From Idea to Architecture

Subjectively Approaching Clean Code through Elicitation

The Customer

Software systems solve a problem, automate a solution, create something new, or many other things.

!Make very !absolutely !completely sure you know what the customer wants!

Elicitation

Elicitation is used in requirements engineering but also can be a useful tool for creating a robust and stable architecture.

The scope and complexity of the environment your software will be executed is a great step to get started.

Remember to keep in mind elicitation is about asking questions from the customer as well as brainstorming.

Where to Start

For agile development, gathering information from the customer starts with a big picture and the design phase happens almost immediately after this information is given. What problem is the software solving?

To start, we need to separate what can change from what will stay the same (concrete).

This separation begins the first part of design and can almost be based entirely on the “big picture”

When brainstorming an architecture, make sure your design can withstand big changes while being able to grow incrementally.

Your developers will appreciate the fact they can create a new class they know will synergize with your architecture.

Design Patterns

Design patterns can be original and unique to your problem or can be based on patterns made by someone else. (GoF?)

Patterns introduce complexity into the software so use design patterns as tools for decreasing coupling and increasing cohesion and not as a way for the entire software to work.

Clean Design = Clean Code

Anticipating every single change that could ever be required will force you to keep your design simple.

Your modules are meant to be refined and interchangeable with a brand new module.

Intuitively express your intent on how the code is separated and relates to the overall purpose of the software, and developers will passively create clean code.

(Java)

Testing

A clean design requires attention to detail. This means you must be able to develop skeleton code or DSL's that can be tested almost immediately.

Important for agile software development and extreme programming.

Imperative for test-driven software development.

Recap

- Customer has the money
- Elicitation = brainstorming and mind reading
- Start somewhere, start getting concrete
- “Architecture - destruction”
- Design patterns
- Clean code
- Testing