Me against me, my experience with TDD

Alcor - Software Craftsmanship Training – presentation day - Giovanni Saba

Topics

Me against me, my experience in TDD

- Relying on a development practice being aware of ist benefits by going against working practices and habits grown in past experiences.

- My personal experience.

The way I have gone till I started rely on automated testing

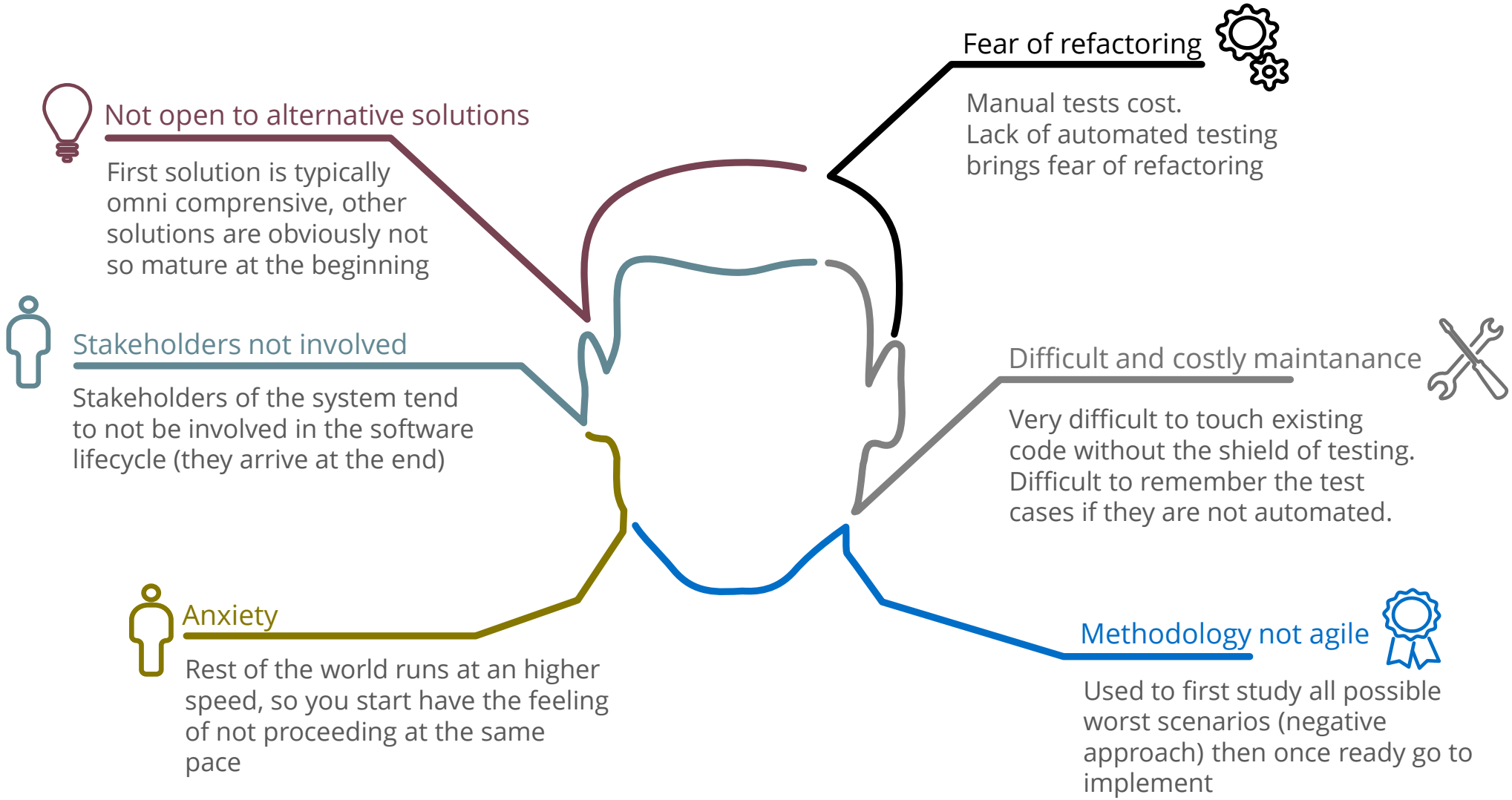
- Requirements Engineering
- Write specifications

- Think of best solution
- Take decisions based on past experience
- Try to **anticipate possible future issues**

- Leverage most stable and **less risky** development design patterns
- Develop as less as possible -> do not reinvent the wheel

- Manual tests
- Ensure developed stuff works
- **Reduce my anxiety**

This led to...


Fear of refactoring
Manual tests cost.
Lack of automated testing
brings fear of refactoring
Not open to alternative solutions
First solution is typically
omni comprensive, other
solutions are obviously not
so mature at the beginning
Stakeholders not involved
Stakeholders of the system tend
to not be involved in the software
lifecycle (they arrive at the end)
Difficult and costly maintanance
Very difficult to touch existing
code without the shield of testing.
Difficult to remember the test
cases if they are not automated.
Anxiety
Rest of the world runs at an higher
speed, so you start have the feeling
of not proceeding at the same
pace
Methodology not agile
Used to first study all possible
worst scenarios (negative
approach) then once ready go to
implement

One day we took a decision

Few years ago me and my team started a new project **from scratch**.

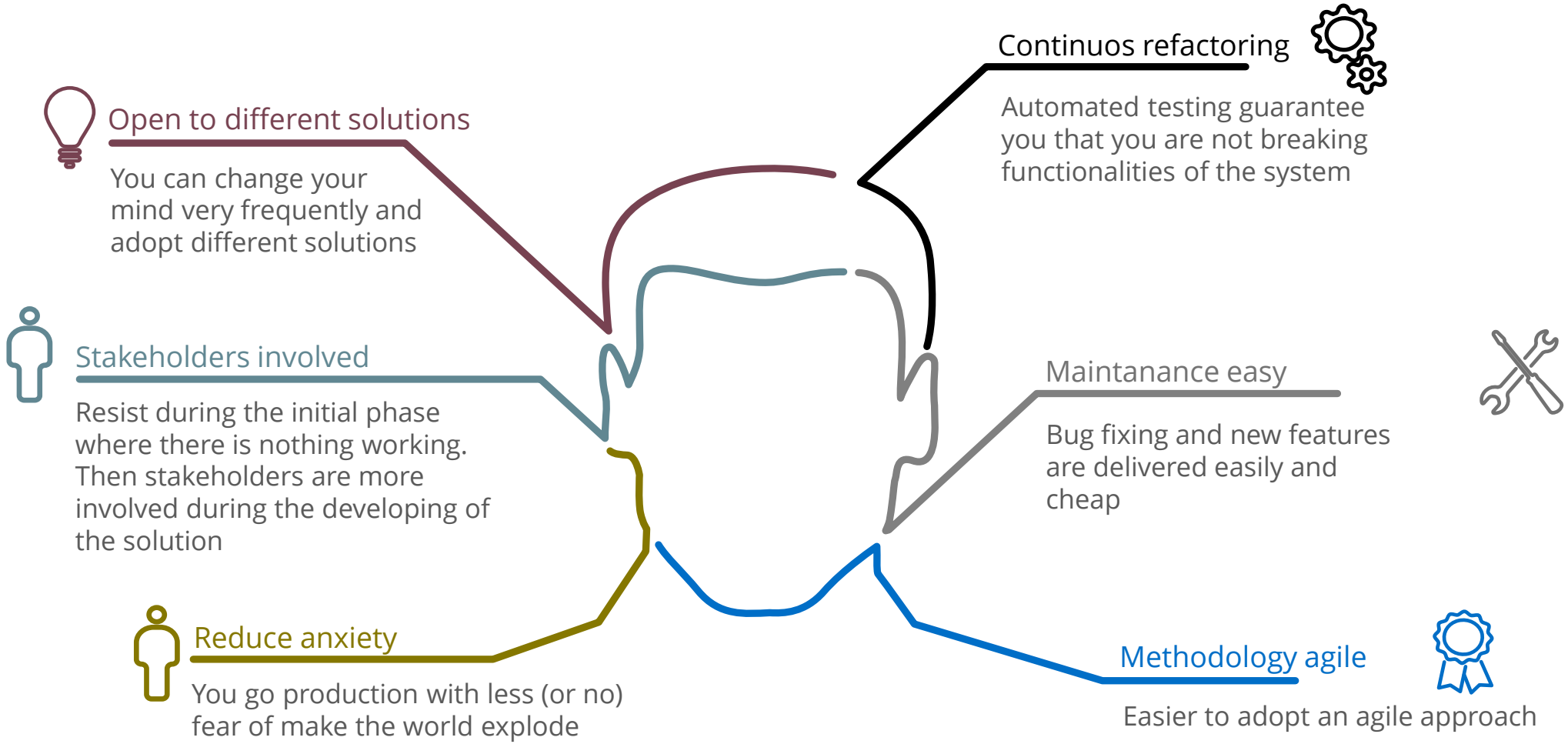
The environment and circumstances were appropiate to invest in quality and not speed (basically we had not so strictly delivery time constraints).

We were aware that after the first release we should have mantained the solution with continous new feature delivery (It was agrred in the contract, and such new feature development was not paid so well, so we should have spent less time as possible developing them).

So the key factors were:
- **no regression**
- **speed of delivery of new feature**

As a team we decided to **invest a lot at the beginning** in
- **automated testing**
- auto generation of code for the interfaces (the component exposed a REST interface agreed with the FE components)
- Adopt good design patters

This led to...


Continuos refactoring
Automated testing guarantee you that you are not breaking functionalities of the system
Open to different solutions
You can change your mind very frequently and adopt different solutions
Stakeholders involved
Resist during the initial phase where there is nothing working. Then stakeholders are more involved during the developing of the solution
Maintanance easy
Bug fixing and new features are delivered easily and cheap
Reduce anxiety
You go production with less (or no) fear of make the world explode
Methodology agile
Easier to adopt an agile approach

… and now TDD … my conclusion

Next step: TDD

First impressions:

- I like a lot starting from testing because this make you **think first on functionalities** and not focus on implementation
- Have a defined methodology helps you when you feel lost in the middle of developing (analysis paralysis) -> have a **continuous pace**
- The refactor phase make you have **defined phase were you simply focus in applying good code design patterns**, forget about functionalities for a while (those are covered by green tests)
- Writing tests make wrong development rise up (it makes them more evident)

Even if:

- My mind-set still works thinking first at the solution (good design) and later to tests.
- I feel a sort of resistance coming from my habits and experiences.

It is clear it is a matter of practicing, the more you do TDD more you change and get confidence.

Thanks

any questions?