TA Training

Facilitator: Code Review

University of Twente. 6 March 2023.

Dr. Vadim Zaytsev aka @grammarware. Programme Director.

UNIVERSITY OF TWENTE.

Teaching Assistants
Roles and Competences

Expert

(passive)	(active)
grade 7.0+	grade 9.0+
basics	in-depth
junior	senior
mastery	excellence
skilled	professional

Coach

(passive)	(active)
run labs	manage
do groups	groupwork
accommodate	dynamics
guide	diversity
consult	inclusive

Grader

(passive)	(active)
summative	align LOs
assess	test review
correct	argue
sign off	contend
use rubrics	admit
be fair	handle

Assistant

(passive)	(active)
assist	engage
motivate	contribute
help	signal
behave	exemplify
care	take charge

Analyst

(passive)	(active)
study	uncertainty
inspect	write manual
interpret	avoid bias
abstract	validate
scrutinise	connect
identify	find issues

Facilitator

(passive)	(active)
formative	acknowledge
descriptive	support
structural	encourage
activate	promote
review code	confer

U BEVINDT ZICH HIER

Shaper

(passive)	(active)
propose	solve
update	divergent
challenge	lateral
be creative	develop
draft	coordinate

UNIVERSITY OF TWENTE.

Grader vs Facilitator?

Feedback: Summative vs Normative

- evaluates
  - product
  - outcome
- given after the project
- focuses on the quality
- examples:
  - conformance test
  - acceptance test

- evaluates
  - process
  - methodology
- given during the project
- focuses on effectiveness
- examples:
  - best practices
  - code quality advice

Why Code Review?

HOW TO MAKE A GOOD CODE REVIEW
geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Why Code Review?

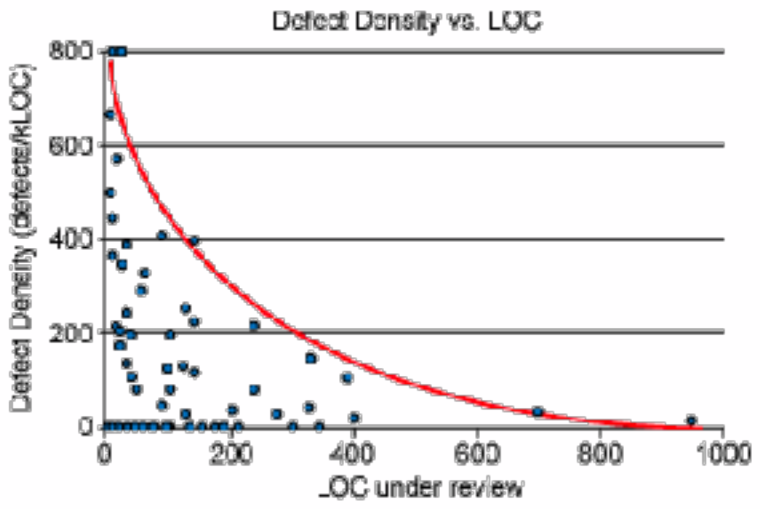
- ✓ catch bugs
- ✓ improve code quality
- ✓ share knowledge
- ✓ enforce standards
- ✓ improve estimates
- ✓ mentor new engineers
- ✓ build a better team

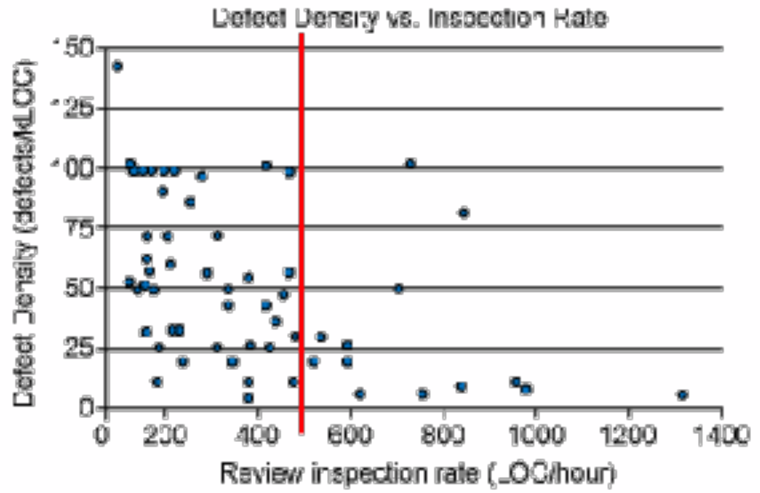
Types of Code Review

- tool assisted
  - linters!
- instant
  - pair programming
- live/synchronous
  - over-the-shoulder
- asynchronous
  - lightweight
- team review
  - hour of code
- formal review
  - artefact evaluation

Key Elements of Code Review

- clear shared objectives
- process defined & followed
- constructive actionable feedback
- collaborative respectful environment
- attention to detail





```
class HtmlWriter {
    void setXhtmlMode(boolean as_xhtml);
    void write(File f, String txt);
}

class HtmlWriter {
    static void write(File f, String txt, boolean as_xhtml);
}

class HtmlWriter {
    HtmlWriter(File f, String txt, boolean as_xhtml);
    void write();
}
```

Java

- ✓ naming.Conventions.namingConventions()
- ✓ collapse variables, move constants to static/enum
- ✓ // clean up dead/debugging code
- ✓ prefer streams and lambdas to for/if
- ✓ choose data structures & build mutable strings
- ✓ switch/case >>> if/else/if/else/…
- ✓ throws Exception
- ✓ equals & hashCode
- ✓ does it have to be public?
- ✓ interface: do or do not
- ✓ beware of pointer leaks
- ✓ JSL/JUnit/Commons/Maven/Log5j/Slf4j/Jackson/Guava/JAXB

Python

- ✓ pythonicity!
- ✓ spacing in both dimensions
- ✓ naming conventions: x, _x, __x__(), FooBar, FOO_BAR, …
- ✓ x = x + 1 # increment x
- ✓ for l in I[0:]:
- ✓ group only from in import
- ✓ strings and f-strings
- ✓ trailing commas
- ✓ comprehensions, zip(), all(), any() >>> for loops
- ✓ if … is not >>> if not … is
- ✓ __eq__: all six of none at all
- ✓ type hints

https://peps.python.org/pep-0008/

Haskell

- ✓ redundant brackets, `$`s, `\`s
- ✓ η-reduction and `` `infix` ``
- ✓ `otherwise = False`
- ✓ `f >>= return`
- ✓ use long camelCase names
- ✓ do not mix IO with computations
- ✓ group `import`
- ✓ map, foldr, foldl, … >>> recursion
- ✓ min, minimum, minimumBy
- ✓ more functions >>> one big function
- ✓ pattern matching >>> guards
- ✓ `data T = … deriving (Eq, Ord, Enum)` >>> `type T = Int`

General Advice

- pay attention to details // do not nitpick
- take your time – code takes effort
- review commit messages
- point out non-atomic commits/PRs
- use/advise tools
- check readability & documentation
- check security & leaks
- check concurrency & peformance
- check for side effects on existing code

```java
public static int dayOfYear(int month, int dayOfMonth, int year) {
    if (month == 2) {
        dayOfMonth += 31;
    } else if (month == 3) {
        dayOfMonth += 59;
    } else if (month == 4) {
        dayOfMonth += 90;
    } else if (month == 5) {
        dayOfMonth += 31 + 28 + 31 + 30;
    } else if (month == 6) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31;
    } else if (month == 7) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30;
    } else if (month == 8) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31;
    } else if (month == 9) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31;
    } else if (month == 10) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30;
    } else if (month == 11) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31;
    } else if (month == 12) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 31;
    }
    return dayOfMonth;
}
```

Homework

https://web.mit.edu/6.005/www/fa15/classes/04-code-review/

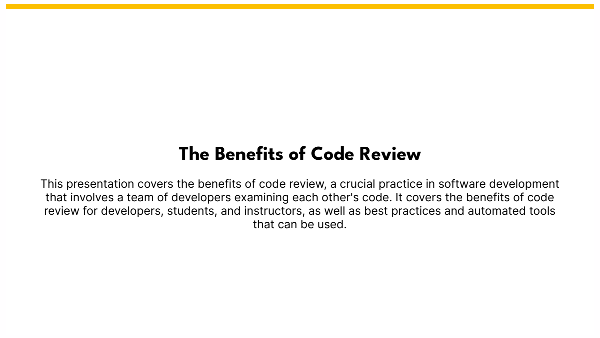
Takeaways

- Code review improves the **code**
- Good code review improves the **coder**
- The devil is *always* in the details
- Everything happens for a **reason**
- Much **research** is needed/done
  - https://doi.org/10.1109/ICSE.2015.131:
    - 50% maintainability, 15% functionality
  - https://doi.org/10.1145/3524842.3527997:
    - missed bugs: semantic (53%), build (16%), analysis checks (9%), compatibility (7%), concurrency (4%), config (4%)
  - https://atomsofconfusion.com

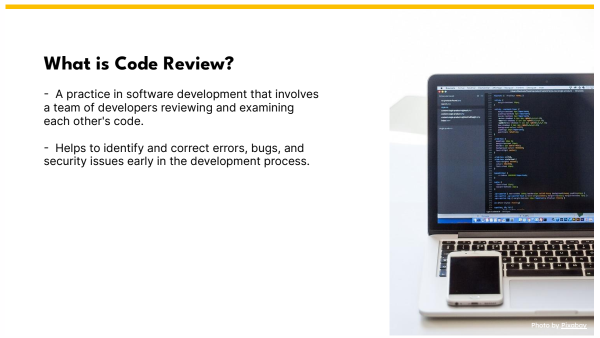
Bonus: ChatGPT + SlidesAI Version

Click to add title

Click to add subtitle

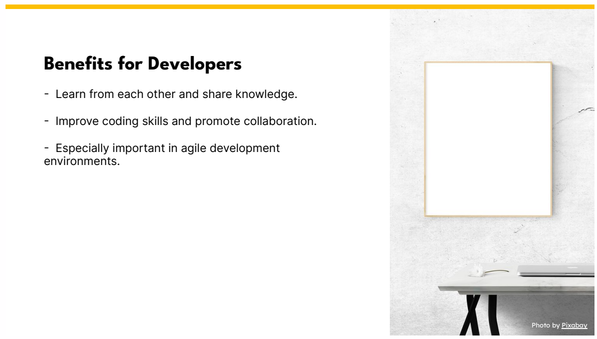

The Benefits of Code Review

This presentation covers the benefits of code review, a crucial practice in software development that involves a team of developers examining each other's code. It covers the benefits of code review for developers, students, and instructors, as well as best practices and automated tools that can be used.


What is Code Review?

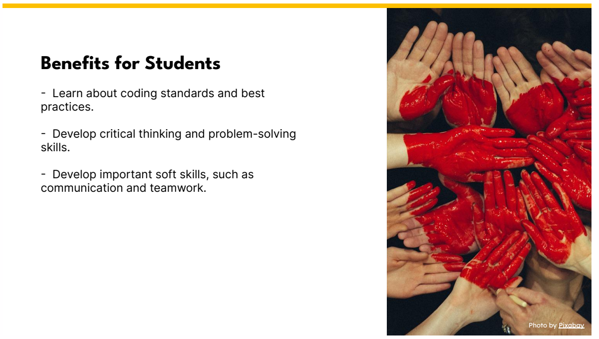
- A practice in software development that involves a team of developers reviewing and examining each other's code.
- Helps to identify and correct errors, bugs, and security issues early in the development process.

Photo by Pixabay


Benefits for Developers

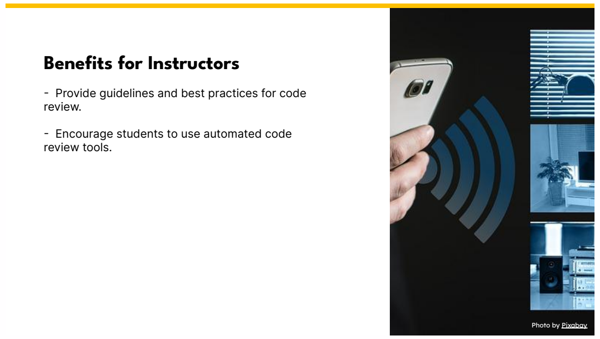
- Learn from each other and share knowledge.
- Improve coding skills and promote collaboration.
- Especially important in agile development environments.

Photo by Pixabay


Benefits for Students

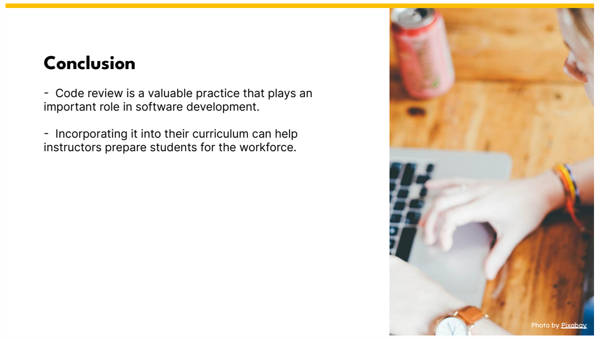
- Learn about coding standards and best practices.
- Develop critical thinking and problem-solving skills.
- Develop important soft skills, such as communication and teamwork.

Photo by Pixabay


Benefits for Instructors

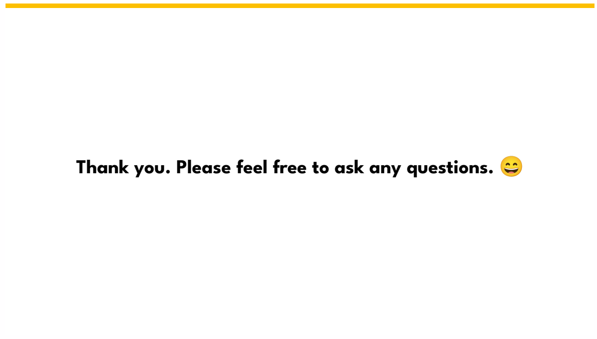
- Provide guidelines and best practices for code review.
- Encourage students to use automated code review tools.

Photo by Pixabay


Conclusion

- Code review is a valuable practice that plays an important role in software development.
- Incorporating it into their curriculum can help instructors prepare students for the workforce.

Photo by Pixabay


Thank you. Please feel free to ask any questions. 😄