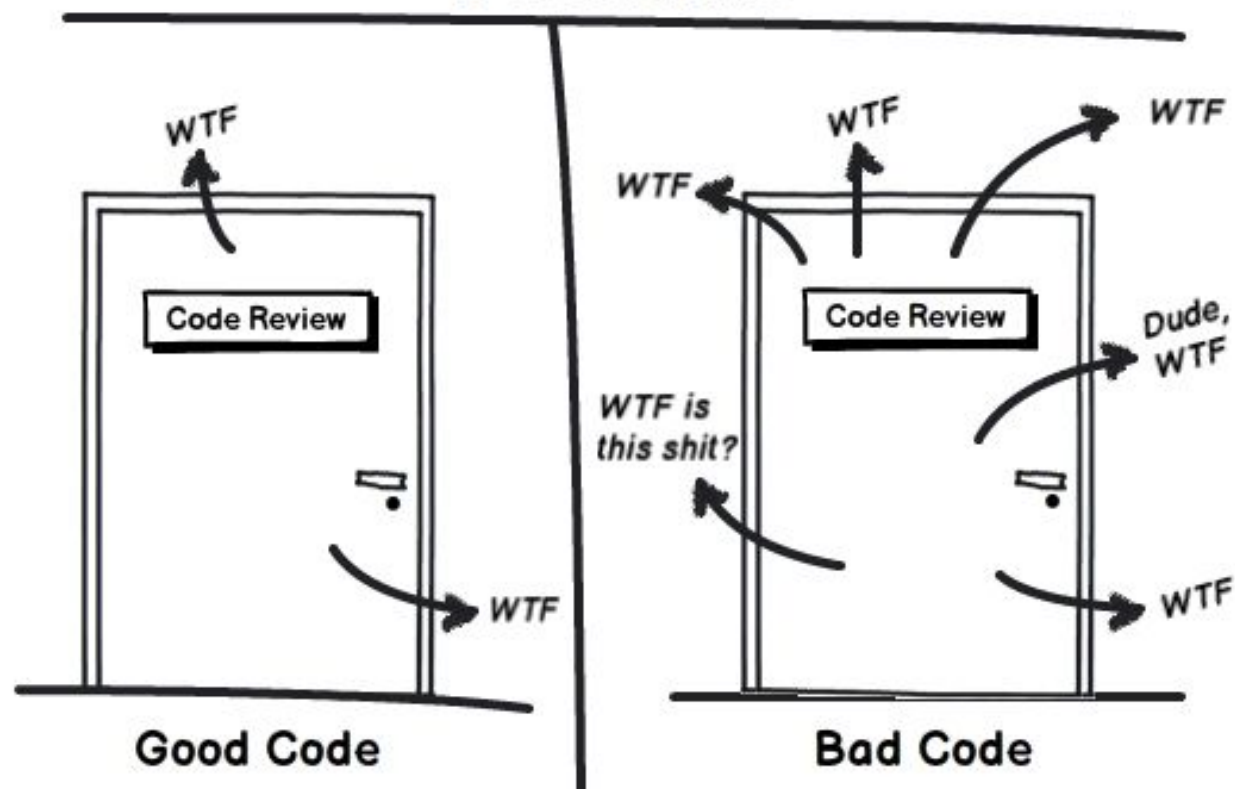




Code Quality Analysis

Reduce Technical Debt

Code Quality Measurement: WTFs/Minute





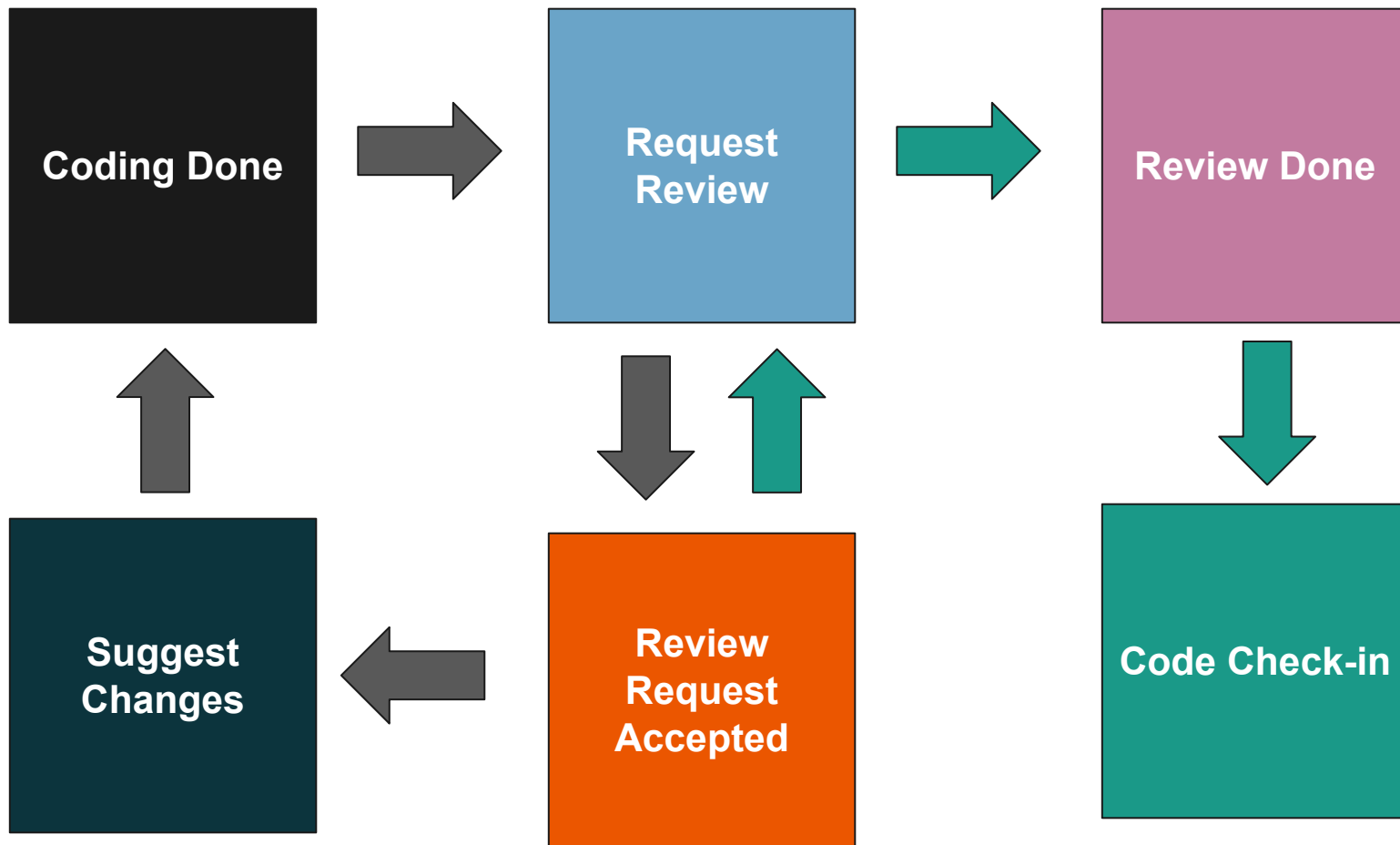
What is Code Review

1. Systematic examination of computer source code.
2. Find and fix mistakes overlooked in the initial development phase.
3. Improves the overall quality of software and the developer's skills.



Why do we need Code Review

1. Mistakes are caught before they become a part of product.
2. Ensures that programs are written following organization's coding guidelines.
3. Repetitive code blocks are caught and gets refactored.
4. Common vulnerabilities like format string exploits, race conditions, memory leaks, etc are found early
5. Knowledge sharing and mentoring for junior developers





What to check in a code

1. Flaws or Potential flaws
2. Consistency with the overall program design
3. Quality of comments
4. Adherence to coding standards



Code Review Checklist

- Naming of variables
 - Variable names should follow camelCasing
 - **Bad** - customerorderhistory
 - **Good** - customerOrderHistory
 - Variable names should be meaningful and complete. Avoid short names.
 - **Bad** - xx, xyz, i, j, k, cusOrdHist, amt, prc, inv
 - **Good** - amount, price, invoice
 - Global variables should be prefixed with underscore(_)
 - **Bad** - globalCustomerName
 - **Good** - _customerName



Code Review Checklist

- Naming of methods
 - Method names should follow camelCasing
 - **Bad** - getOrderdetails
 - **Good** - getOrderDetails
 - Method names should be meaningful and complete. Avoid short names.
 - **Bad** - getAmt, getInv
 - **Good** - getAmount, getInvoice
 - Async methods should have suffix async
 - **Bad** - asyGetStock
 - **Good** - getStockAsync



Code Review Checklist

- Constants should be in all CAPS
 - Method names should follow PascalCase
 - **Bad** - secretkey, secret_key
 - **Good** - SECRET_KEY
- Namespaces and folders should be in PascalCasing
 - **Bad** - namespace app_resources, namespace appResources
 - **Good** - namespace AppResources
- Unit tests have been written for functions and methods



Code Review Checklist

- Use comments to provide input and output parameter types
- Use comments to explain about the function
- Keep adding bug fixes in the comment with BugId, Developer name, date and short description
 - //ID:230 Divyanshu Das 11.04.2019 Changed logic to avoid race condition in invoice numbering
- Long functions and code blocks should be split into smaller functions
 - **Each function should do one thing and one thing only**
- Hardcoded values should be defined in constant or config file



Code Review Checklist

➤ Meaningful error messages

- **Bad** -

- Error in Application
- Something went wrong
- There is an error

- **Good** -

- Failed to update database. Please make sure login id and password are correct
- Your access token has expired.



Code Review Checklist

- Try catch should be used for exception handling not flow control

Bad Practice

```
1 try {  
2     userData.setName(requestData.getName());  
3 } except(Exception e) {  
4     userData.hasName = false;  
5 }
```

Good Practice

```
1 if (!requestData.getName().equals(null)) {  
2     userData.setName(requestData.getName());  
3 } else {  
4     userData.hasName = false;  
5 }
```

- Blank Catch block is not used
- Large try catch blocks should be split into smaller ones



Code Review Checklist

- Unit tests have been defined for the functions and methods
 - **Bad** -
 - Error in Application
 - Something went wrong
 - There is an error
 - **Good** -
 - Failed to update database. Please make sure login id and password are correct
 - Your access token has expired.



Effective Code Review Practices

- Should only last 60 minutes
- Should not be more than 200 LOC (200 lines of code)
- Review at every pull request.
- Use linters, code sniffers, testing tools, etc.



Tools For Code Review

- Always use IDEs
- Sublime is good but IDEs are better
- Sonar Lint
- TsLint
- PHPUnit, Junit, Karma, Jasmine, etc.
- Sonarqube
- Sonar Scanner



Setting up Sonarqube analysis for Angular

1. `npm install tslint typescript -g`
2. `npm install tslint-sonarts -save-dev`
3. Add tslint to tslint.json
 - a. `"extends": [`
 - b. `"tslint:recommended",`
 - c. `"tslint-sonarts"`
 - d. `]`
4. Create sonar-project.properties



Sonar-project.properties

```
5. sonar.projectKey=Demo:sonar-ts-demo
6. sonar.projectName=Sonar TS Demo
7. sonar.projectVersion=1.0
8. sonar.sourceEncoding=UTF-8
9. sonar.sources=src
10. sonar.exclusions=**/node_modules/**,**/*.spec.ts
11. sonar.tests=src
12. sonar.test.inclusions=**/*.spec.ts
13. sonar.ts.tslintconfigpath=tslint.json
14. sonar.ts.lcov.reportpath=test-results/coverage/coverage.l
    cov
```



tsconfig.json

```
{  
  "include": [  
    "src/**/*"  
  ],  
  "exclude": [  
    "node_modules",  
    "**/*.spec.ts"  
  ]  
}
```



Further steps

1. `tslint --project tsconfig.json -c tslint.json`
2. `npm install sonarqube-scanner --save-dev`
3. Create `gulpfile.js`
 - a. `var gulp = require('gulp');`
 - b. `var sonarqubeScanner = require('sonarqube-scanner');`
 - c.
 - d. `gulp.task('sonar', function(callback) {`
 - e. `sonarqubeScanner({`
 - f. `serverUrl : "http://sq.moglix.com",`
 - g. `options : {`
 - h. `}`
 - i. `}, callback);`
 - j. `});`
4. Run -> `gulp sonar`




Thank you



Install SonarLint in IntelliJ

- 1.1 go to <https://plugins.jetbrains.com/plugin/7973-sonarlint/versions>
- 1.2 go to Versions tab
- 1.3 download [3.5.1.2759](#) version (This version is chosen as to be compatible with server installed sonarqube)
- 1.4 Open IntelliJ Idea Settings/Preferences
- 1.5 Go To plugins on your left hand list of categories
- 1.6 Click on install plugins from disk
- 1.7 plugin installed; restart ide



Configure Sonarqube server on your local plugin (Optional step, only required if you have server credentials and want to run project on server and analyze complete report, required for team leads and project-managers)

2.1 go to SonarLint General Settings

2.2 Click on + to add new binding

2.3 Select SonarQube instance; add <http://sq.moglix.com> ; click next

2.4 Use token or Username/Password authentication Method

2.5 go to SonarLint Project Settings

2.6 Click on checkbox to enable binding to the server

2.7 Select Moxlix Sonar from dropdown

2.8 Search your project in the list and select it



Install SonarLint Plugin Eclipse

3.1 Go to Eclipse market place

3.2 search for SonarLint, install the plugin



Install SonarLint in VisualStudio Code

- 4.1 Click on search icon box
- 4.2 search for SonarLint
- 4.3 Install the plugin