TDD AND MOB PROGRAMMING - RETHINKING AND RELEARNING

Luiza da Silva

THINK AGAIN!

Think Again, by Adam Grant

Our mindset is usually one of three:

- **Preacher**
  When we promote our ideas and defend them from criticism
- **Prosecutor**
  When we scrutinize other's opinions to win an argument
- **Politician**
  When we try to win over others for approval and agreement

THINK AGAIN!

Think Again, by Adam Grant

AND YET we should foster this mindset:

- **Scientist**
  - Expect to doubt what we already know and be curious about the unknown
  - When faced with new information, we update our views
  - Rely on evidence, not jump to conclusions

WHY?

Because we search for reasons for why
we are right,

instead of why we are wrong

“

YES, LEARNING REQUIRES FOCUS.

BUT,

UNLEARNING AND RELEARNING REQUIRES MUCH MORE—

IT REQUIRES CHOOSING COURAGE OVER COMFORT.

”

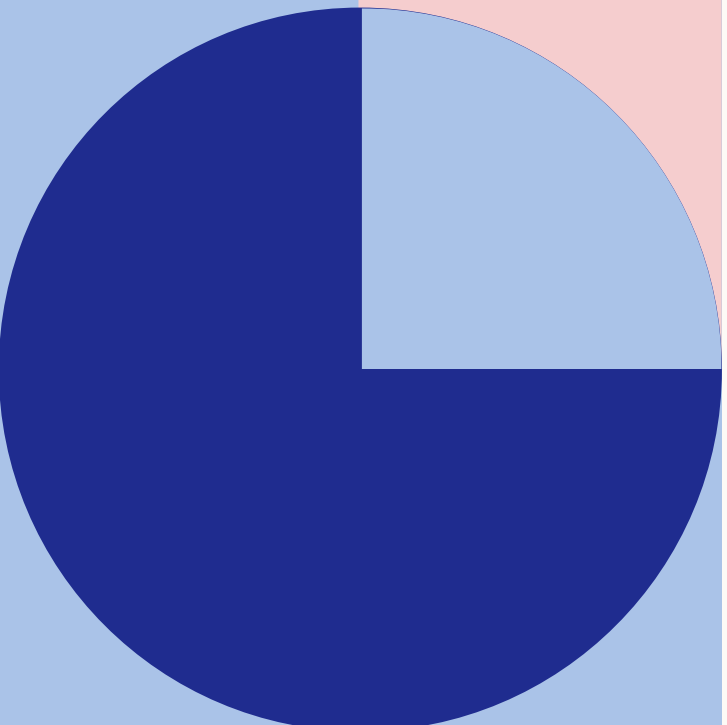
Brene Brown, author of *Dare to Lead*

RETHINKING CYCLE

Confident Humility -> Doubt -> Curiosity -> Discovery

- Have faith in abilities but retain doubt, curiosity and flexibility to recognize that we might be wrong
- Detach from our beliefs - Find the joy in being wrong
- Learn how to deal with conflict
  - With honesty, not just being agreeable
- Ask questions to understand the point of view of others

RECAP: MOB PROGRAMMING



Going beyond pair programming - the Strong-style pairing technique

Driver must trust the navigator

- Become comfortable with incomplete information

Navigator must express ideas clearly with words first

Even tighter feedback loop!

RECAP: TDD

What?

Test behavior (just one per test) and let that drive implementation

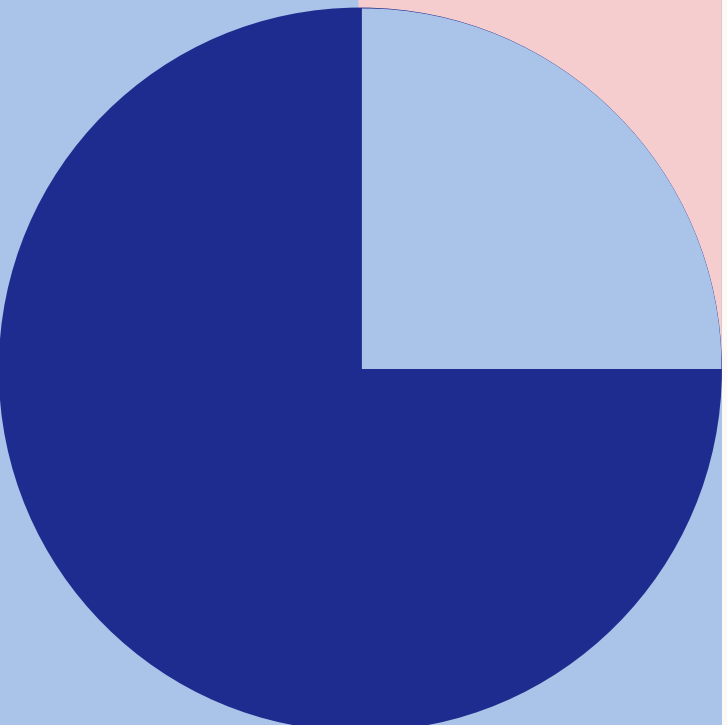
Using rules to internalize principles

Why? Benefits

- Simpler design
- Unit tests are behavior documentation that stay updated
- Reduce debugging time

How? Three steps

- Red: Write a failing test
- Green: Write just enough code to make it pass
- Refactor: Improve code while passing the test



RECAP: EVOLVING CODE

- Fake implementation
  Make test pass by returning the expected result

- Obvious implementation
  - Move a step beyond faking it
  - What is the next step?
    Use Transformation Priority Principle

Transformation Priority Premise - What is "Obvious implementation" ?

#	TRANSFORMATION	STARTING CODE	FINAL CODE
1	{} => **nil**		`return nil`
2	nil => **constant**	`return nil`	`return "1"`
3	constant => **constant+**	`return "1"`	`return "1" + "2"`
4	constant => **scalar**	`return "1" + "2"`	`return argument`
5	statement => **statements**	`return argument`	`return arguments`
6	unconditional => **conditional**	`return arguments`	`if(condition)return arguments`
7	scalar => **array**	`dog`	`[dog, cat]`
8	array => **container**	`[dog, cat]`	`{dog = "DOG", cat = "CAT"}`
9	statement => **recursion**	`a + b`	`a + recursion`
10	conditional => **loop**	`if(condition)`	`while(condition)`
11	recursion => **tail recursion**	`a + recursion`	`recursion`
12	expression => **function**	`today - birthday`	`CalculateAge()`
13	variable => **mutation**	`day`	`var day = 10; day = 11;`
14	**switch case**		

- Triangulation with next test - Make behavior more generic

=> Object Calisthenics for Refactoring

Object Calisthenics rules

- ➔ Only one level of indentation per method
- ➔ Don't use the ELSE keyword
- ➔ Wrap all primitives and strings
- ➔ First class collections (wrap all collections)
- ➔ Only one dot per line ~~`dog.Body.Tail.Wag()`~~ => `dog.ExpressHappiness()`
- ➔ No abbreviations
- ➔ Keep all entities small
  [10 files per package, 50 lines per class, 5 lines per method, 2 arguments per method]
- ➔ No classes with more than two instance variables
- ➔ No public getters/setters/properties

SUMMARY

I learned a new cycle of Rethinking and Relearning!

By mob programming and using TDD

- We are acting like scientists
- Setting up hypotheses and testing our beliefs and understanding of behavior
- Avoiding jumping to solutions and conclusions
- … while learning from others and receiving feedback

SUMMARY

So...

"Eat that hot pepper" and constantly reevaluate

- Avoid being stuck with our beliefs as developers
- Avoid being stuck with our ways of thinking and habits
- Accept that programming with others can be a wonderful tool for relearning

All this is the start for building a **collaborative mindset**

THANK YOU!