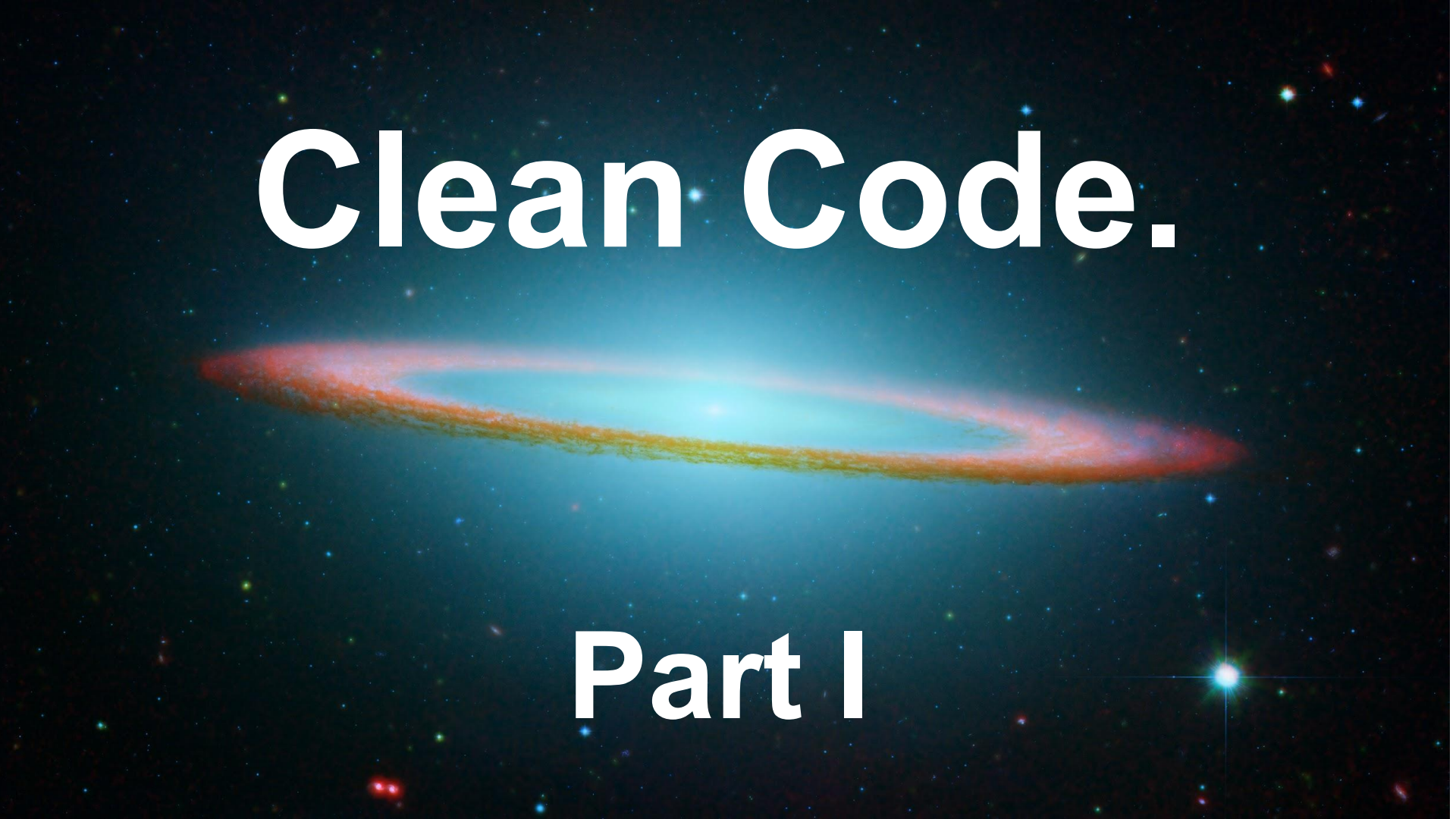
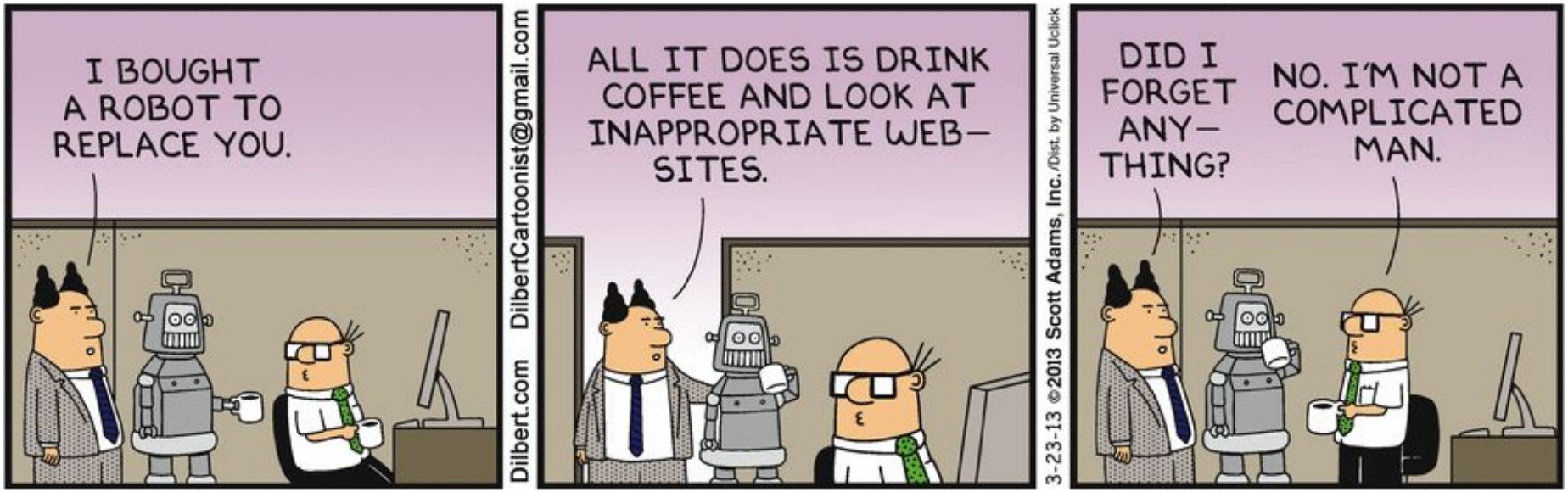

Clean Code.
Part I

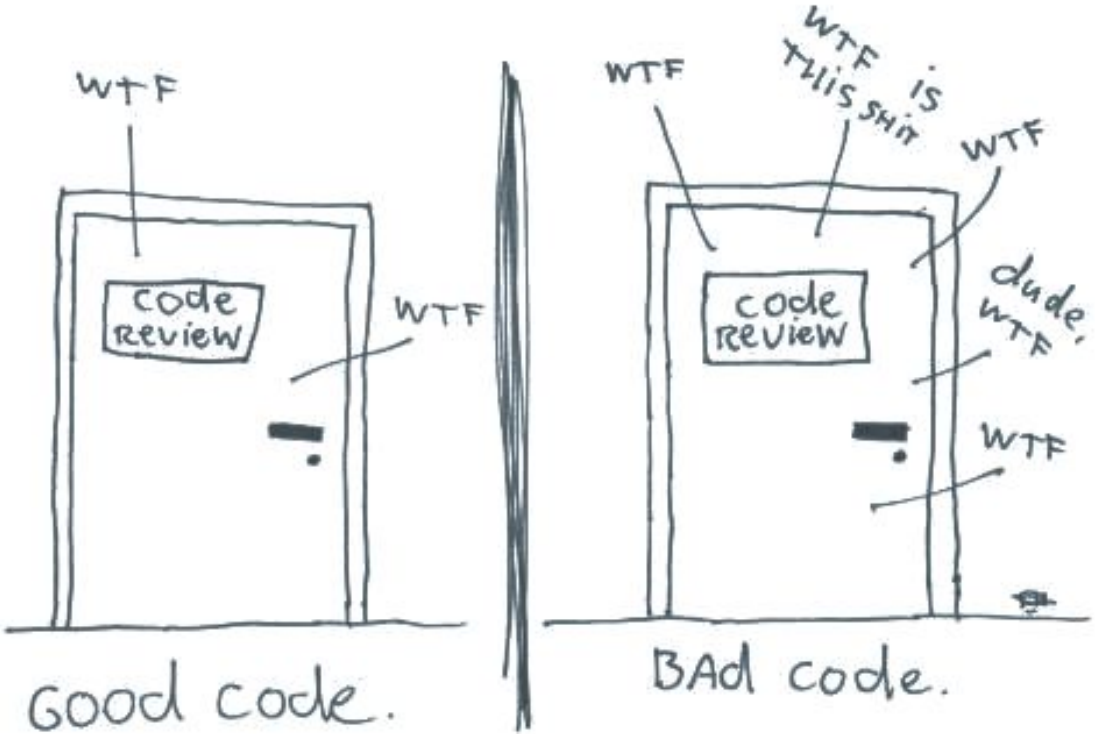


Be future-proof

Honesty in small things is not a small thing.

Какво е лош код?

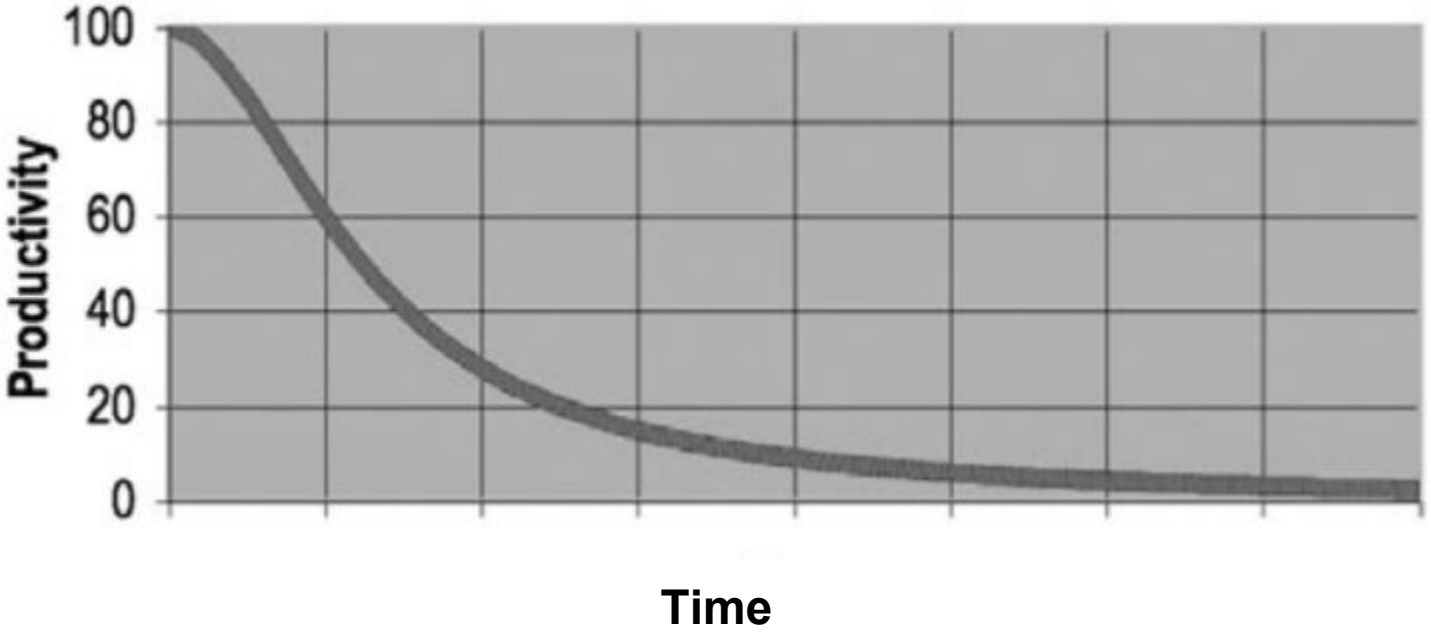
The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE



LeBlanc’s Law:
“Later equals never.”

Продуктивност

на проект с лош код



Смислени имена

- Имена разкриващи намерение
- Не дезинформирайте
- Лесни за търсене имена
- Тип част от името
- Префикси - за добро и лошо
- Интерфейси
- Без едно наум
- Бъдете консистентни

- int d; // elapsed time in days
- public List<int[]> getThem() {

```
    List<int[]> list1 = new ArrayList<int[]>();
    for (int[] x : theList)
        if (x[0] == 4)
        list1.add(x);
    return list1;
}
```

- Names such as accountList
- public static void copyChars(char a1[], char a2[])

```
{
    for (int i = 0; i < a1.length; i++) {
            a2[i] = a1[i];
    }
}
```

PhoneNumber phoneString;

getActiveAccount();

getActiveAccounts();

getActiveAccountInfo();

Class Product, ProductInfo or ProductData

`firstName`, `lastName`, `street`, `houseNumber`, `city`, `state`, and `zipcode`

```
for (int j=0; j<34; j++) {
    s += (t[j]*4)/5;
}
```

```
int realDaysPerIdealDay = 4;
const int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for (int j=0; j < NUMBER_OF_TASKS; j++) {
    int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
    int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);
    sum += realTaskWeeks;
}
```

FUNCTIONS SHOULD DO ONE THING.

THEY SHOULD DO IT WELL.

THEY SHOULD DO IT ONLY.

Методи

- МАЛКИ!
- Идентация, не повече от две
- Правят едно нещо. Правят го добре. Но. Само едно нещо.
- Трябва да разказват история
- Дескриптивни имена
- Аргументи - в най-лошия случай - два
- Не връщайте стойност в аргумент
- Страничните ефекти на вашите функции, са лъжи, с които мамите вас и другите около вас.
- Функцията или променя обект или прави проверка за обект. Никога и двете.
- Try/catch е отделно действие

```
Circle makeCircle(double x, double y, double radius);
Circle makeCircle(Point center, double radius);


public boolean checkPassword(String userName, String password) {
    User user = UserGateway.findByName(userName);
    if (user != User.NULL) {
        String codedPhrase = user.getPhraseEncodedByPassword();
        String phrase = cryptographer.decrypt(codedPhrase, password);
        if ("Valid Password".equals(phrase)) {
            Session.initialize();
            return true;
        }
    }
    return false;
}
```

```
public void delete(Page page) {
    try {
            deletePageAndAllReferences(page);
    } catch (Exception e) {
            logError(e);
    }
}

private void deletePageAndAllReferences(Page page) throws Exception {
    deletePage(page);
    registry.deleteReference(page.name);
    configKeys.deleteKey(page.name.makeKey());
}

private void logError(Exception e) {  logger.log(e.getMessage());  }
```

Много нива на абстрактност наведнъж

- Сложете пред името на метода to и обяснете какво прави
- Пример:
- TO RenderPageWithSetupsAndTeardowns, we check to see whether the page is a test page and if so, we include the setups and teardowns. In either case we render the page in HTML.

To include the setups and teardowns, we include setups, then we include the test page content, and then we include the teardowns.

To include the setups, we include the suite setup if this is a suite, then we include the regular setup.

To include the suite setup, we search the parent hierarchy for the "SuiteSetUp" page and add an include statement with the path of that page.

To search the parent…

DRY - Don’t Repeat Yourself

Duplication may be the root of all evil in software. Many principles and practices have been created for the purpose of controlling or eliminating it.

Duplication is a problem because it bloats the code and will require many-fold modifications should the algorithm ever have to change. It is also a many-fold opportunity for an error of omission.

The art of programming is, and has always been, the art of language design.

Master programmers think of systems as stories to be told rather than programs to be written.

- Uncle Bob

Коментари

“Don’t comment bad code—rewrite it.”

- Нищо не е по-полезно от коментар на място
- Нищо не пречи повече от коментар не на място
- Коментарите се пишат при нашия провал да изразим мислите си чрез код
- Коментарите лъжат, кода - не
- Добрият коментар, е този който си намерил начин да не напишеш
- TODO or not TODO
- Кой коментира код..

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) && (employee.age > 65)) { … }

// Returns an instance of the Responder being tested.
  protected abstract Responder responderInstance();

// Utility method that returns when this.closed is true. Throws an exception
  // if the timeout is reached.
  public synchronized void waitForClose(final long timeoutMillis)  throws Exception  {
     if(!closed) {
              wait(timeoutMillis);
              if(!closed)
                    throw new Exception("MockResponseSender could not be closed");
     }
  }
```

```
/**
 * Default constructor.
 */
protected AnnualDateRule() {
}

/**
 * Returns the day of the month.
 *
 * @return the day of the month.
 */
public int getDayOfMonth() {
  return dayOfMonth;
}

// Actions ////////////////////////////////
```

```
this.bytePos = writeBytes(pngIdBytes, 0);
    //hdrPos = bytePos;
    writeHeader();
    writeResolution();
    //dataPos = bytePos;
    if (writeImageData()) {
        writeEnd();
        this.pngBytes = resizeByteArray(this.pngBytes, this.maxPos);
    }
    else {
       this.pngBytes = null;
    }
    return this.pngBytes;
```

Форматиране

- Кодът се мени, стилът остава
- Пишете новини
- Използвайте празните редове, но никога не оставяйте допълнителни
- Смислово свързан код трябва да е групиран заедно вертикално
- Декларирайте променливи, близо до мястото на използване
- Функции, които се викат една друга, трябва да са близо
- Кратки редове
- Правилата на племето

Юнит тестове

- Качеството на тестовете трябва да е толкова добро, че и по-добро, от качеството на кода
- От поддържането на тестове зависи дали кода може да се променя, дали е лесен за поддръжка, дали е преизползваем
- Четимост
- One assert per test
- Single concept per test

Test Driven Development

First Law. You may not write production code until you have written a failing unit test.

Second Law. You may not write more of a unit test than is sufficient to fail, and not compiling is failing.

Third Law. You may not write more production code than is sufficient to pass the currently failing test.

Ресурси

- https://en.wikipedia.org/wiki/Aether_theories
- http://www.fourmilab.ch/etexts/einstein/specrel/specrel.pdf
- http://ricardogeek.com/docs/clean_code.pdf