

SMARTRHINO 2018

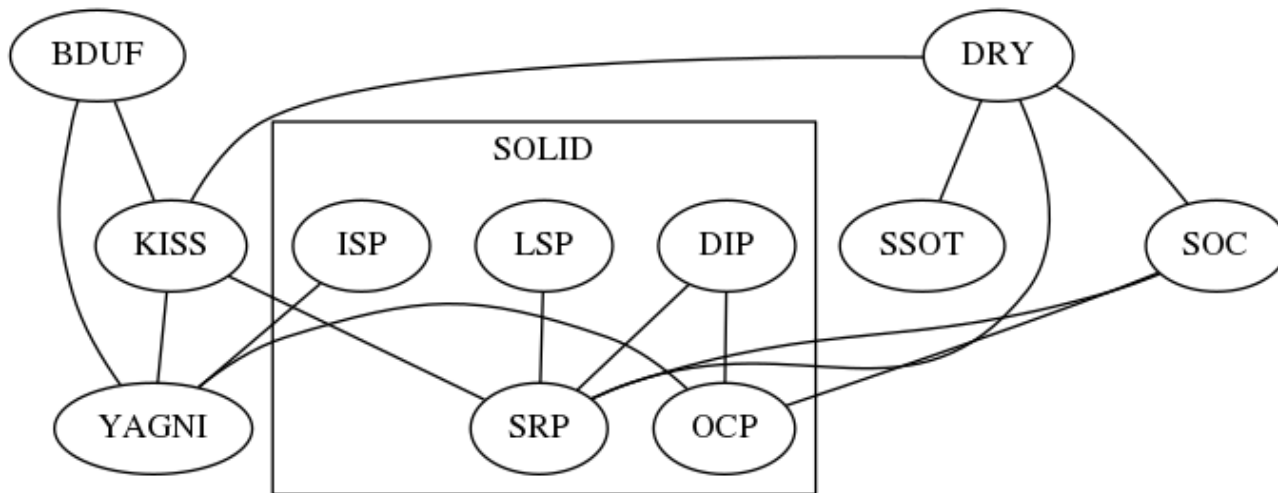


BEST PRACTICES

Влад Чесноков

Тема спорная

- много противоречивых советов;
- тяжело постичь в теории;
- простор для холиваров;
- тяжело воспринимать объективно;

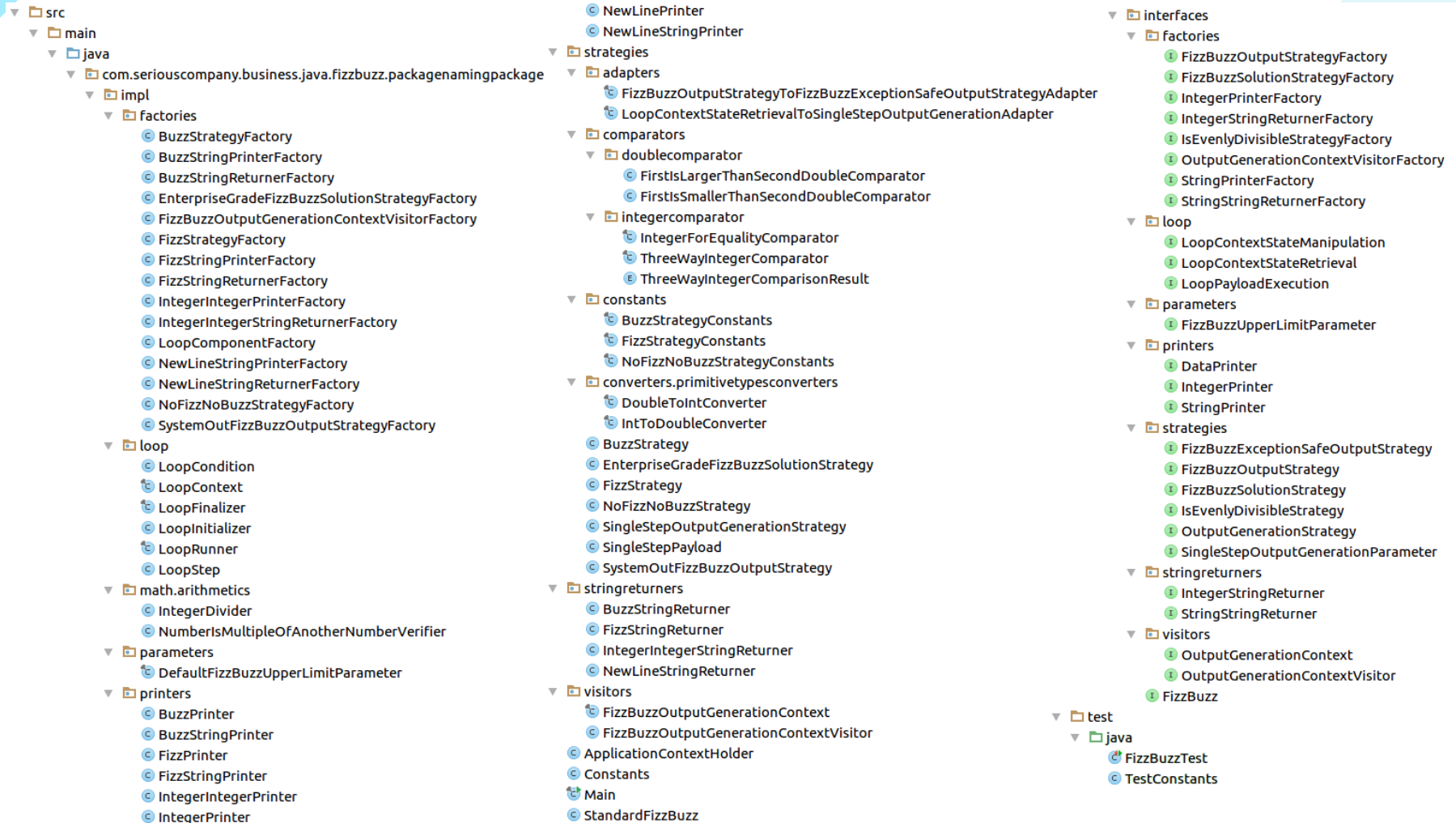


Серебряных пуль не существует



Только Ситхи все возводят
в абсолют.

Слепое следование практикам



Слепое следование практикам

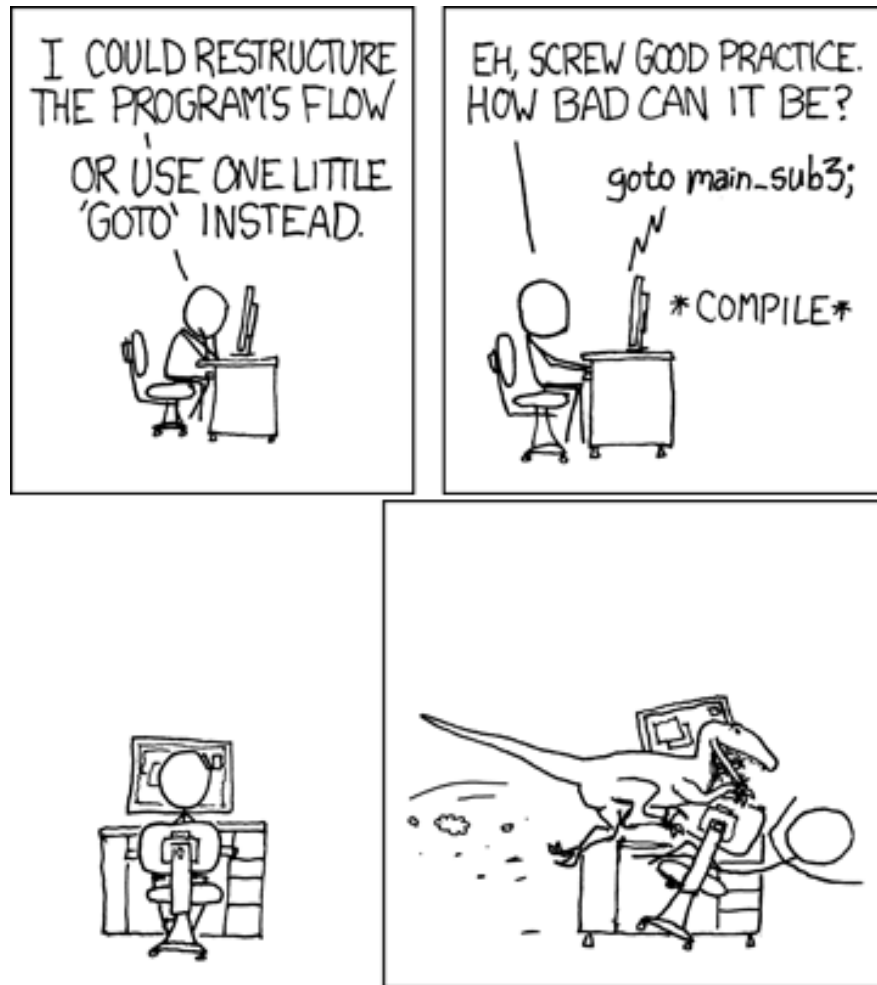
```
public class FizzBuzz{  
    public static void main(String[] args){  
        for(int i= 1; i <= 100; i++){  
            if(i % 15 == 0){  
                System.out.println("FizzBuzz");  
            }else if(i % 3 == 0){  
                System.out.println("Fizz");  
            }else if(i % 5 == 0){  
                System.out.println("Buzz");  
            }else{  
                System.out.println(i);  
            }  
        }  
    }  
}
```

Слепое следование практикам


Singleton:

- самый недопонятый паттерн;
- первая буква в STUPID;
- тяжело тестировать;
- скрытые зависимости;
- большая связность;
- беда в многопоточном приложении.

Слепое следование практикам: goto



Слепое следование практикам: goto



openssl/open...

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Code 439

Commits 22

Issues 154

Wikis

Languages

C X

Pod 3

C++ 2

DIGITAL Command Language 2

Perl 2

Perl 6 1

439 code results in openssl/openssl

[crypto/bn/bn_sqrt.c](#)
Showing the top two matches Last indexed 14 days ago

```
28         if (ret == NULL)
29             ret = BN_new();
30         if (ret == NULL)
31             goto end;
32         if (!BN_set_word(ret, BN_is_bit_set(a, 0))) {
...
48         if (ret == NULL)
49             goto end;
50         if (!BN_set_word(ret, BN_is_one(a))) {
51             if (ret != in)
```

[crypto/dsa/dsa_gen.c](#)
Showing the top three matches Last indexed 9 hours ago

```
95         if ((mont = BN_MONT_CTX_new()) == NULL)
96             goto err;
```

[Advanced search](#) [Cheat sheet](#)

Что дальше?

- «Если нельзя, но очень хочется, то можно»?
- Как понять, стоит ли применять практику или нет?
- «Почему это вообще должно меня заботить»?
- «Работает – и ладно».

1. Цените время программиста

- «время программиста дороже процессорного»;
- цените время ДРУГИХ программистов;
- другие программисты: вы в будущем, ваши коллеги, психопат с топором, который знает, где вы живете;
- цените время программистов-ЧИТАТЕЛЕЙ.

1. Цените время программиста-читателя



Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

— *Martin Fowler* —

AZ QUOTES

Код = рассказ*

- черновик = прототип;
- вычитка на ляпы = проверка логики;
- орфография/синтаксис = code style;
- тестовые читатели = code review;
- публикация = залить в репозиторий.

*как и любая аналогия – это плохая аналогия

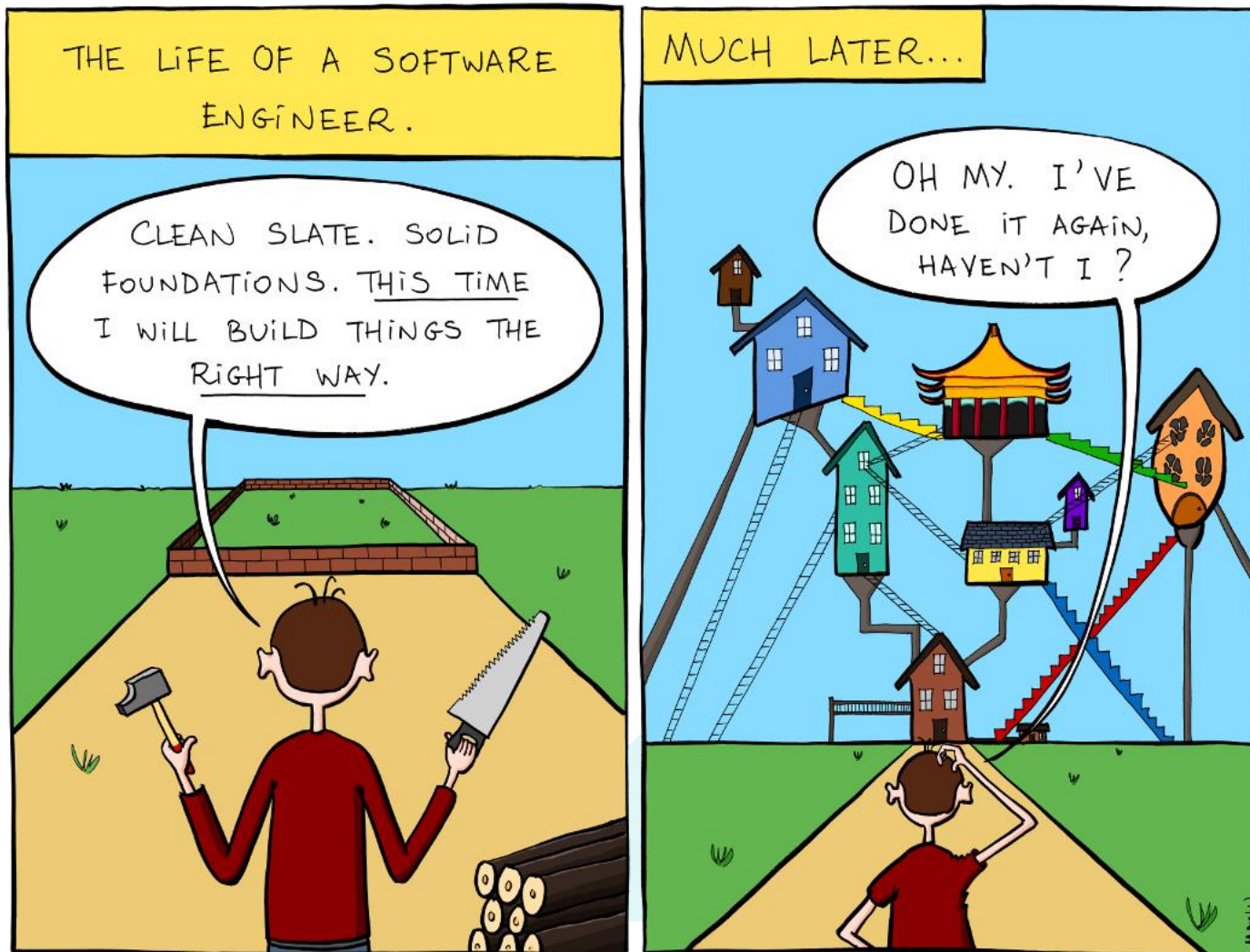
Для читателей надо стараться

- быстро ≠ хорошо;
- после написания – покрыть тестами и отрефакторить.

Образ разработчика, проектирующего программу рациональным безошибочным способом на основе ясно сформулированных требований, совершенно нереалистичен. Никакая система так никогда не разрабатывалась и, наверное, не будет разрабатываться.

Дэвид Парнас и Пол Клементс (David Parnas and Paul Clements)

Выделяйте время на рефакторинг



Иногда есть обстоятельства



Но потом надо переделать или сжечь,
пока не стало продакшеном или легаси!



2. Понимайте, почему вы это делаете

- Зачем вы применяете совет/практику?
- Что вы получите, если примените?
- Что плохого может произойти, если НЕ примените?
- Старайтесь учиться на чужих ошибках, а не на своих.

2. Понимайте, почему вы это делаете

- совет может устареть;
- могут появиться новые технологии;
- совет может оказаться догмой.



3. Изучите доводы против

- Только плюсы = серебряная пуля => дичь
- «Вы все делаете неправильно» => дичь
- Нет наглядных примеров/не могут объяснить => дичь
- У проблемы может быть другое решение - сравните

4. Это случится с вами*

- «Ну, так ошибиться может только дебил»
- «Я же не настолько тупой»



*только ситхи все возводят все в абсолют!

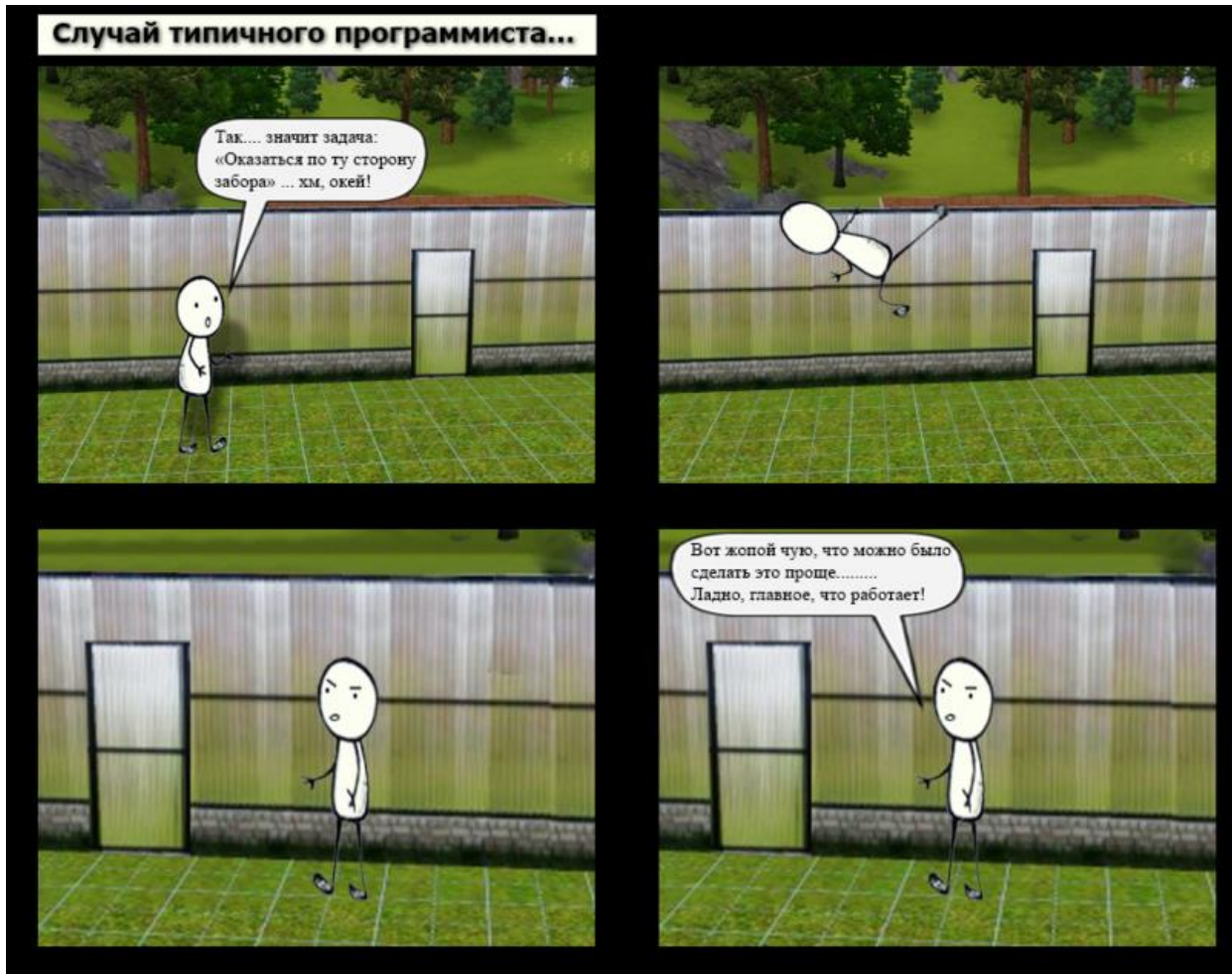
Организация работы

- команда разработчиков;
- тесты;
- система контроля версий;
- баг-трекер / issue-трекер;
- workflow / методология разработки;
- среда разработки;
- инструменты сборки;
- легаси-что-угодно-*;
- документация;
- языки программирования, библиотеки и т.п.;
- поддержка;
-

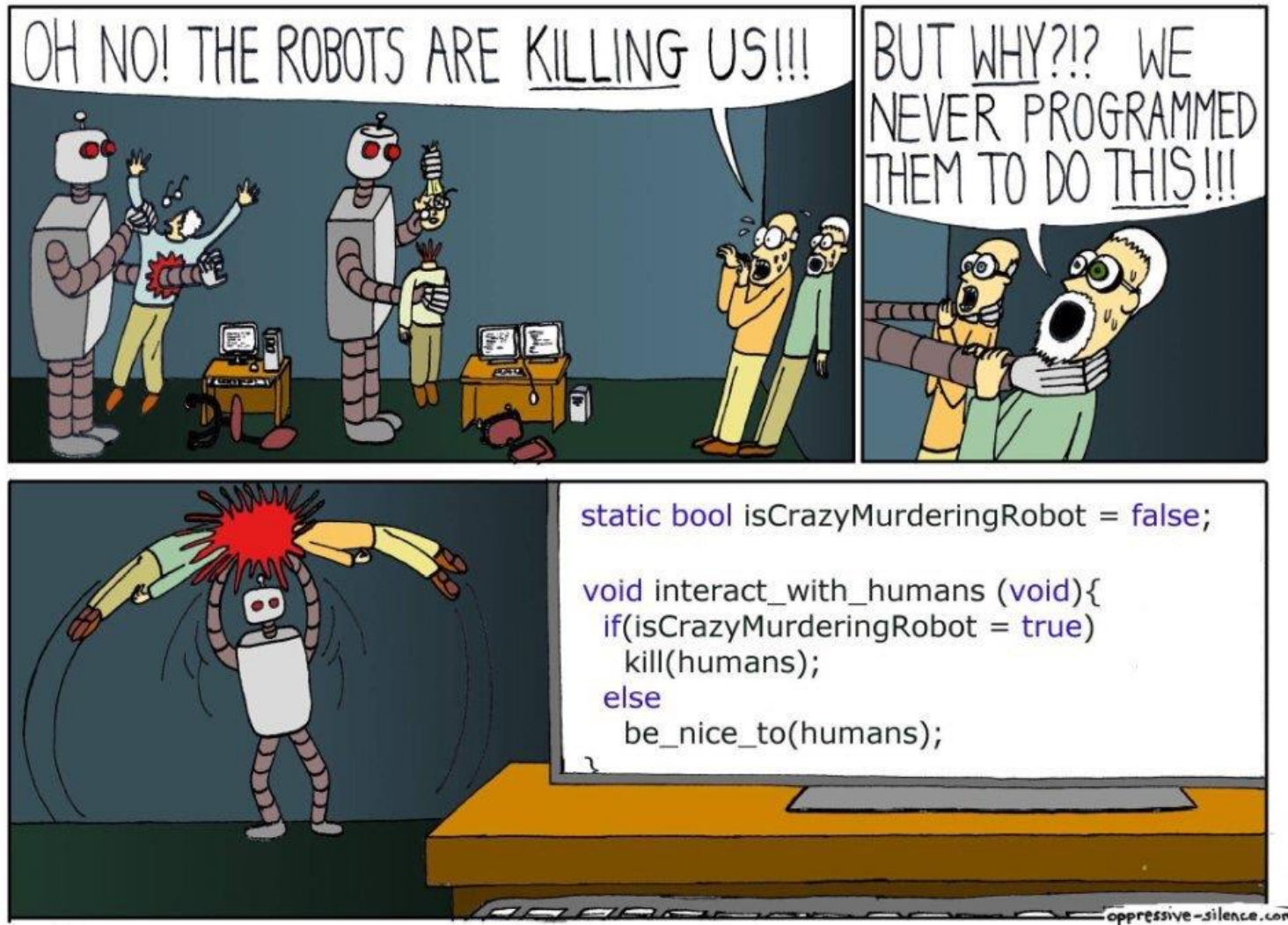
Команда разработчиков

- ваш код смотрят другие люди;
- внезапно, есть другое мнение;
- внезапно, вас и/или ваш код могут не понять;
- внезапно, надо искать компромиссы, совместно решать проблемы, понимать других, перенимать опыт...
- вы можете оказаться неправы;
- Soft skills важны!

Команда разработчиков



Тесты



Тесты

- Тесты надо писать.
- Программу вряд ли будет тестировать кто-то другой.
- Если повезет – будет команда тестирования, которая не разбирается в коде.
- Заказчик не будет тестировать («все сломалось»).
- На клиентах тестировать нельзя.



Тесты

Хорошие тесты – это:

- первое использование вашего кода;
- (хоть какая-то) документация к коду;
 - + документация мелких особенностей
- возможность отрефакторить код с меньшей болью.

Писать хорошие тесты – тяжело.

Хороший программист ≠ хороший тестер.

Тесты

- Покрытие 100% - не панацея.

```
1 def average(lst):  
2     return sum(lst) / len(lst)  
3  
4 assert average([1,2,3,4,5]) == 3
```

- Максимально реальные данные!
- Проверяйте, что тесты не работают на неправильных программах.

```
1 regex = re.compile("[A-Za-z_0-9]*")  
2 assert regex.matches("login")  
3 assert regex.matches("under_score_login")  
4 assert regex.matches("login24")  
5 assert regex.matches("LOGIN")  
6 assert regex.matches("_login_")
```

Тесты

- Покрытие 100% - не панацея.

```
1 def average(lst):  
2     return sum(lst) / len(lst)  
3  
4 assert average([1,2,3,4,5]) == 3
```

- Максимально реальные данные!
- Проверяйте, что тесты не работают на неправильных программах.

```
1 regex = re.compile("[A-Za-z_0-9]*")  
2 assert regex.matches("login")  
3 assert regex.matches("under_score_login")  
4 assert regex.matches("login24")  
5 assert regex.matches("LOGIN")  
6 assert regex.matches("_login_")  
7 regex = re.compile(".*")
```

Тесты

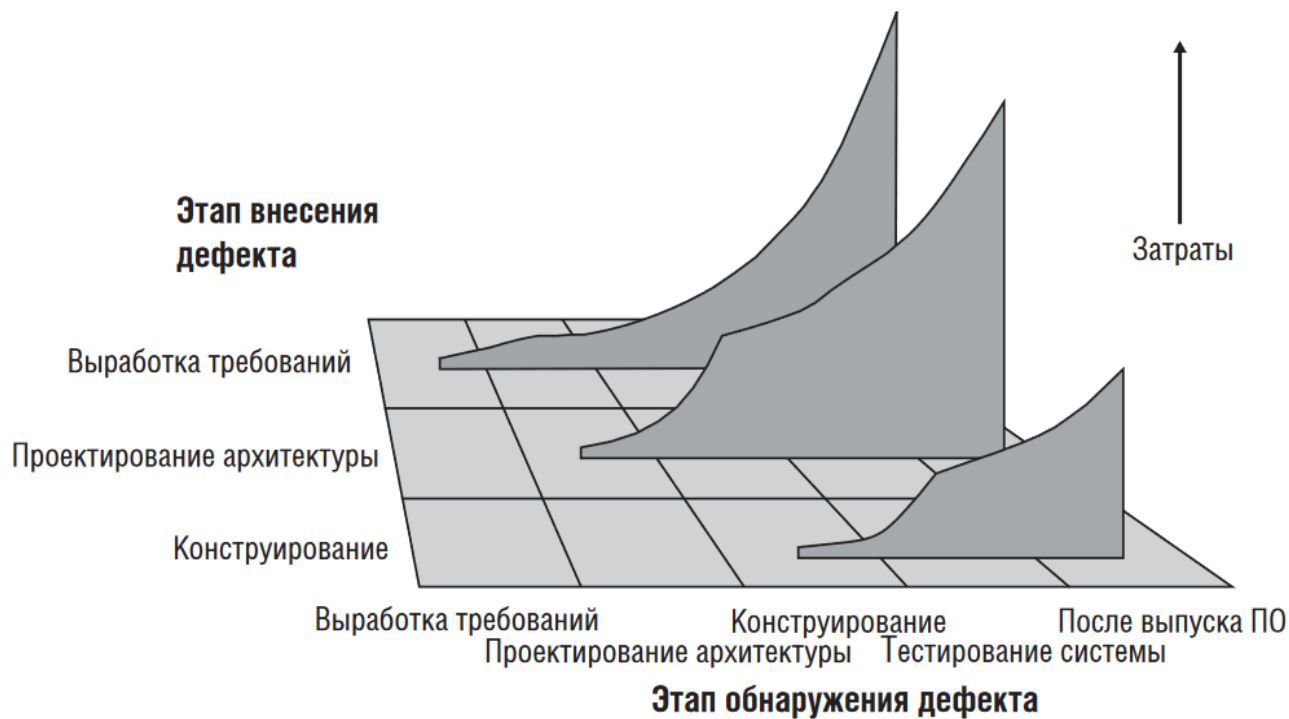


Рис. 3-1. С увеличением интервала между моментами внесения и обнаружения дефекта стоимость его исправления сильно возрастает. Это верно и для очень последовательных проектов (выработка требований и проектирование на 100% выполняются заблаговременно), и для очень итеративных (аналогичный показатель равен 5%)

Система контроля версий

- тяжело придумать, почему она не нужна;
- позволяет эффективно совместно работать над кодом;
- хранит историю изменений;
- можно откатить свои изменения;
- хранит авторство;
- связывает изменения кода и задачи;
- интеграция с билд-сервером;
- и другое.

Workflow

- зависит от множества факторов;
- записывайте задачи и проблемы, которые не можете сделать сейчас (так, чтобы понять через год!);
- не делайте много задач сразу;
- «как-нибудь сделаю» = «получится как-нибудь»;
- много связей – хорошо (помогает в расследованиях):

код <-> коммит <-> тикет <-> документация

Инструменты



Знайте свои инструменты и как они устроены.

Выбирайте инструмент под задачу.



Выбирайте не за достоинства, а по результатам сравнения.

Автоматизируйте билд

- проще тестировать;
- понятно, как работает;
- не нужны мутные инструкции;
- проблемы легче воспроизвести.

5. You'll have several errors, because you need to build the projects once with ant (on the command line: `cd /path/to/processing/build && ant`) so that the "generated" folder is created. After doing that, select the processing-app project in Eclipse and hit F5 to refresh it.

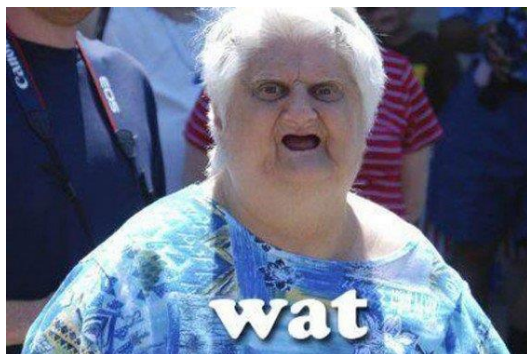
Автоматизируйте рутину

- повод изучить новое;
- экономия времени*;
- рутинные задачи лень делать;
- быстрая обратная связь.

* см. xkcd #1319



Легаси



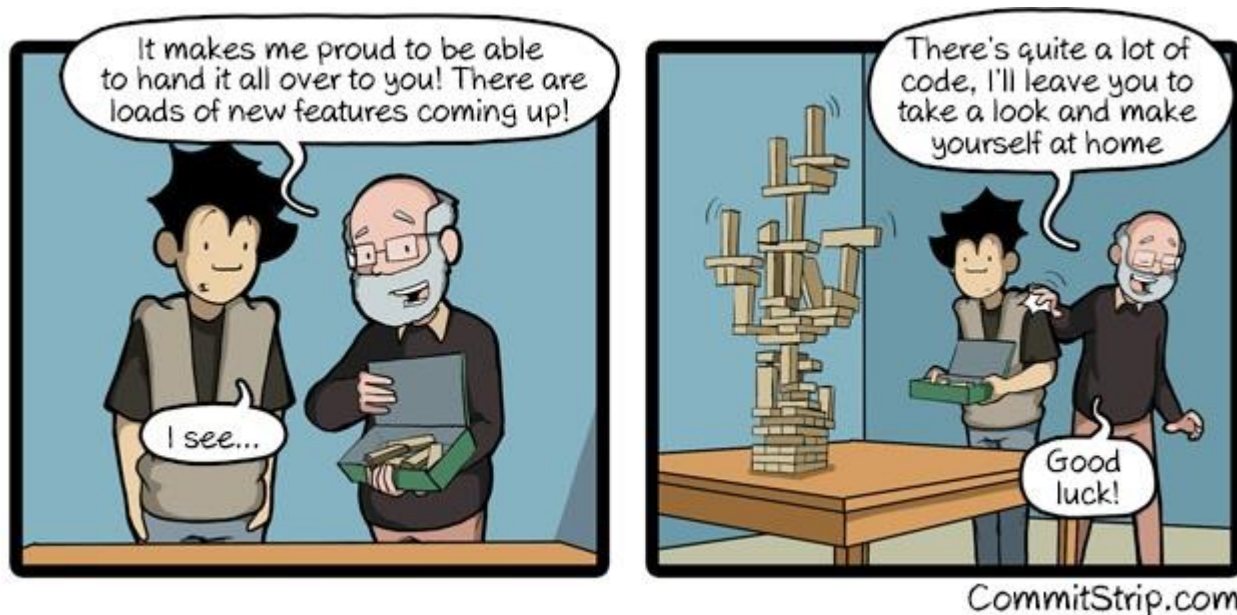
- документация на арі, языки, фреймворки;
- примеры кода;
- устаревшие фичи в языке;
- устаревшие фреймворки, протоколы, и т.д.

При этом:

- старый ≠ устаревший;
- новый ≠ хороший ≠ перспективный.

Легаси

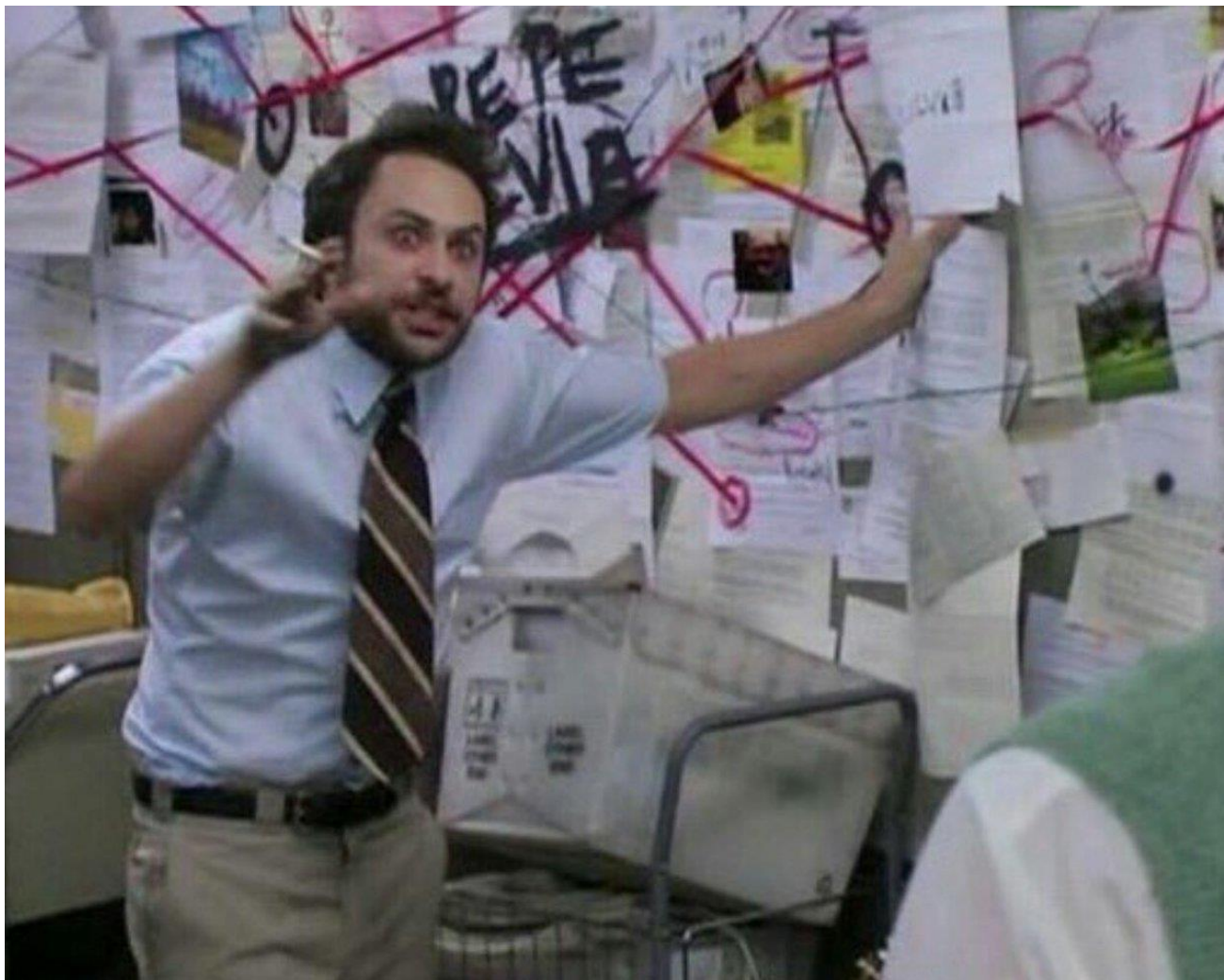
- не тяните устаревшее в новый код;
- постепенно избавляйтесь от легаси;
- но не «чините» то, что работает.



Документация и общение

- Если хочешь что-то понять – объясни другому.
- Код с документацией лучше, чем без нее.
- Пересказывайте код в формате документации, рефакторинга, отчетов по ДЗ и т.д.
- Легко читать код => легко объяснять => легко документировать.

Документация и общение



Окружение

- каждый проект уникален;
- окружение меняется со временем;

Проектирование — эвристический процесс



Так как проектирование не детерминировано, методы проектирования чаще всего являются эвристическими методами, т. е. «практическими правилами» или «способами, которые могут сработать», а не воспроизводимыми процессами, которые всегда приводят к предсказуемым результатам. Проектирование — метод проб и ошибок. Инструменты или методы проектирования, оказавшиеся эффективными в одном случае, в другой ситуации могут оказаться куда менее эффективными. Универсальных методик проектирования не существует.

Наконец-то про код!

Нестареющая классика:

- KISS – **K**ee**P** It **S**imple, **S**tupid.
- YAGNI – **Y**ou **A**in't **G**onna **N**eed It.
- не усложняйте (вам еще потом это поддерживать);
- требования могут меняться резко и непредсказуемо;
- код жаль выкидывать – вот и не пишите лишнего;
- чуть-чуть наперед все-таки надо думать.

Декомпозиция

- Разбивайте задачу на более мелкие.
- Не бросайтесь все написать сразу.

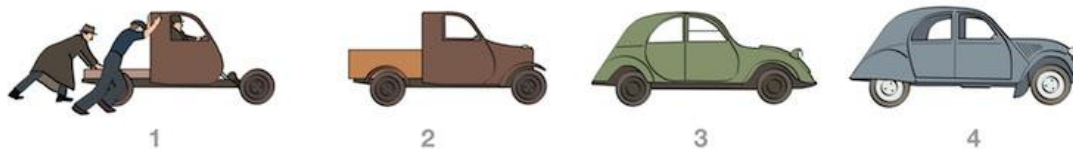
HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT



ALSO HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT



HOW TO BUILD A MINIMUM VIABLE PRODUCT



FRED VOORHORST

WWW.EXPRESSIVEPRODUCTDESIGN.COM

Модульность

Разбивайте вашу программу на
небольшие логические элементы

- «Зачем мне выделять этот код в отдельный метод?
Я же его использую только один раз!»
- «Лишняя переменная – лишняя память!»
- «Зачем миллион файлов, и так норм.»

Модульность

```
64 int main (int argc, char *argv[])
65 {
66     // считаем частоты символов
67     int vibor = 0;
68     if(argc<2) {
69         cout << "Сжать файл - 1" << endl << "Разжать ф
70         cin >> vibor;
71     } else {
72         vibor = atoi(argv[1]);
73     }
74
75     if(vibor == 1 || vibor == 2) {
76         string MyFile;
77         if(argc<2) {
78             cout << "Введите название файла:" << endl;
79             cin >> MyFile;
80         } else {
81             MyFile = argv[2];
82         }
83
84         ifstream f(MyFile+".txt", ios::out | ios::bina
85
86         map<char,int> m; //буква и ее счетчик
87
88         int stretch1 = 0;
89         bool reset = true;
90         while (!f.eof()) {
91             // пробегаемся по всему тексту и заносим в ма
92             // пробегаемся по тексту один раз. Грубо гово
93             if(vibor==1) {
94                 char c = f.get();
95                 m[c]++;
96             } else {
97                 char c1,c2,c3;
98                 if(reset) {
99                     ofstream Temp("~TempFile.tmp", ios
100                     while(!f.eof()) {
101                         c1 = f.get();
102                         if (c1 == '\n') {
103                             c1 = f.get();
104                             if(c1 == '#') {
105                                 reset = false;
106                                 break;
107                             }
108                         }
109                         Temp<<c1;
```

```
129 // записываем начальные узлы в список list
130 list<UzelDereva*> t; //список массива указателей для работы
131 for( map<char,int>::iterator itr=m.begin(); itr!=m.end(); ++i
132     UzelDereva *p = new UzelDereva;
133     //"Разбиваем" map
134     p->c = itr->first; //буква
135     p->a = itr->second; //число
136     t.push_back(p); //кладем все в список
137 }
138
139 // создаем дерево
140
141 while (t.size()!=1) {
142
143     UzelDereva *root = t.front(); //root - указатель на вершин
144
145     // создаем пары 'символ-код':
146
147     BuildThree(root); //проход по дереву
148
149     // Выводим коды в файл output.txt сжатый формат
150
151     f.clear();
152     f.seekg(0); // перемещаем указатель снова в начало файла
153     if(vibor == 1) {
154         if(argc<2) {
155
156             ofstream Output(MyFile+".txt", ios::out | ios::binary);
157
158             int count=0;
159             char buf=0;
160             while (!f.eof()) {
161                 Output<<endl;
162
163                 Output.close();
164
165                 ofstream fil(MyFile+".txt", ios::app | ios::binary); //
166                 for( map<char,int>::iterator itr=m.begin(); itr!=m.end();
167                 fil<<"#";
168                 fil.close();
169
170             // Выве код в бинарном виде в файл binary.txt в виде 0 и 1
171             cout<<"Сжаты ли данные?<endl><<"<<endl><<"<<endl><<
172             int g;
```

Модульность

- читаемость;
- быстрый поиск места для правок;
- проще отлаживать;
- проще тестировать;
- видны связи по аргументам;

```
35 int main (int argc, char *argv[]){
36     if(argc != 4 || argv[1][0] != 'c' && argv[1][0] != 'd')
37         printf("Неверный формат: <сжать/расжать (c/d)> <файл> <файл>");
38     return 1;
39 }
40 FILE *ifile, *ofile;
41 ifile = fopen(argv[2], "r");
42 ofile = fopen(argv[3], "w");
43 CheckFiles(ifile, ofile);
44
45 if (argv[1][0] == 'c'){
46     Zip(ifile, ofile);
47 } else{
48     UnZip(ifile, ofile);
49 }
50 return 0;
51 }
52 void Zip(FILE *ifile, FILE *ofile){
71 void WriteBin(FILE *ifile, FILE *ofile, hash_map<char, int> &frequencies, hash_map<char, int> &tree){
108 void UnZip(FILE *ifile, FILE *ofile){
134 void WriteText(FILE *ifile, FILE *ofile, hash map<char, int> &frequencies, hash map<char, int> &tree){
157 UzelDereva Huffman(hash map<char, int> &frequencies, hash map<char, int> &tree){
193 void BuildThree(UzelDereva *root, hash map<char, int> &frequencies, hash map<char, int> &tree){
212 void insertSort(UzelDereva* a, int size) {
228 void ShiftDown(UzelDereva *object, int index, int size){
252 void Sort(UzelDereva *object, unsigned int size){
277 void CheckFiles(FILE *ifile, FILE *ofile){
```

Самодокументирующийся код

- читатели должны понимать, что происходит, без дополнительной информации;
- говорящие имена;

```
1 ▾ for (int n = 0; n < M; n++){  
2     fgets(buff, 255, FIN);  
3     int k1 = (buff[0]=='?')?rand() %  
4     fgets(buffer, 255, FIN);  
5     int k2 = (buff[0] == '?') ? rand(  
6     if (k2 <= k1) k2 = k1 + 1;  
7     if (k2 >= N) k2 = N;  
8     F[k1].next = k2;
```



```
1 ▾ for (int cell = 0; cell < cellsCount; cell++){  
2     fgets(buffer, MAX_STRING_SIZE, inputFile);  
3     int startCell = parseCell(buffer);  
4     fgets(buffer, MAX_STRING_SIZE, inputFile);  
5     int finishCell = parseCell(buffer);  
6 ▾ if (finishCell <= startCell){  
7         finishCell = startCell + 1;  
8     }  
9 ▾ if (finishCell >= fieldSize){  
10         finishCell = fieldSize;  
11     }  
12     field[startCell].next = finishCell;
```



Комментарии

- код говорит сам за себя => чаще всего не нужны;
- не капитаньте;

```
1 average_count = sum(values) / len(values) | # get average count of values
```

- неочевидные вещи => ссылка на документацию;

```
if (user == null){  
    return ""; // backward compatibility, see #9931 :(  
} else {  
    return user;  
}
```

- код в комментариях хранить не надо – у вас есть история в репозитории (или он не нужен по YAGNI);
- документация в комментариях – зависит от проекта.

SRP – одна ответственность

```
1 function getCancelMessage(data){  
2     ...  
3     return message  
4 }
```

SRP – одна ответственность

```
1 function getCancelMessage(data){  
2     ...  
3     gui.toggleSchedulesButtons("NOT_READY");  
4     return message;  
5 }
```

- побочный эффект + неочевидное изменение;
- неподготовленный читатель может психануть, пока найдет нужное место для правок;
- если тяжело назвать – может, что-то не так?

```
if (uncompressedSizes.length > 2000) {  
    HighlyCompressedMapStatus(loc, uncompressedSizes)  
} else {  
    new CompressedMapStatus(loc, uncompressedSizes)  
}
```

Стиль кода

- простор для холивара;
- привычный стиль повышает скорость чтения;
- лучше безобразно, но единообразно;
- визуальное распознавание типичных кусков.

Обычно решается автоформатингом + небольшим количеством соглашений

Стиль кода

```
1 public TaskExecutor(IFilesService filesService, ISQLClient sqlClient,  
2     IParser parser, IBlacklist blacklist, IReadOnlyCollection<IFileWorker>  
3     fileWorkers, IReadOnlyCollection<IFileWorker> validators){  
4     IFilesService _tempFilesService;  
5     int MaxContextDepth = 5;  
6  
7     this.FilesService = filesService;  
8     this.SqlClient = sqlClient;  
9     this.Parser = parser;  
10    this.Blacklist = blacklist;  
11    this.FileWorkers = fileWorkers;  
12    this.Validators = validators;  
13    ....
```

Стиль кода

```
1  public TaskExecutor(  
2      IFilesService filesService,  
3      ISQLClient sqlClient,  
4      IParser parser,  
5      IBlacklist blacklist,  
6      IReadOnlyCollection<IFileWorker> fileWorkers,  
7      IReadOnlyCollection<IValidator> validators){  
8      IFilesService _tempFilesService;  
9      int MaxContextDepth = 5;  
10  
11     this.FilesService = filesService;  
12     this.SqlClient = sqlClient;  
13     this.Parser = parser;  
14     this.Blacklist = blacklist;  
15     this.FileWorkers = fileWorkers;  
16     this.Validators = validators;  
17     ...
```

Стиль кода

```
1  public TaskExecutor(  
2      IFileService fileService,  
3      ISQLClient sqlClient,  
4      IParser parser,  
5      IBlacklist blacklist,  
6      IReadOnlyCollection<IFileWorker> fileWorkers,  
7      IReadOnlyCollection<IValidator> validators  
8  ){  
9  
10     IFileService _tempFileService;  
11     int MaxContextDepth = 5;  
12  
13     this.FileService = fileService;  
14     this.SqlClient = sqlClient;  
15     this.Parser = parser;  
16     this.Blacklist = blacklist;  
17     this.FileWorkers = fileWorkers;  
18     this.Validators = validators;  
19     ...
```

Стиль кода

```
1  if (shouldSkip(item)){  
2      // много кода  
3      // всякого-разного  
4      // тут могла быть  
5      // ваша реклама  
6      // вот  
7  } else  
8      log.info("Skipping")
```

Стиль кода

```
1  if (shouldSkip(item)){  
2      // много кода  
3      // всякого-разного  
4      // тут могла быть  
5      // ваша реклама  
6      // вот  
7  } else  
8      log.info("Skipping")  
9      cache.clear()
```

У читателя ограниченная память

- Пример: инициализация при первом использовании.

```
1 DBConnection dbConnection;  
2 Cache usersCache;  
3 Response response;  
4 int tmp1, tmp2;  
5 ....  
6 ....  
7 ..../* 15 minutes later*/  
8 ....  
9 ....  
9010 dbConnection = SqlDriver.open(connectionString);
```

У читателя ограниченная память

Пример: глобальные переменные

- надо помнить текущее состояние;
- надо помнить, кто может изменить;
- увеличивают связность кода;
- тяжело покрывать тестами.

У читателя ограниченная память

Пример: ранний возврат

```
1  if (dbConnection != null){  
2      ...  
3      ...  
4      /* 15 minutes later*/  
5      ...  
6      ...  
7  } else {  
8      throw new IllegalArgumentException("Db connection is null")  
9  }
```

```
1  if (dbConnection == null){  
2      throw new IllegalArgumentException("Db connection is null")  
3  }  
4  
5  ...  
6  ...  
7  ...
```


Не дублируйте код (DRY)



- тяжело менять / поддерживать;
- надо думать, в чем разница;
- PVS Studio: очень много ошибок при копипасте;
- дублирование в тестах: $A==A$.

Не оптимизируйте заранее

- «Я так пишу потому что так быстрее»
Set the m-th bit of n to 0 (from low to high)

```
n & ~(1 << (m-1));
```

n + 1

```
~n
```

- не делайте работу компилятора (см. про мартышек);
- оптимизируйте с профилировщиком;
- используйте подходящие алгоритмы и структуры данных.

Имену́йте константы

- повышает читабельность: 140 vs MAX_TWEET_SIZE;

(если нейминг адекватный)

```
1 // http://shitcode.net/96
2 const int TWENTY_EIGHT = 28;
```

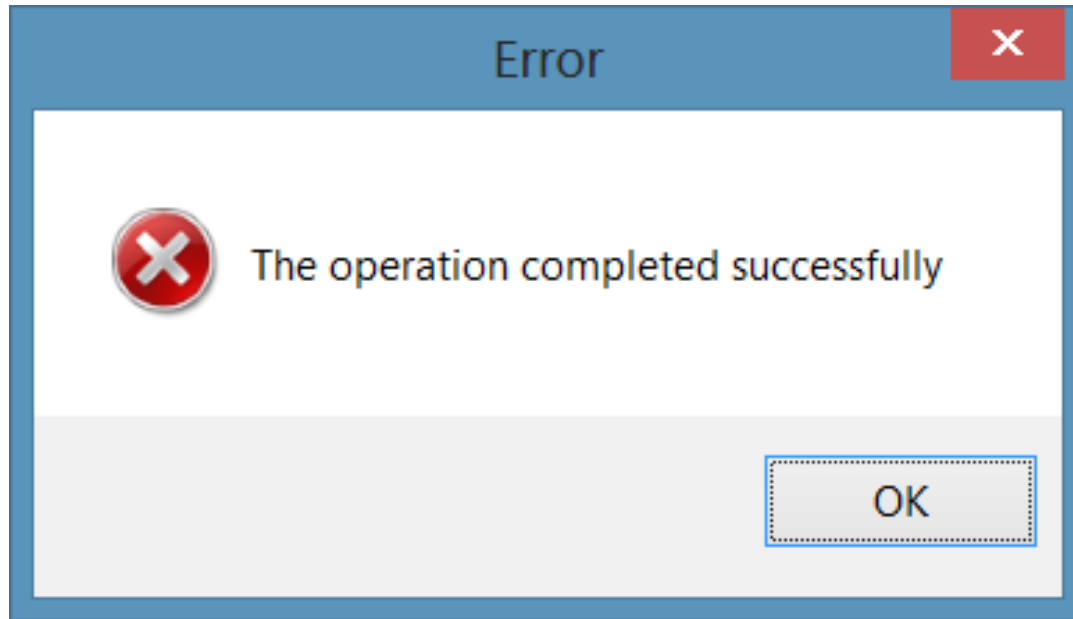
- облегчает рефакторинг;
- семантическое разделение:

одинаковое значение ≠ одинаковый смысл

- без именованя – это дублирующийся код.

Специальные значения

- `const unsigned int NO_ERROR = 0;`



- -1 (INT_MIN), 1970 год, "" и т.п.
- Разделяйте ошибку и успех: код возврата, nullable, Optional, Either, ...

Правильные типы

- используйте адекватные типы;
- ограниченное число значений => enum;

```
1 def join(right: Dataset[_], usingColumns: Seq[String], joinType: String): ...
2
3 object JoinType {
4   def apply(typ: String): JoinType = typ.toLowerCase(Locale.ROOT).replace("_",
5     case "inner" => Inner
6     case "outer" | "full" | "fullouter" => FullOuter
7     case "leftouter" | "left" => LeftOuter
8     case "rightouter" | "right" => RightOuter
9     ...
}
```

- компилятор вам может помочь:

```
1 val shirtColor = when(gender){
2   MALE -> Colors.BLUE, // не скомпилируется, если
3   FEMALE -> Colors.PINK // появится пол APACHE_HELICOPTER
4 }
```

- используйте синонимы типов:

```
6 typealias MyHandler = (Int, String, Any) -> Unit
7 typealias Properties = Map<String, Any>
```

Обработка ошибок

- описание ошибки должно быть понятным;
- пример:

NullPointerException из-за опечатки в конфиге;

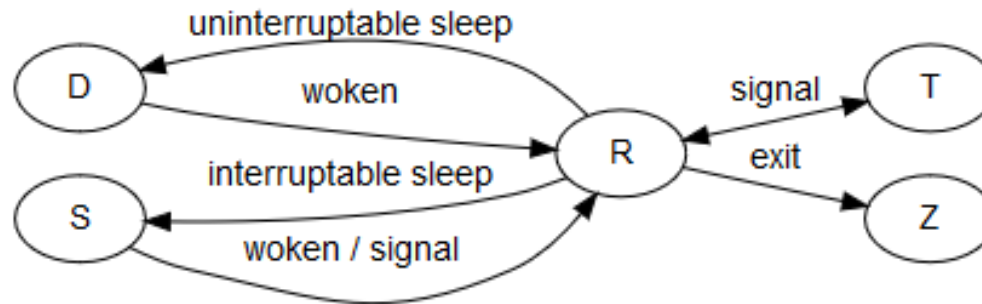
```
1 val zoneManagers: Map<String, ZoneManager> = Config.load();  
2 val zoneManager = zoneManagers[zoneName]!!
```

- не глотайте исключения (привет, OOM):

```
protected static void safeExecute(Runnable task) {  
    try {  
        task.run();  
    } catch (Throwable t) {  
        logger.warn("A task raised an exception. Task: {}", task, t);  
    }  
}
```

Условия

- если поведение зависит от состояния => FSM;



- табличка истинности для сложных комбинаций;
- не делайте много инверсий/запутанности:

```
1  if (size1 < 5 || a != b || !(b == c && c == d))
2      return false;
3
4  if (size1 > 5 && a == b && b == c && c == d)
5      return true;
```

Функциональное программирование

- читаемость – короткие, узнаваемые конструкции;
- иммутабельность:
 - меньше надо помнить,
 - легче дебажить,
 - легче тестировать,
 - легче параллелить;
- но все это не бесплатно.

Функциональное программирование

```
1  var categorizedProducts = new Dictionary<string, List<Product>>();
2  foreach (var product in this.Data.Products){
3      if (product.IsEnabled){
4          if (!categorizedProducts.ContainsKey(product.Category){
5              categorizedProducts.Add(product.Category, new List<Product>());
6          }
7          categorizedProducts[product.Category].Add(product);
8      }}
9  foreach (var productsInCategory in categorizedProducts){
10     var minimumPrice = double.MaxValue;
11     var maximumPrice = double.MinValue;
12     foreach (var product in productsInCategory.Value){
13         if (product.Price < minimumPrice){
14             minimumPrice = product.Price;
15         }
16         if (product.Price > maximumPrice){
17             maximumPrice = product.Price;
18         }
19     }
20     yield return new PricesPerCategory(category: productsInCategory.Key, minimum
21 }
```

Функциональное программирование

```
1  return this.Data.Products
2      .Where(product => product.IsEnabled)
3      .GroupBy(product => product.Category)
4      .Select(productsInCategory =>
5          new PricesPerCategory(
6              category: productsInCategory.Key,
7              minimum:  productsInCategory.Value.Min(product => product.Price),
8              maximum:  productsInCategory.Value.Max(product => product.Price)
9          )
10 );
```

Расследуйте баги



Заключение

- это все было очень поверхностно;
- только ситхи все возводят в абсолют;
- понимайте, зачем следовать практике;
- код для читателя;
- учитывайте контекст и особенности проекта;
- рассматривайте плюсы и минусы, сравнивайте;
- учитесь на ошибках (лучше на чужих);

Заключение – что дальше?

- больше программируйте;
- изучайте другие языки и парадигмы;
- избегайте радикализации своих подходов;
- постоянно изучайте что-то новое*;
- читайте книги и блоги, чтобы упорядочить свой опыт.

* Тут могла быть ваша шутка про js-фреймворки.

Литература

- Весь список есть здесь:

<http://pastebin.com/FdLzyF8g>

- Макконнелл «Совершенный код».
- Мартин «Чистый код».
- Фаулер «Рефакторинг»
- <https://github.com/97-things/97-things-every-programmer-should-know>
- Кнут о goto <http://www.clifford.at/cfun/cliffdev/p261-knuth.pdf>
- <https://github.com/EnterpriseQualityCoding/FizzBuzzEnterpriseEdition>
- <http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code>
- <https://code.google.com/archive/p/google-singleton-detector/wikis/WhySingletonsAreControversial.wiki>
- <https://habrahabr.ru/post/350742/>
- Как менять лампочку <https://www.youtube.com/watch?v=AbSehcT19u0>



Спасибо за внимание!

Вопросы?