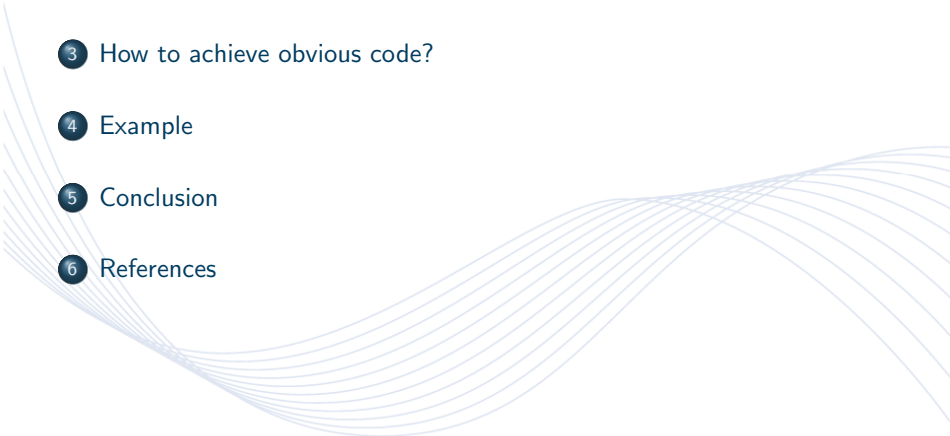


John Ousterhout: A Philosophy of Software Design

Code Should be Obvious

Vikas Ramaswamy

7. Mai 2023

- 1 Introduction
 - 2 Importance for code
 - 3 How to achieve obvious code?
 - 4 Example
 - 5 Conclusion
 - 6 References
- 

Introduction



Introduction to A Philosophy of Software Design I

- A Philosophy of Software Design is a book by John Ousterhout that presents a set of principles and practices for designing high-quality software. Ousterhout is a well-known computer scientist and software engineer who has made significant contributions to the development of various programming languages and software systems.
- The book argues that software design is more about understanding the problem domain than about implementing solutions. It emphasizes the importance of simplicity, clarity, and maintainability in software design, and provides practical guidance on how to achieve these goals.

Introduction to A Philosophy of Software Design II

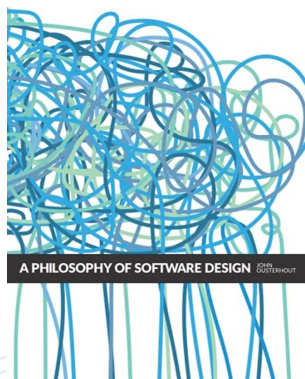


Abbildung: Cover Page [Ous18]

Introduction to Code Should be Obvious I

According to "A Philosophy of Software Design" by John Ousterhout, for code to be obvious, it should be easy to understand and follow, even for someone who is not familiar with the codebase. The book argues that writing code that is self-documenting is key to achieving this goal. This means that the code should be written in a way that clearly communicates its purpose and behavior, with clear and concise names for functions, variables, and classes.

Introduction to Code Should be Obvious II

The book provides several practical tips for making code obvious, including:

- Use clear and descriptive names for variables, functions, and classes. Avoid overly complex code that is difficult to understand. Break down code into smaller, more manageable functions and modules.
- Use comments and documentation sparingly, and only when necessary. Ensure that code is consistent in style and structure throughout the codebase. Avoid using overly clever or complex coding techniques that may be difficult to understand.

Introduction to Code Should be Obvious III

- Overall, the book emphasizes the importance of simplicity, clarity, and maintainability in software design, and making code obvious is a key component of achieving these goals. By writing code that is easy to understand and follow, software engineers can reduce the likelihood of bugs and errors, improve the maintainability of the codebase, and ultimately deliver higher-quality software. [Man14]

Importance for code

A series of light blue, wavy lines that sweep across the bottom half of the slide, creating a sense of motion and design.

Importance for code I

- There are several reasons why it is important for code to be obvious. First, obvious code is easier to understand and maintain. This means that bugs can be fixed more quickly and new features can be added more easily. This can save time and money in the long run.
- Second, obvious code is more robust. When code is obvious, it is easier to test and debug. This means that it is less likely to contain bugs that could cause problems later on. Obvious code is also less likely to break when changes are made to it.
- Third, obvious code is more scalable. When code is obvious, it is easier to refactor and extend. This means that the code can be modified to meet changing requirements without causing problems elsewhere in the system. This can make it easier to add new features or modify existing ones.

How to achieve obvious code?



How to achieve obvious code?

- **Keep it simple**

One of the key principles of good software design is to keep things simple. This means avoiding unnecessary complexity or abstraction. Simple code is easier to understand and maintain, and it is less likely to contain bugs.

- **Use meaningful names**

Good variable names and function names can make a big difference in how easy code is to understand. Names should be descriptive and meaningful, and they should accurately reflect the purpose of the variable or function.

- **Write clear comments**

Comments can be a powerful tool for making code more obvious. Good comments explain what the code does, why it is necessary, and how it works. Comments should be concise and clear, and they should not repeat the code.

- **Follow consistent coding conventions**

Consistent coding conventions can make code easier to read and understand. This means using consistent indentation, formatting, and naming conventions. Consistent conventions can also make it easier to collaborate with other developers.

- **Use modular design**

Modular design is a key principle of good software design. This means breaking code down into smaller, more manageable components. Modular design can make code more obvious by reducing complexity and making it easier to understand.[ZGS18]

Example



Example I

- Let's consider an example of a function that calculates the factorial of a given number:

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n-1)  
  
result = factorial(5)  
print("The factorial of 5 is", result)
```

The factorial of 5 is 120.

The above implementation is recursive and it's quite obvious what it does.

Example II

- However, we can make it more obvious by using a loop:

```
def factorial(n):  
    result = 1  
    for i in range(1, n+1):  
        result *= i  
    return result
```

This implementation is more obvious because it's using a loop instead of recursion, which makes it easier to read and understand. The use of a loop also eliminates the risk of hitting the recursion limit for large values of n .

Example III

- Furthermore, we can make it even more obvious by adding comments and meaningful variable names:

```
def factorial(n):  
    # Set the initial value of result to 1  
    result = 1  
    # Use a for loop to iterate over the range 1 to n  
    for i in range(1, n+1):  
        # Multiply the current value of result by i  
        result *= i  
    # Return the final value of result  
    return result
```

Conclusion



Conclusion I


Code should be obvious is a key concept in good software design. Obvious code is easier to understand and maintain, more robust, and more scalable. Achieving obvious code requires a combination of good design principles and coding practices, including keeping things simple, using meaningful names, writing clear comments, following consistent coding conventions, and using modular design. By following these principles, developers can create software that is easier to understand, maintain, and extend.

References



References I

- [Man14] Sandro Mancuso. *The Software Craftsman: Professionalism, Pragmatism, Pride*. Pearson, 2014. ISBN: 9780134052625.
DOI: 10.1007/978-1-4302-6584-4.
- [Ous18] John K. Ousterhout. *A Philosophy of Software Design*. Addison-Wesley Professional, 2018. ISBN: 978-1732102217.
DOI: 10.1145/3234235.
- [ZGS18] S. Kevin Zhou, Hayit Greenspan und Dinggang Shen. *Deep Learning for Medical Image Analysis*. 1st. Academic Press, 2018. ISBN: 9780128104088. URL: <https://www.sciencedirect.com/book/9780128104088/deep-learning-for-medical-image-analysis>.



Thank you
for your attention