Clean Code Summary

Kevin Löffler

April 2023

Contents

1 About the book

1.1 Introduction

Often ranked as the most important book to read for software engineers "Clean Code" by Robert C. Martin covers fundamental software engineering practices. Martin explains how to write clean code that is, readable, easy to modify, and maintainable by providing practical tips and examples that developers can apply to their projects. The book is organized into chapters that cover different aspects of clean coding, from naming conventions to code structure, and also includes case studies and real-world examples.

In this summary I attempt to capture some of the most important concepts, often with practical examples, because its often easiest to let the code speak for itself. An idea that I believe Robert C. Martin would agree with.

1.2 Clean Code

First we need to agree that good code is important. Bad code can be detrimental to a project, product and ultimately a business itself. Martin tells stories about companies that collapsed because their code base became unmanageable. He identifies several problems with code that make working with it unfeasible and unenjoyable.

1.2.1 Readability

Developers spend a much bigger portion of their time reading code than writing (10:1). CITATION?? Code that is hard to read therefore slows the creation of new features down. The more code that is hard to read the less progress is made overall.

1.2.2 Maintainability

In the beginning of a project teams often progress extremely quickly. Later theirs speed often gets destroyed by unmaintainable code. It manifests itself when a change at one point in the program leads to refactors in several completely different parts of the code. Resulting in the fact that no change is trivial and a developer needs to understand every part of a system to be able to work on it.

1.2.3 Care

Martin says that the core of the problem is not the reasons we often hear: to little time, a budget that is too small, shifting requirements. He proclaims the problem to be more fundamental: unprofessionalism or a lack of care from the developers.

Martin calls Code that satisfies these principles 'clean code'.

2 Naming

2.1 Descriptive Names

Names should always describe why they exist and what they do. If a variable declaration needs a comment it has already failed.

Bad and good vriable names

```
let d: int;  // elapsed time in days
let elapsedTimeInDays: int;  // descriptive = no comment needed
```

Additionally, variables should always have names that are pronounceable and searchable. A variable named '`x`' is very hard to ever find again.

2.2 Honest Names

When using programming concepts in our names, developers should be careful not to misuse a name. For example a variable that is called '`accountList`' should really be a list and not some other data structure. Much better would be an implementation agnostic name like '`accounts`'.