Название: "Практические методы повышения качества

программного кода"

Департамент: Кафедра Программной Инженерии (КПИ)

Период реализации: 1,2,3 модули 2023/2024

Язык: Русский

Видео: YouTube

Слайды: GitHub

Охват аудитории: Для всего кампуса

Объем дисциплины: 48 часов семинары + 66 часов самостоятельная

работа

Онлайн курс: —

Технологии реализации: Лекции: офлайн-занятия, практика:

офлайн-занятия

Разработчики: Бугаенко Егор Георгиевич

Утверждение: —

# 1 Цели курса

В лекционной части курса рассматриваются инструменты оценки качества кода и методы его повышения. Теоретическое изложение не привязано к конкретным языкам программирования. Практическая часть курса предполагает самостоятельную работу студентов над анализом крупных open source проектов с целью выявления закономерностей и проблем с качеством.

Ожидается, что по окончанию курса студент будет:

- Понимать, из чего состоит оценка качества программных продуктов;
- Уметь проводить научные исследования в сфере анализа репозиториев с программным кодом;
- Уметь пользоваться инструментами анализа качества кода.

Предполагается, что к началу курса студент умеет:

- Самостоятельно программировать на одном из популярных языков, таких как Java, JavaScript, Python или C++;
- Самостоятельно управлять репозиторием проекта с помощью Git;
- Самостоятельно моделировать архитектуру программного проекта, используя UML.

Лучшие студенты опубликуют результаты своего исследования на одной из научных конференций.

## 2 Лекции

Всего в курсе 24 лекции по два академических часа (80 минут каждая лекция):

- 1. Lines of Code
- 2. McCabe's Cyclomatic Complexity
- 3. Cognitive Complexity
- 4. Halstead Complexity
- 5. Maintainability Index
- 6. Coupling
- 7. LCOM 1, 2, 3, 4, 5
- 8. TCC and LCC
- 9. CAMC and NHD
- 10. Object Dimensions
- 11. Clone Coverage
- 12. Dead Code
- 13. Code Churn
- 14. Tech Debt
- 15. Code Coverage
- 16. Mutation Coverage
- 17. Function Points
- 18. Bugs
- 19. Stars
- 20. Contributors
- 21. Forks and Pull Requests
- 22. Instruction path length
- 23. Number of methods, classes, etc.
- 24. Build Time & Stability
- 25. Algorithmic complexity Big-O
- 26. Neural Metrics

Порядок лекций может меняться, а тематика корректироваться в зависимости от интереса аудитории.

Тип аудиторий: лекционные (для лекционных занятий).

Оснащение аудиторий: наличие Wi-Fi.

#### 3 Аттестация

В рамках курса студенты должны разделиться на группы по 1—3 человека. Каждая группа должна выбрать одну из исследовательских тем, предложенных на первой лекции. По выбранной теме необходимо провести исследование и оформить его результаты в научную статью на английском языке, объемом 7—12 страниц формата <a href="mailto:acmart/sigplan">acmart/sigplan</a>. Статья должна содержать обязательные разделы: Abstract, Introduction, Related Work, Method, Experimental Results, Discussion, Conclusion и References.

Оценка за курс складывается из суммы баллов:

Результат	Баллы
Студент представил черновик статьи (2+ страниц) до	1
конца первого модуля	
Студент представил черновик статьи (4+ страниц) до	1
конца второго модуля	
Студент посетил 10 или более лекций	1
Студент посетил 20 или более лекций	3
Студент отправил статью на конференцию после	1
разрешения преподавателя	
Статья принята на конференции CORE-B	7
Статья принята на конференции CORE-A	10

Пересдача невозможна.

## 4 Самостоятельная работа

Работая самостоятельно над темой исследования, ожидается, что студент выполнит следующее:

- Изучит существующие статьи и книги, касающиеся темы исследования;
- Сформулирует research question;
- Определит метод исследования;
- Проведет исследования и соберет данные;
- Проанализирует результаты;
- Выявит слабые стороны и опишет их;
- Сделает вывод;
- Оформит статью.

Дополнительную помощь можно получить в этой статье: Research Paper Simple Template (и в материалах, на которые статья ссылается).

### 5 Литература

Len Bass et al., Software Architecture in Practice

Paul Clements et al., <u>Documenting</u>
<u>Software Architectures: Views and</u>
<u>Beyond</u>

Karl Wiegers et al.,  $\underline{\text{Software}}$  Requirements

Alistair Cockburn, Writing Effective Use Cases

Steve McConnell, Software Estimation: Demystifying the Black Art

Robert Martin, <u>Clean Architecture: A</u>
<u>Craftsman's Guide to Software Structure</u>
and Design

Steve McConnell, <u>Code Complete</u>
Frederick Brooks Jr., <u>Mythical</u>
Man-Month, The: Essays on Software
Engineering

David Thomas et al., <u>The Pragmatic</u> Programmer: Your Journey To Mastery

Robert C. Martin, <u>Clean Code: A</u>
<u>Handbook of Agile Software</u>
Craftsmanship

Grady Booch et al., <u>Object-Oriented</u> Analysis and Design with Applications

Bjarne Stroustrup, Programming: Principles and Practice Using C++

Brett McLaughlin et al., <u>Head First</u>
Object-Oriented Analysis and Design: A
Brain Friendly Guide to OOA&D

David West, <u>Object Thinking</u>
Eric Evans, <u>Domain-Driven Design:</u>
Tackling Complexity in the Heart of

Eric Evans, <u>Domain-Driven Design:</u>
Tackling Complexity in the Heart of
<u>Software</u>

Yegor Bugayenko, <u>Code Ahead</u>
Yegor Bugayenko, <u>Elegant Objects</u>
Michael Feathers, <u>Working Effectively</u>
with Legacy Code

 $\frac{\text{Martin Fowler, } \underline{\text{Refactoring: Improving}}}{\text{the Design of Existing Code}}$ 

Erich Gamma et al., <u>Design Patterns:</u> Elements of Reusable Object-Oriented Software

Martin Fowler, <u>UML Distilled</u>
Anneke Kleppe et al., <u>MDA Explained:</u>
The Model Driven Architecture: Practice

and Promise

Jez Humble et al., <u>Continuous Delivery:</u>
Reliable Software Releases through
Build, Test, and Deployment
Automation

Michael T. Nygard, <u>Release It!: Design</u> and Deploy Production-Ready Software