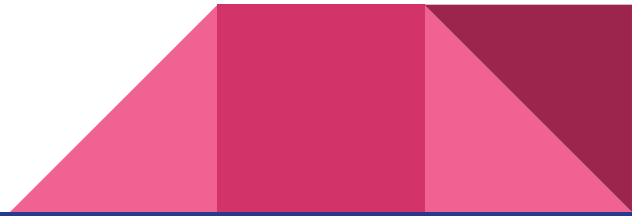
Effective Code Reviews

David Forshner
DForshner@gmail.com

Why have code reviews?

What are some benefits of code reviews?

- Knowledge sharing
- Proofreading – Hard to spot problems when reading your own work.
- Redefining “done” – The code both works and someone unfamiliar with the work can figure out what it does.
- Converge towards a common set of language features, idioms, libraries, design patterns, etc.
- Move away from personal/team ownership of “the code”.



What are some limitations of code reviews?

- Only seeing a tiny facet of a much larger system.
- Hard to see higher level (class/project) duplication of functionality/data.
- Reviewers outside of a team probably can’t spot problems with domain/business logic.
- Probably too late to address design and/or major structural problems.

What are some anti-patterns?

- **Bikeshedding/Wadler's law** - Getting stuck on trivial stuff while larger issues are ignored.
- **Intellectual Violence** – “someone who understands a theory, technology, or buzzword uses this knowledge to intimidate … may happen inadvertently due to the normal reticence of technical people to expose their ignorance.”
- Focusing on how you would have done it.
- Public shaming

How to Find Issues

Where to spend your time

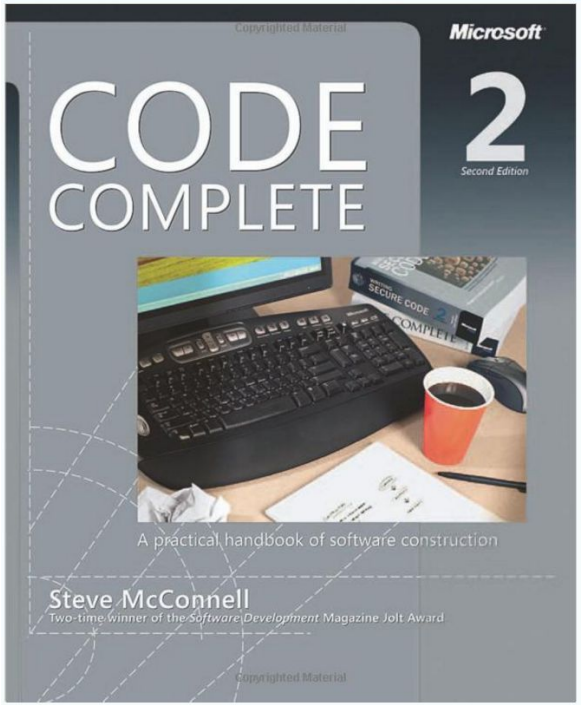
I have no idea what this does

- Focus on changed code in isolation.
- Can I get a rough idea of what this code is doing?
- Are there confusing names or comments?
- Are there opportunities to simplify and/or reduce the amount of code.
- Are there style problems with syntax, indentation, spacing, etc.

I know this domain/project

- Focus on how the change fits with the rest of project.
- Does the class/interface design fit with the rest of the project?
- Are there errors in the domain(business) logic?
- Is there duplication?
  - Functionality
  - Information (single source of truth)
- How is this going to perform?
  - Ex: How many database/network calls are triggered each time this runs?

Read a book/take a course



Pretend to be: A maintenance programmer

- Magic

```
return -44;  // \_(ツ)_/¯
```

- Confusing or misleading naming/comments

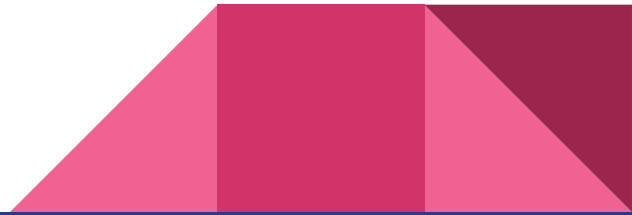
```
// Get from DB or web service
Order cat = CreateCustomer();
```

- Less lines of code!

```
return vals.Select(x => x.ToUpper()).Where(x => validVals.Con…
```

- Precedence/order of operations puzzles

```
if ({ a == true || { { { b == true && c == false } || { c = …
```



Pretend to be: A computer

- Are there execution paths that lead to unexpected states/edge cases?

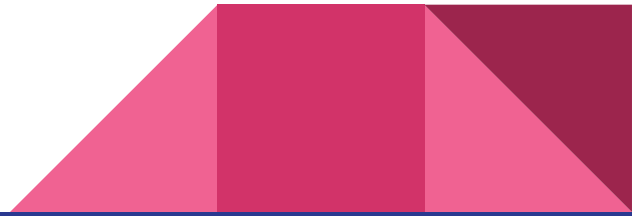
```
var cat = GetCats.FirstOrDefault();
var name = cat.Name;
```

- Are there misleading checks?

```
var cats = GetCats.ToList();
if (cats == null)
```

- Is it “efficient enough”?

```
foreach(var cat in allTheCatsInTheEntireWorld) {
  Owner servant = context.FindById(cat.Id).Owner;
  Address palace = context.FindById(cat.Id).Address;
```



Pretend to be: Someone who unit tests

- Can I fake or mock the dependencies?

```
public Cat MethodIWantToTest(CatShow show) {
  var today = DateTime.Now;
  Cat current = StaticGlobalSingleton.GetWinner(show.Id, date);
  var lastyear = today.AddYear(-1);
  Cat previous = new HistoryDbAccessThing.GetWinner(show.Id, today);
```

- Can I tell why they wrote this test?

```
[TestMethod]
public void RadiationLevel() {
  ReactorControl a = new ReactorControl(0.4443, “S2");
  Assert.AreEqual(0.042, a.RadiationLevel);
```

How to Comment

Constructive Criticism

Prefer

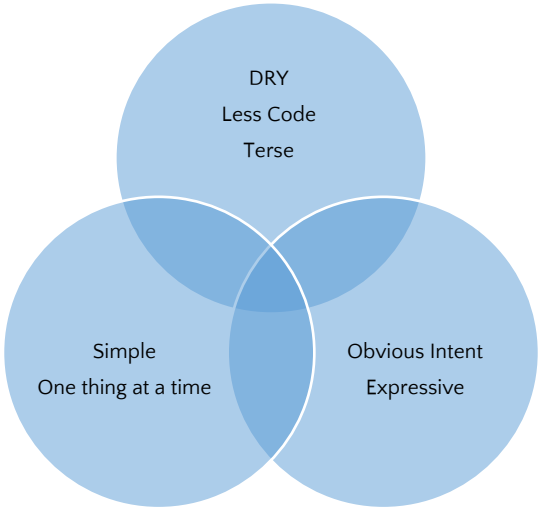
- Impersonal – This/the code
- Explaining the “why” – Consider X because Y
- Providing examples
- Asking questions - Is this used?

Avoid

- Personal – You/your code
- Focusing on what people can and cannot do (enforcement).
- Implying someone “should know this”

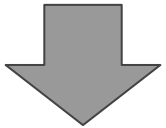
Selling it (encouraging laziness)

- Two facts I made up:
  - 80% of code will need multiple changes over its lifetime.
  - 80% of developers can’t remember what they wrote a month ago
- Long term laziness



One-liner

```
var bobCat = null;
foreach(var cat in cats) {
        if (cat.Name == “Mr. Bob")
                bobCat = cat;
}
```



```
var bobCat = cats.FirstOrDefault(x => x.Name == “Mr. Bob");
```

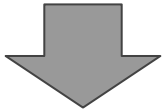
Simpler/Less code/More obvious

```
var food = null;
if (cat.Age < ONE_YEAR)
        food = new KittenStuff();
else
        food = new RegularStuff();
```

```
var food = (cat.Age < ONE_YEAR) ? new KittenStuff() : new RegularStuff();
```

Show intent/reduce coupling

```
if (cat.Type == 3 && cat.Status == "H")
```



```
if (cat.Type == CatTypes.PUREBRED && cat.Status == HEALTHY)
```

```
if (cat.isEligibleForShow)
```

End

(did I mention a good book?)