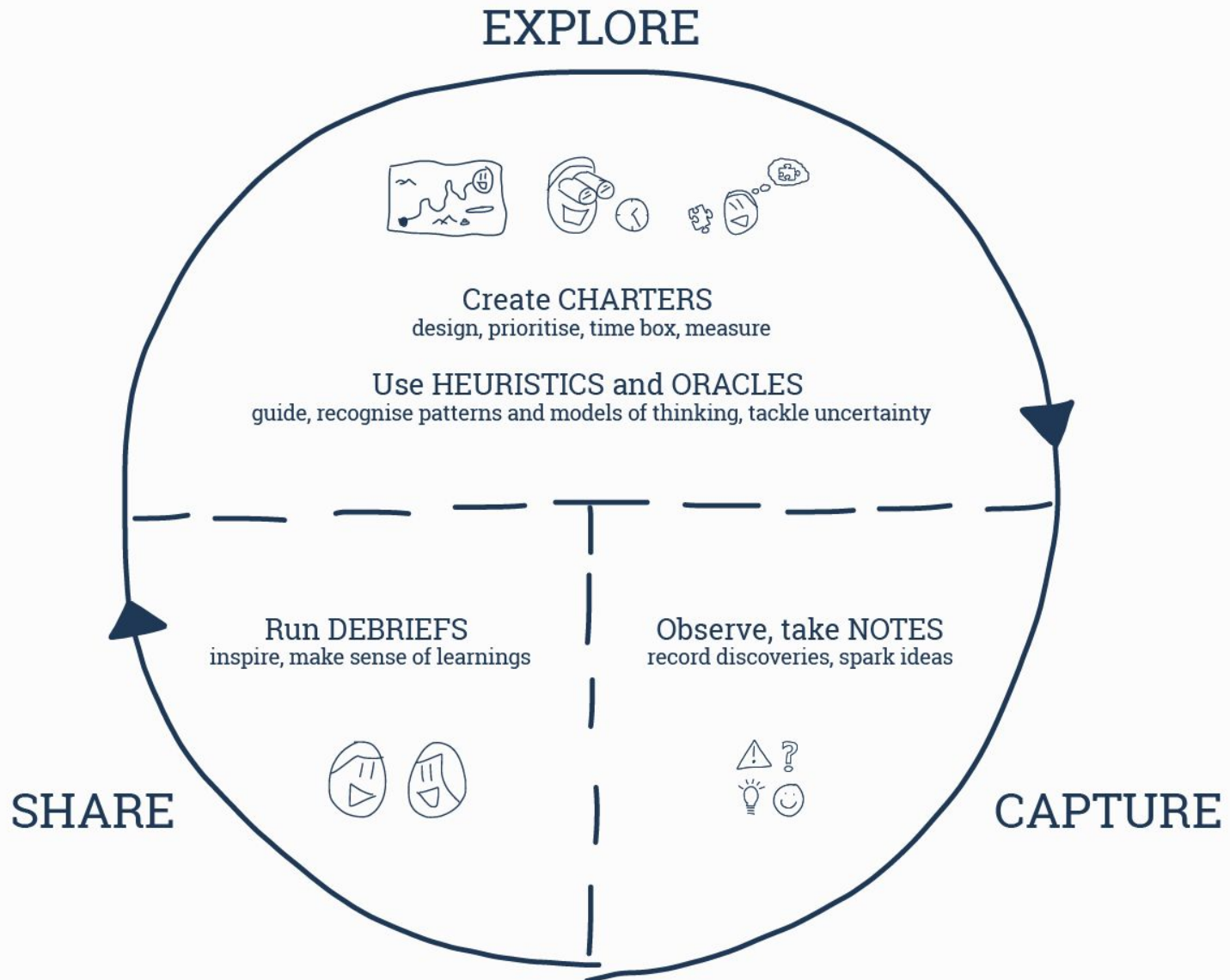
The session was performed with purpose to investigate the functionality of the taxi ordering in the mobile app "Example App".

### Features:

- Make an instant order
- Make a postponed order
- Cancel the order
- Select a pick up point
- Select a finish point
- Select intermediate points

Full example is [here](#)

# THE “TESTING IS LEARNING” LOOP





# Conclusions




- Exploratory testing is a powerful approach
  - But it's not a silver bullet
  - Don't be afraid to try something new
  - Always remember about continuous improvement
-

# Thank you!

It's time for questions :)

---

[pinchuk.diana@gmail.com](mailto:pinchuk.diana@gmail.com)

 [@diana\\_pinchuk](https://twitter.com/diana_pinchuk)