Programming best practices and other stuff about the barbarians

(the first day)

The first SW principle

“F**k you my code is better than yours”

Principle that is always valid no matter contexts, technologies or skills.

(2016 M. Merola)

Goal

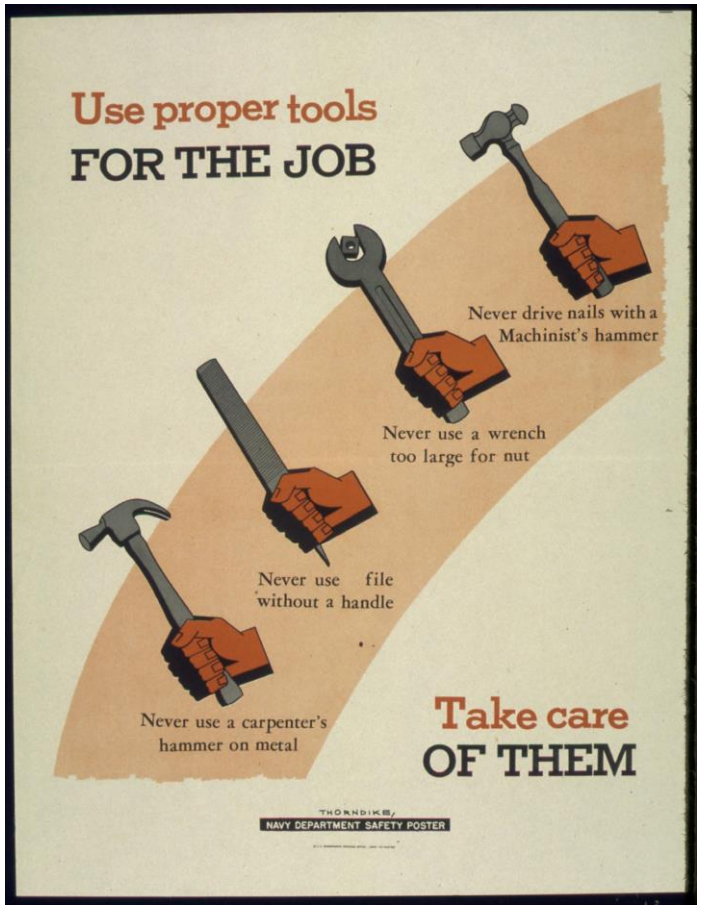
Best practices before starting
Errors while you are writing
Debugging the system

Before starting

Tools

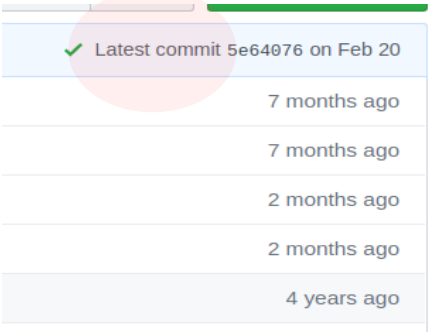
The first step

Use proper tools for the job



Check new technology with care

My hidden hint



The right tools

example

Android development

Android is Java native

The naming convention and the programming rules

The second step before start a new project is to choose (and to write) clear rules

Order first

- Naming conventions
- Programming rules

My hidden hint → https://google.github.io/styleguide/cppguide.html

The comments and the names

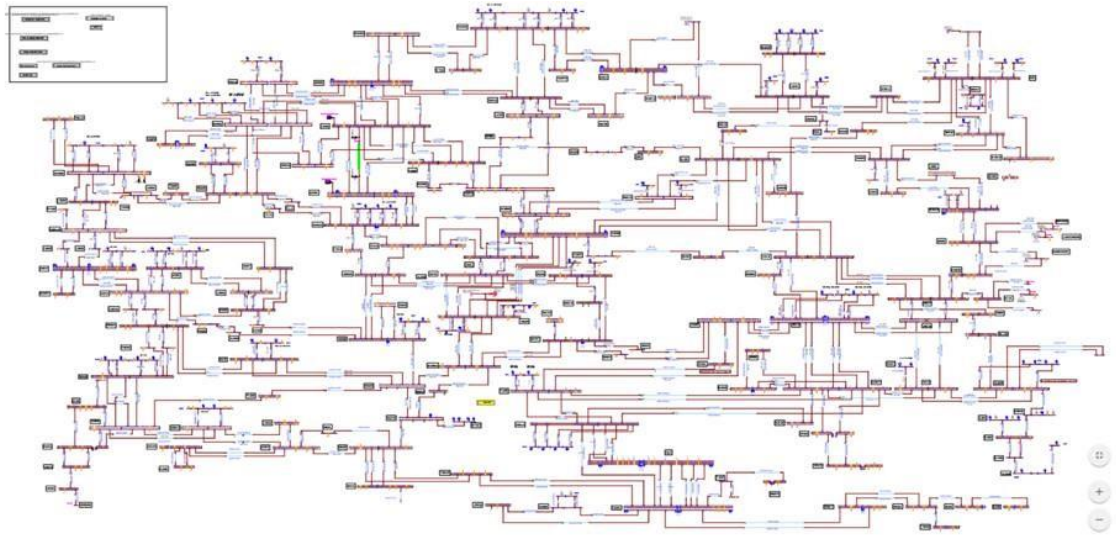
To comment or not to comment?

The key: choose right names

Note interesting by Joel:
the infamous Hungarian naming convention
https://www.joelonsoftware.com/2005/05/11/making-wrong-code-look-wrong/

Software architecture

Mostly if you are on Agile group



Write down only the main part and the general lines of your software

- Choose UML language to be generic
- Be careful with class diagram
- The architecture will change for sure in the next weeks
- A too complex and rigid architecture won't let you make the new changes
- Prefer deployment diagram, and design at components level

Now is developing time

Code your own wheel

Close code wheel

Check licenses
Check library lifetime and OS support
Check cost
Test functionality
Test code efficiency
Test code fault
Test and test

Open code wheel (GitHub)

Check licenses
Check library lifetime and OS support
Test functionality
Test code efficiency
Test code fault
All is under my control (*I hope*)

My wheel

Never use it, but you can learn
All is under my control

The Mongolian horde

Some delay occur → PM add more programmers to the team → Usually adding people to a delayed software project delays it even more

My hidden hint → Add people only if you have a good planning

The second SW principle

“Compila quindi funziona”

Again, principle that is always valid no matter contexts, technologies or skills.

(2007 L. Nardella)

Software quality I (compile so it works)

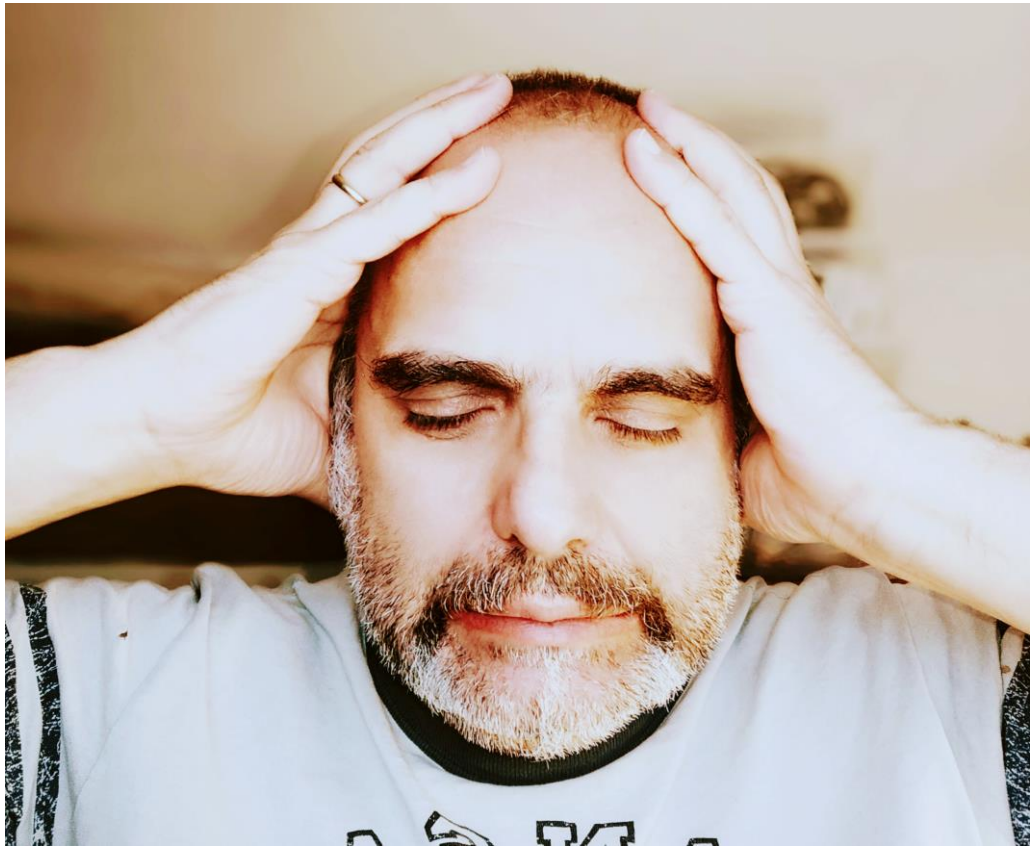
The Coke problem (Siemens 2010, OEE market)

The third SW principle

“It works on my machine”

Again, principle that is always valid no matter contexts, technologies or skills.

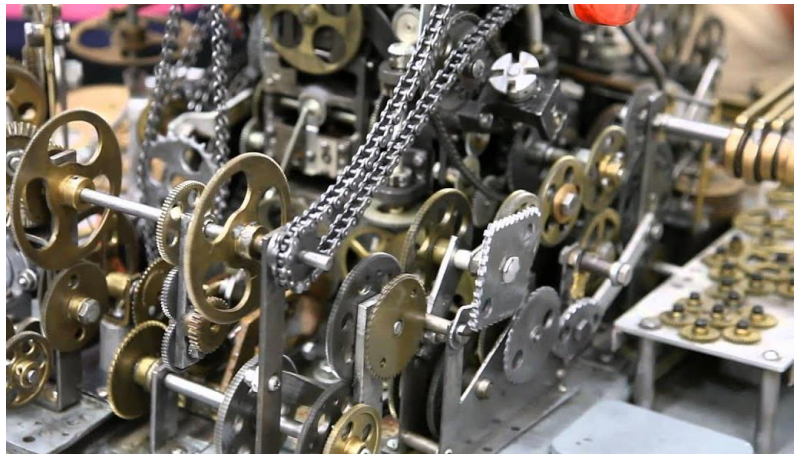
Software quality II (my machine)
The delivery problem (Siemens 2012, OEE market)



Always think generic

(not only what you guess they can ask you in the future)

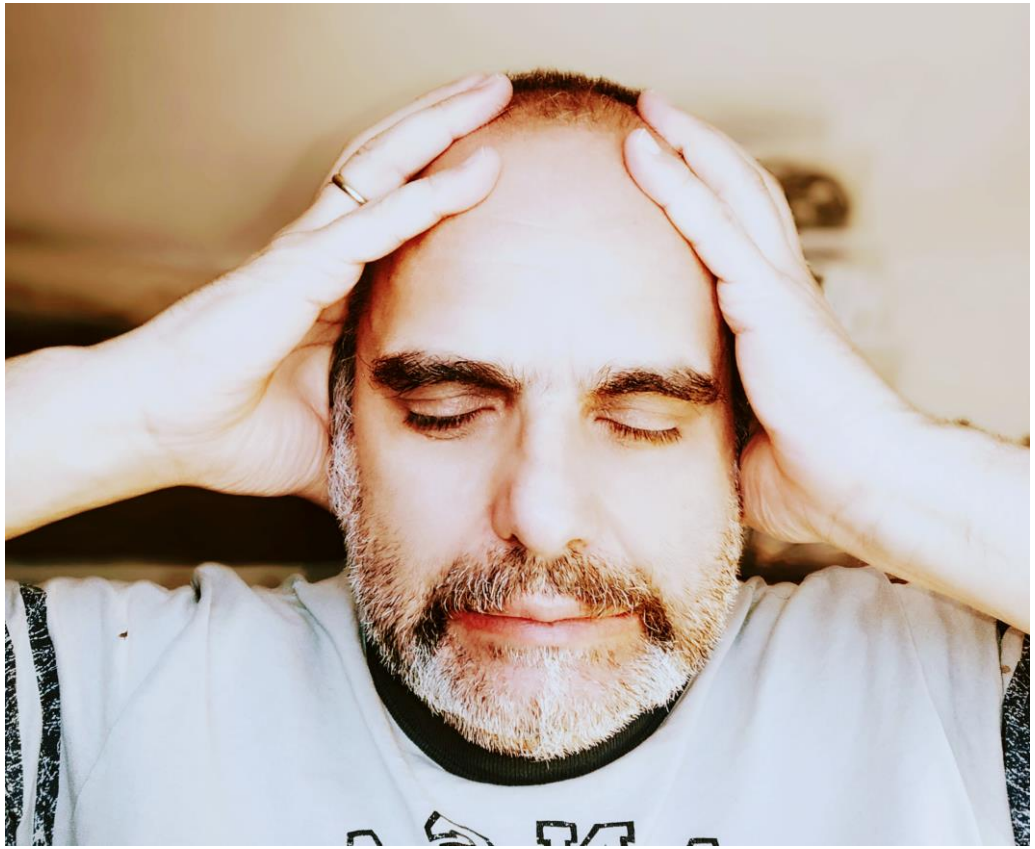
How can a banana fit in a gearbox?



Too late now

My hidden hint → Add or remove complexity?

Software quality III (be generic)
The dynamometric wrench problem (Atlas-copco 2017)



Refactoring (the code smells)

Why and When

To keep your code clean, **the code smells like a wet camel**
To save yourself time and **money** in the future
To reduce your technical **debt** (what is this?)

Why not

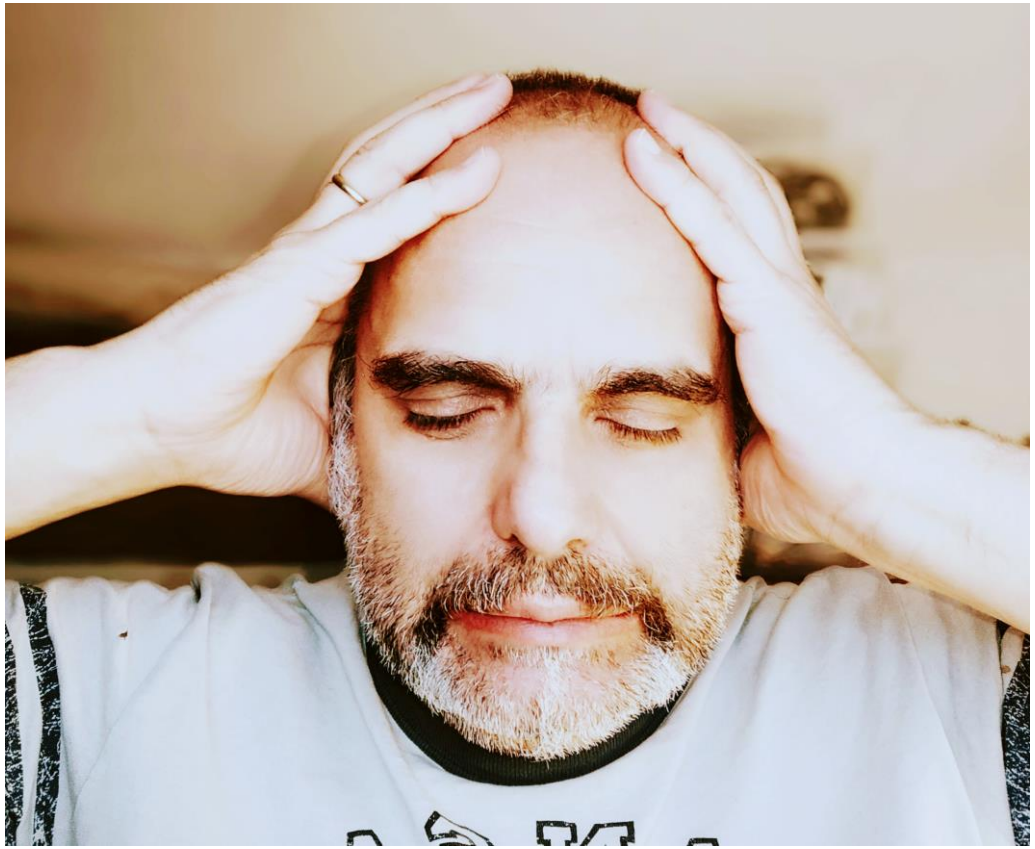
You have some new features to develop
You don't have tests

My hidden hint → Only if you can test the software

Example of a technical debit

Product name	Code name	Release date	Version number	Latest Update Version	Latest Update Date	Support Ends	Supported .NET Framework (no add-on)	Supported .NET Core (no add-on)
Visual Studio 2019	*Dev16*	2019-04-02[62]	16.0	16.7.0[63]	2020-08-05	**April 10, 2029**[65]	4.0 - 4.8	1.1, 2.1, 2.2, 3.0, 3.1
Visual Studio 2017	*Dev15*[66]	2017-03-07[67]	15.0	15.9.25[68]	2020-07-14	April 13, 2027[69]	3.5 - 4.7.2	1.0-1.1, 2.0, 2.1
Visual Studio 2015	*Dev14*[70]	2015-07-20[71][72]	14.0	Update 3[73]	2016-06-27	October 14, 2025[74]	2.0 - 4.6.1	1.0
Visual Studio 2013	*Dev12*[70]	2013-10-17[75][76]	12.0	Update 5[71]	2015-07-20	April 9, 2024[77]	2.0 - 4.5.1	N/A
Visual Studio 2012	*Dev11*[70]	2012-09-12[78][79][80]	11.0	Update 5[81]	2015-08-24	January 10, 2023[82]	2.0 - 4.5	N/A
Visual Studio 2010	*Dev10*[83]	2010-04-12[84][85]	10.0	Service Pack 1[86][87]	2011-03-10	July 14, 2020[88]	2.0 - 4.0	N/A
Visual Studio 2008	*Orcas*[89]	2007-11-19[90]	9.0	Service Pack 1[91]	2008-08-11	April 10, 2018[92]	2.0, 3.0, 3.5	N/A
Visual Studio 2005	*Whidbey*[93]	2005-11-07[94]	8.0	Service Pack 1[95]	2006-12-15	April 12, 2016[96]	2.0	N/A
Visual Studio .NET 2003	*Everett*[97]	2003-04-24[98]	7.1	Service Pack 1[99]	2006-08-15	October 14, 2013[100]	1.1	N/A
Visual Studio .NET (2002)	*Rainier*[101]	2002-02-13[102]	7.0	Service Pack 1[103]	2005-03-08	July 14, 2009[104]	1.0	N/A
Visual Studio 6.0	*Aspen*[105]	1998-09-02[106][107]	6.0	Service Pack 6[108]	2004-03-29	September 30, 2005[109][110]	N/A	N/A
Visual Studio 97	*Boston*[111]	1997-03-19[112][113][114]	5.0	Service Pack 3	1997-12-04	June 30, 2003[115][116]	N/A	N/A

Software quality IV (refactoring)
The Tetra nightmare (Marconi 1998, the TETRA standard with Nokia)



Code optimization

(Clear code first)

Optimization code rules(Michael A. Jackson):

1. Don’t do it.
2. *(For experts only!)* Don’t do it *yet*.

Why and When

If you really have performance problems

If you want to add some bugs to your code

Why not

For joke

Someone has already done this (stl,boost ...)

Be careful

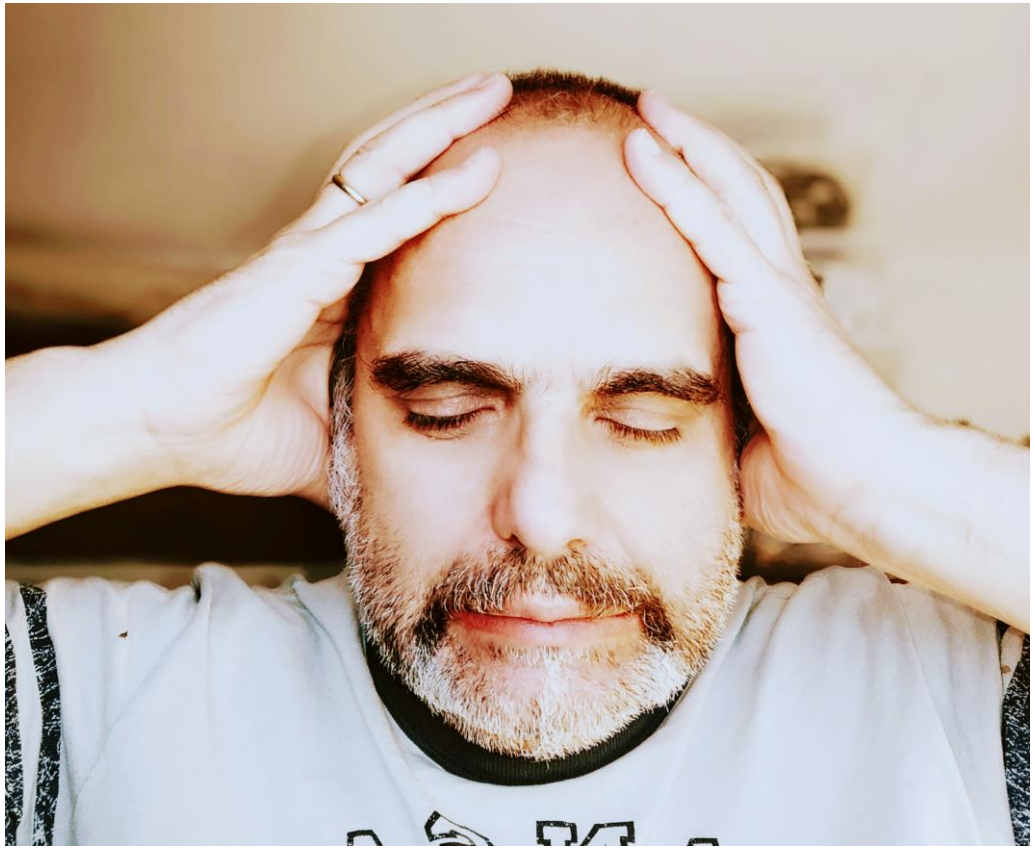
Use the right tools, don't waste your time

Writing less code is not optimization!

Optimization vs Readability

My hidden hint Check where to optimize

Software quality V (optimization)
100K Dollars (Ratioconsulta 2009, the telephone calls market)



Testing and Debugging

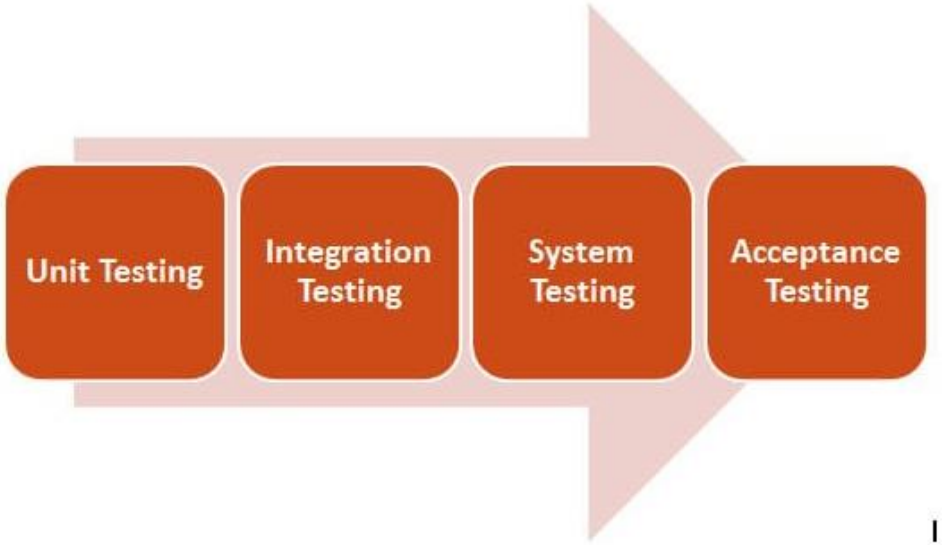
- Agile group collaboration vs Waterfall approach
- Unit test (Java, C++)
- Integration test
- System test

My hidden hint

Create mix test/devel group

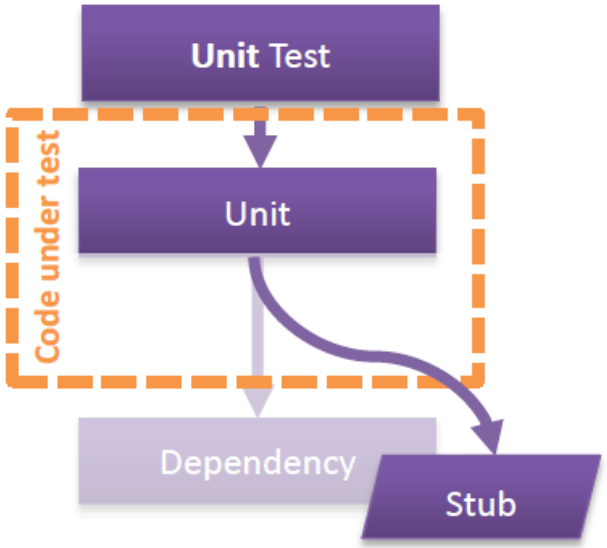
https://www.fingent.com/blog/collaborating-testers-and-developers-in-hybrid-agile-environment

Test checklist



Unit test – Test the smaller part of a system

Mock and Stub

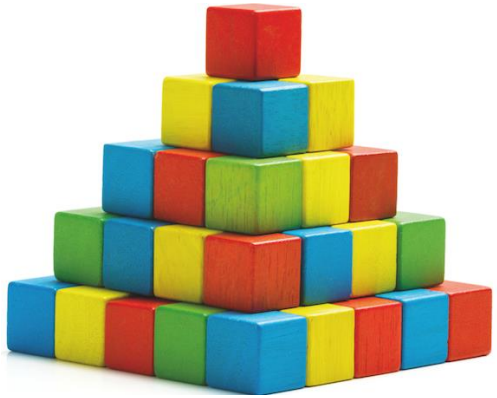


System test - Test a complete system

Testing a plane for breaks while landing

Automation test – repeatable, save and easy

Easy to use
Close to natural language as test case
Modular
Safe – each block is correct



Blocktest

Debugging time

Debugger VS Logger

Simple development

Complex development
Production

My hidden hint → Learn to use good logging library

Practices, break the stones