



# EFFECTIVE CODE REVIEW

DAVE LIDDAMENT

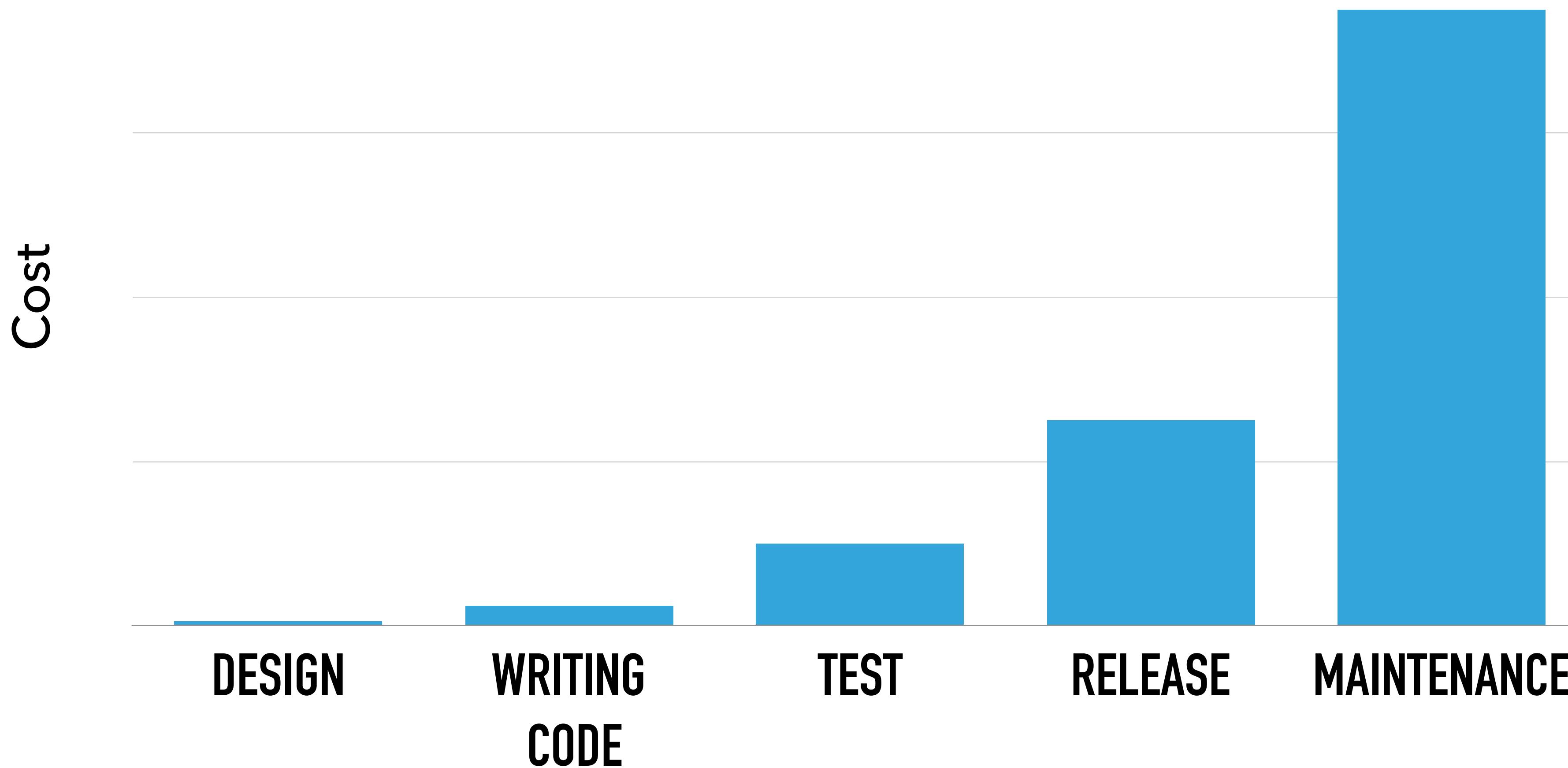
Lamp Bristol

# EFFECTIVE CODE REVIEW REDUCES OVERALL COST OF SOFTWARE DEVELOPMENT

## WHY THIS TALK?

---

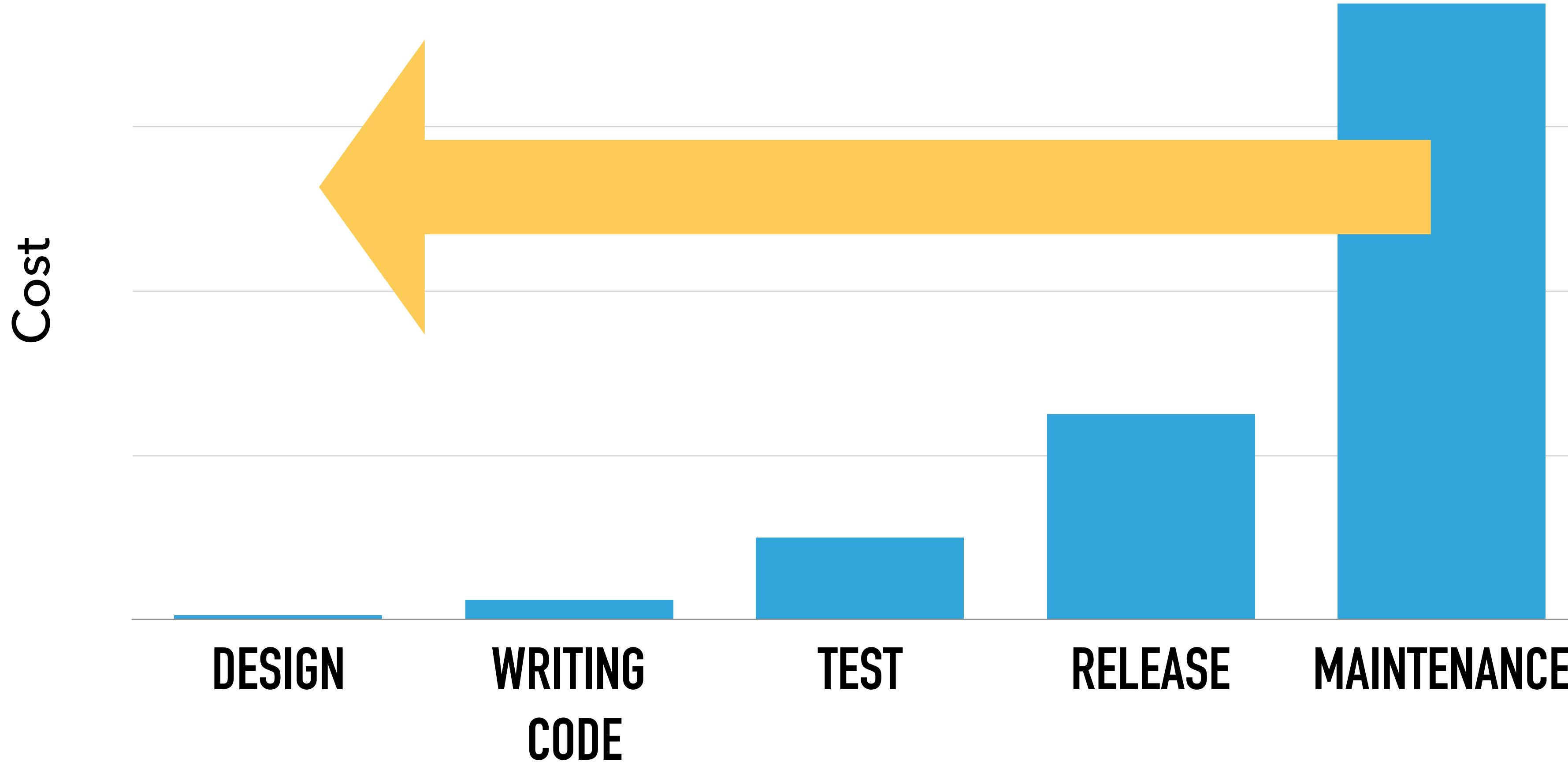
# REDUCE COST OF BUG – FIND IT SOONER

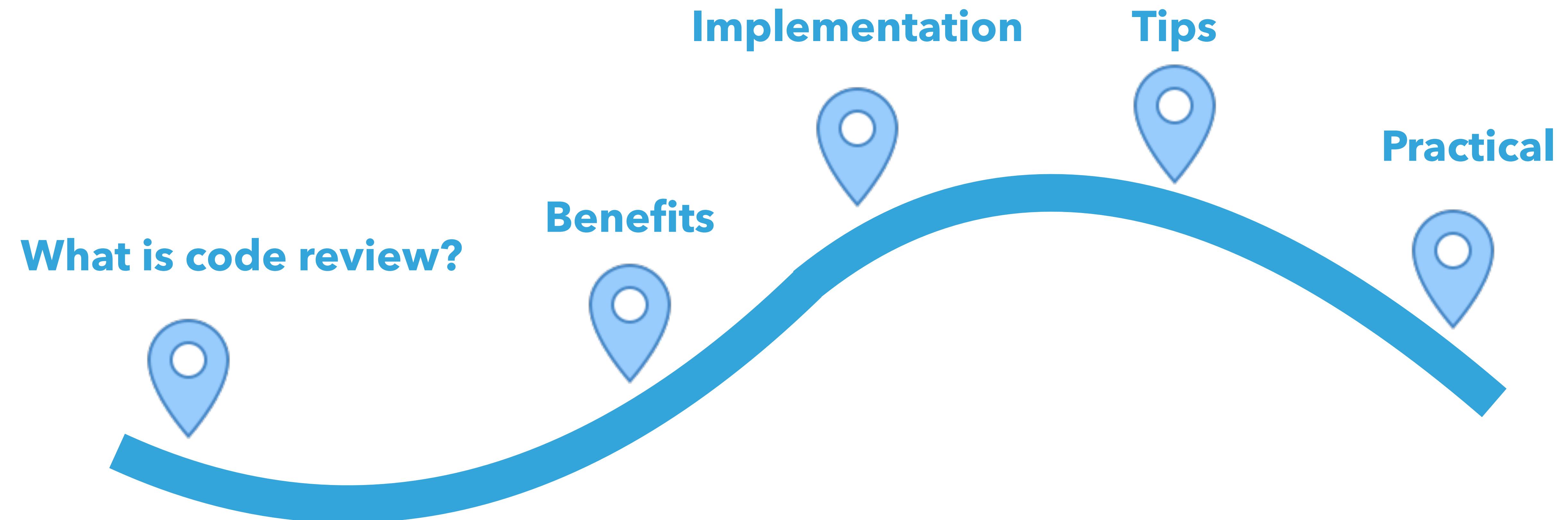


## WHY THIS TALK?

---

# REDUCE COST OF BUG – FIND IT SOONER





**What is code review?**



## WHAT IS CODE REVIEW?

---

CODE REVIEW IS THE  
SYSTEMATIC EXAMINATION OF  
SOURCE CODE...

**Wikipedia**

IT IS INTENDED TO FIND MISTAKES  
OVERLOOKED IN SOFTWARE  
DEVELOPMENT, IMPROVING THE  
OVERALL QUALITY OF SOFTWARE.

Wikipedia

WHAT IS CODE REVIEW?

---

HOW IS IT DONE?

WHAT IS CODE REVIEW?

---

## HOW IS IT DONE?



# WHAT IS CODE REVIEW?

---

## HOW IS IT DONE?



A photograph of two people, a man and a woman, sitting at a desk and looking at a computer monitor. The woman is pointing at the screen, which displays a GitHub pull request interface.

The GitHub pull request interface shows:

- Title: FEATURE add sponsor type to sponsor #46
- Status: Merged
- Author: opdavies
- Merged On: Feb 26
- Conversation: 0
- Commits: 2
- Files changed: 15
- Changes from 1 commit
- Jump to... +116 -18
- Unified | Split | Review changes
- Commit details:
  - FEATURE add sponsor type to sponsor (develop #46)
  - DaveLiddament committed on Feb 26
  - commit 86070f4c08780c8a167bef2b44e09a00609915d6
- Code diff view showing changes in Sponsor.php:

```
59 @@ -6,6 +6,16 @@
6
7 class Sponsor
8 {
9+ /**
10+ * Full sponsor.
11+ */
12+ const SPONSOR_FULL = 'full';
13+
14+ /**
15+ * Sponsor only covers occasional events.
16+ */
17+ const SPONSOR_EVENT = 'event';
18+
19 /**
20 * @var string
21 *
```

@daveliddament

What is code review?

Benefits

Implementation

Tips

Practical



## CODE REVIEW BENEFITS 1/5

---

## WHAT ARE DEFECTS?

## WHAT ARE DEFECTS?

Bugs

## WHAT ARE DEFECTS?

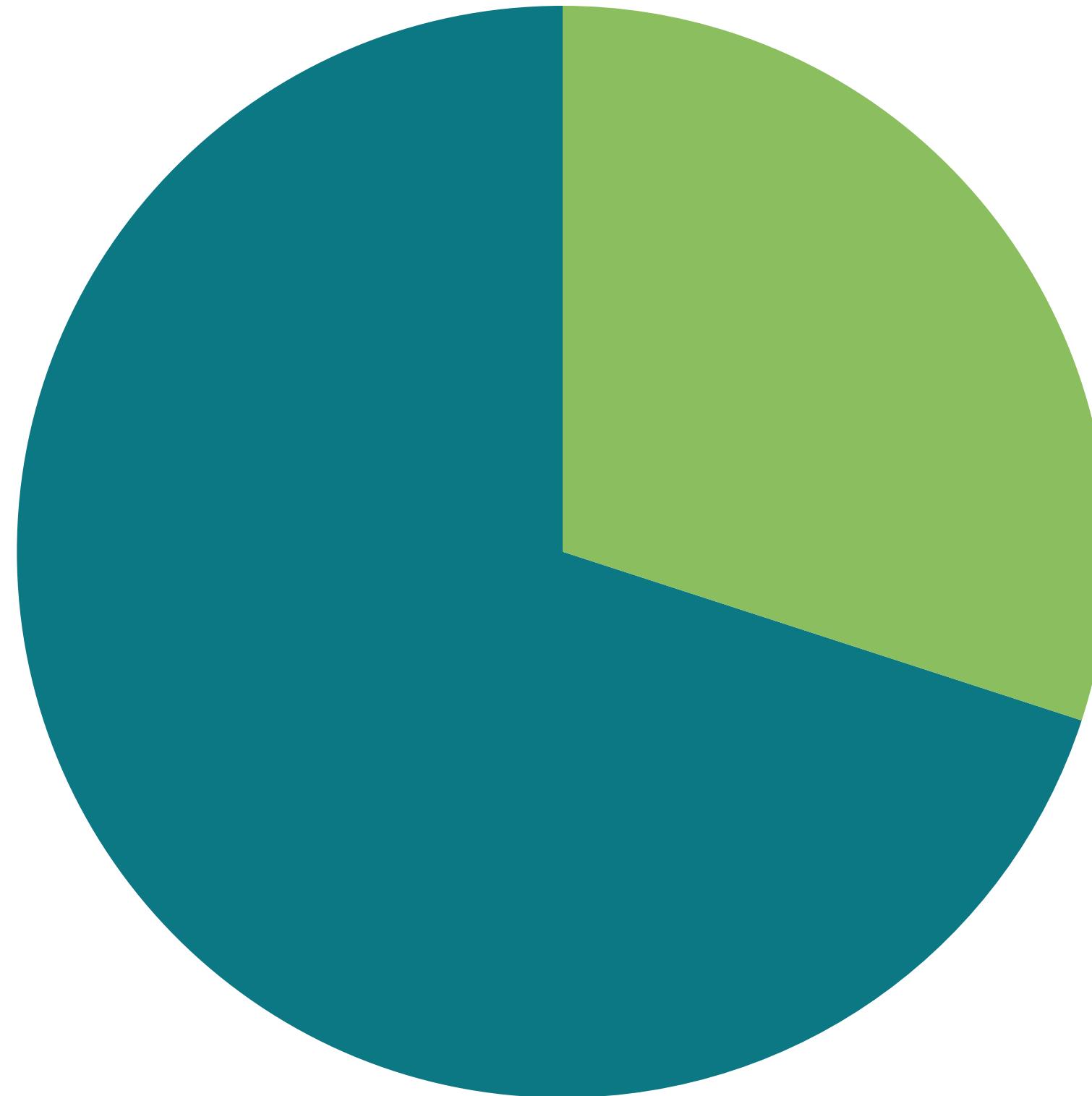
Bugs

Evolvability

## AN EVOLVABILITY DEFECT IS...

Code that makes code base less compliant with standards, more error prone, or more difficult to modify, extend or understand.

## WHAT ARE DEFECTS?



Evolvability

[1, 2]

Bugs

## EVOLVABILITY IS IMPORTANT

- ▶ Low evolvability costs money:
  - ▶ New features took 28% longer to implement [3]
  - ▶ Fixing bugs took 36% longer [3]
- ▶ Software structure may account for 25% of total maintenance costs [4]

## COST TO DEVELOP SIMILAR SIZED FEATURE OVER TIME

## COST TO DEVELOP SIMILAR SIZED FEATURE OVER TIME

---

---

---

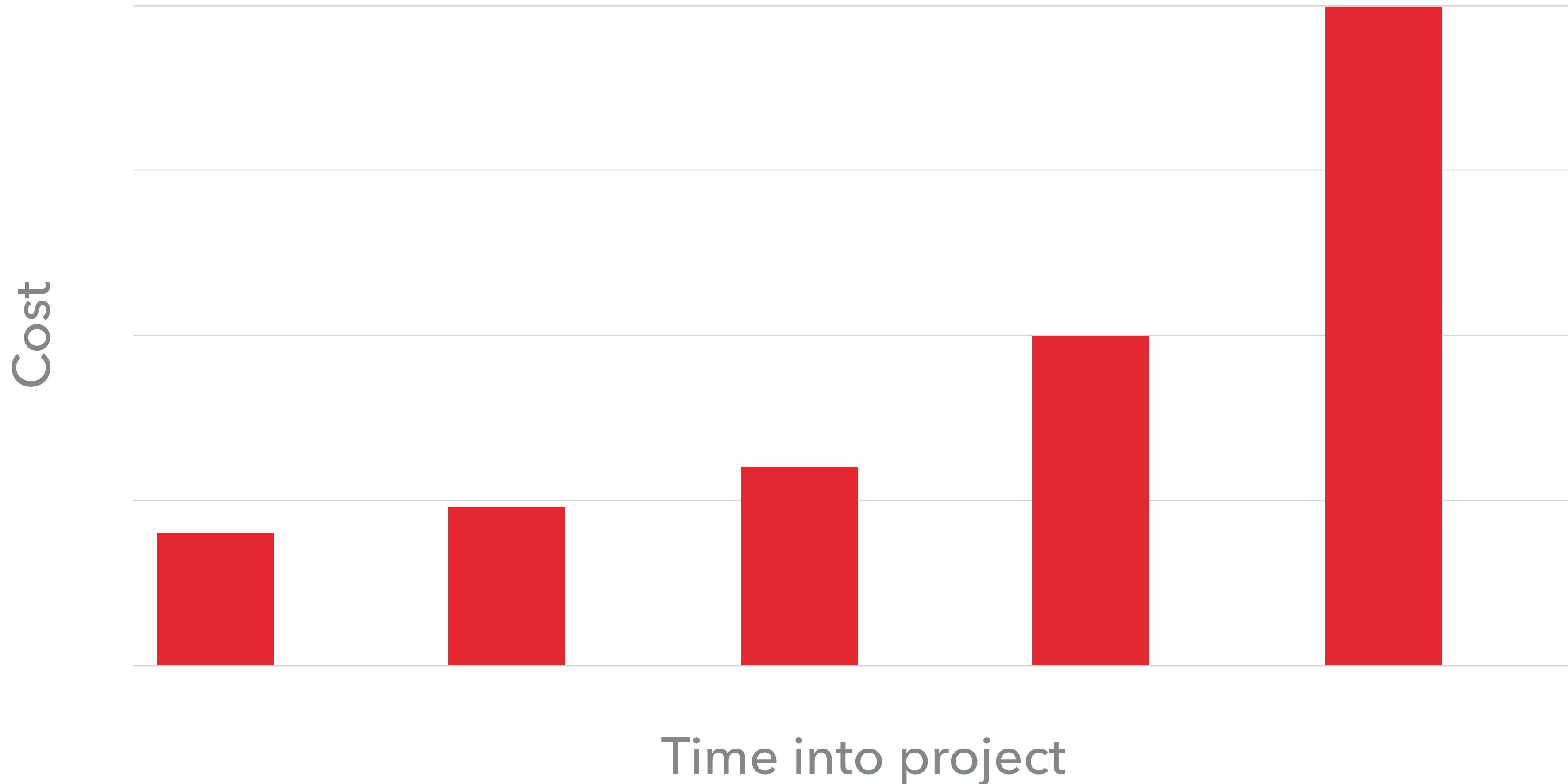
---

---

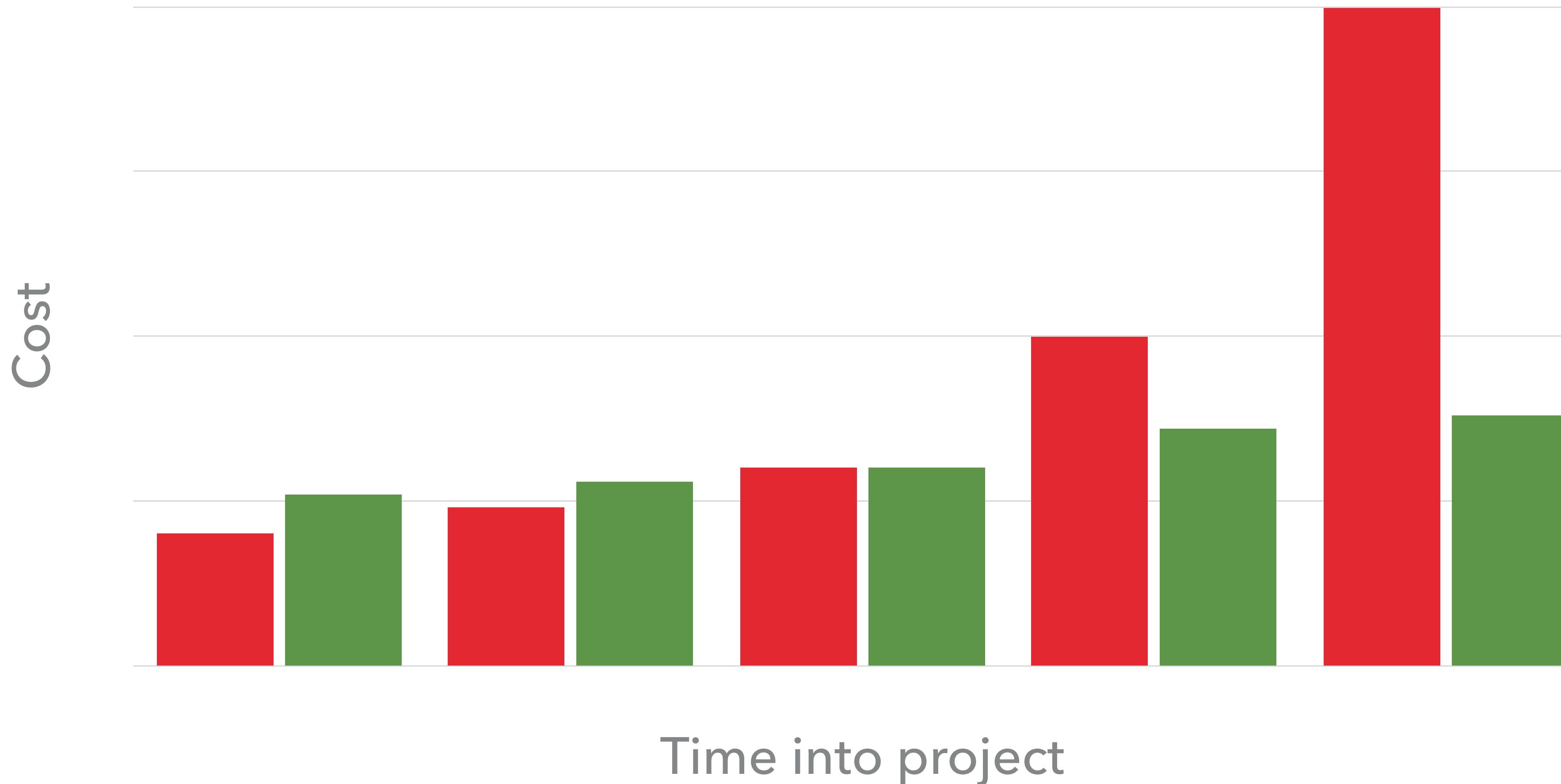
Cost

Time into project

## COST TO DEVELOP SIMILAR SIZED FEATURE OVER TIME



## COST TO DEVELOP SIMILAR SIZED FEATURE OVER TIME



### TAKE AWAY

- ▶ The majority of code review comments will be evolvability defects.
- ▶ OK not to find lots of “bugs”
- ▶ Most comments will be code improvements.
- ▶ Reduces evolvability defects. Lower overall cost.
- ▶ Studies back this up.
- ▶ Remember to sell the right metric to management.

# SECURITY REVIEW

## SECURITY REVIEW

- ▶ Writing sensitive data to logs (e.g. password)

## SECURITY REVIEW

- ▶ Writing sensitive data to logs (e.g. password)
- ▶ Files that shouldn't be there? (e.g. malware)

## SECURITY REVIEW

- ▶ Writing sensitive data to logs (e.g. password)
- ▶ Files that shouldn't be there? (e.g. malware)
- ▶ OWASP top 10

## SECURITY REVIEW

- ▶ Writing sensitive data to logs (e.g. password)
- ▶ Files that shouldn't be there? (e.g. malware)
- ▶ OWASP top 10
- ▶ OWASP top 10 cheat sheet

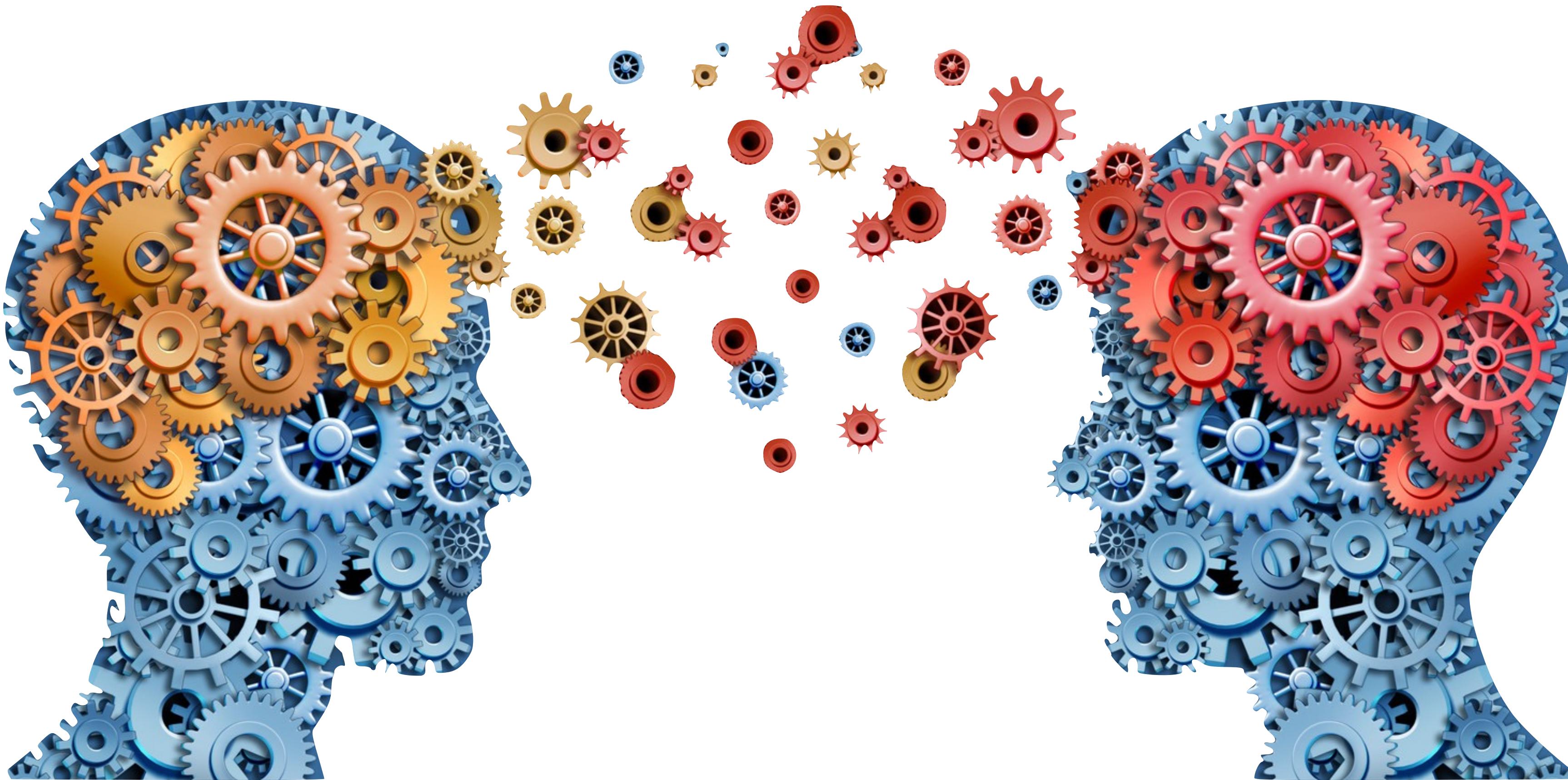
## SECURITY REVIEW

- ▶ Writing sensitive data to logs (e.g. password)
- ▶ Files that shouldn't be there? (e.g. malware)
- ▶ OWASP top 10
  - ▶ OWASP top 10 cheat sheet
- ▶ Not / weakly hashing passwords

## SECURITY REVIEW

- ▶ Writing sensitive data to logs (e.g. password)
- ▶ Files that shouldn't be there? (e.g. malware)
- ▶ OWASP top 10
  - ▶ OWASP top 10 cheat sheet
- ▶ Not / weakly hashing passwords
- ▶ Rolled your own authentication / hashing / encryption algorithms

## SPREAD THE KNOWLEDGE



### NO MORE SILOS



## NO MORE SILOS



## NO MORE SILOS



## NO MORE SILOS



## NO MORE SILOS



## MENTORING



TIL

```
if ($act1 === $act2) {  
    return $scene1 <=> $scene2;  
}  
return $act1 <=> $act2;
```

TIL

```
if ($act1 === $act2) {  
    return $scene1 <=> $scene2;  
}  
return $act1 <=> $act2;  
  
return [$act1, $scene1] <=> [$act2, $scene2]
```

**YOU'RE BEING WATCHED!**



YOU'RE BEING WATCHED!



[5]

# BENEFITS

- ▶ Reduce defects
  - ▶ Studies that back this up
- ▶ Find security vulnerabilities
- ▶ Spread knowledge
- ▶ Mentoring
- ▶ Peer pressure improves code quality

**EFFECTIVE CODE REVIEW  
REDUCES OVERALL COST OF  
SOFTWARE DEVELOPMENT**

**What is code review?**



**Benefits**



**Implementation**



**Tips**



**Practical**

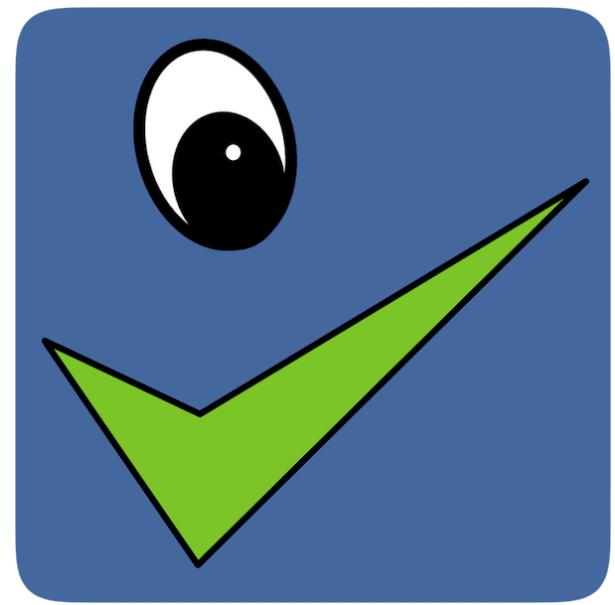
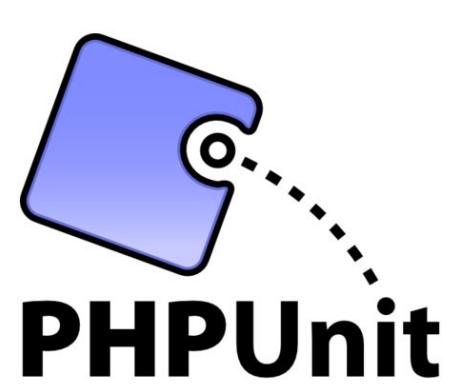


# AUTOMATE AS MUCH AS POSSIBLE

# AUTOMATE AS MUCH AS POSSIBLE



# AUTOMATE AS MUCH AS POSSIBLE

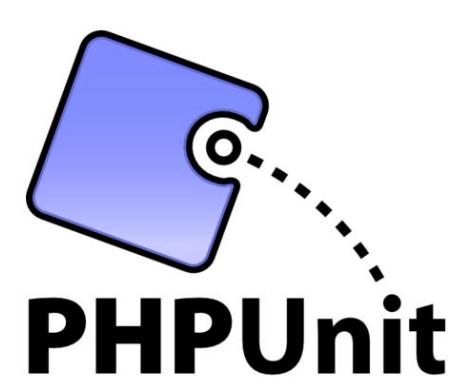


**Check  
Style**



**Psalm**

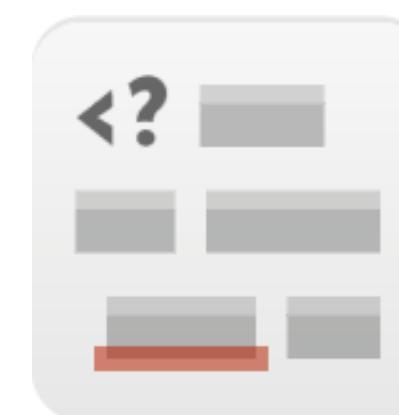
## AUTOMATE AS MUCH AS POSSIBLE



**PHPUnit**

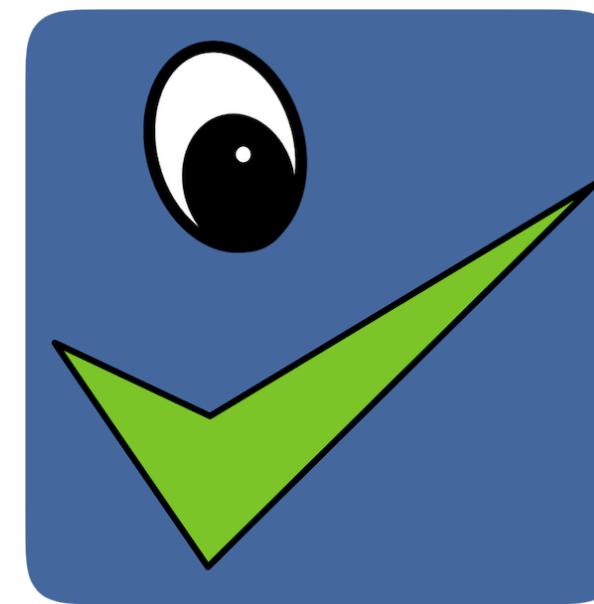
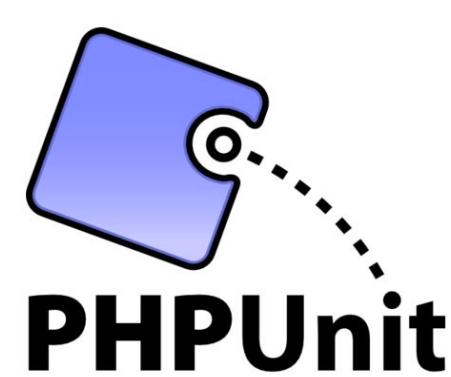


**Check  
Style**

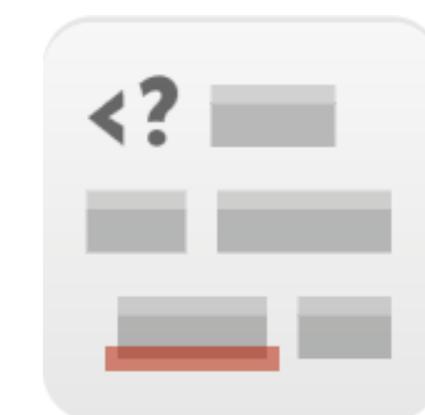


**Psalm**

## AUTOMATE AS MUCH AS POSSIBLE



Check  
Style



Psalm

<https://github.com/DaveLiddament/skeleton-ci-project>

## IMPLEMENTATION

---

# WHAT ARE WE REVIEWING?



DaveLiddament commented on 26 May 2019

Owner

+ 😊 ...

*No description provided.*

# WHAT ARE WE REVIEWING?



DaveLiddament commented on 30 Dec 2018

Owner + 😊 ...

Previously SARB generated a diff from the baseline commit and the most recent commit. However static analysis might be run on the current state of the code. These 2 might be different.

When running on CI the current state of the code and latest commit will *probably* be the same.

Developing locally this will probably not be the case. Developers will want to make sure no errors have been introduced since the baseline before committing code. To improve DX SARB will look at diff between the baseline commit and HEAD.

It is assumed the process will be something like this:

- developer edits code
- developer runs static analysis tool
- developer runs SARB
- developer removes any issues raised since the baseline
- once all post baseline issues are removed developer commits code

Hence the need for the diff to be taken against current state of code rather than last commit.

## ARE THE TESTS TESTING THE RIGHT THING?

Input	Expected Output
my blog	my-blog
hello Dave	hello-Dave

## ARE THE TESTS TESTING THE RIGHT THING?

Input	Expected Output
my blog	my-blog
hello Dave	hello-Dave

## ARE THE TESTS ADEQUATE?

Input	Expected Output
my blog	my-blog
it's Saturday	its-saturday

## ARE THE TESTS ADEQUATE?

Input	Expected Output
my blog	my-blog
it's Saturday	its-saturday

# ARE THE TESTS ADEQUATE?

Input	Expected Output
my blog	my-blog
it's Saturday	its-saturday

# HOW MANY TESTS DO WE NEED?

```
/**  
 * @throws NotFoundException  
 */  
public function isAllowed(int $userId) : bool  
{  
    .. some implementation ..  
}
```

# WILL I UNDERSTAND THIS CODE IN 6 MONTHS?

```
$userFields = [  
    'Username',  
    'Email',  
    'FirstName',  
    'LastName',  
    'Phone',  
];  
  
foreach ($userFields as $key) {  
    if ($userDetails->{ 'get' . $key } ()) {  
        $user->{ 'set' . $key } ($userDetails->{ 'get' . $key }());  
    }  
}
```

# WILL I UNDERSTAND THIS CODE IN 6 MONTHS? (2)

```
if ($userDetails->getUsername()) {  
    $user->setUsername($userDetails->getUsername());  
}  
if ($userDetails->getEmail()) {  
    $user->setEmail($userDetails->getEmail());  
}  
if ($userDetails->getFirstName()) {  
    $user->setFirstName($userDetails->getFirstName());  
}  
if ($userDetails->getLastName()) {  
    $user->setLastName($userDetails->getLastName());  
}  
if ($userDetails->getPhone()) {  
    $user->setPhone($userDetails->getPhone());  
}
```

“THE RATIO OF TIME SPENT READING VERSUS WRITING IS WELL OVER 10 TO 1. WE ARE CONSTANTLY READING OLD CODE AS PART OF THE EFFORT TO WRITE NEW CODE. .... [THEREFORE,] MAKING IT EASY TO READ MAKES IT EASIER TO WRITE.”

**Robert C. Martin (Clean Code)**

# ARE THE COMMENTS STILL CORRECT?

```
class Location
{
    /**
     * Returns type (City, town, village)
     */
    public function getUrl(): string
    {
        return $this->url;
    }

}
```

# ARE THE COMMENTS STILL CORRECT?

```
class Location
{
    /**
     * Returns type (City, town, village)
     */
    public function getUrl(): string
    {
        return $this->url;
    }
}
```

# ARE THE COMMENTS USEFUL?

```
/**  
 * @return string url  
 */  
public function getUrl(): string  
{  
    return $this->url;  
}
```

# IS A COMMENT MISSING?

```
function compareTeams(Team $a, Team $b):int  
{ ... }
```

# IS A COMMENT MISSING?

```
/**  
 * Used to sort order of teams in a league table.  
 *  
 * Sort order:  
 * - points (3 for win, 1 for draw)  
 * - goal difference  
 * - goals for  
 * - team name  
 */  
  
function compareTeams(Team $a, Team $b):int  
{ ... }
```

# IS A COMMENT MISSING?

```
/**  
 * Used to sort order of teams in a league table.  
 *  
 * Sort order:  
 * - points (3 for win, 1 for draw)  
 * - goal difference  
 * - goals for  
 * - team name  
 */  
function compareTeams (Team $a, Team $b) :int  
{ ... }
```

compareTeamsByPoints3ForWin1ForDrawThenGoalDifferenceThenGoalsForThenTeamName

# HOMEWORK

- ▶ Find code >6 months old. Can you understand it?
- ▶ Find “self documenting code” > 6 months old.
- ▶ Can you understand it?
- ▶ Can someone else understand it?

# HOW DO WE MAKE THIS MORE OBVIOUS?

```
class MarketingCampaign
{
    public function addAddress(
        string $address
    ): void {
        .. some implementation ..
    }
}
```

### HOW DO WE MAKE THIS MORE OBVIOUS (2)

```
class MarketingCampaign
{
    public function addEmailAddress(
        string $emailAddress
    ): void {
        .. some implementation ..
    }
}
```

### HOW DO WE MAKE THIS MORE OBVIOUS (3)

```
class MarketingCampaign
{
    /**
     * Adds email address, person will then be
     * messaged as part of the campaign.
    */
    public function addEmailAddress(
        string $emailAddress
    ): void {
        .. some implementation ..
    }
}
```

# HOW DO WE MAKE THIS MORE OBVIOUS (4)

```
class MarketingCampaign
{
    /**
     * Adds email address, person will then be
     * messaged as part of the campaign.
     */
    public function addEmailAddress(
        EmailAddress $emailAddress
    ): void {
        .. some implementation ..
    }

}
```

# ARE WE FOLLOWING PROJECT CONVENTIONS?

```
interface LocationRepository
{
    public function findClosestTo($point);
    public function findByName($name);
    public function findBySlug($slug);
    public function searchForLocation($name, $type);
    public function findAllByType($type);
}
```

# ARE WE FOLLOWING PROJECT CONVENTIONS?

```
interface LocationRepository
{
    public function findClosestTo($point);
    public function findByName($name);
    public function findBySlug($slug);
    public function searchForLocation($name, $type);
    public function findAllByType($type);
}
```

## DOCUMENT PROJECT CONVENTIONS

### #coding-standards

☆ | 84 | ⚡0 | Add a topic

---



**dave** 10:55 AM

Naming: Do not use abbreviations



8 replies Last reply 3 months ago



**dave** 11:29 AM

# HAS DEFENSIVE CODING BEEN USED?

```
/**  
 * Set status (one of started|finished|quit)  
 */  
public function setStatus(string $status) : void  
{  
    $this->status = $status;  
}
```

# CORRECT NAMING

### CORRECT NAMING

- ▶ Language for domain or project you're working on

# CORRECT NAMING

- ▶ Language for domain or project you're working on
- ▶ Design patterns

# HAS TECHNICAL DEBT BEEN DOCUMENTED?

```
// TODO: Refactor to method https://trello.com/c/Aaa123  
... some hacky code ...
```

# IS ARCHITECTURE OK?

# IS ARCHITECTURE OK?

- ▶ Make big architectural decisions before coding

# IS ARCHITECTURE OK?

- ▶ Make big architectural decisions before coding
- ▶ Over architected for “future problems”?

# IS ARCHITECTURE OK?

- ▶ Make big architectural decisions before coding
- ▶ Over architected for “future problems”?
- ▶ Correct level of abstraction?

# IS ARCHITECTURE OK?

- ▶ Make big architectural decisions before coding
- ▶ Over architected for “future problems”?
- ▶ Correct level of abstraction?
- ▶ Encapsulation / leakage?

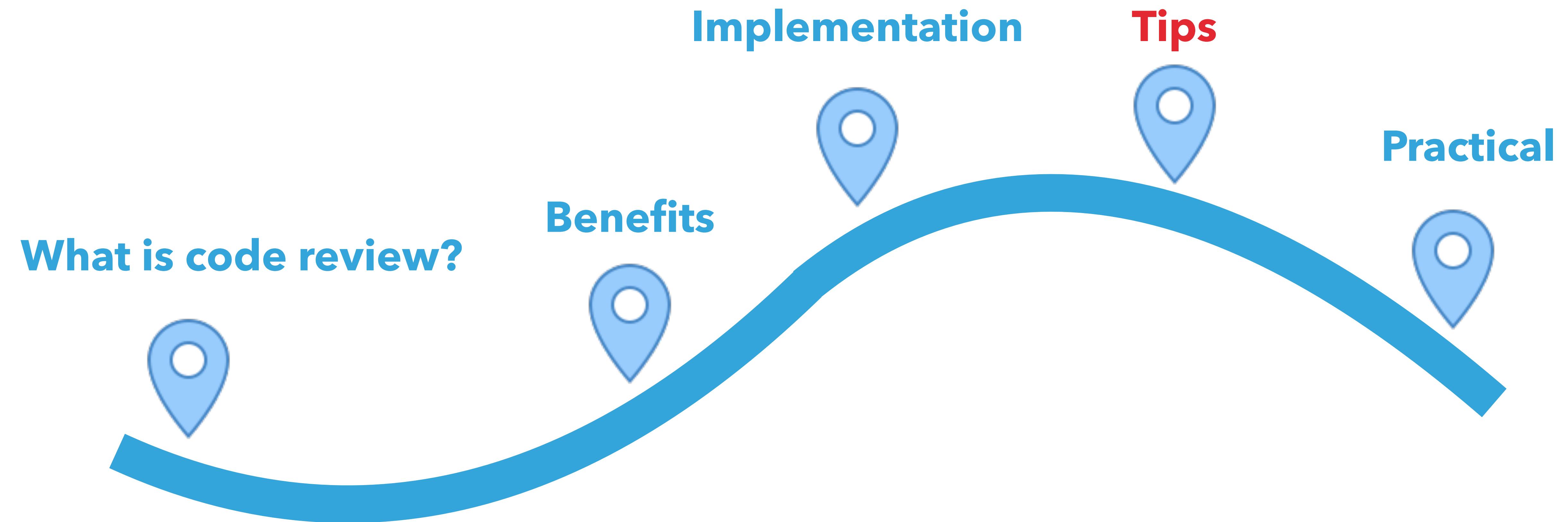
**ARE THERE ANY BUGS?!**

### CHECK LIST

- ▶ What are we reviewing?
- ▶ Do the tests fully test the required functionality?
- ▶ Will I understand this code in 6 months?
- ▶ Do comments match the code?
- ▶ Is the code obvious and explicit?
- ▶ Does the code follow project conventions?
- ▶ Has defensive coding been used?
- ▶ Has technical debt been documented?
- ▶ Can architecture be improved?
- ▶ Are there any bugs?

### CHECK LIST: TLDR

- ▶ Correct tests
- ▶ Clean code
- ▶ No bugs



# CODE REVIEW TIPS

---

## EVERYONE SHOULD CODE REVIEW



ASK PROGRAMMERS TO  
REVIEW 10 LINES OF CODE  
THEY'LL FIND 10 ISSUES...

Anyone who's done code review

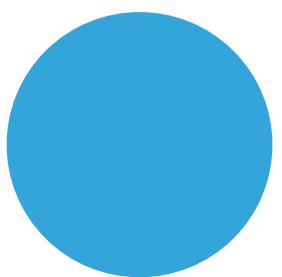
**ASK THEM TO DO 500 LINES  
THEY'LL SAY IT'S GOOD TO GO**

**Anyone who's done code review**

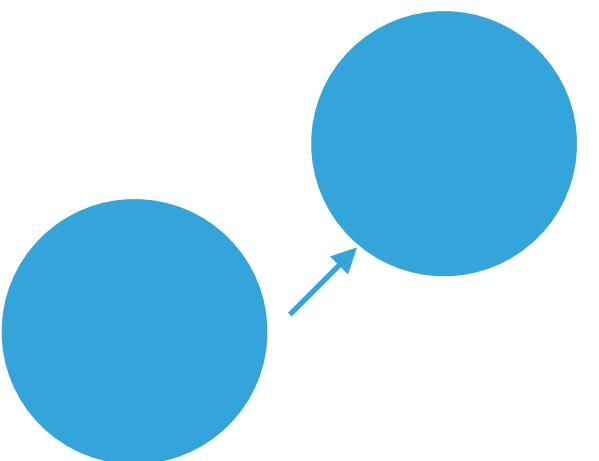
## HOW MUCH SHOULD YOU REVIEW IN ONE GO?

- ▶ Fewer than 400 lines of code at a time [6]
- ▶ Under 500 line of code reviewed per hour [6]
- ▶ Max 1 hour review at a time [6]

## SMALL COMMITS

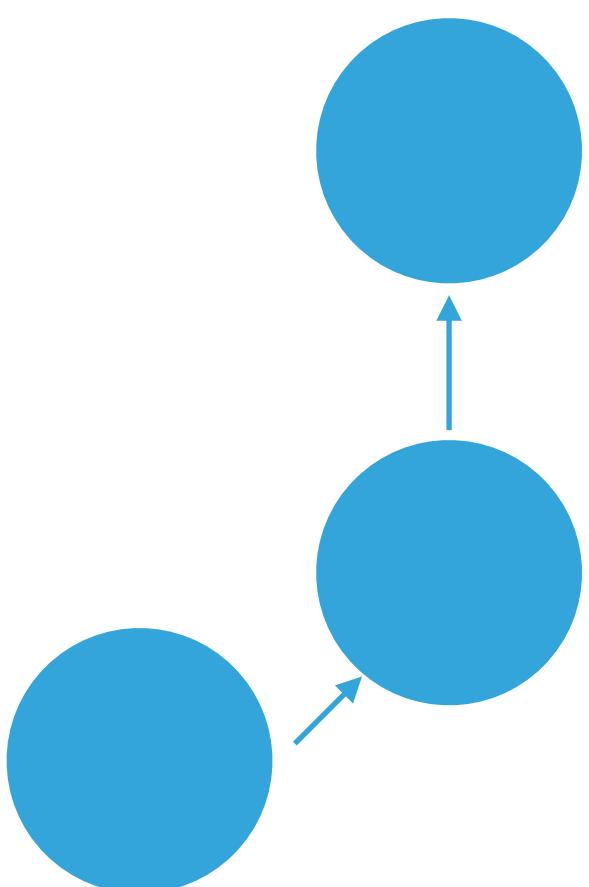


## SMALL COMMITS



DEPRECATE: The price calculation service

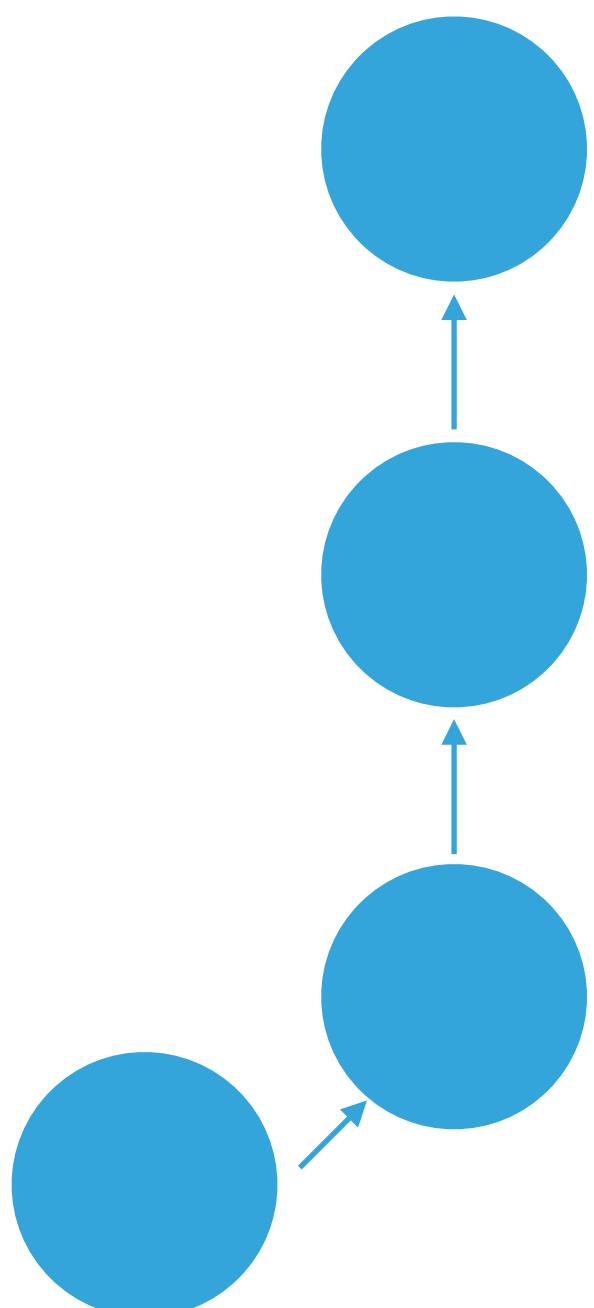
## SMALL COMMITS



ADD: Facade to 3rd party price calculation service

DEPRECATE: The price calculation service

## SMALL COMMITS

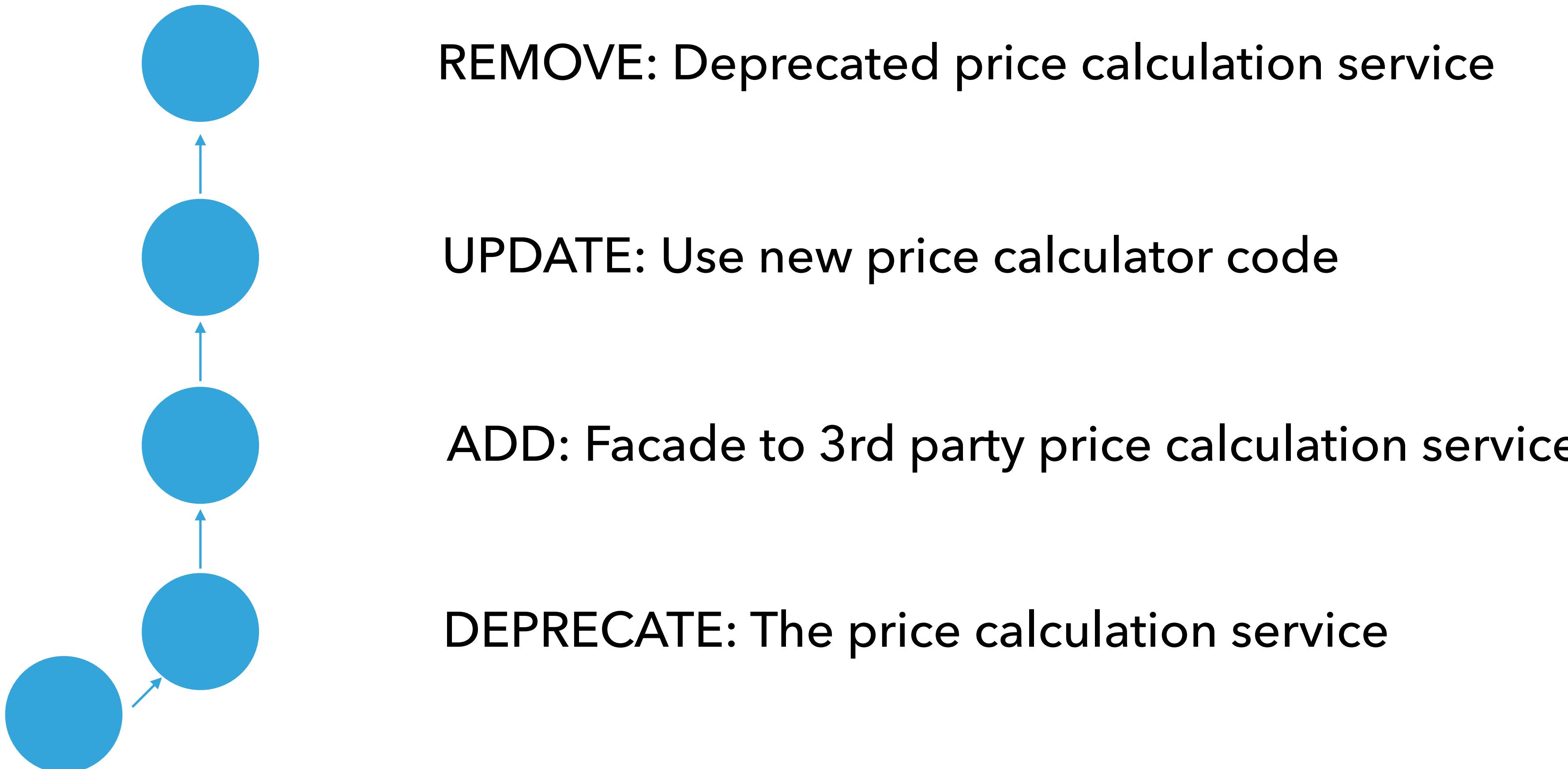


UPDATE: Use new price calculator code

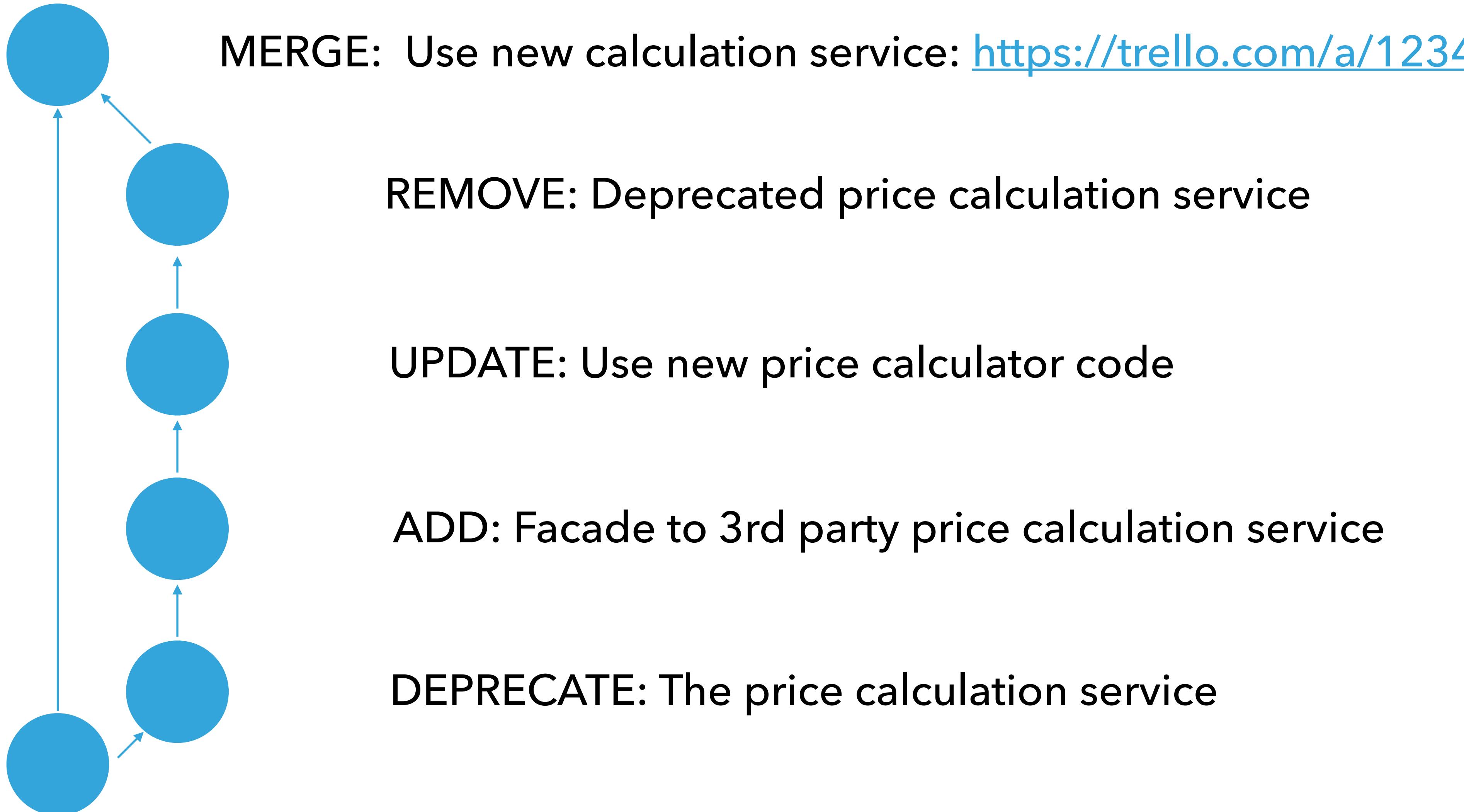
ADD: Facade to 3rd party price calculation service

DEPRECATE: The price calculation service

## SMALL COMMITS



## SMALL COMMITS



# GOOD COMMITS

# GOOD COMMITS

- ▶ Small

### GOOD COMMITS

- ▶ Small
- ▶ Focus on one thing

### GOOD COMMITS

- ▶ Small
- ▶ Focus on one thing
- ▶ Use individual commits for:

### GOOD COMMITS

- ▶ Small
- ▶ Focus on one thing
- ▶ Use individual commits for:
  - ▶ whitespace changes

## GOOD COMMITS

- ▶ Small
- ▶ Focus on one thing
- ▶ Use individual commits for:
  - ▶ whitespace changes
  - ▶ class renames or moves

## GOOD COMMITS

- ▶ Small
- ▶ Focus on one thing
- ▶ Use individual commits for:
  - ▶ whitespace changes
  - ▶ class renames or moves
  - ▶ composer updates

## GOOD REVIEW COMMENTS

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.
- ▶ Not critical of the author. (Use we rather than you).

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.
- ▶ Not critical of the author. (Use we rather than you).
- ▶ State problem and solution. (Maybe as a commit).

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.
- ▶ Not critical of the author. (Use we rather than you).
- ▶ State problem and solution. (Maybe as a commit).
- ▶ Link to Stack Overflow, blog, etc.

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.
- ▶ Not critical of the author. (Use we rather than you).
- ▶ State problem and solution. (Maybe as a commit).
- ▶ Link to Stack Overflow, blog, etc.
- ▶ Use: "Let's chat".

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.
- ▶ Not critical of the author. (Use we rather than you).
- ▶ State problem and solution. (Maybe as a commit).
- ▶ Link to Stack Overflow, blog, etc.
- ▶ Use: "Let's chat".
- ▶ Use: "Question".

## GOOD REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude.
- ▶ Not critical of the author. (Use we rather than you).
- ▶ State problem and solution. (Maybe as a commit).
- ▶ Link to Stack Overflow, blog, etc.
- ▶ Use: "Let's chat".
- ▶ Use: "Question".
- ▶ Compliment.

# RECEIVING REVIEW COMMENTS

## RECEIVING REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude

## RECEIVING REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude
- ▶ Don't take offence

## RECEIVING REVIEW COMMENTS

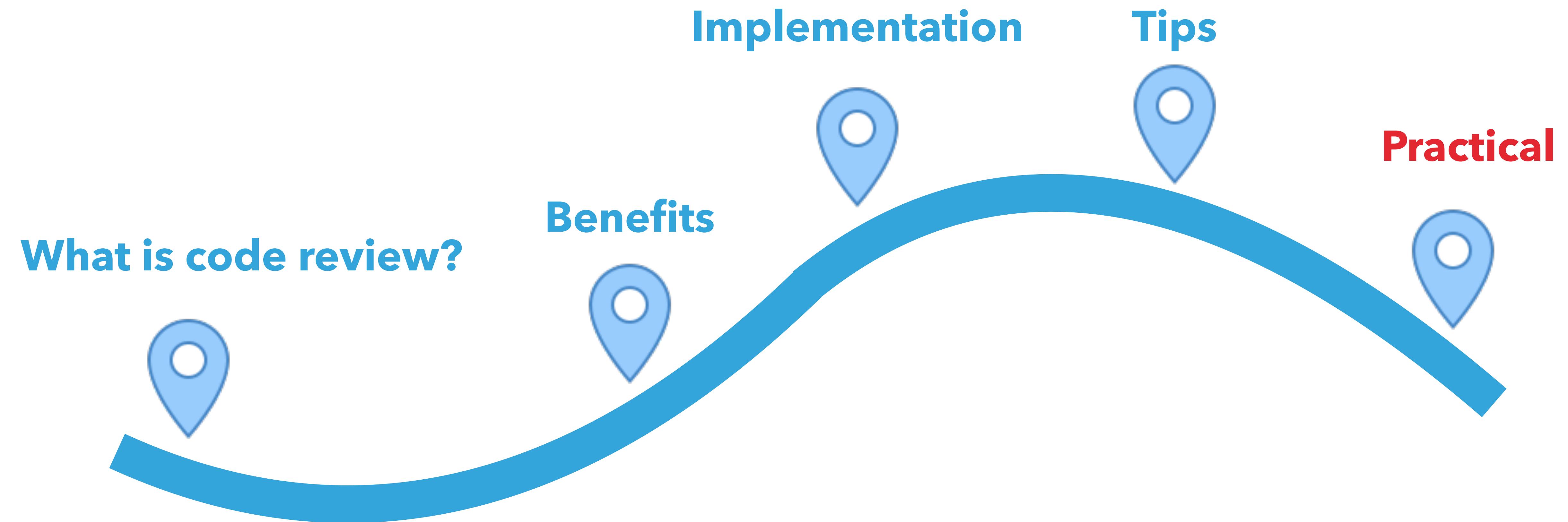
- ▶ Don't be an idiot. Don't be rude
- ▶ Don't take offence
- ▶ Do say if you disagree

## RECEIVING REVIEW COMMENTS

- ▶ Don't be an idiot. Don't be rude
- ▶ Don't take offence
- ▶ Do say if you disagree
- ▶ Compliment

## CODE REVIEW TIPS

- ▶ Get everyone involved
- ▶ Keep commits small
- ▶ Be constructive in code review comments
- ▶ Link to relevant bugs / stories / Trello cards



### CODE CAN ONLY BE DEPLOYED IF:

- ▶ CI passes
- ▶ Code review passes

# SETUP GITHUB

---

The screenshot shows the GitHub repository settings page for 'DaveLiddament / Demo'. The left sidebar has a vertical navigation menu with options: Options, Collaborators, **Branches**, Webhooks, Integrations & services, and Deploy keys. The 'Branches' option is currently selected, indicated by an orange border.

**Default branch**

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

master ▾   Update

**Protected branches**

Protect branches to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to protected branches? [Learn more](#).

Choose a branch... ▾

No protected branches yet.

# SETUP GITHUB

---

The screenshot shows the GitHub repository settings page for 'DaveLiddament / Demo'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Settings (highlighted with a red box), and Insights (0). The sidebar on the left lists Options, Collaborators, Branches (selected), Webhooks, Integrations & services, and Deploy keys. The main content area starts with a 'Default branch' section, which explains that the default branch is the 'base' branch for pull requests and code commits. It shows a dropdown menu set to 'master' and an 'Update' button. Below this is a 'Protected branches' section, which allows users to protect branches by disabling force pushing and preventing deletion. A dropdown menu labeled 'Choose a branch...' is shown, along with a message stating 'No protected branches yet.'

# SETUP GITHUB

---

The screenshot shows the GitHub repository settings page for 'DaveLiddament / Demo'. The 'Settings' tab is selected. On the left, a sidebar menu includes 'Options', 'Branches' (which is highlighted with a red box), 'Webhooks', 'Integrations & services', and 'Deploy keys'. The main content area is titled 'Default branch' and describes it as the 'base' branch for pull requests and commits. It shows a dropdown set to 'master' and an 'Update' button. Below this is the 'Protected branches' section, which is currently empty, displaying a message 'No protected branches yet.' A 'Choose a branch...' dropdown is present.

DaveLiddament / Demo

Unwatch 1 | Star 0 | Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Options

Branches

Webhooks

Integrations & services

Deploy keys

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

master ▾ Update

Protected branches

Protect branches to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to protected branches? [Learn more](#).

Choose a branch... ▾

No protected branches yet.

# SETUP GITHUB

The screenshot shows the GitHub repository settings for 'DaveLiddament / Demo'. The 'Branches' tab is selected in the sidebar. The main content area is titled 'Default branch' and describes it as the 'base' branch for pull requests and commits. A dropdown menu shows 'master' with a 'master' button and an 'Update' button. Below this, the 'Protected branches' section is shown, featuring a red box around the 'Choose a branch...' dropdown menu. A message indicates no protected branches have been created yet.

DaveLiddament / Demo

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Options

Collaborators

Branches

Webhooks

Integrations & services

Deploy keys

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

master ▾ Update

Protected branches

Protected branches enable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to protected branches? [Learn more](#).

Choose a branch... ▾

No protected branches yet.

# SETUP GITHUB

The screenshot shows the GitHub repository settings page for the 'master' branch of the 'DaveLiddament / Demo' repository. The left sidebar menu is visible, showing options like 'Options', 'Collaborators', 'Branches' (which is selected), 'Webhooks', 'Integrations & services', and 'Deploy keys'. The main content area is titled 'Branch protection for master' and contains several configuration sections with checkboxes:

- Protect this branch**: Describes how it disables force-pushes and prevents deletion.
- Require pull request reviews before merging**: Explains that commits must be made to a non-protected branch, submitted via a pull request with at least one approved review, and no changes requested before merging.
- Dismiss stale pull request approvals when new commits are pushed**: States that new reviewable commits will dismiss existing pull request review approvals.
- Require review from Code Owners**: Describes requiring an approved review in pull requests for files with a designated code owner.
- Require status checks to pass before merging**: Allows choosing which status checks must pass before merging. It notes that commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.
- Require branches to be up to date before merging**: Ensures the branch has been tested with the latest code on master.

At the bottom of the main content area, there is a message: "Sorry, we couldn't find any status checks in the last week for this repository." followed by a link: "Learn more about status checks on GitHub."

# SETUP GITHUB

DaveLiddament / Demo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

**Branch protection for master**

**Protect this branch**  
Disables force-pushes to this branch and prevents it from being deleted.

**Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into master.

**Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a branch will dismiss pull request review approvals.

**Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

**Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into master. When enabled, commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.

**Require branches to be up to date before merging**  
This ensures the branch has been tested with the latest code on master.

Sorry, we couldn't find any status checks in the last week for this repository.  
[Learn more about status checks on GitHub.](#)

# SETUP GITHUB

 [DaveLiddament / Demo](#) Unwatch ▾ 1 Star 0 Fork 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Settings](#) [Insights ▾](#)

[Options](#) [Collaborators](#) [Branches](#) [Webhooks](#) [Integrations & services](#) [Deploy keys](#)

## Branch protection for master

**Protect this branch**  
Disables force-pushes to this branch and prevents it from being deleted.

**Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into master.

**Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a branch will dismiss pull request review approvals.

**Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

**Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into master. When enabled, commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.

**Require branches to be up to date before merging**  
This ensures the branch has been tested with the latest code on master.

Sorry, we couldn't find any status checks in the last week for this repository.  
[Learn more about status checks on GitHub.](#)

# SETUP GITHUB

 [DaveLiddament / Demo](#) Unwatch  1 Star  0 Fork  0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Settings !\[\]\(52bb9d4622c72bd96f6e4ea4a34ad902\_img.jpg\)](#) [Insights !\[\]\(9c99f1e5270988f12cf319eba71df4f8\_img.jpg\)](#)

[Options](#)

[Collaborators](#)

**Branches**

[Webhooks](#)

[Integrations & services](#)

[Deploy keys](#)

## Branch protection for master

**Protect this branch**  
Disables force-pushes to this branch and prevents it from being deleted.

**Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one review. This can be bypassed by force-pushing to the branch or by pushing directly to master.

**Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a branch will dismiss pull request review approvals.

**Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

**Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into master. When enabled, commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.

**Require branches to be up to date before merging**  
This ensures the branch has been tested with the latest code on master.

Sorry, we couldn't find any status checks in the last week for this repository.  
[Learn more about status checks on GitHub.](#)

# SETUP GITHUB

 [DaveLiddament / Demo](#)

 Unwatch 1     Star 0     Fork 0

[Code](#)    [Issues 0](#)    [Pull requests 0](#)    [Projects 0](#)    [Wiki](#)    [Settings](#)    [Insights ▾](#)

[Options](#)

[Collaborators](#)

**Branches**

[Webhooks](#)

[Integrations & services](#)

[Deploy keys](#)

## Branch protection for master

**Protect this branch**  
Disables force-pushes to this branch and prevents it from being deleted.

**Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into master.

**Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a branch will dismiss pull request review approvals.

**Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

**Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into master. When enabled, commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.

**Require branches to be up to date before merging**  
This ensures the branch has been tested with the latest code on master.

Sorry, we couldn't find any status checks in the last week for this repository.  
[Learn more about status checks on GitHub.](#)

# SETUP GITHUB

 [DaveLiddament / Demo](#)

 Unwatch 1     Star 0     Fork 0

[Code](#)    [Issues 0](#)    [Pull requests 0](#)    [Projects 0](#)    [Wiki](#)    [Settings](#)    [Insights](#)

**Options**

**Collaborators**

**Branches**

**Webhooks**

**Integrations & services**

**Deploy keys**

## Branch protection for master

**Protect this branch**  
Disables force-pushes to this branch and prevents it from being deleted.

**Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into master.

**Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a branch will dismiss pull request review approvals.

**Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

**Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into master. When enabled, commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.

**Require branches to be up to date before merging**  
This ensures the branch has been tested with the latest code on master.

Sorry, we couldn't find any status checks in the last week for this repository.  
[Learn more about status checks on GitHub.](#)

# SETUP GITHUB

The screenshot shows the GitHub repository settings page for the 'master' branch of the 'DaveLiddament / Demo' repository. The left sidebar includes options like Options, Collaborators, Branches (selected), Webhooks, Integrations & services, and Deploy keys. The main content area is titled 'Branch protection for master' and lists several configuration options:

- Protect this branch**: Describes how it prevents force-pushes and deletion.
- Require pull request reviews before merging**: Explains that commits must be made to a non-protected branch via a pull request with at least one approved review.
- Dismiss stale pull request approvals when new commits are pushed**: Details how new commits will dismiss existing approvals.
- Require review from Code Owners**: Requires an approved review in pull requests for files with a designated code owner.
- Require status checks to pass before merging**: Allows selecting which status checks must pass before merging.
- Require branches to be up to date before merging**: Ensures the branch has been tested with the latest code on master.

A red box highlights the status check section, which displays the message: "Sorry, we couldn't find any status checks in the last week for this repository. Learn more about status checks on GitHub."

# SETUP GITHUB

---

## **Require status checks to pass before merging**

Choose which [status checks](#) must pass before branches can be merged into master. When enabled, commits must first be pushed to another branch, then merged or pushed directly to master after status checks have passed.

## **Require branches to be up to date before merging**

This ensures the branch has been tested with the latest code on master.

Status checks found in the last week for this repository

ci/circleci

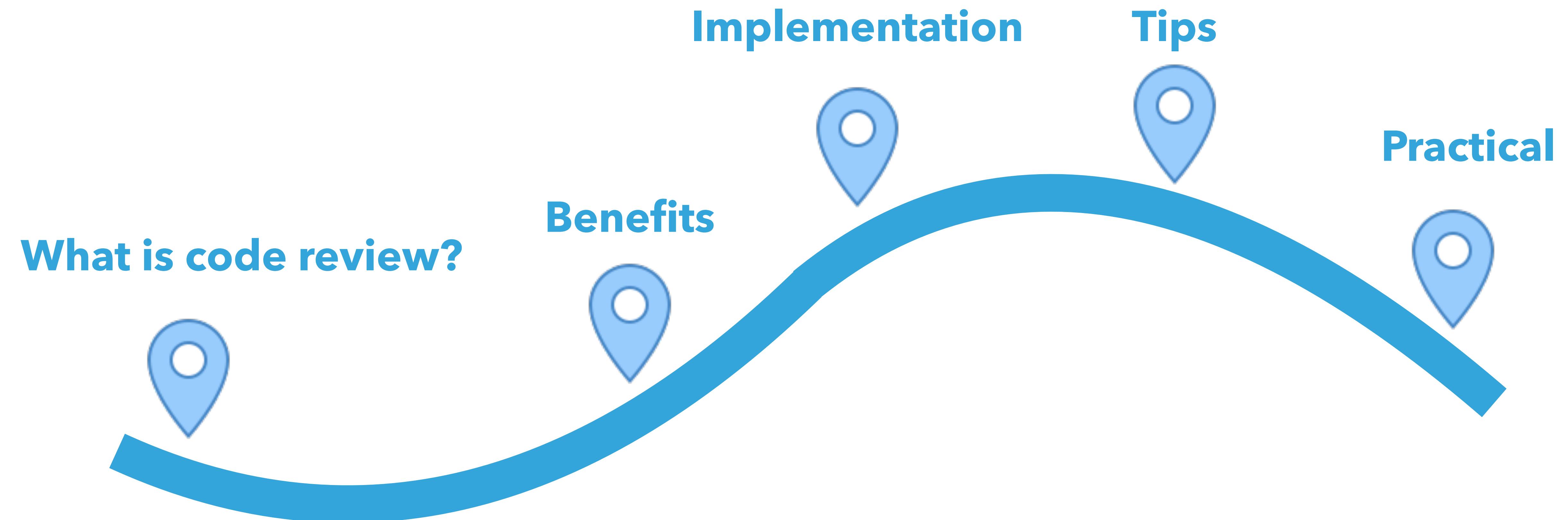
Required

## **Include administrators**

Enforce all configured restrictions for administrators.

## INTEGRATING CODE REVIEW INTO PROJECT WORKFLOW

- ▶ Easy with tools like Github
- ▶ No excuse not to start using today



**EFFECTIVE CODE REVIEW  
REDUCES OVERALL COST OF  
SOFTWARE DEVELOPMENT**

# Dave Liddament

Lamp Bristol

@daveliddament

Organise PHP-SW and Bristol PHP Training  
Author of Static Analysis Results Baseline (SARB)  
18 years of writing software (C, Java, Python, PHP)

Dave Liddament

Lamp Bristol

Thank you for  
listening

@daveliddament

Organise PHP-SW and Bristol PHP Training  
Author of Static Analysis Results Baseline (SARB)  
18 years of writing software (C, Java, Python, PHP)



# REFERENCES

- ▶ [1] Mika V. Mantyla and Casper Lassenius "What Types of Defects Are Really Discovered in Code Reviews?" IEEE Transactions on Software Engineering
- ▶ [2] Harvey Siy, Lawrence Votta "Does The Modern Code Inspection Have Value?"
- ▶ [3] R.K. Bandi, V.K. Vaishnavi, and D.E. Turk, "Predicting Maintenance Performance Using Object-Orientated Design Complexity Metrics"
- ▶ [4] R.D. Bunker, S.M. Datar, C.F. Kemerer, and D. Zweig, "Software Complexity and Maintenance Costs,"
- ▶ [5] <https://www.bbc.co.uk/news/uk-37502136>
- ▶ [6] <https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/>