Clean Code Colloquium: What's in a name?

Who am I?

- Founder and CTO at Tracworx
- Immersive SE advisory board member
- Building digital infrastructure for returnable assets
- Built patient tracking systems
- Built COVID-19 contact tracing solution

What's in a name?

- Names are identifiers for concepts and objects
- Good names are a vital part of Clean Code
- Context gives extra information to names
- Good names reveal intent
- Computers don't care about good/bad names...
- ...but programmers should!

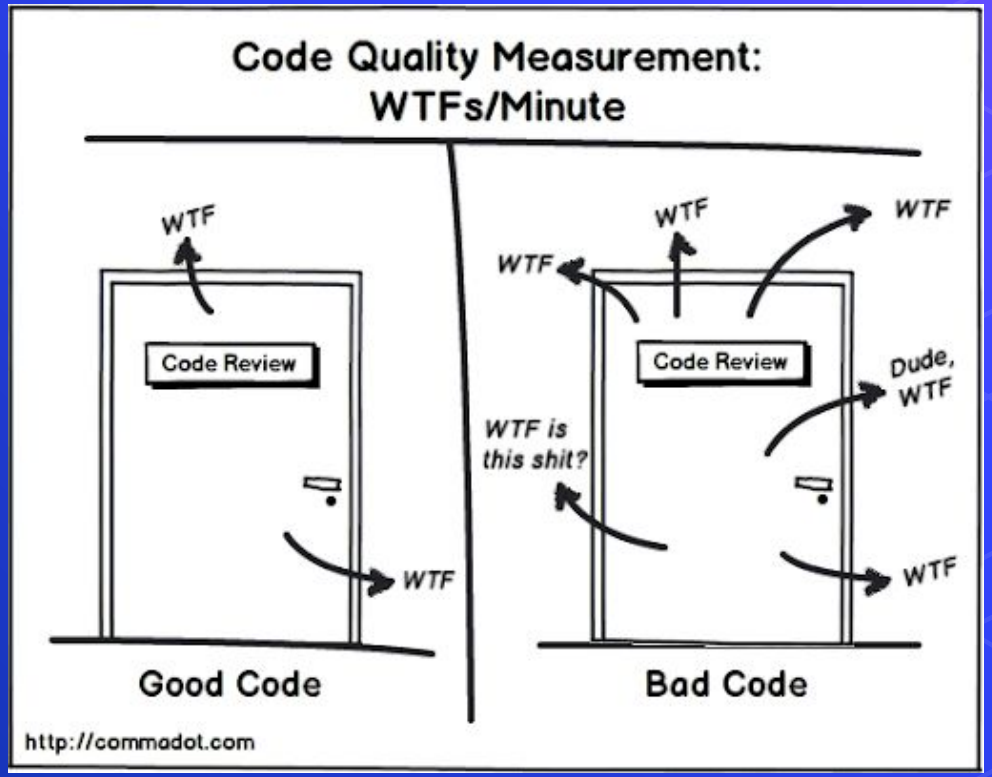
Human-friendly naming

- Who reads code? People!
- So names should be human-readable
- If you’re explaining, you’re losing
- Context is king
- Acronyms and abbreviations
- Don’t make up crazy terminology

Where do names come from?

- From the solution domain:
  - Linked list
  - Decorator
  - Service
- From the problem domain:
  - Learn from the client
  - Workshop with the users
- Names are for humans — not just those writing code!

What makes a good name?



Some bad names

- a
- x1, x2, x3, x4
- foo
- bar
- data
- uptime
- theString
- lllllllll

Clarity over all

- One word per concept — no puns!
- What's the difference between `remove` and `delete`?
- What's the difference between `add` and `create`?
- Semantics must be consistent
- Be skim-friendly!

Disinformation

- If the name doesn't fit, don't use it
- Be careful with fixed meanings
  - A `userMap` should be a map data structure
- Don't add meaningless noise (**a**, **an**, **the**…)
- Don't switch between US and British spellings
  - `optimise()` vs `optimize()`

Searching for names

- `Ctrl+F` and `grep` are your friends
- Searchability is just as important as readability
- Meaningful names make feature location easier
- Magic numbers cause problems
  - `pi` is better than `3.14159`
- Name length is proportional to scope width

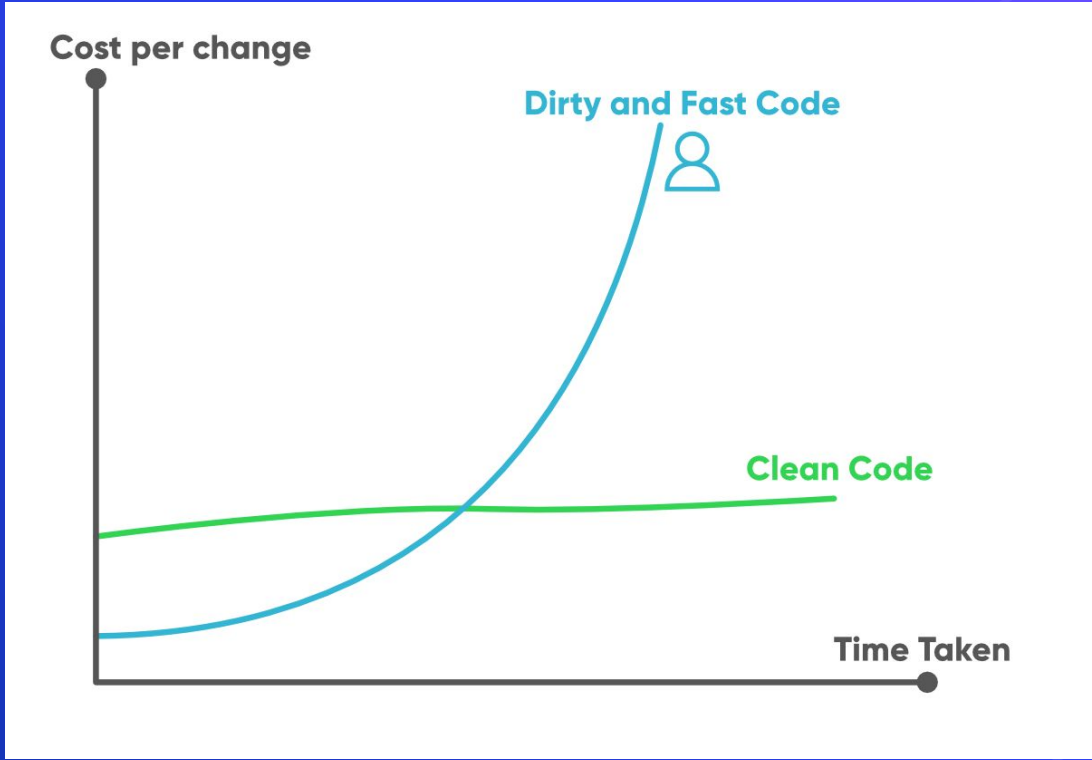
No jokes!

- Say what you mean and mean what you say
- In-jokes can be fun…
- …but crashes in production aren’t fun
- …and some people won’t get the joke
- `UserManager.isUserAGoat()`

Maintaining names

- Code will change over time
- Stay DRY — don’t repeat yourself!
- Don’t include types: `szGivenName`
- Don’t include access levels: `mFamilyName`
- Don’t give too much context: `product_db_table`
- Don’t fight the IDE
- Don’t be afraid to rename things

The cost of names



Comments

- Good comments show intent
- Bad comments explain code and add noise
- `a = 5; // Set the value of a to 5`
- `// I don't know why this works`
- Do I really need a comment... or just a better name?
- `int x = 280; // Character limit`
- `int characterLimit = 280;`
- Maintain comments like code!

Conclusion

- Good naming conventions are vital!
- Intent and context
- Maintainability and extensibility
- Keep your code clean and your tech debt low

Questions?

eoin@tracworx.ai