# Unit and Integration tests for Angular components. Part 2 out of 3.

Anastasia Dvorkina
May 18, 2018 · 3 min read

## How deep integration tests should be?

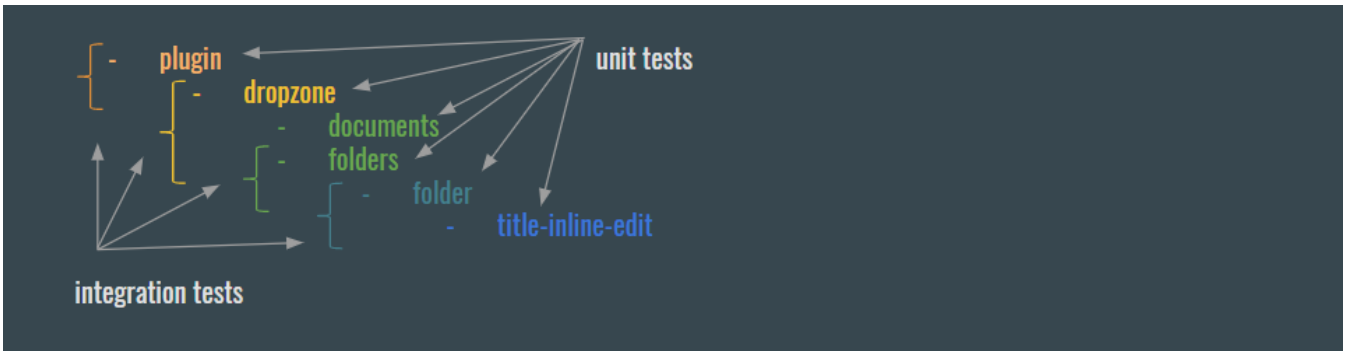
In my previous article I've pointed out the difference between tests for Angular components. During my research I've discovered a couple of interesting posts from Martin Fowler and Google Testing blog. See the links below.

As you may already know **integration tests** ensure that different units of the application are **communicating** with each other correctly. So you can imagine that one integration test may go through the whole application, testing more and more units working together in a perfect flow.

unit tests said it's working

But do we really want it? Think about software as a chain of events. Software would be good and robust when all the chains are good and robust, but it's too hard to examine the whole chain at once. That's why we split it and examine first every link (with unit tests) and later examine closest links connections (with integration tests).

How might it be applied to Angular? First do shallow tests for each Angular component and later add integration tests to examine component interaction with its first-level children. Let's take a look at the following components structure.




The dropzone template looks:

```
<span [class.isActive]="isActive()">{{title}}</span>
<folders [folders]="folders"
         [activeFolder]="activeFolder"
         (folderSelected)="folderSelected($event)">
</folders>
<documents [documents]="documents"
           (documentOpened)="documentOpened($event)">
</documents>
```

**dropzone.component.html** hosted with ❤ by **GitHub** **view raw**

Unit test configuration for dropzone component would be:

```
  beforeEach(
    async(() => {
      TestBed.configureTestingModule({
        providers: [SomeProvider],
        declarations: [DropzoneComponent],
        schemas: [CUSTOM_ELEMENTS_SCHEMA]
      }).compileComponents();
    })
  );
```

**dropzone.component.spec.ts** hosted with ❤ by **GitHub** **view raw**

So to achieve shallow unit test template you only declare tested component inside of the TestBed and nothing else. This means that all the elements in the template will be treated as simple DOM nodes, and only common directives (e.g., ngIf and ngFor) will be applied.

Integration test configuration:

```
1    beforeEach(
2      async(() => {
3        TestBed.configureTestingModule({
4          providers: [SomeProvider],
5          declarations: [DropzoneComponent, DocumentsComponent, FoldersComponent],
6          schemas: [CUSTOM_ELEMENTS_SCHEMA]
7        }).compileComponents();
8      })
9    );
```

**dropzone.component.ispec.ts** hosted with ❤ by **GitHub** **view raw**

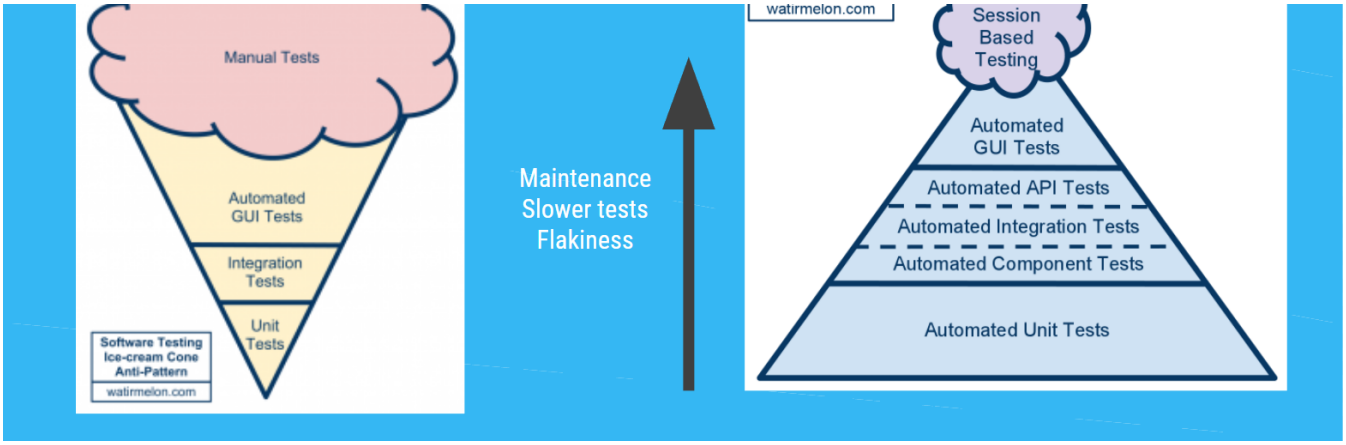
In unit tests all the possible component scenarios are examined:

- correct classes are applied to elements
- correct values are put into HTML nodes
- correct events are triggered on mouse events and etc.

in integration tests the points of integration are tested

- correct output events chain
- correct input values chain

Don't forget about the wisdom from Google and 70/20/10 rule. And remember that integration tests only examine the flow and don't try to test functionality which should be done by good old unit tests.




click on me!

Happy testing to you, my coding friends! And here are the articles for you from the smart guys.

**bliki: IntegrationTest**

test categories tags: Integration tests determine if independently developed units of software work...

martinfowler.com

**bliki: UnitTest**

test categories · extreme programming tags: Unit testing is often talked about in software...

martinfowler.com

**Just Say No to More End-to-End Tests**

by Mike Wacker At some point in your life, you can probably recall a movie that you and your friend...

testing.googleblog.com

Software Development Angular Testing Integration Testing Unit Testing