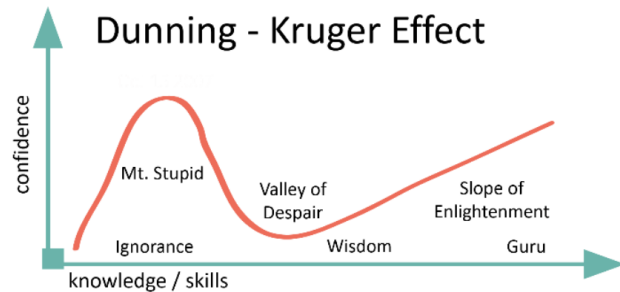


TDD methodologies: The GREY side of the Moon

By Pierluigi Fornoni

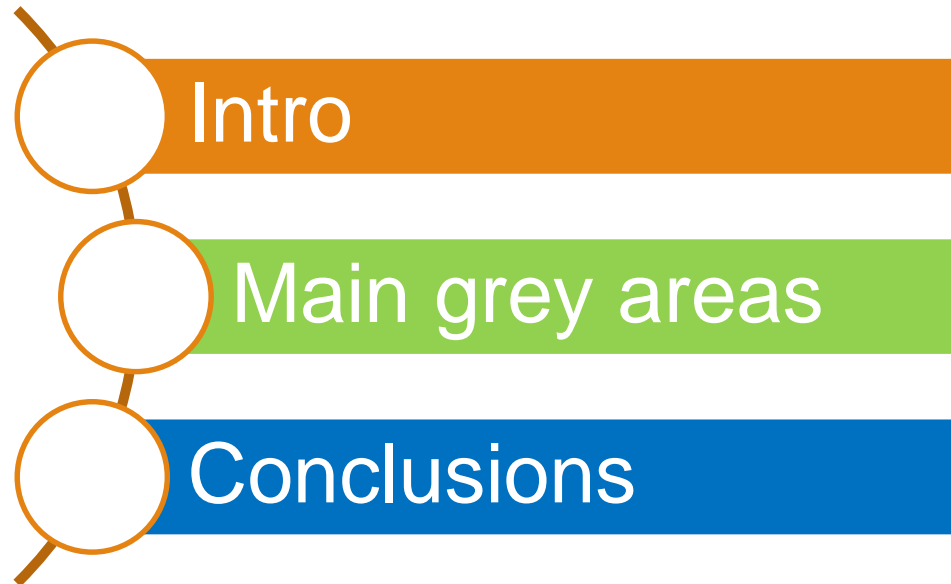
Is everything black or white?

Now we attended 18 lessons, we have been introduced to all patterns and methodologies and all should be finally clear. Can we walk with our own feet without perplexity?



Life is about the gray areas. Things are seldom black and white, even when we wish they were and think they should be, and I like exploring this nuanced terrain
-- Emily Giffin

Table of Contents



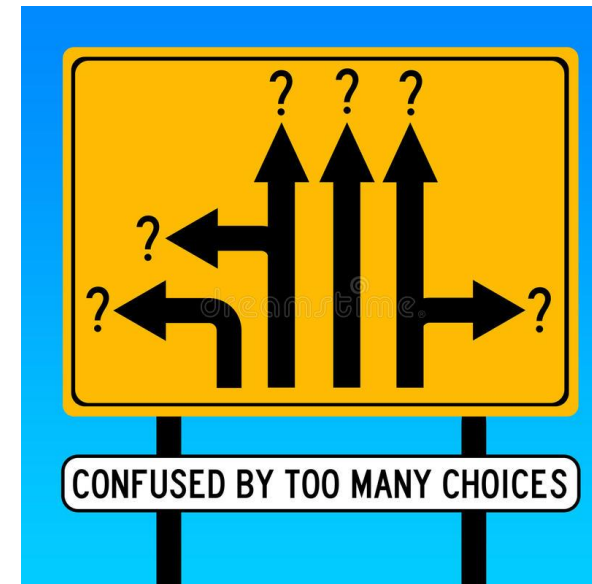
Intro
Main grey areas
Conclusions

Grey areas

Sometime a door is open and new ideas comes. But when many roads are possible and different actors are involved it is sometime hard to choose the right one.

When for a problem:

- different solutions are possible
 - no solution works perfectly
- ... then we probably found a **grey area**.



Solution: «just use common sense»

Sometime it is easy to get lost.

How many times have you heard this?
If you are in trouble... just use common sense!

Generally this means.

Just use MY common sense.

*(and if something goes wrong AFTER having seen results I will
blame you for your choices)*



Main grey areas

Following arguments are subjected to grey area

- Requirement Analysis (level of details)
- Implementation (quality and test coverage)
- Methodology (collaboration)



Requirement analysis

This part of software production has a lot of human interaction level. Many ideas, discussions, agreements.



- ❖ how far should we go with our analysis before starting coding and avoiding **analysis paralysis**?
- ❖ how many **scenario** have to be cleared before producing the first line of code?
- ❖ what is the best way to **document** business requests?

Requirement analysis

Through ***outside-in*** architecture and ***high-level acceptance tests*** approach together on having business actors in meetings (***3 amigos?***) we learned how our analysis phase can improve by removing communication barriers and misunderstandings.

```
//Given
I_Bought_Shares(OldSchoolWaterfallSoftwareLimited, 1000, new DateTime(1990, 2, 14));
I_Bought_Shares(CrafterMastersLimited, 400, new DateTime(2016, 6, 9));
I_Bought_Shares(XpPractitionersIncorporated, 700, new DateTime(2018, 12, 10));
I_Sold_Shares(OldSchoolWaterfallSoftwareLimited, 500, new DateTime(2018, 12, 11));
Current_share_value_of(OldSchoolWaterfallSoftwareLimited, 5.75m);
Current_share_value_of(CrafterMastersLimited, 17.25m);
Current_share_value_of(XpPractitionersIncorporated, 25.55m);

//When
I_print_the_portfolio();

//Then
string expectedOutcome =
    "company | shares | current price | current value | last operation\n" +
    "Old School Waterfall Software LTD | 500 | $5.75 | $2,875.00 | sold 500 on 11/12/2018\n" +
    "Crafter Masters Limited | 400 | $17.25 | $6,900.00 | bought 400 on 09/06/2016\n" +
    "XP Practitioners Incorporated | 700 | $25.55 | $17,885.00 | bought 700 on 10/12/2018";
The_outcome_should_be(expectedOutcome);
```



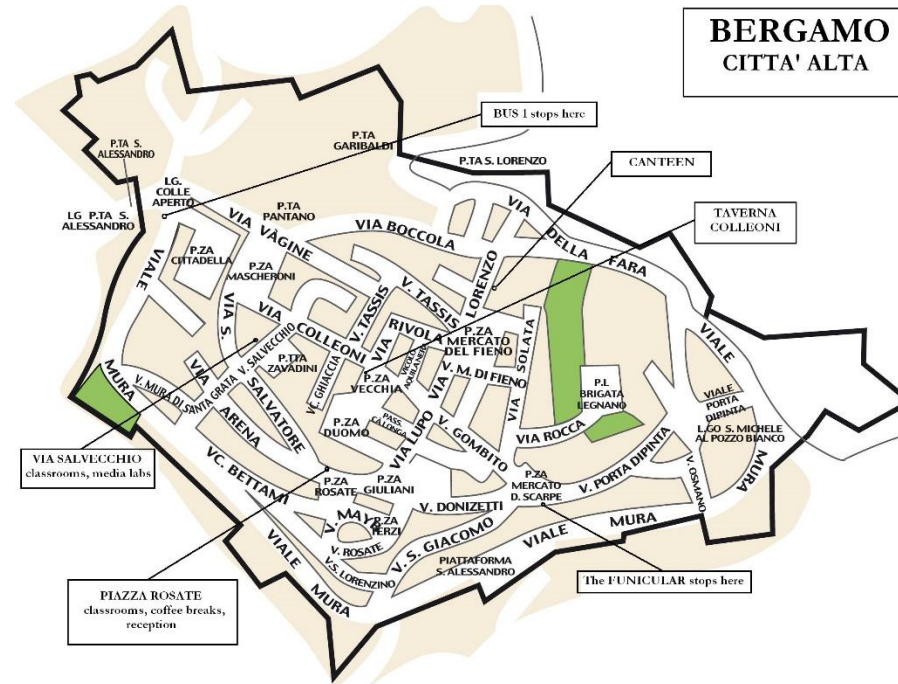
Does it means all grey area are cleared?



Of course not.
But really is this a so huge problem?

Start a project is like starting a trip

You can wait for a complete and detailed map before leaving...



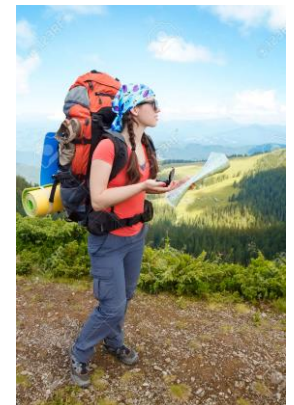
Start a project is like starting a trip

...but it would need a lot of time to prepare it.

In that time a road can be closed, or a new nice place to see not initially considered can pop up.



On the other hand you can just prepare a minimal set of “necessaire” and leave. And during trip check the **compass** often.



Is making a single step in the wrong direction still a problem?

Development/Design

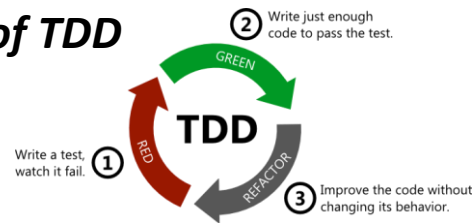
During course we had the opportunity to touch the various problems and face the most appropriate strategies to solve them



Development/Design

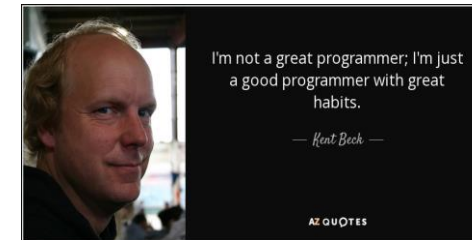
How many tests / which part should be tested?

❖ *The three lows of TDD*



Which test first which after?

❖ *One degree of freedom* (having good habits)



How measure code quality?

- ❖ *Cognitive effort (transformation priority premise + object calisthenics rules)*
- ❖ *Coupling and cohesion*
- ❖ *Connascence*



Look Ma!
I'm a
behaviour
attractor!!

Development/Design

How indagate for code problems and how to fix them?

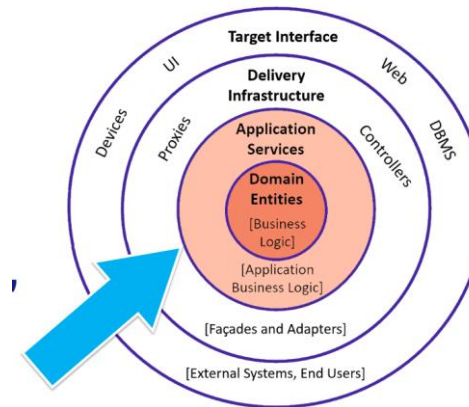
❖ *Code smells + solid principles*



How should be a great design?

❖ *Simple design (using test doubles)*

❖ *Outside in*



Development/Design

But are there still open points that can be interpreted depending on different factors and don't follow a 1-1 rule



Dev false hopes

Don't expect that your way of doing and progressing is the only correct way of doing.
Don't expect that in 1 year no one would regret some decision.
Don't expect that nobody never say «I would do it some other way».



This should not be a problem!

SO WHAT?



WHO CARES?

Whenever a theory appears to you as the only possible one, take this as a sign that you have neither understood the theory nor the problem which it was intended to solve.

-- Popper

Here are some examples

What a beautiful thing TDD and now I will use it during development of my GUI.
But how should I do it? Does ***only-under-the-skin*** tests means ***no-over-the-pants*** checks?

Everybody understands what ***YAGNI*** stand for, so no extra code till the rule of three is satisfied. But... interface segregation + dependency inversion? Should I implement interfaces from the beginning for all classes? Moreover, do I really need to do 100 refactorings to join the same architecture I focused will be useful from beginning?

When creating a ***walking skeleton***, which basic scenario would be the best to choose?

We know that we never should provide production code without tests for it. But what about a fix to a old legacy tool that has been untouched for 2 years and probably will be replaced in next 6 months?

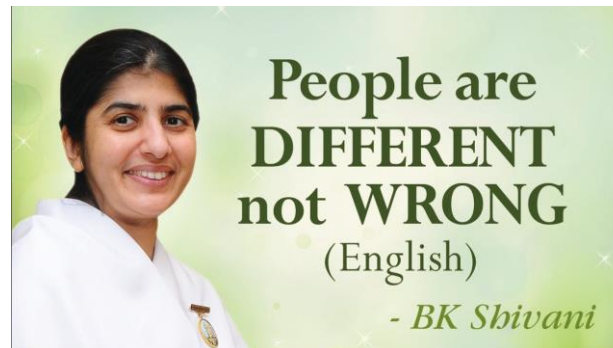


Methodology

Software engineers are humans with specific culture, ideas, and **psychology**.

Software development force them to **collaborate**.

What could be a way to improve productivity and reduce **misunderstandings**?



Methodology

Pair programming / Mob programming force multiple actor in share knowledge. This is a great alternative to pool requests and multiple mails.

But...

- ❖ What if 2 people don't get along?
- ❖ What if they best work in complementary time zones (morning/afternoon)?
- ❖ What if one is extroverted and the other introverted?



- Un inglese è un lord, due inglesi sono un team, tre inglesi sono un club.
 - Un italiano è un latin lover, due italiani un casino, tre italiani fanno quattro partiti.
- Beppe Grillo

Methodology



Susan Cain (1968) is an American writer and lecturer, and author of the 2012 non-fiction book **Quiet: The Power of Introverts in a World That Can't Stop Talking**, which argues that modern Western culture misunderstands and undervalues the traits and capabilities of introverted people.



Calvin C. Newport (born 1982) is an American non-fiction author and associate professor of computer science at Georgetown University. Coined the term "*deep work*," in his book **Deep Work** (2016 which refers to studying for focused chunks of time without distractions such as email and social media .

Methodology

Deep Work: Professional activities performed in a state of distraction-free concentration that push your cognitive capabilities to their limit. These efforts create new value, improve your skill, and are hard to replicate.

To learn hard things quickly, you must focus intensely without distraction.



Methodology

Does it means that Pair/Mob programming is bullshit and all developers should never talk each other?

No it means that whatever collaboration way we decide, it should **not be considered as a rule of thumb**. It could work great in some periods for some features for some people. But it does not mean it works in all occasions.

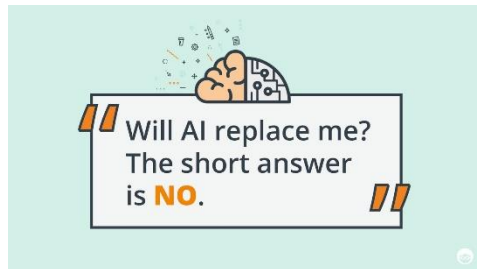


Force 2 people to collaborate in strictly way or set them near in the same office do not automatically improve their performance. On the contrary it could reduce it.

Is still having grey areas a problem?

Just think at it.

If software production process would be straightforward, a machine could do it automatically and world did not need software developer anymore.



Software Development is a learning process. Working Code is a Side effect
-- Alberto Brandolini

Thank you

References:

- ❖ [Agile Technical Practices Distilled: A learning journey in technical practices and principles of software design](#)
- ❖ [Quiet: The Power of Introverts in a World That Can't Stop Talking Paperback – January 29, 2013](#)
- ❖ [Deep Work: Rules for Focused Success in a Distracted World](#)



Pierluigi Fornoni, software engineer-eoc
pierluigi.fornoni@eoc.ch