The testing pyramid

Maurício F. Aniche

M.F.Aniche@tudelft.nl
That’s what we have been calling Unit Testing.
Some definitions

• ISQTB: “Searches for defects in, and verifies the functioning of software items (e.g., modules, programs, objects, classes, etc) that are separately testable”.

• Osherove: “A unit test is an automated piece of code that invokes a unit of work in the system and then checks a single assumption about the behavior of that unit of work. [...] A unit of work is a single logical functional use case in the system that can be invoked by some public interface (in most cases). A unit of work can span a single method, a whole class or multiple classes working together to achieve one single logical purpose that can be verified.”
Advantages

• Very fast
• Easy to control
• Easy to write

Disadvantages

• Less real
• Some bugs can’t be reproduced at such level
When do we need more reality?
What can we do to gain reality?
We can start testing more pieces together

• Maybe test 2 or more classes together.
• Test the integration with the database.
• Test the web application via its UI.

It can be very challenging!
Let’s do an integration test!

String sql = "SELECT * FROM TABLE WHERE A > ? ...");
conn.execute(sql);

It does not make sense to write a unit test!
Integration means...

• Testing the interaction of one component (part of your system) to another component.
  • Your architecture defines what a component is.
• One component to an external infrastructure, such as a database or the operational system.
• ISQTB: “Tests interfaces between components, interactions to different part of a system such as OS, file system, hardware or interfaces between systems.”
It’s more real, but harder to be done!

- We need a DB!
- Make sure the DB has the right schema
- Set up the database state (INSERTs, ...)
- Make sure one test does not interfere in the other
  - Clean up everything after the test

ItemDAO.java

```java
String sql = "SELECT * FROM TABLE WHERE A > ? ...";
```

ItemDAOTest.java

It does not make sense to write an unit test!

conn.execute(sql);
Unit, integration... Can we get even more real?
We can do **System Testing!**

BigTest.java
<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Very realistic</td>
<td>• Slow</td>
</tr>
<tr>
<td>• Captures the user perspective</td>
<td>• Hard to write</td>
</tr>
<tr>
<td></td>
<td>• Flaky</td>
</tr>
</tbody>
</table>
Testing pyramid

- Unit tests
- Integration tests
- System tests
- Manual

More reality
More complexity
How I (Maurício) do the trade-off

- **Unit tests**: All business rules should be tested here.
- **Integration tests**: Complex integrations with external services.
- **System tests**: Main/Risky flow of the app tested.
- **Manual**: Exploratory tests.
The ice-cream cone anti-pattern

Manual

System tests

Integration tests

Unit tests

Manual GUI tests

System tests

Integration tests

Unit tests
The Practical Test Pyramid

The "Test Pyramid" is a metaphor that tells us to group software tests into buckets of different granularity. It also gives an idea of how many tests we should have in each of these groups. Although the concept of the Test Pyramid has been around for a while, teams still struggle to put it into practice properly. This article revisits the original concept of the Test Pyramid and shows how you can put this into practice. It shows which kinds of tests you should be looking for in the different levels of the pyramid and gives practical examples on how these can be implemented.
References


• Osherove, R. (2015). The art of unit testing. MITP-Verlags GmbH & Co. KG.

• The practical test pyramid: https://martinfowler.com/articles/practical-test-pyramid.html