Refactoring

based on *Code Complete* by Steve McConnell

Philipp Trucksäß

June 17, 2015

All successful software gets changed.

—Fred Brooks

All successful software gets changed.

—Fred Brooks

Myth

Software is written once and works flawlessly

All successful software gets changed.

—Fred Brooks

Myth

Software is written once and works flawlessly

Reality

Code evolves and changes dramatically during development

Why
is refactoring necessary

All successful software gets changed.

—Fred Brooks

Myth

Software is written once and works flawlessly

Reality

Code evolves and changes dramatically during development

Solution

Refactor to adapt to changes and improve the overall quality

Cardinal Rule of Software Evolution

Internal quality should improve with code evolution

Why
is refactoring necessary

Cardinal Rule of Software Evolution

Internal quality should improve with code evolution

Method

The tool to achieve this: Refactoring

Cardinal Rule of Software Evolution

Internal quality should improve with code evolution

Method

The tool to achieve this: Refactoring

> *[Refactoring is] a change made to the internal structure of the software to make it easier to understand and cheaper to modify without changing its observable behavior.*
>
> —*Fowler(1999)*

Code Smells

Signs that code is bad or has degenerated through changes

Code Smells

Signs that code is bad or has degenerated through changes

Duplicate Code

Some C++ Code:

```
Vector3D normalizedA=a/sqrt(a.x*a.x+a.y*a.y+a.z*a.z);

                                ⋮

Vector3D normalizedB=b/sqrt(b.x*b.x+b.y*b.y+b.z*b.z);
```

Code Smells

Signs that code is bad or has degenerated through changes

Duplicate Code

Refactored C++ Code:

```
Vector3D normalizedA=a.normalized();
Vector3D normalizedB=b.normalized();

                                  ⋮

Vector3D Vector3D::normalized(){
   return this/sqrt(x*x+y*y+z*z);
}
```

Code Smells

Signs that code is bad or has degenerated through changes

Setup & Takedown Code

Some C++ Code:

```
WithdrawalTransaction withdrawal;
withdrawal.SetAccountBalance(accountBalance);
withdrawal.SetWithdrawalAmount(withdrawalAmount);
ProcessWithdrawal(withdrawal);
accountBalance = withdrawal.GetAccountBalance();
withdrawalAmount = withdrawal.GetWithdrawalAmount();
```

Code Smells

Signs that code is bad or has degenerated through changes

Setup & Takedown Code

Refactored C++ Code:

```
ProcessWithdrawal(accountBalance,withdrawalAmount);
```

Code Smells

Signs that code is bad or has degenerated through changes

Many Parameters

Some C++ Code:

```
draw(box.getNumberVertices(),box.getVertices(),
    box.getIndices(),box.getVertexColors(),
    box.getModelMatrix());
```

Code Smells

Signs that code is bad or has degenerated through changes

Many Parameters

Refactored C++ Code:

```
box.draw();
```

80/20 Rule

Do the 20% of possible refactorings that provide 80% of the benefit

80/20 Rule

Do the 20% of possible refactorings that provide 80% of the benefit

When

- After additions
- After fixes

80/20 Rule

Do the 20% of possible refactorings that provide 80% of the benefit

When

- After additions
- After fixes

Targets

- High complexity
- Error-prone

There is no code so big, twisted, or complex that maintenance can't make it worse

—Gerald Weinberg

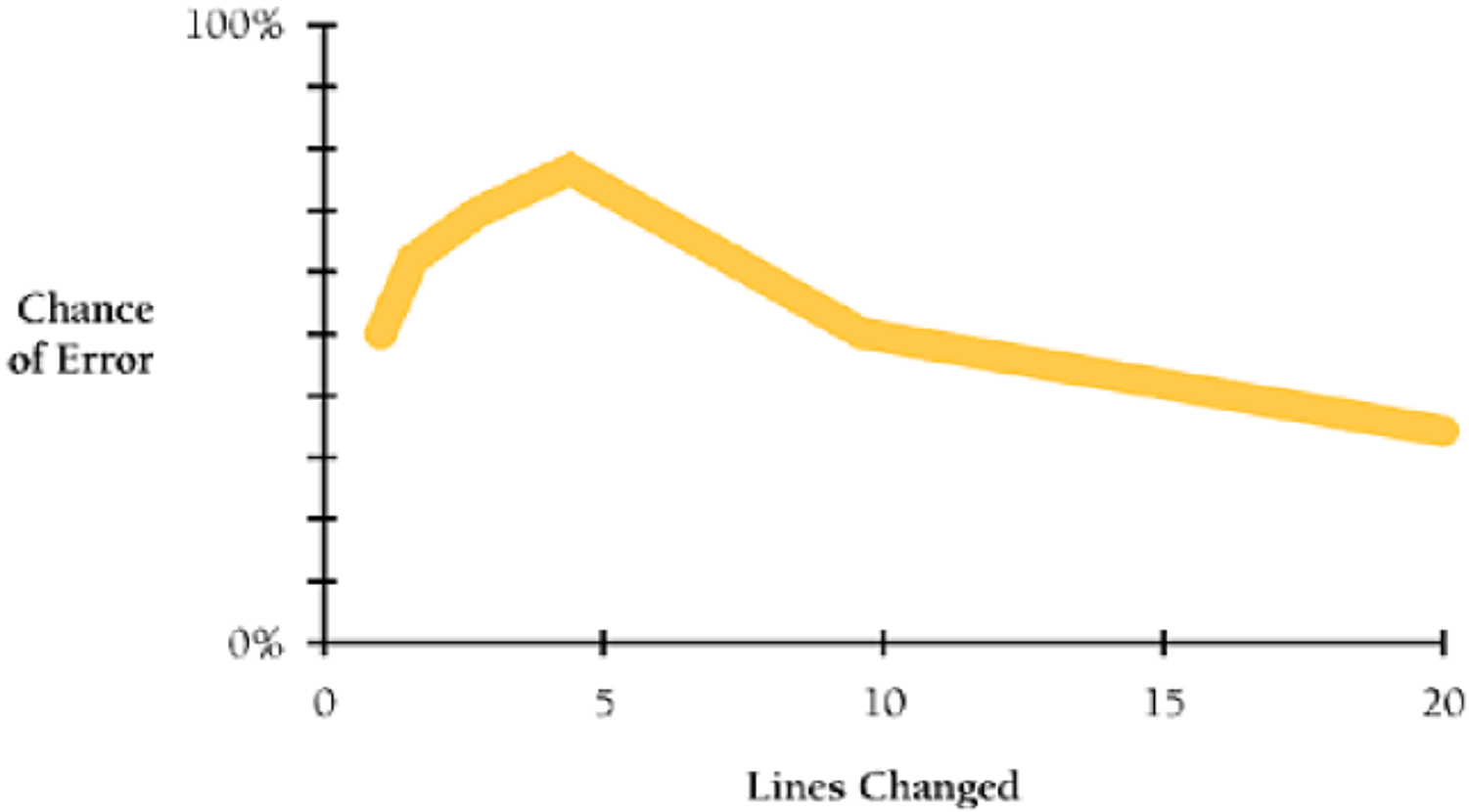
There is no code so big, twisted, or complex that maintenance can't make it worse

—*Gerald Weinberg*

Refactoring Safety

Refactoring can cause more harm than good. Take precautions to prevent missteps

How
to refactor safely



—Weinberg(1983)

How
to refactor safely

- Backup

How
to refactor safely

- Backup
- Small changes

How
to refactor safely

- Backup
- Small changes
- One at a time

How
to refactor safely

- Backup
- Small changes
- One at a time
- "Parking lot"

How
to refactor safely

- Backup
- Small changes
- One at a time
- "Parking lot"
- Retest & Review

- Backup
- Small changes
- One at a time
- "Parking lot"
- Retest & Review

Bad Times to Refactor

Refactoring does not affect a program's behavior. Tweaking broken code to make it work is hacking.

Conclusion

Refactoring during development is the best chance you'll get to improve your program, to make all the changes you'll wish you'd made the first time.

—Steve McConnell