FREE eBook

LEARNING testing

Free unaffiliated eBook created from
Stack Overflow contributors.

#testing

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: testing

It is an unofficial and free testing ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official testing.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with testing

Remarks

This section provides an overview of what testing is, and why a developer might want to use it.

It should also mention any large subjects within testing, and link out to the related topics. Since the Documentation for testing is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting testing set up or installed.

Test case

A test case or test script is a more or less formal description of the actions needed to prove that a requirement is met. It has a descriptive title, may start off with some preparation and ends with an expected result. Usually the test cases for a *system under test* are organised into *test suites*.

Log on to system

Step	Result
Start system	Startup screen shows, logon asked
Type username	username shows
Type password	'*' characters show the number of characters entered
Press Enter	Main screen shows

Test case

TEST CASE –

A test case is a set of conditions and steps that when followed/applied, a tester can determine whether a feature, an application or a software system is working as originally designed.

A test case usually contains References to the original task, Pre-conditions (if the pre-conditions are not met, testing can not continue), Steps (steps describe what a tester needs to do in order to get a result), Expected Result (what should happen after following all the described steps) but may also contain Estimation, Priority, Environment on which to perform test.

Example of a test case :-

Title – Perform a login action on mail.google.com with an invalid password

Preconditions:

1. Access to supported browsers: IE 11, Firefox, Chrome.
2. Internet connection
3. A valid gmail account

Test steps :

1. Launch the URL "mail.google.com" on one of the supported browsers
2. Enter your email address in the email field
3. Click the NEXT button.
4. Enter an incorrect password in the Password field
5. Click the Sign-In button.

Expected Result – A error message will be displayed under the Password field with red letters stating "Wrong password. Try again"

Read Getting started with testing online: https://riptutorial.com/testing/topic/4880/getting-started-with-testing

Chapter 2: Software Testing Techniques - Boundary Value Analysis

Remarks

This technique should be used whenever you have boundaries defined into a spec. It is a great idea to apply it to any rule based on time, values, any kind of counting or scale to be triggered.

It also ensure and helps finding n + 1 errors. And yes, it is an expansion from Equivalence Partition concepts. Your decision to apply this or the other should be the boundaries existing and being clearly defined.

Using the EP

```
      June    |           July          |    August
... 28 29 30 | 01 02 03 ... 29 30 31 | 01 02 03 ...
    ^                      ^                    ^
```

Using the BVA

```
      June    |           July          |    August
... 28 29 30 | 01 02 03 ... 29 30 31 | 01 02 03 ...
           ^ | ^                   ^ | ^
```

Examples

Time-boxed offers

The rule

An e-commerce have a 20% off offer valid for entire month of July.

How to apply the technique

Using the Equivalence Partition technique we could divide this as

- Inside July
- Is outside July

But we can do a little better with this case. We can use the boundaries of the partition to ensure we are covering the set as follows:

- June 30th (Out)
- July 1st (In)
- July 31st (In)
- August 1st (Out)

Using this specific inputs into our test cases design, we are going to ensure that the offer is valid inside July, and not outside it.

Progressive discount

The rule

Another common case we usually find and where it is useful to apply this technique are progressive rates based on values.

An e-commerce has an offer that is like:

- 10% off for orders above US$ 80.00
- 15% off for orders above US$ 150.00
- 25% off for orders above US$ 200.00

How to apply the technique

Rule of thumb: we always will pick the two sides of each boundary. For this case we will need to test with orders with the following values:

- US$ 79.99 (no discount)
- US$ 80.00 (10% off)
- US$ 149.99 (10% off)
- US$ 150.00 (15% off)
- US$ 199.99 (15% off)
- US$ 200.00 (25% off)

This is enough to cover all intervals with a great confidence level.

Read Software Testing Techniques - Boundary Value Analysis online:
https://riptutorial.com/testing/topic/10817/software-testing-techniques---boundary-value-analysis

Chapter 3: Software Testing Techniques - Equivalence Partition

Remarks

This technique divides input data into data classes to reduce test cases amount to validate a rule. The idea is that given a set of possible equivalent values, using just one of those values will be enough to design a test case.

An advantage of this approach is reduction in the time required for testing a software due to lesser number of test cases.

It strives to find errors that may arise based on information classes and reduce to the minimum the effort needed in terms of test case design as well as test data mass.

Examples

Login validation

The Rule:

The user will input credentials (email and password).

If credentials are valid, redirect to home page. Show an error message otherwise.

How to apply technique:

You can have a huge valid list of emails that could be used:

- admin@email.com
- foo@system.com
- bar@example.com
- ...

As well as a huge list of invalid emails

- xxx123
- foobar
- noreply@example.com
- ...

For passwords, you can have:

- a valid password for a specific and valid user
- non-valid password (one that does not exists on the system)

Equivalence Partition means that any element on a given individual set should be enough to design a test case.

So for represent a valid email we can choose **bar@example.com**. There is no need to design test cases using **admin@email.com** or another from the set of valid emails to ensure a good coverage.

For represent an invalid email we can choose **xxx123**

Same logic applies to password

Shipping fee based on zipcode

The Rule

An e-commerce have a shipping rule based on zipcodes. All shipping to California is eligible to free shipping. All other west cost are US$ 10.00 and The rest of USA is US$ 20.00

How to apply technique

We all know there are lots of zipcodes by state, and we are not going to use them all. For this case, we need:

1. One valid zipcode in California
2. One valid zipcode in West Coast, other than California
3. One valid zipcode in outside west coast
4. One invalid zipcode *

Thus, we reduce from thousands of zipcodes to only 4. A lot better now.

Despite it is not outlined in the rule, I have put the 4th one, the invalid zipcode.

This is because this is the **most important test**: testing the spec itself. In this case, thinking about dividing data into categories, we found a class that is not in the spec, and has no behaviour or expectations clearly set. I then include this class here and ask for partners/stakeholders/business analysts, what should be the behaviour for this case and then amend the spec or I can review and rip off this data class from my tests.

Read Software Testing Techniques - Equivalence Partition online:
https://riptutorial.com/testing/topic/10815/software-testing-techniques---equivalence-partition

Chapter 4: Software Testing Techniques - State Transition

Remarks

This technique should be used when you have any workflow in place, and should consider positive test cases (transitions that can happen), as well as negative test cases (transitions that are not allowed).

Any rule that can be described, thinked, scratched as a state transition diagram, workflow, lifecycle can have their test cases designed using this technique.

This technique can be also works to find completeness problems inside worflows and diagrams during documentation analysis.

HOT TIP

If a **state transition like rule** is provided as a series of statements instead of a table or graphic diagram, you can proceed as following:

1. Make yourself a state transition table as in the ticket system example
2. Add question marks for unclear transitions
3. Add an extra column at the end for broad discussion with business - raise questions and take notes

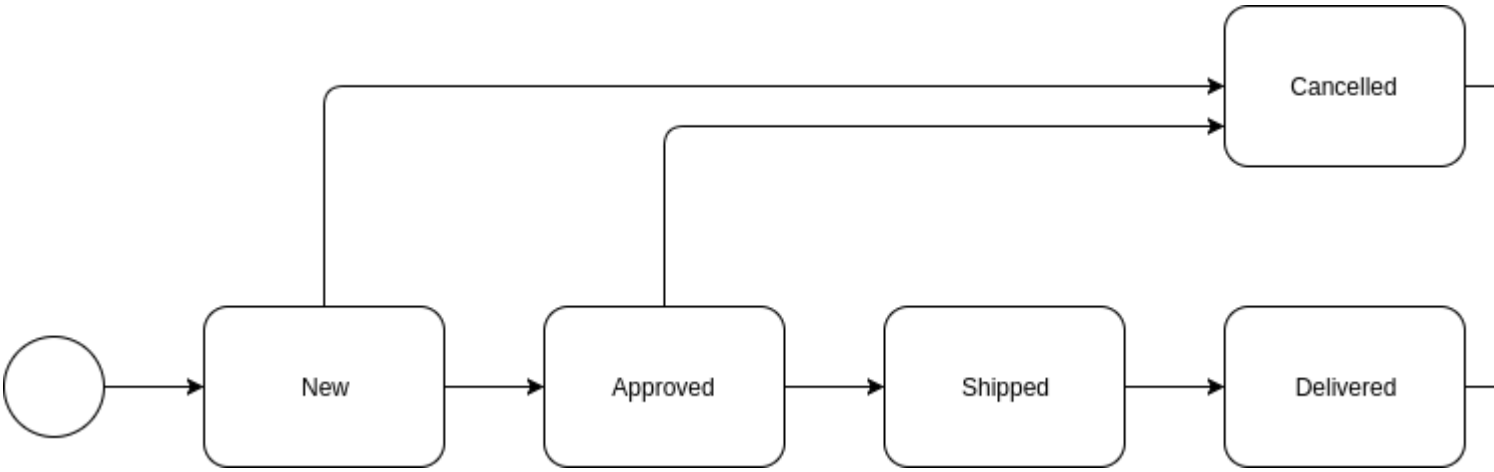
This is a way common situation we face on daily basis, and this practice can be handful to gain confidence on what is being developed and how it should be tested

Examples

Order phases

The rule

An order lifecycle at any e-commerce follows roughly the workflow:



How to apply the technique

From the diagram we see that it has the following allowed (positive) transitions:

- From New to Cancelled
- From New to Approved
- From Approved to Cancelled
- From Approved to Shipped
- From Shipped to Delivered

And all the other unmentioned transitions should be treated as invalid (negative) transtitions

- From New to Shipped (negative)
- From New to Delivered (negative)
- From Approved to New (negative)
- From Approved to Delivered (negative)
- From Shipped to New (negative)
- From Shipped to Approved (negative)
- From Shipped to Cancelled (negative)
- From Delivered to New (negative)
- From Delivered to Approved (negative)
- From Delivered to Shipped (negative)
- From Delivered to Cancelled (negative)
- From Cancelled to New (negative)
- From Cancelled to Approved (negative)
- From Cancelled to Shipped (negative)

- From Cancelled to Delivered (negative)

Ticket system

The rule

A ticket system have their valid transitions documented in the following table, where **O** represents allowed and **X** represents not allowed.

From \ To	Reported	Open	In Progress	In Review	Delivered	Rejected	Reopen
Reported	-	O	X	X	X	O	X
Open	X	-	O	X	X	O	X
In Progress	X	X	-	O	X	O	X
In Review	X	O	O	-	O	O	X
Delivered	X	X	X	X	-	X	O
Rejected	X	X	X	X	X	-	X
Reopen	X	X	O	X	X	X	-

How to apply technique

There is no much secret applying this technique to design test cases.

1. Each transition show be be represented by one test case
2. There is no transition from one state to itself (diagonal line with dots)
3. The number of test cases is always equals to ((n * n) - n)

For this case we have 7 states (n = 7). So we will have 42 test cases, as follows:

1. From Reported to Open (positive)
2. From Reported to In Progress (negative)
3. From Reported to In Review (negative)
4. From Reported to Delivered (negative)
5. From Reported to Rejected (positive)
6. From Reported to Reopen (negative)
7. From Open to Reported (negative)
8. From Open to In Progress (positive)
9. From Open to In Review (negative)
10. From Open to Delivered (negative)
11. From Open to Rejected (positive)
12. From Open to Reopen (negative)

13. From In Progress to Reported (negative)
14. From In Progress to Open (negative)
15. From In Progress to In Review (positive)
16. From In Progress to Delivered (negative)
17. From In Progress to Rejected (positive)
18. From In Progress to Reopen (negative)
19. From In Review to Reported (negative)
20. From In Review to Open (positive)
21. From In Review to In Progress (positive)
22. From In Review to Delivered (positive)
23. From In Review to Rejected (positive)
24. From In Review to Reopen (negative)
25. From Delivered to Reported (negative)
26. From Delivered to Open (negative)
27. From Delivered to In Progress (negative)
28. From Delivered to In Review (negative)
29. From Delivered to Rejected (negative)
30. From Delivered to Reopen (positive)
31. From Rejected to Reported (negative)
32. From Rejected to Open (negative)
33. From Rejected to In Progress (negative)
34. From Rejected to In Review (negative)
35. From Rejected to Delivered (negative)
36. From Rejected to Reopen (negative)
37. From Reopen to Reported (negative)
38. From Reopen to Open (negative)
39. From Reopen to In Progress (positive)
40. From Reopen to In Review (negative)
41. From Reopen to Delivered (negative)
42. From Reopen to Rejected (negative)

Chapter 5: UI Testing

Introduction

The focus of this topic is basically cover most important aspects of UI testing. This post would be mostly beneficial for freshers since they get assignments to work on and many times it so happens that few of the important points are missed. Let's help the freshers by contributing to this topic :)

Examples

[UI Testing] How to test a website?

Following points are to be kept in mind always:

1. Take the PRD (Product Requirement Document from Business Analyst/Project Manager). If they don't have it, insist on a documentation since that will be the bible for your testing
2. Once you get it, make sure to read it thoroughly.
3. You will get complete idea about what the system actually does and how the UI should be, in different cases.
4. Make sure to take into consideration the different OS / Browsers combos on which the system is expected to be tested. Ask PM or client (if needed) for this part, but let's be very clear about this in the early phase itself.
5. Once you are clear with the OS/Browser combinations, make sure you have it in your workplace. If not, start with procuring those combos.
6. Make sure the link is accessible and the code is deployed on appropriate environment (staging, pre-staging etc.)
7. Hit the link on the required browsers on required environements
8. Observe the speed at which the site loads on each OS/browser combo and make a note
9. Observe the images/text loaded correctly or not. Also make sure to have the correct font in place for all the browsers.
10. Explore through all the pages and make sure each and every thing is at its place. No distortion in UI should be observed.
11. (If the website is responsive (mobile friendly), make sure that it loads accurately when you resize the browser window or check it on larger screen resolutions.
12. (Partial functional testing) Make sure that all the buttons/links are clickable and are redirecting to the correct pages.
13. Also observe the functionality of the system as per the UI developed.

Read UI Testing online: https://riptutorial.com/testing/topic/9134/ui-testing

Credits

S. No	Chapters	Contributors
1	Getting started with testing	Bookeater, Community, Devmati Wadikar, edward.eas, Stephen Leppik
2	Software Testing Techniques - Boundary Value Analysis	Dave
3	Software Testing Techniques - Equivalence Partition	Dave
4	Software Testing Techniques - State Transition	Dave
5	UI Testing	talktokets