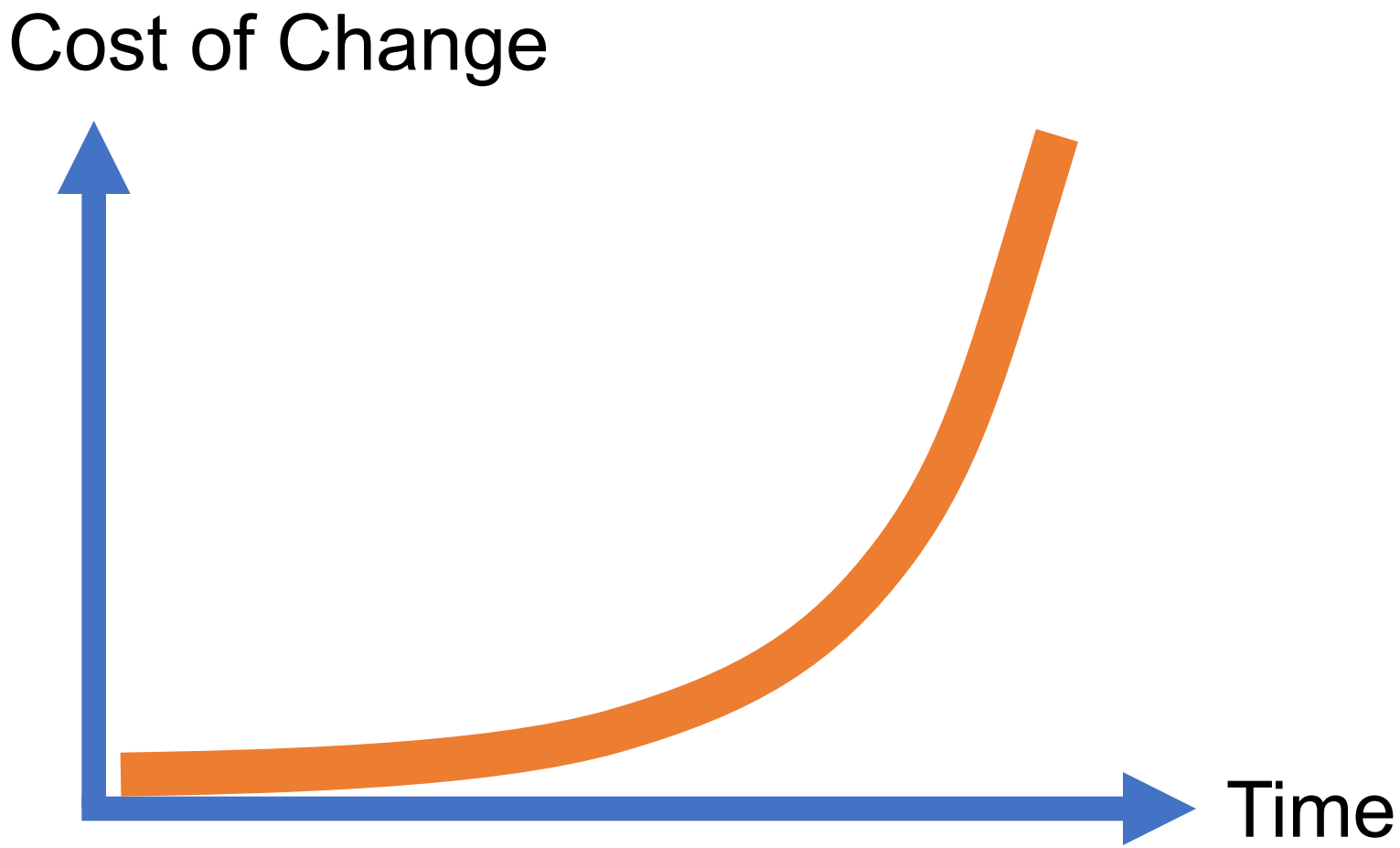
The Code Hoover

a close look on clean code

Andreas Wintersteiger

@awintersteiger | hello@mpirics.com

Why?



Bad Codebase

Fear Driven Development

Hate Driven Development

Ultimate GOAL

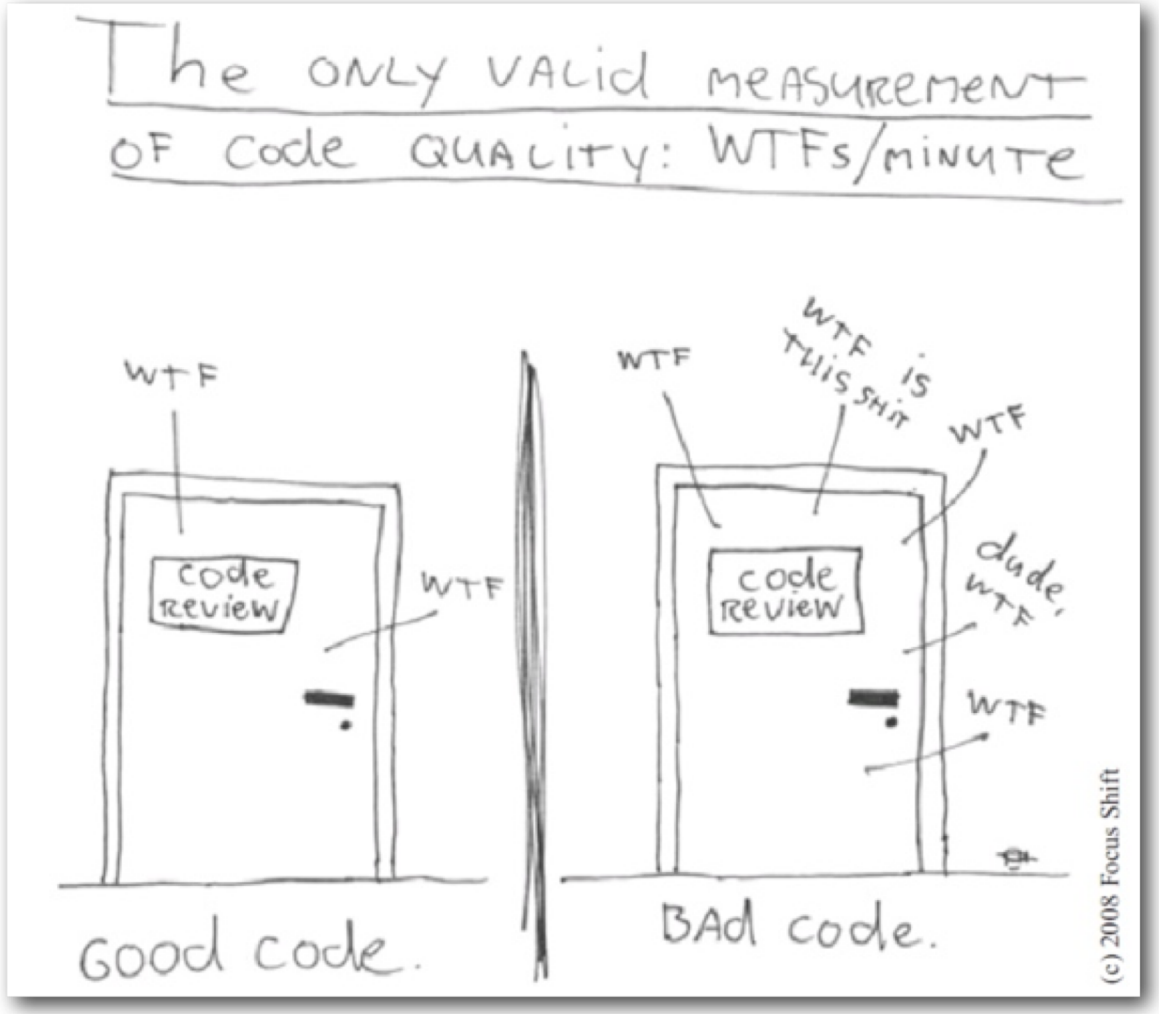
„Changeable“ Software

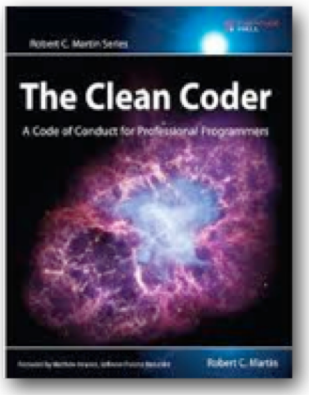
Example 1: what‘s ugly here?

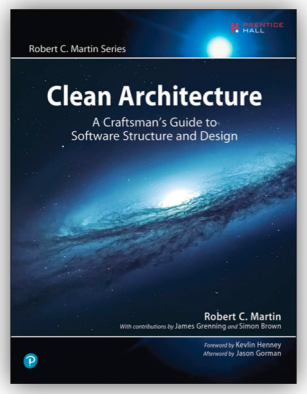
```
private List<NotesUrlParameter> addAuftraege(List<NotesUrlParameter> params, List<Pair<Isin, OrderItemUI>> auftraege, DauerAuftragAnlageUrlKey urlKey) {
    List<Pair<Isin, OrderItemUI>> auftraegeCopy = new ArrayList<>(auftraege);

    Pair<Isin, OrderItemUI> auftrag = auftraegeCopy.remove(0);
    params.add(new NotesUrlParameter(urlKey, buildAuftragValue(auftrag.getKey().getValue(), auftrag.getValue())));

    if (!auftraegeCopy.isEmpty()) {
        return addAuftraege(params, auftraegeCopy, urlKey.getNext());
    }

    return params;
}
```

How would you feel if you are supposed to change that code?

What is „Clean Code“







What resembles „clean“ code?

- Meaningful names
- Methods/Functions
  - Small, obvious and telling a story
  - Do only one thing and do it well without any side effects
  - Command-Query separation
  - niladic and monadic, then dyadic, avoid more than two arguments
  - Avoid output parameters
  - Exceptions instead of error codes
- Comments are failures to express in code

Principles for writing good code - design

- Principle of least Astonishment
- DRY – don’t repeat yourself
- KISS – Keep it simple, stupid!
- Separation of Concerns
- Avoid Premature Optimization
- Favor Composition over Inheritance (FCoI)
- TDA - Tell, don’t ask!
- Law of Demeter
- Information Hiding Principle
- YAGNI – you ain’t gonna need it

Example 2

Code cannot be shown in print mode

Example 3

```
protected boolean allOrderItemsOfWertpapierDescriptionAreProcessed(WertpapierDescription wertpapierDescription) {
        boolean allSwisLinksAreExecuted = true;
        for (OrderItem orderItem : wertpapierDescription.getOrderItems()) {
            if (isBuySellLinkDisabled(wertpapierDescription, orderItem)) {
                continue;
            }
            allSwisLinksAreExecuted = allSwisLinksAreExecuted && orderItem.isProcessed();
        }
        return allSwisLinksAreExecuted;
    }
```

Example 4

```
public void onUpdateMeetingLocation(Meeting meeting) {
    if (!Features.BANK2i_TEAM_SPRINT_93_ORT_DER_BERATUNG_BEI_BANKKGESCHAEFTSRAUM.isActive()) {
        if (!isOtherLocationDetailsRequired(meeting)) {
            meeting.setOtherLocationDetails(null);
        }
    } else {
        meeting.setOtherLocationDetails(null);
    }
}
```

Single Responsibility Principle (SRP)

- First principle of the SOLID-principles aka “cohesion”
- If a class has more then one responsibility, then the responsibilities become coupled. Changes to one responsibility may impair or inhibit the class’ ability to meet the others. This kind of coupling leads to fragile designs that break in unexpected ways when changed.
- Smell: There is more than one reason to change!
- Always separate responsibilities into multiple classes

x

Open Closed Principle (OCP)

> Software entities should be open for extension and closed for modification.

- We can extend the behavior (what it does)
- Extending the behavior does not change the source/binary
- How? We use Abstraction!

Liskov Substitution Principle (LSP)

Subtypes
must be substitutable
for their base types.

- OOPL rely on polymorphism
- If method “m” expects type “A” as its input for processing
- “m” is expected to behave unchanged on subtypes of “A”
- It must not know about specifics in subtypes
- It must not break with special cases

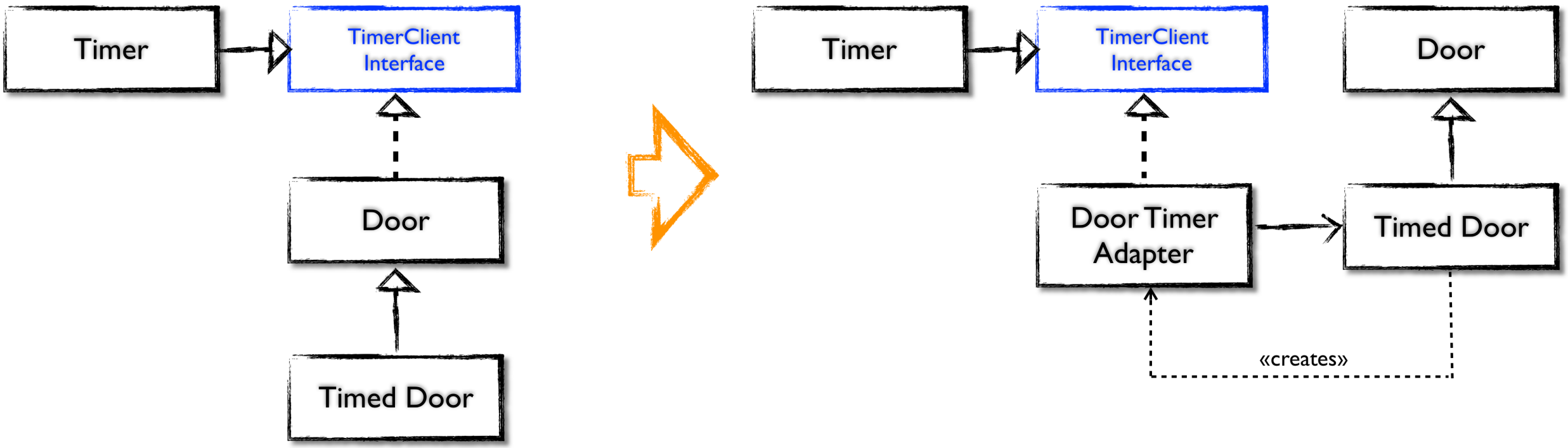
Interface Segregation Principle (ISP)

- Clients should not be forced to depend on methods they don't use.

- Problem:
  - Classes whose interfaces are not cohesive have "fat" interfaces
  - Fat interfaces have groups of methods serving different clients
  - Separation of interfaces is often not possible

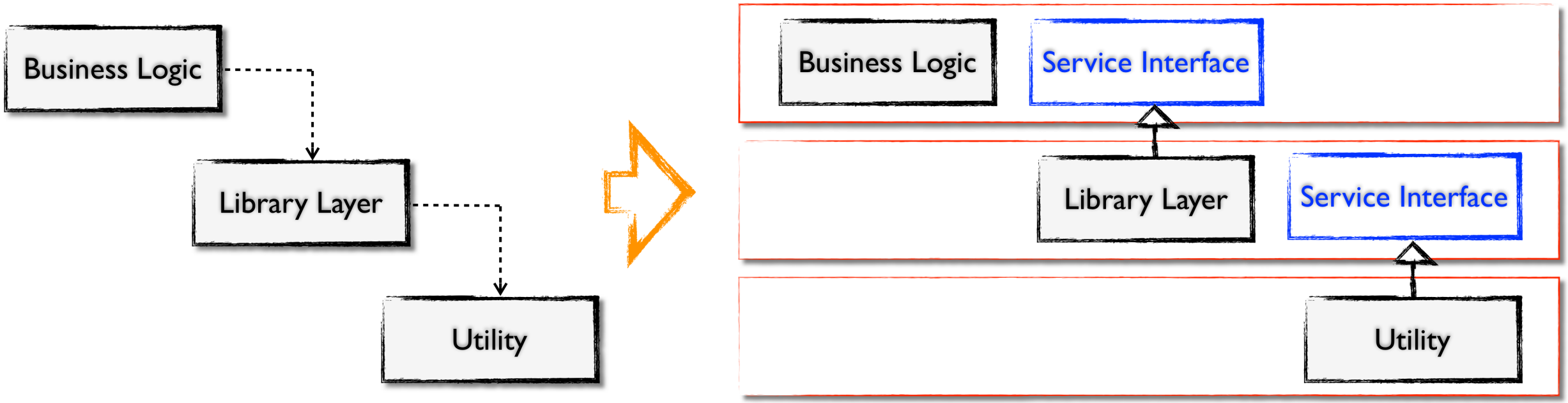
X

Use Delegation to conform ISP



X

Dependency Inversion Principle



Example 5

```ruby
def entry_date
  employment_terms.order(:sequence_nr).first.try(:from_date)
end

def exit_date
  employment_terms.order(:sequence_nr).last.try(:to_date)
end
```

Example 6

```ruby
module TimeZoneService
  extend ActiveSupport::Concern

  class_methods do

    def employees_having_midnight_now
      midnight_zones_ids = TimeZoneInfo.where(iana_name: timezones_having_midnight_now).pluck(:id)
      employee_ids = EmployeeSetting.where(default_time_zone_id: midnight_zones_ids).pluck(:employee_id)
      Employee.active.where(id: employee_ids)
    end

    def timezones_having_midnight_now
      ActiveSupport::TimeZone.all.find_all { |time| time.now.hour.zero? }.map{ |tz| tz.tzinfo.name }.uniq
    end

  end

end
```

Example 7

```ruby
# returns Employees of one or more departments
# Parameters:
#   - department_ids: an Array of department ids
#   - heads: boolean value if heads should be returned
#   - members: boolean value if members should be returned
#   - only_active_employees: boolean value if only active employees should be returned
# if heads and members are not passed as arguments, head AND members will be returned
def self.assigned_employees_for(department_ids:, heads: false, members: false, by_date: Date.current, only_active_employees: true)
  query = only_active_employees ? Employee.active(by_date).joins(:department_assignments) : Employee.joins(:department_assignments)
  query = query.where(department_assignments: { department_id: department_ids })
  query = query.where("department_assignments.from_date <= :by_date and (department_assignments.to_date >= :by_date or \
        department_assignments.to_date is null)", by_date: by_date) if only_active_employees
  query = query.where(department_assignments: { head: (heads || !members) }) if ((heads && !members) || (!heads && members))
  query = query.order(:last_name, :first_name)
end
```

Clean Code is essential, put it into the center!

... but only one small contribution to becoming agile:

- Coding Culture (e.g. Broken Window Syndrome)
- Test Automation & TDD
- Continuous Refactoring
- Measure technical debt
- Hoover the code daily!

TRANSPARENCY ABOUT THE CODE BASE‘S STATUS