SOLID
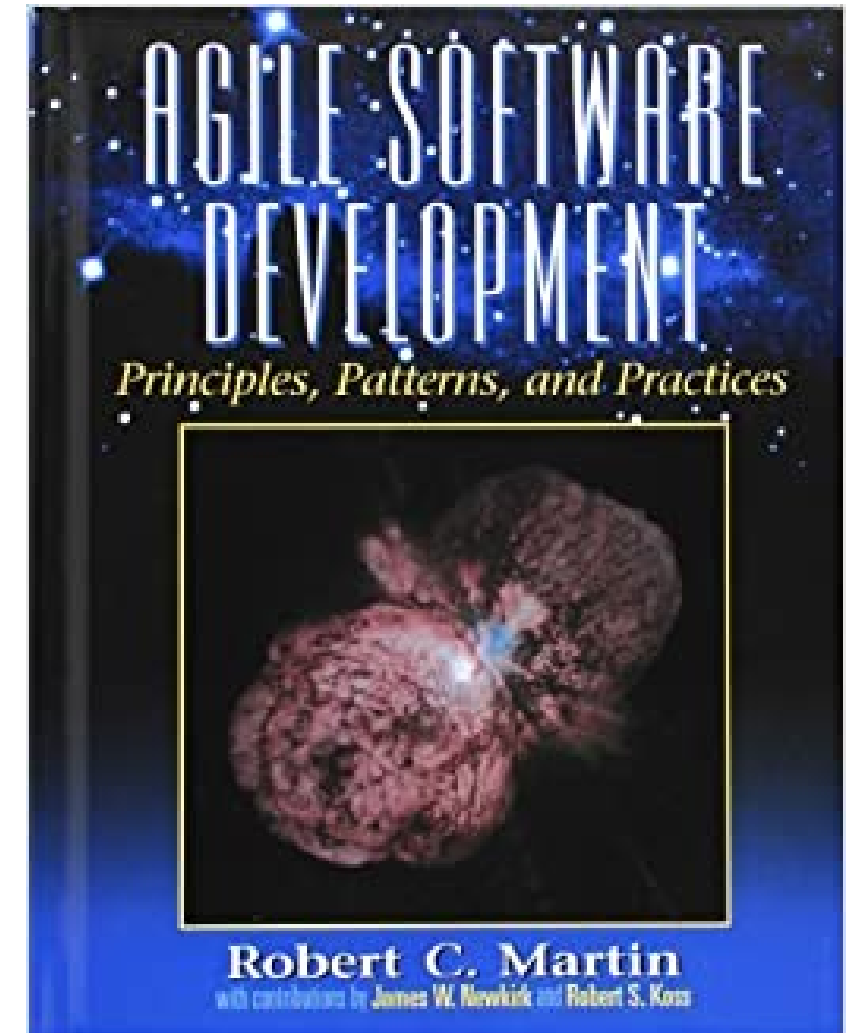
Single Responsibility Principle (SRP)

Single Responsibility Principle

> *"A class should only have one reason to change."* [Robert C. Martin]

Why?

- Increase of Maintainability
- Decrease of Rigidity

Single Responsibility Principle

Probably first mentioned during the 1970's in:
David L. Parnas, *On the Criteria To Be Used in Decomposing Systems into Modules.* Commun. ACM 15(12): 1053-1058 (1972)

It means

- **Separate things...**
  ...which are likely to change because of *different reasons*.
- **Group things...**
  ...which are likely to change because of *the same reason*.

Reasons to change software/code

- **Fixing a bug**
  Old requirements have not been met.

- **Adding a feature**
  New requirements have not (yet) been met.

There exists only a single reason to change software: **Conformance to requirements**.

Where do requirements originate from?

Requirements and Responsibility

The origin of requirements and the meaning of responsibility are related:

Definitions of Responsibility:

> *"(1.) The state or fact of having a duty to deal with something or of having control over someone.*
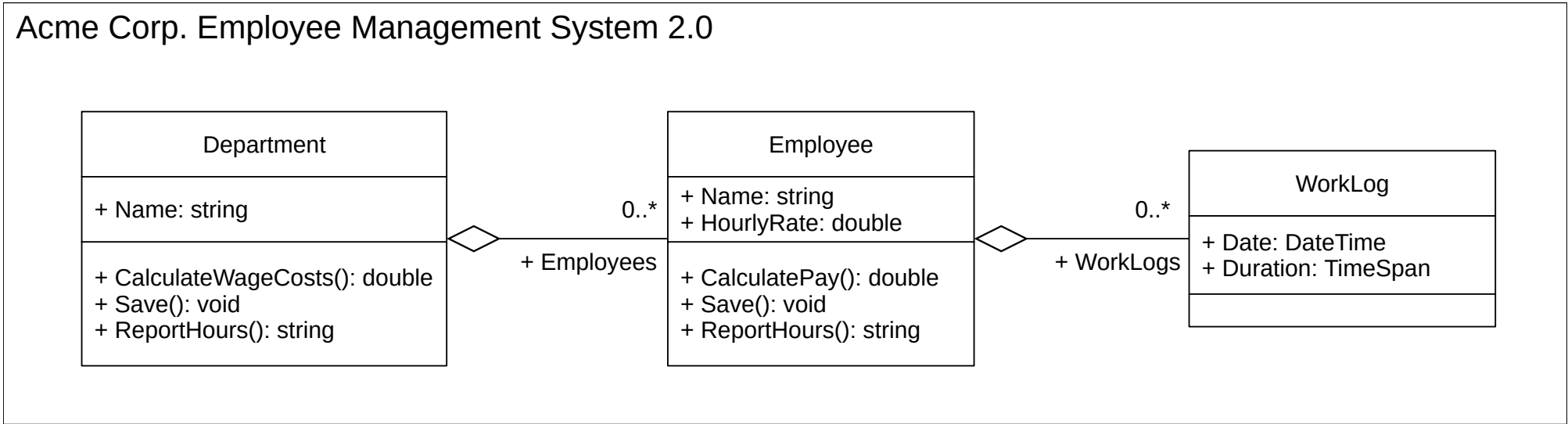> **(2.) The state or fact of being accountable or to blame for something.**
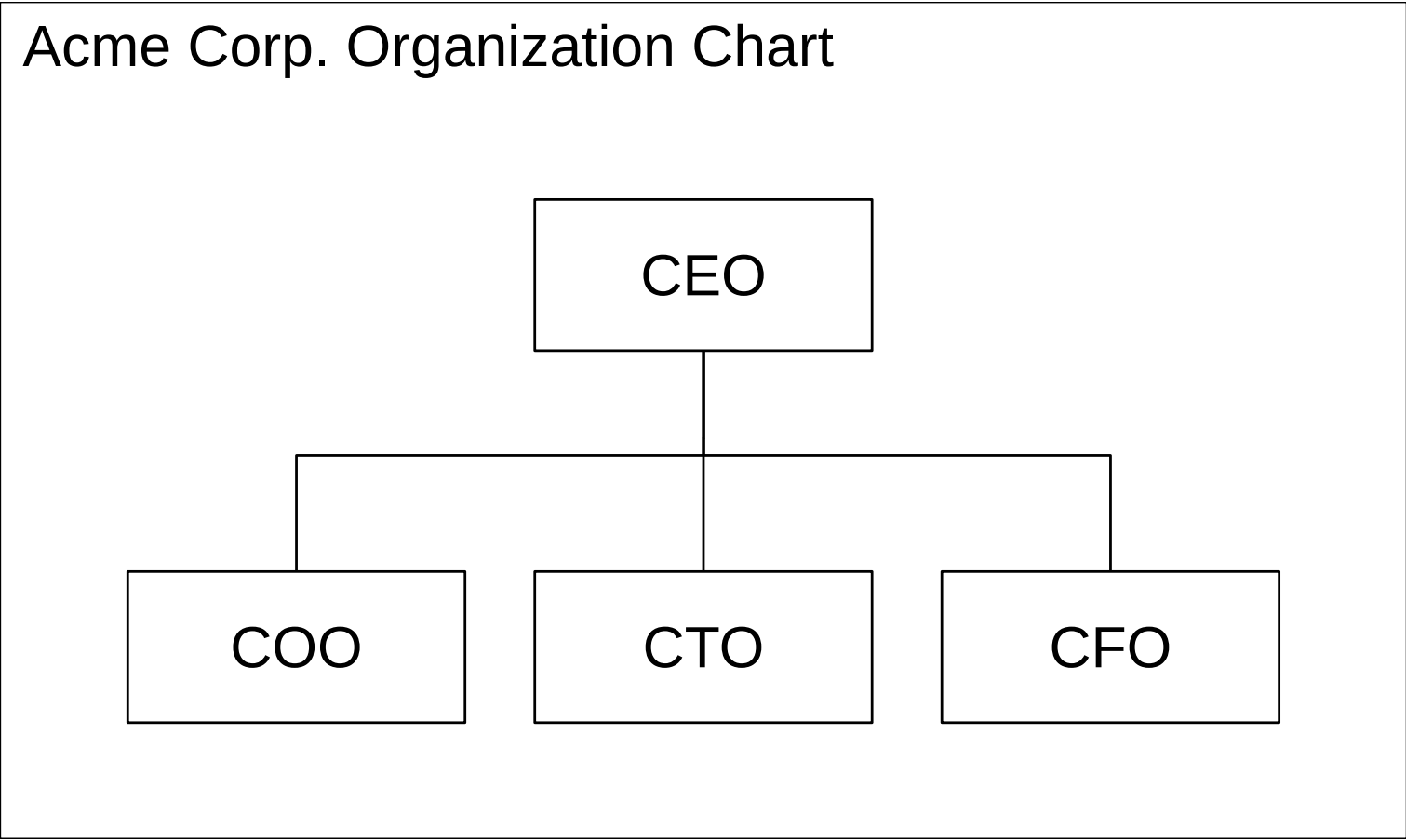> *(3.) The opportunity or ability to act independently and take decisions without authorization."*
> [https://www.lexico.com/definition/responsibility]

- Who to respond to?
- Accountable to whom?

Example: Emplyoee Management System

Acme Corp. Employee Management System 2.0

Department
+ Name: string
+ CalculateWageCosts(): double + Save(): void + ReportHours(): string

0..*
+ Employees

Employee
+ Name: string + HourlyRate: double
+ CalculatePay(): double + Save(): void + ReportHours(): string

0..*
+ WorkLogs

WorkLog
+ Date: DateTime + Duration: TimeSpan

Example: Stakeholders

Acme Corp. Organization Chart

```
                    ┌──────────┐
                    │   CEO    │
                    └──────────┘
          ┌──────────────┼──────────────┐
   ┌──────────┐   ┌──────────┐   ┌──────────┐
   │   COO    │   │   CTO    │   │   CFO    │
   └──────────┘   └──────────┘   └──────────┘
```

Example without SRP

Stakeholders may create *axis of change*.
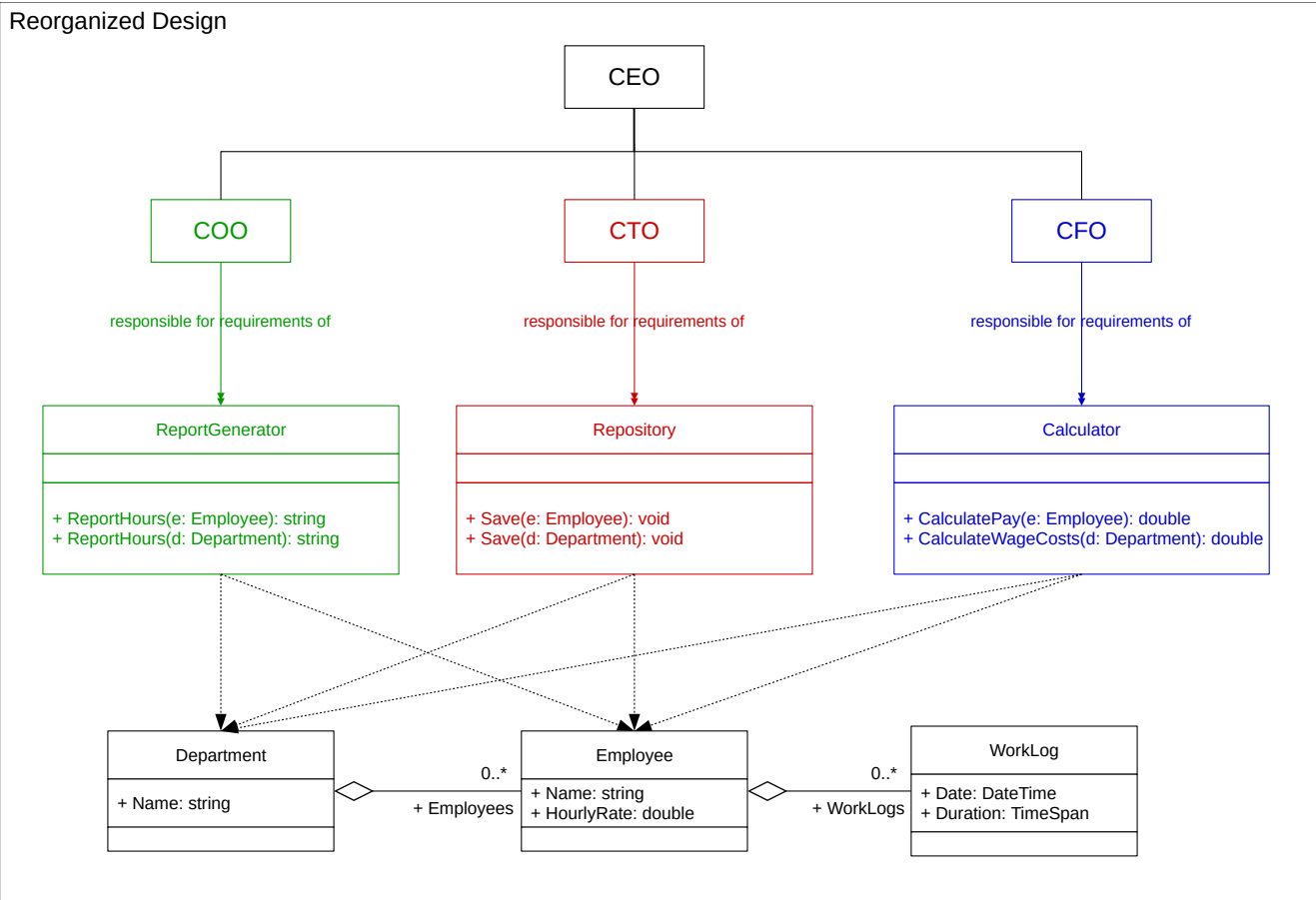
Problem: Coincidental Cohesion

```
class Employee
{
  public double CalculatePay()
  {
    ...
    HelperMethod();
    ...
  }
  public string ReportHours()
  {
    ...
    HelperMethod();
    ...
  }
  private void  HelperMethod()
  {
    ... // CHANGES HERE MAY HAVE UNPREDICTABLE EFFECTS OUTSIDE THE AXIS OF CHANGE!
  }
}
```

A change to `HelperMethod` caused by requirements of the COO stakeholders may break requirements of CFO stakeholders.

Example with SRP

Reorganized Design

CEO

COO — responsible for requirements of

CTO — responsible for requirements of

CFO — responsible for requirements of

ReportGenerator
+ ReportHours(e: Employee): string
+ ReportHours(d: Department): string

Repository
+ Save(e: Employee): void
+ Save(d: Department): void

Calculator
+ CalculatePay(e: Employee): double
+ CalculateWageCosts(d: Department): double

Department
+ Name: string

0..* + Employees

Employee
+ Name: string
+ HourlyRate: double

0..* + WorkLogs

WorkLog
+ Date: DateTime
+ Duration: TimeSpan

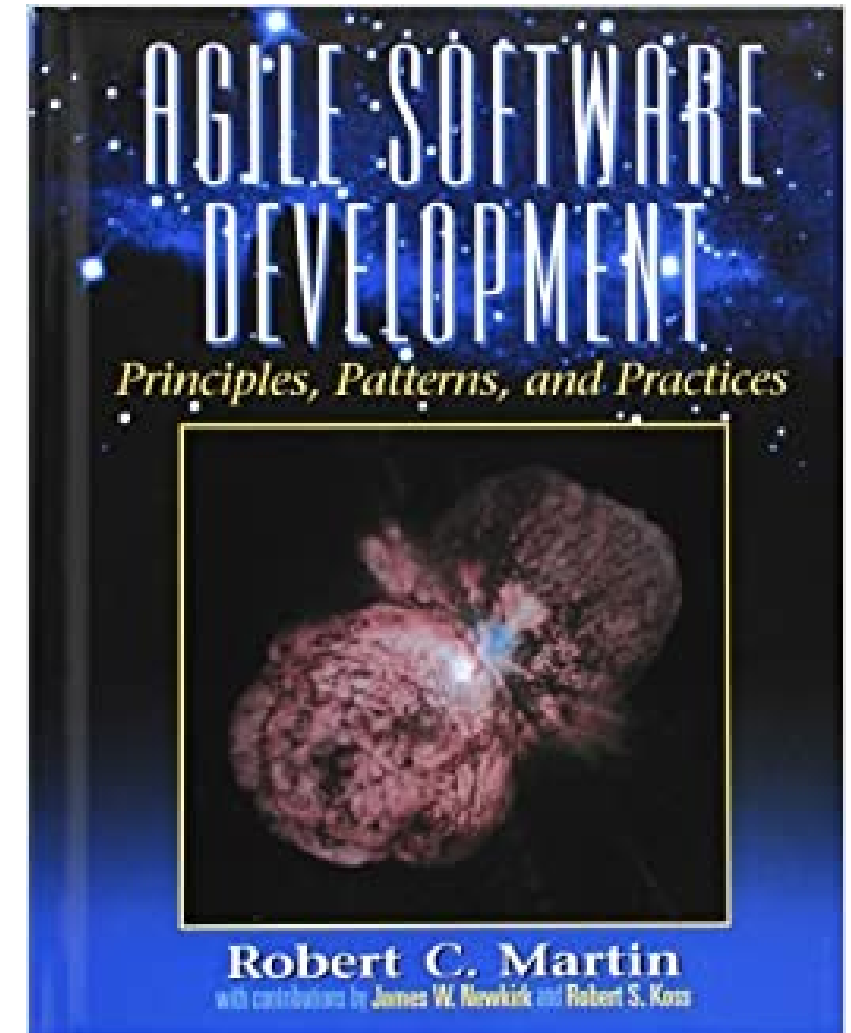# **Example with SRP**

- `ReportGenerator`
  Responds to COO stakeholders.

- `Repository`
  Responds to CTO stakeholders.

- `Calculator`
  Responds to CFO stakeholders.
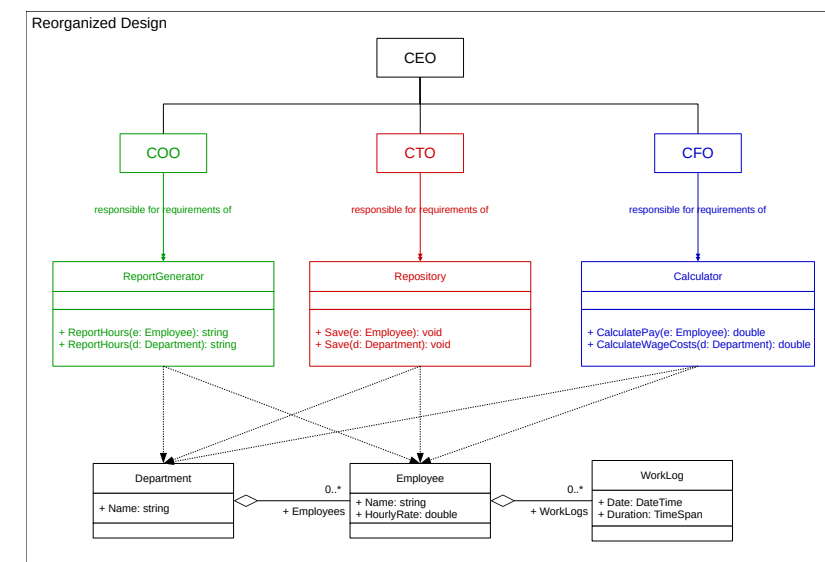
A Word of Warning

> *"An axis of change is an axis of change only if the changes actually occur."* [Robert C. Martin]

The SRP should not be applied without good cause!

Thanks!