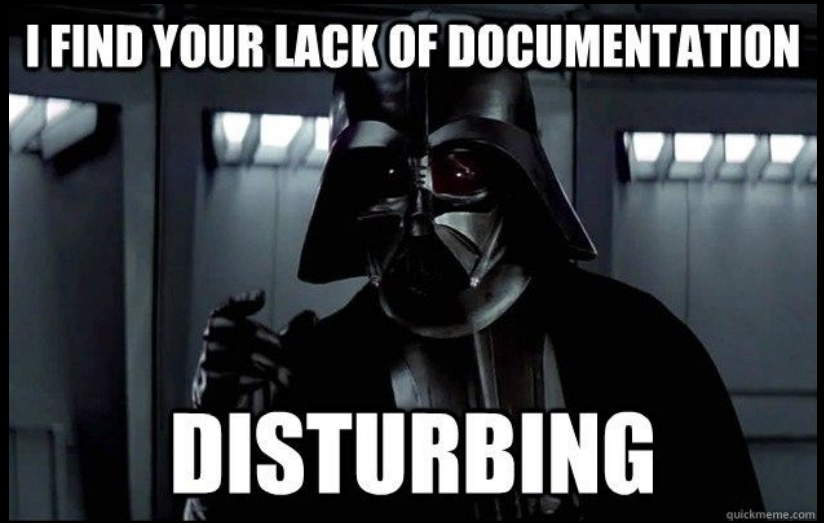
Documenting ET



Lector: Viktor Slavchev
E-mail: mr.slavchev@gmail.com
Twitter: @TheTestingTroll
Blog: mrslavchev.com

www.pragmatic.bg

PRAGMATIC

Test notes

We need a way to keep track of the findings we make.

- Test charter
- Document our strategy
- Document the methods we use
- Heuristics
- Personas
- Observations
- Bugs
- Anything suspicious

General advice TN

- As in bugs, keep them concise, but comprehensive
- Think in perspective - 1 day/1 week/1 year
- What would you like to know if you were staring to test this? Write it
- Use examples.
- Write findings, results, conclusions.
- Ask for a review from a colleague.

Elements: Test charter

- Meant to set the goals of your testing session
- Must be as inclusive as possible
- To invoke exploration and experimentation
- Short
- Meaningful
- Transparent

How I failed...

- Normally written at the end of the session.
- Answers the question - “what has not been tested”.
- Might be used as a “to do” list for next sessions.
- Might contain goals that were not visible in first place.

Strategy

The essential part of your testing that you want to record. Make it a complete story

- What you tested? How? What data have you used?
- What variables have you manipulated
- What experiments?
- Examples?
- Data?
- Discoveries? Conclusions.

Notes

- Anything that you believe might have value
- Anything suspicious
- Not necessarily a bugs
- Questionable behavior
- Improvements

Bugs

Logging a bug properly takes 5-10 mins. It’s a distraction we don’t want.

- Mark bugs in telegram style
- Later you can add links to these
- You can use them if you want to report what bugs you logged.

Problem:

Task:

You are asked by your manager to provide detailed report (written or verbal) of “what you tested”.
As bullet points list what you think have to be included there?

Time to complete: 10 mins

We suck at storytelling

What stories do we like

While walking along the street one day in London I happened to look down and notice a gold and silver band. I picked it up and noticed that it had very small curvy writing engraved on it. I couldn't read what it said but it was clearly not english. I put it in my pocket and carried on quite happy with my find. A couple of days later I asked my cousin's husband if he knew what language the writing was in. Here I am thinking that I've found some neat ring with maybe Arabic (or something to that effect) writing on it. He looked at me and started laughing. It turns out that my possibly expensive ring was a mass-produced freebie children's gift that you'd find in a Macdonalds happy meal! It was 'the ring' from The Lord of the Rings!!! Good grief - only me!!!

What stories do we tell as testers

- I was in London
- Saw a bridge, a castle, 3 churches
- It was raining
- I walked

The testing story

Taken from: Braiding The Stories

3 stories:

1. **The product story** - qualitative report, how the product work, fails or might work.
2. **The testing story** - what and how we tested - coverage, oracles, data, examples
3. **The quality of testing story** - priorities, tested vs. non tested. Cost and risks of testing.

Testing coverage

According to M.Bolton:

“Coverage is the extent to which we’ve tested the program; it’s about where we’ve looked and how we’ve looked, and it’s also about what’s uncovered—where we might not have looked yet, and where we don’t intend to look.”

Common misattribution

Very often test coverage is measured **wrong** by quantitative metrics such as:

- Code coverage
- Statement coverage
- Branch coverage

Cem Kaner writes about 101 types of coverage.

What's worth telling?

Instead, it makes sense to think of:

- What is there to test? Components, modules
- Structural vs. non-structural testing
- Deep vs. shallow
- Playing vs. exploring
- Variables
- Personas
- Contexts

Product coverage outline- PCO

- We can think of the PCO as a model, a map or inventory of the parts of the system that need testing. Look here.
- Might be a list, a mindmap, a flowchart, anything
- Helps is to define different functionalities within the application.
- Helps us split complex system into smaller bits - answer a series of easy questions, rather than a complex question right away.
- Small components are easier to define and experiment with.
- **IMPORTANT!** PCO focuses on visible part, but also invisible from the UI like API calls for example.

SFDIPOT

Heuristic aiming to help solving the problem of decomposition, understanding and documenting a complex system.

Mindmap here: SFDIPOT mindmap

SFDIPOT

- How to make sure that our test plan or product coverage is complete and we are not missing out important areas to be tested?
- James Bach and Michael Bolton authored a document called: Heuristic test strategy model where they offer check list of idea generators.
- It's called SFDIPOT, easily remembered as San Francisco depot.
- It's a mnemonic code.
- It aims to provide us with checks for visible and invisible variations during our testing.

Structure

attributions go to RST by James Bach and Michael Bolton

- **Structure. Everything that comprises the physical product.**
  Code: the code structures that comprise the product, from executables to individual routines.
  Hardware: any hardware component that is integral to the product.
  Non-executable files: any files other than multimedia or programs, like text files, sample data, or help files.
  Collateral: anything beyond software and hardware that is also part of the product, such as paper documents, web links and content, packaging, license agreements, etc

Function

attributions goes to RST by James Bach and Michael Bolton

- **Function. Everything that the product does.**

Application: any function that defines or distinguishes the product or fulfills core requirements.
Calculation: any arithmetic function or arithmetic operations embedded in other functions.
Time-related: time-out settings; daily or month-end reports; nightly batch jobs; time zones; business holidays; interest calculations; terms and warranty periods; chronograph functions.
Transformations: functions that modify or transform something (e.g. setting fonts, inserting clip art, withdrawing money from account).
Startup/Shutdown: each method and interface for invocation and initialization as well as exiting the product.
etc

Data

attributions go to RST by James Bach and Michael Bolton

- **Data. Everything that the product processes.**

Input: any data that is processed by the product.
Output: any data that results from processing by the product.
Preset: any data that is supplied as part of the product, or otherwise built into it, such as prefabricated databases, default values, etc.
Persistent: any data that is stored internally and expected to persist over multiple operations. This includes modes or states of the product, such as options settings, view modes, contents of documents, etc.
Sequences/Combinations: any ordering or permutation of data, e.g. word order, sorted vs. unsorted data, order of tests.
Cardinality: Numbers of objects or fields may vary (e.g. zero, one, many, max, open limit). Some may have to be unique (e.g. database keys).
Big/Little: variations in the size and aggregation of data.

Side note: Interface

Definition (wikipedia):

In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans and combinations of these. Some computer hardware devices, such as a touchscreen, can both send and receive data through the interface, while others such as a mouse or microphone may only provide an interface to send data to a given system.

Interfaces

attributions go to RST by James Bach and Michael Bolton

- **Interfaces. Every conduit by which the product is accessed or expressed.**
  User Interfaces: any element that mediates the exchange of data with the user (e.g. displays, buttons, fields, whether physical or virtual).
  System Interfaces: any interface with something other than a user, such as other programs, hard disk, network, etc. **API(Application program interface)** Any programmatic interfaces or tools intended to allow the development of new applications using this product.
  Import/export: any functions that package data for use by a different product, or interpret data from a different product.

Platform

attributions go to RST by James Bach and Michael Bolton

- **Platform. Everything on which the product depends (and that is outside your project).**
  External Hardware: hardware components and configurations that are not part of the shipping product, but are required (or optional) in order for the product to work: systems, servers, memory, keyboards, the Cloud.
  External Software: software components and configurations that are not a part of the shipping product, but are required (or optional) in order for the product to work: operating systems, concurrently executing applications, drivers, fonts, etc.
  Internal Components: libraries and other components that are embedded in your product but are produced outside your project.

Operations

attributions go to RST by James Bach and Michael Bolton

- **Operations. How the product will be used.**
  Users: the attributes of the various kinds of users.
  Environment: the physical environment in which the product operates, including such elements as noise, light, and distractions.
  Common Use: patterns and sequences of input that the product will typically encounter. This varies by user.
  Disfavored Use: patterns of input produced by ignorant, mistaken, careless or malicious use.
  Extreme Use: challenging patterns and sequences of input that are consistent with the intended use of the product.

Time

attributions go to RST by James Bach and Michael Bolton

- **Time. Any relationship between the product and time.**
  Input/Output: when input is provided, when output created, and any timing relationships (delays, intervals, etc.) among them.
  Fast/Slow: testing with “fast” or “slow” input; fastest and slowest; combinations of fast and slow.
  Changing Rates: speeding up and slowing down (spikes, bursts, hangs, bottlenecks, interruptions).
  Concurrency: more than one thing happening at once (multi-user, time-sharing, threads, and semaphores, shared data).

Practical task

App:http://www.shino.de/parkcalc/index.php
Map: https://goo.gl/XDNroZ