Design Patterns Revisit

Add new features to legacy code through Refactoring, Design Patterns, and Copilot.

โอฬาร อึงอำนวยพร (โอ)

ODDS |

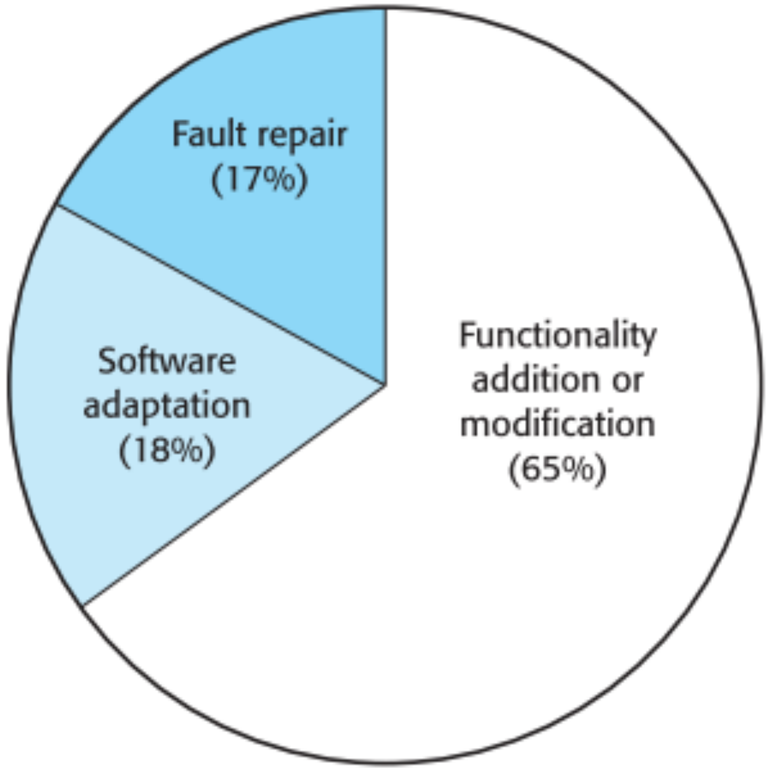
ทำ software ต้องสนุก
เราจะเก่งขึ้นทุกวัน

About me

- Olarn Ungumnuayporn
  olarn.u@gmail.com
- ODDS
  https://web.facebook.com/oddsteam
- Agile Practitioner, Software Developer, Scrum Master, Agile Coach
- Ex-Certified Scrum Master, LeSS Practitioner, Kanban, Flight Levels

Software always change!

if customers / users still want it.

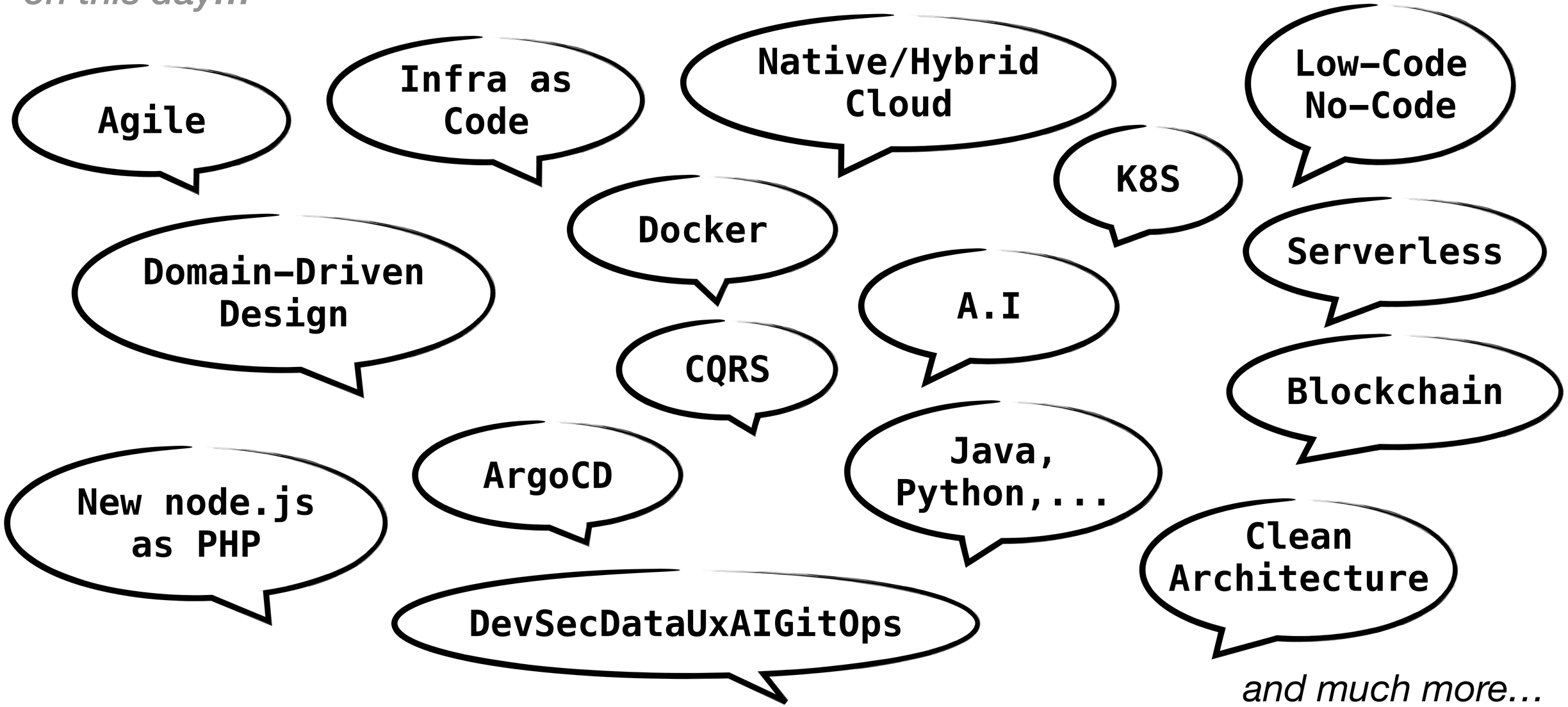


Distribution of Maintenance Effort [15] illustrates that for any software, the changes are always expected and their distribution according to a survey is clarified here thus maintenance is the second major factor for fighting architecture erosion. Lifetime of software is elongated when there is high maintainability thus lowering development risks. Maintainability is also considered as a quality attribute that only focuses on the existing short-range attempts for changes but does not emphasize on the long-standing conservation of the software. For example, the maintainability of a software system can be enhanced by improving the quality of designs and code but it would not have any sufficient affect on the capability of the software for evolvability. Maintenance activities do not consider the structural modifications as they are not involved in the maintainability of the software systems because any addition to software can result in code clones thus reducing maintainability and leading towards architecture erosion. Therefore evolvability should be considered as a separate quality attribute factor necessary for software architecture consistency.

Reference: https://www.researchgate.net/figure/Distribution-of-Maintenance-Effort-15-illustrates-that-for-any-software-the-changes_fig5_271644986

Things we're talking about software development

on this day…



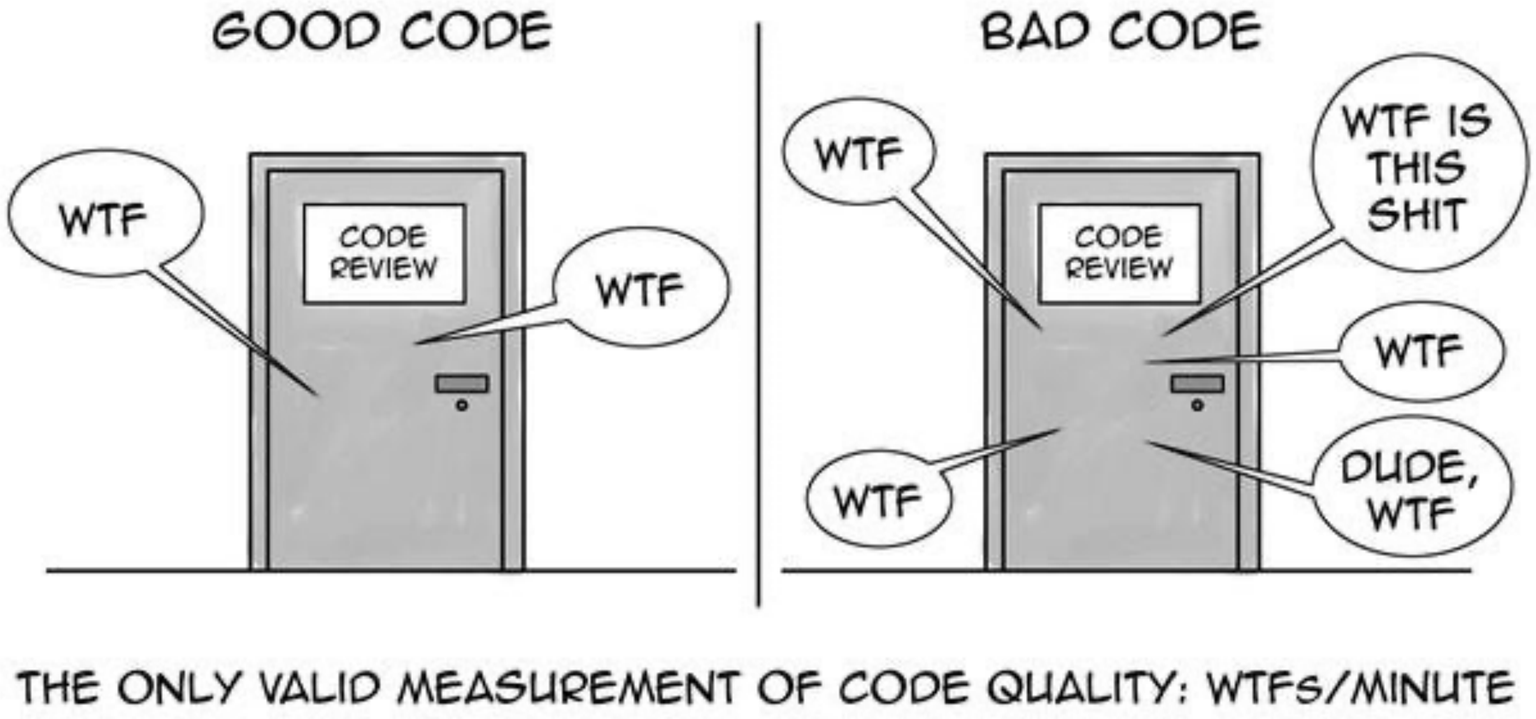
and much more…

…However, one thing that may be unavoidable is …

Legacy Code

Code review…

do you ever…



Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin
https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882

However…

Technical excellence is the responsibility of developers, even in the difficult situations.

Legacy code

will hurt the one who trying to fix it!

How might we…

Code Craftsmanship

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Twelve Principles of Agile Software

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

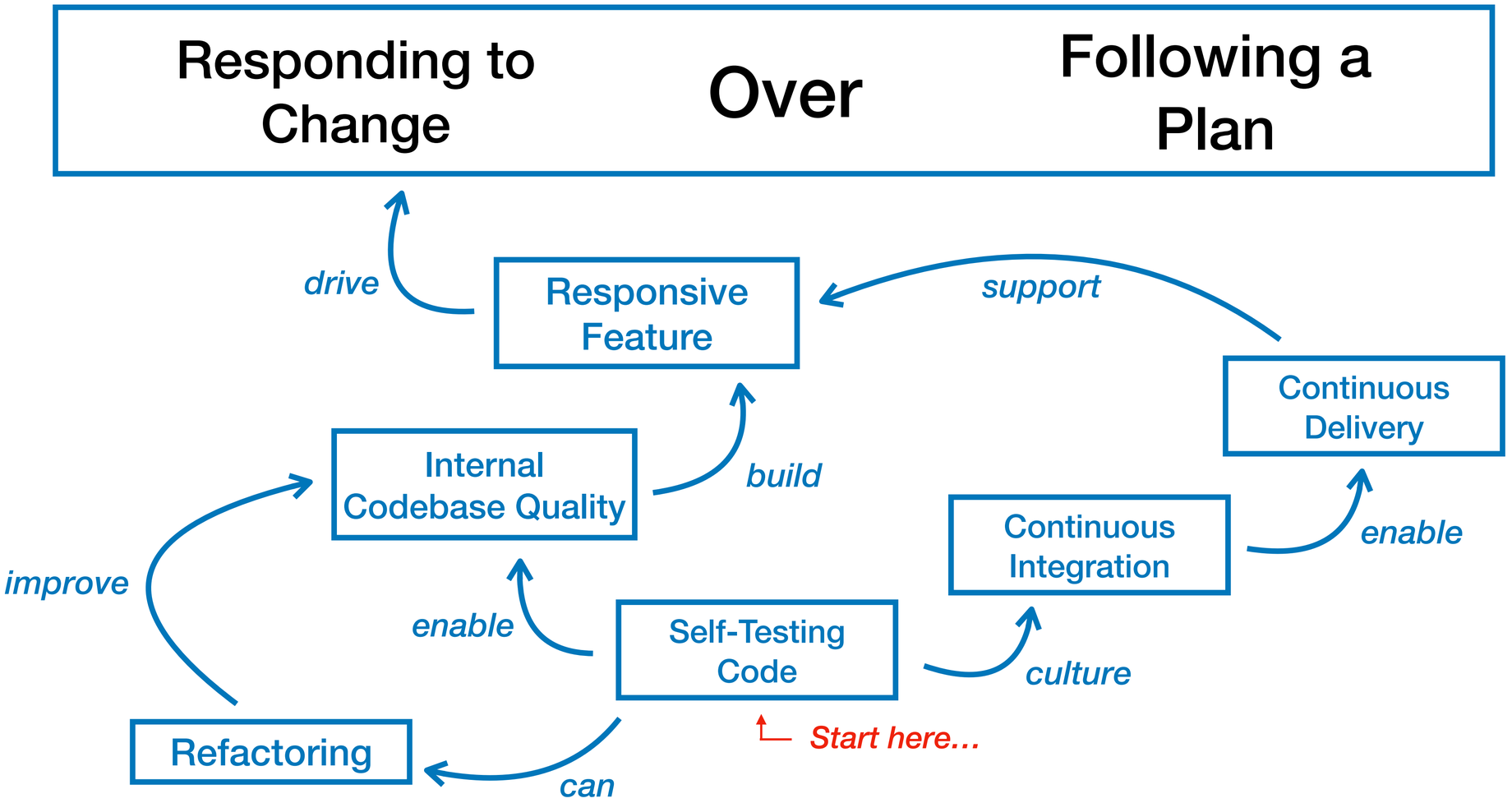
Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

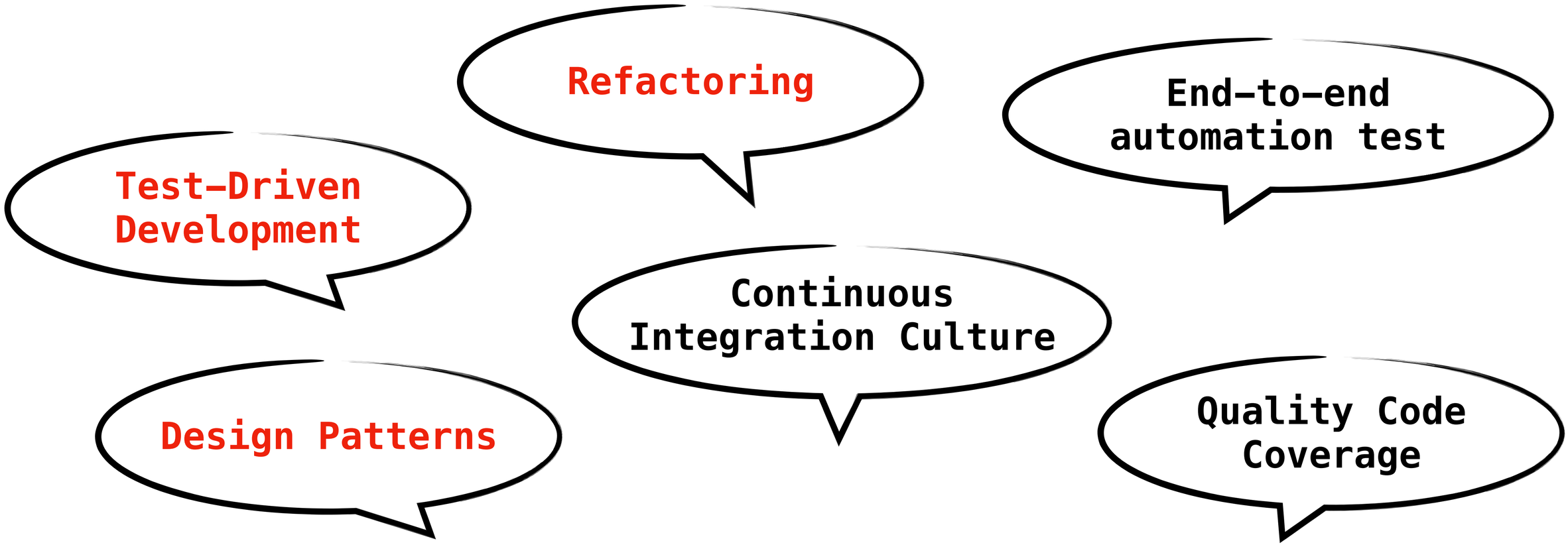
Agility Mentality with Technical Excellence



Martin Fowler - Thoughtworks XConf Bangkok 2018

When you’re dealing with legacy code

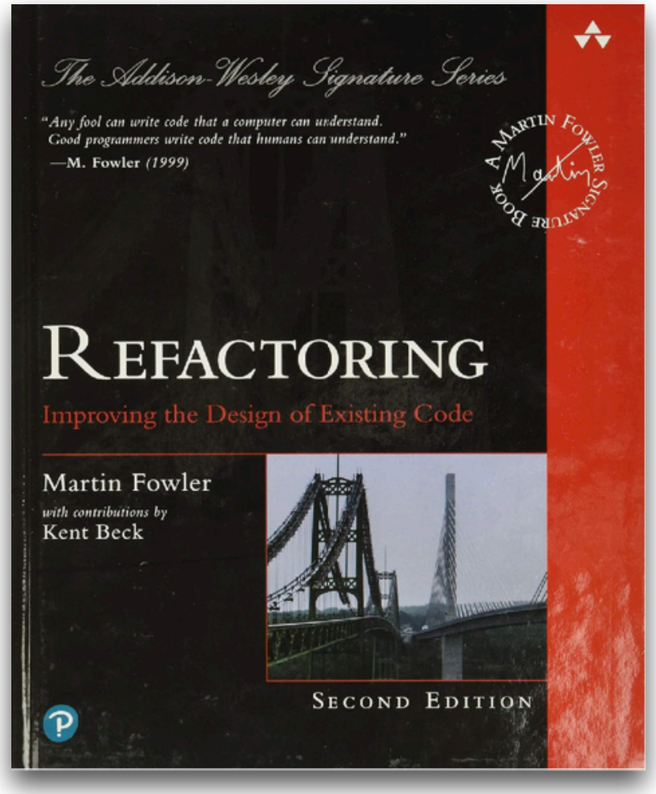
technical excellence is your friend.



Refactoring

require self-tested code.

- Refactoring is the process of restructuring existing computer code **without changing** its **external behavior**.
- Refactoring is a common practice in software development, and it can be used to make code more readable, maintainable, extensible, and efficient.
- Improved code readability and maintainability.
- Increased code extensibility.
- Improved code efficiency.



Martin Fowler with contributions by Kent Back

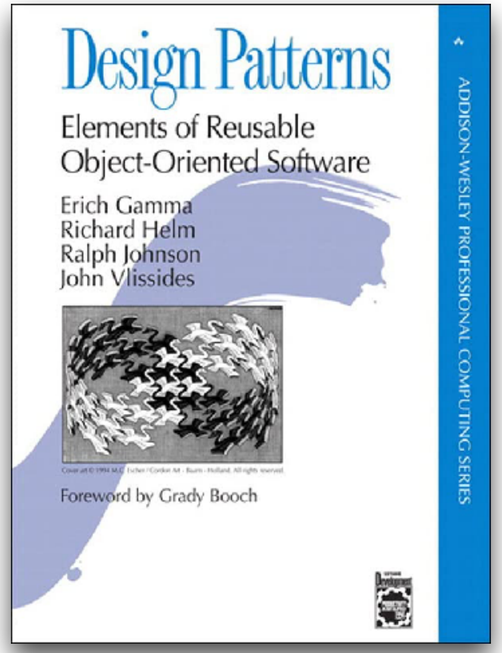
https://shorturl.at/ektMX

GoF Design Patterns

…stand for Gang of Four

- Almost 30 years ago (Oct 31, 1994) the iconic computer science book “Design Patterns: Elements of Reusable Object-Oriented Software” was first published.

- The four authors of the book: Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, have since been dubbed “The Gang of Four”, or GoF.



Demo

let’s Refactoring & use Design Pattern.

```
export enum PlanType {
  UNKNOWN = 'Unknown',
  BASIC = '1 THB per minute. Minimum pay 30THB/month.’,
  VALUE = '0.5 THB/min for first 30 minutes. 1 THB/min thereafter. Min pay 30THB/month.’,

  // new requirements
  NIGHT_PACK = 'free from 6pm to 6am. 1 THB/min thereafter. Min 30 THB/month.’,
  BIZ_VALUE = 'free from 9am to 6pm. 1 THB/min thereafter. Min 30 THB/month.’,
}

export class Billing {
  calculatesMonthlyFee(transactions: Transaction[], plan: PlanType): number { . . . }
```

Demo - The Legacy Code

```
import { Transaction } from './transaction/transaction';
import { PlanType } from './plan/planType';

export class Billing {
  calculatesMonthlyFee(transactions: Transaction[], plan: PlanType): number {
    let totalMinutes = 0.0;
    for (const transaction of transactions) {
      totalMinutes += transaction.totalMinutes();
    }
    const minimumToPay = 30;
    let fee = 0.0;
    if (plan === PlanType.BASIC) {
      fee = totalMinutes;
    } else if (plan == PlanType.VALUE) {
      if (totalMinutes <= 30) {
        fee = totalMinutes * 0.5;
      } else {
        fee = 15 + (totalMinutes - 30);
      }
    } else {
      throw new Error('Unknown plan type');
    }
    if (fee < minimumToPay) {
      return minimumToPay;
    }
    return fee;
  }
}
```

not support for time-basis plan.

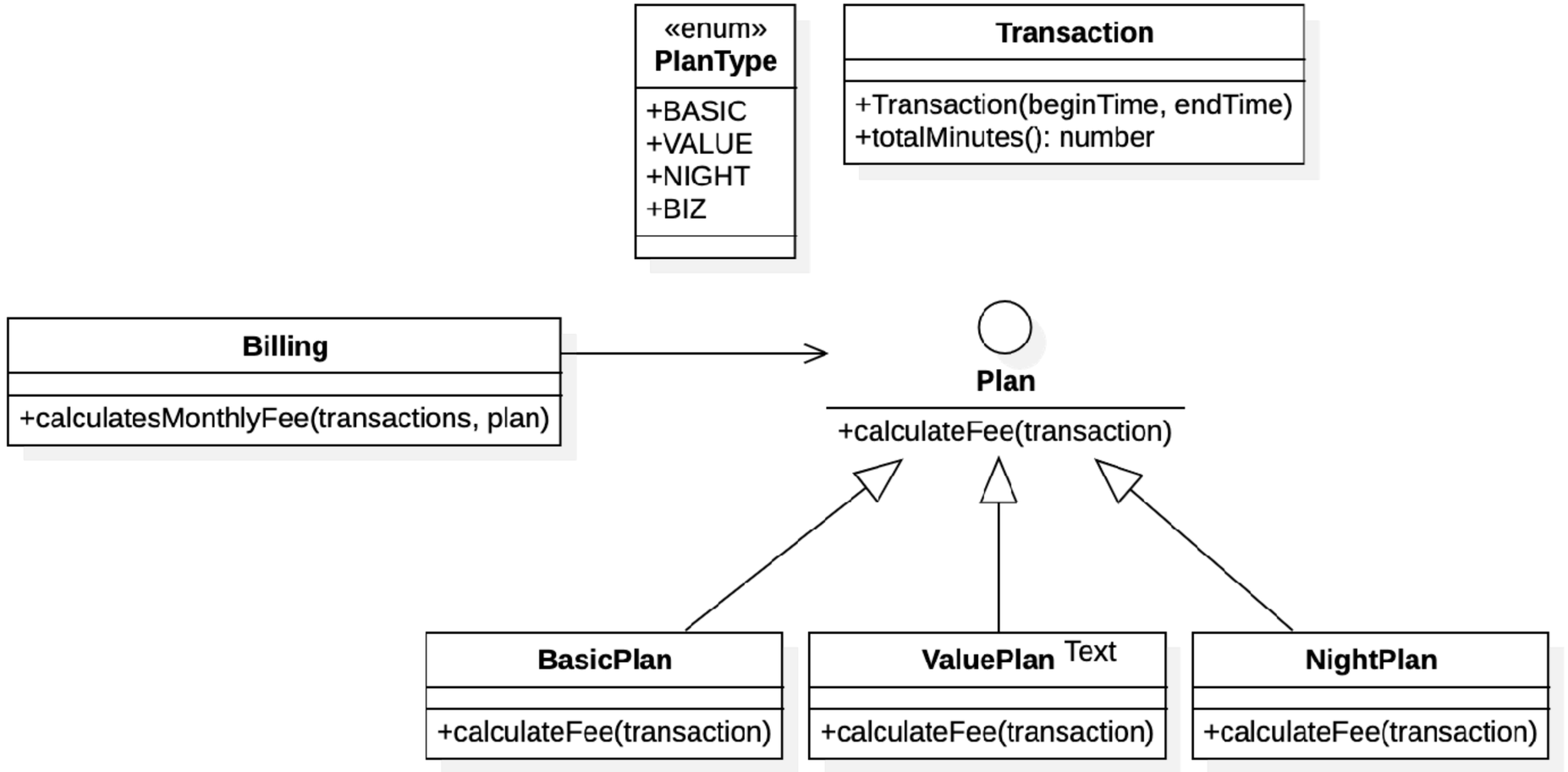
add more plan (features), add more if-else

difficult to read, understand

add one plan, re-test all plan

Demo - Strategy Pattern

in UML Diagram



Conclusion

- Legacy code happens today. Right now when you write bad code.
- Bad code is the code that difficult to add new functionality, fragile, hard to understand.
- Technical excellence is the responsibility of developers, even in the difficult situations.
- Technical craftsmanship take time, effort and practices. Design Patterns is just a small parts of it.

Some tips…

- Build continuous improvement team culture. Be humble, respect, be inspired.
- Learn from someone who know about it. Practice coding kata. bug badge, etc,
- Read books, lots of it.
- Happy coding ❤️

Google Slides

https://shorturl.at/flnM8

GitHub

https://github.com/olarn/GoF-Design-Pattern

Thank you.