Одиннадцатая независимая научно-практическая конференция «Разработка ПО 2015»

22 - 24 октября, Москва

Опыт работы с метриками для обеспечения качества ПО

Колесников Александр

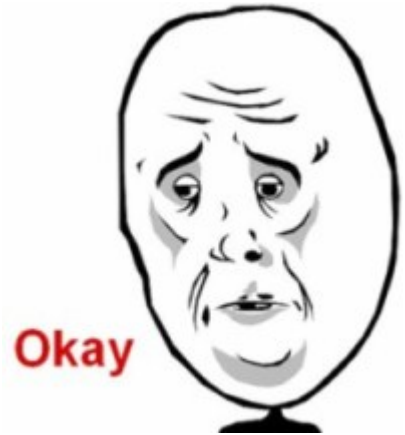
KASPERSKY lab

Допущения

- Доклад не является попыткой охватить весь имеющийся материал по данной тематике
- «Мы не можем управлять тем, что мы не можем измерить» (*Том ДеМарко*)
- Мы можем ловить баги на этапе разработки, до тестирования

Реальный кейс

- Приходит шеф и спрашивает:
  - «А на сколько качественное ПО мы разрабатываем?»
- Ответ:
  - «Очень качественное!»
- Шеф:
  - «А докажи!»



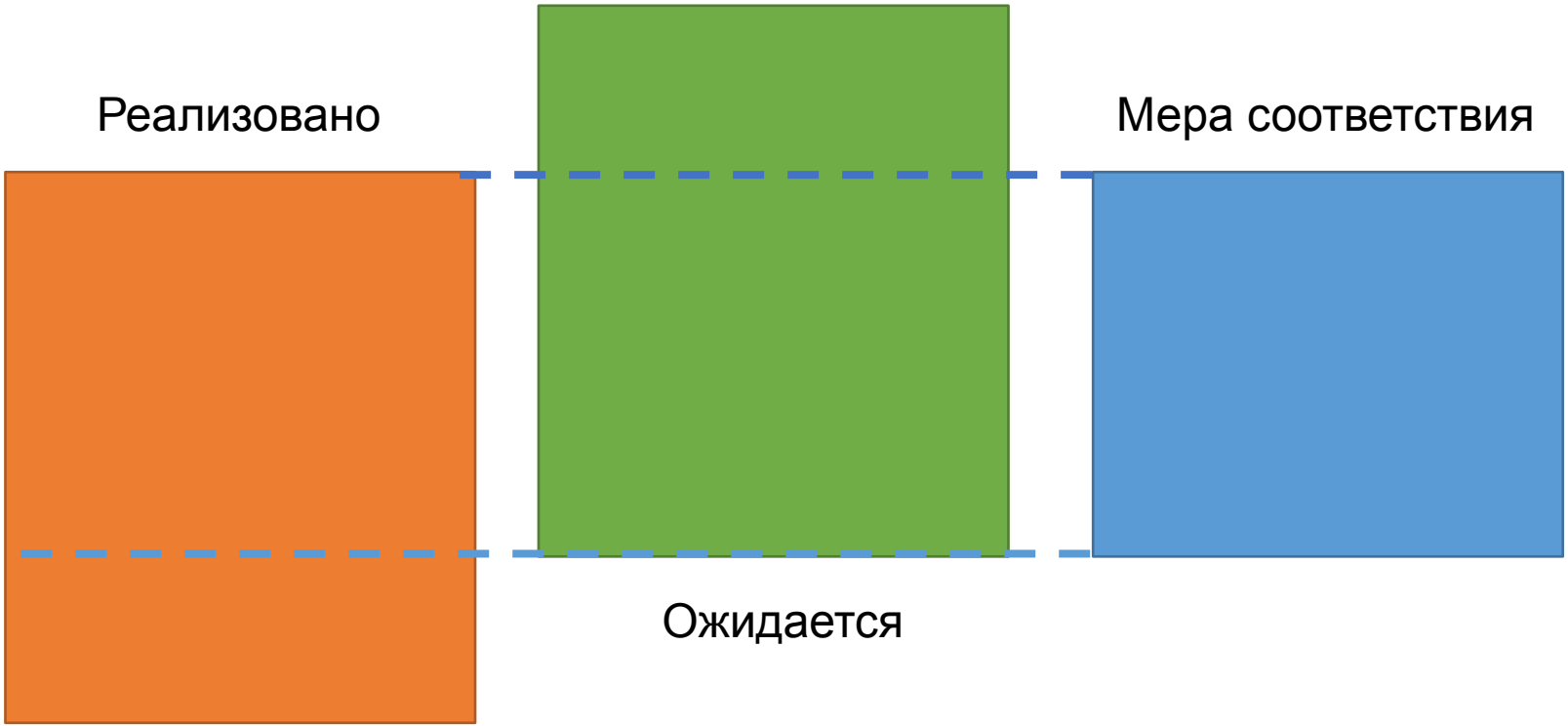
Цель доклада

- Поделиться практическим опытом использования инструментария для обеспечения и контроля качества, построенного на базе Tiobe TICS Framework
- Я расскажу про:
  - Метрики качества
  - Факторы качества
  - Их место в процессе разработке ПО
  - Приведу кейсы

А что такое качество?

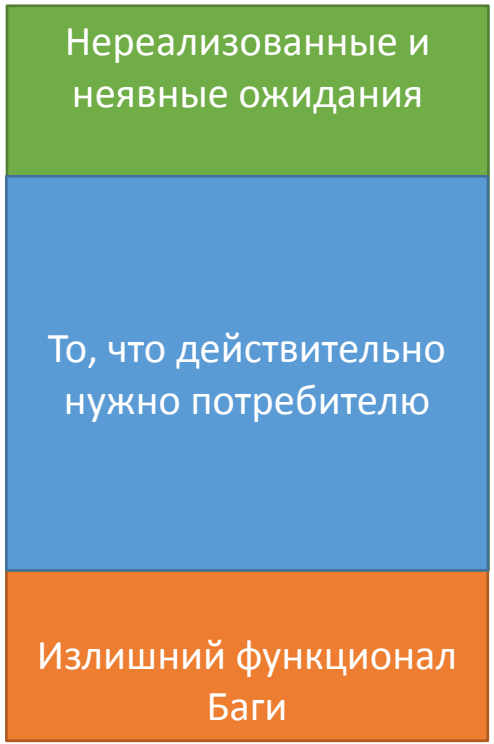
А что такое качество?

ISO 25010, ISO 9000...



А что такое качество?

ISO 25010, ISO 9000...

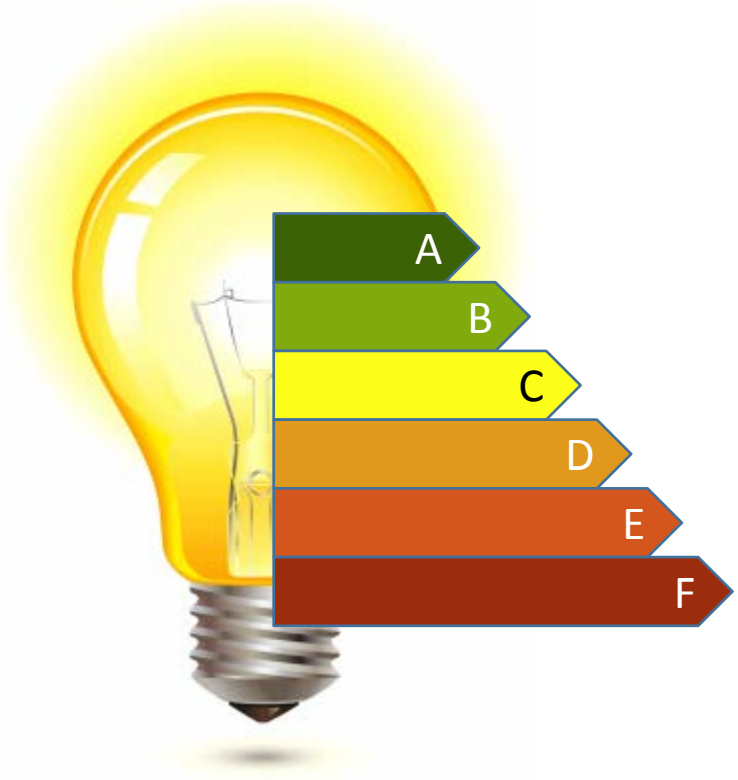


Характеристики качества

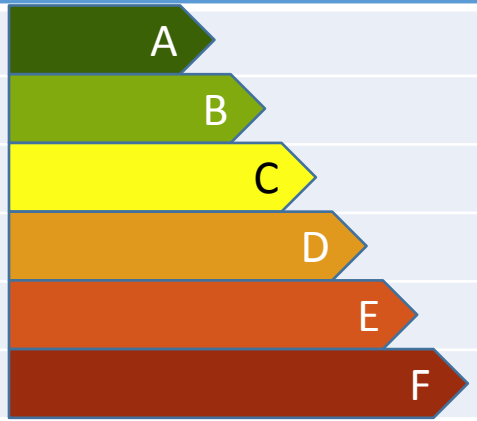
ISO 25010, ISO 9000...

- функциональная пригодность (functional suitability)
- уровень производительности (performance efficiency)
- совместимость (compatibility)
- удобство пользования (usability)
- надежность (reliability)
- защищенность (security)
- сопровождаемость (maintainability)
- переносимость (portability)

Идея



Оценка метрик

Категория	Описание	TQI счет
	Отлично	>= 90%
	Хорошо	>= 80%
	Неплохо	>= 70%
	Умеренно	>= 50%
	Слабо	>= 40%
	Недостаточно	< 40%

Метрики качества

1. покрытие кода тестами (code coverage)
2. ошибки абстрагирования (abstract interpretation)
3. цикломатическая сложность (cyclomatic complexity)
4. предупреждения компилятора (compiler warnings)
5. соответствие стандартам кодирования (coding standards)
6. дублирование кода (code duplication)
7. связность (fan out)
8. мертвый код (dead code)

Веса метрик

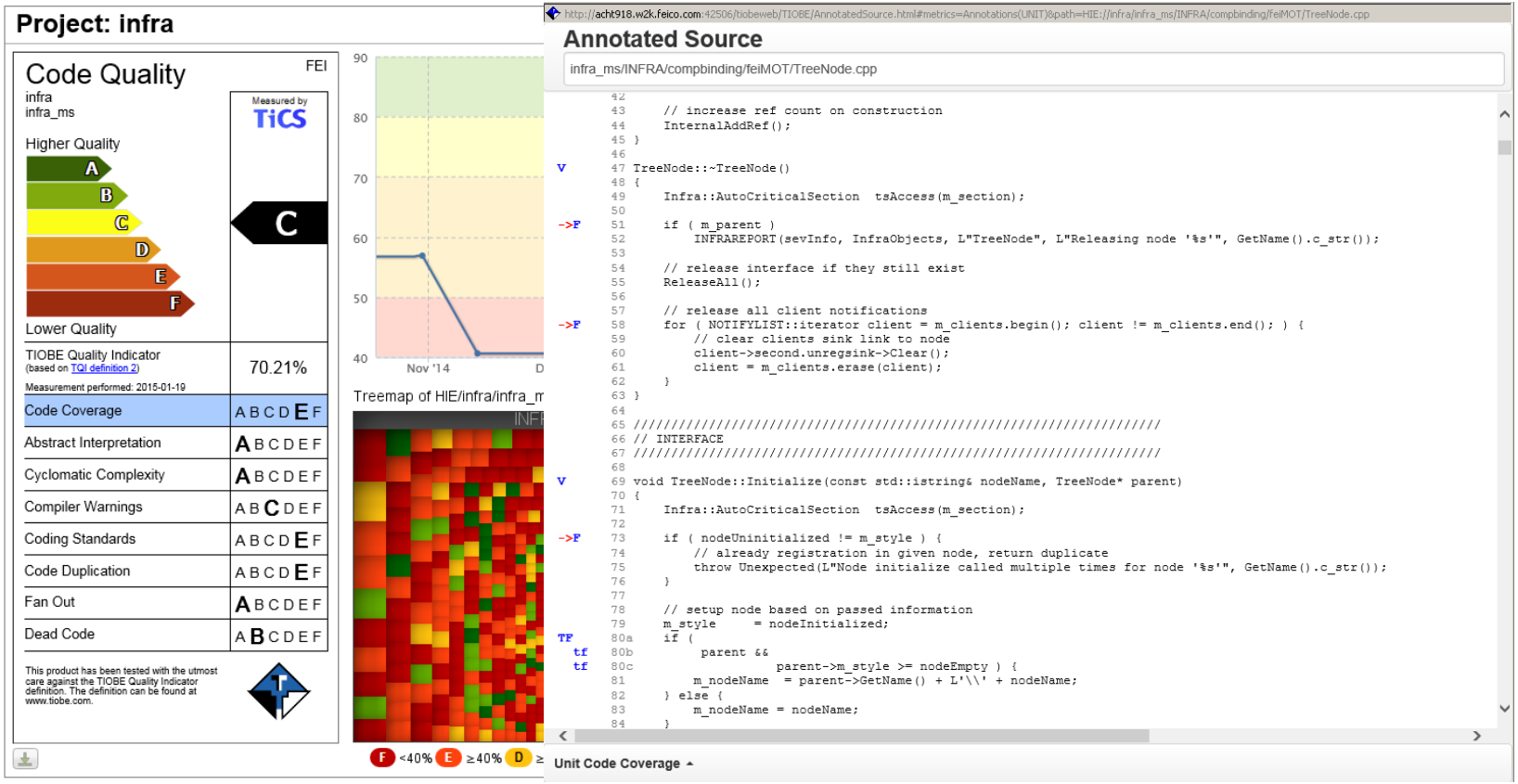
Метрика	Вес
Покрытие кода тестами	20%
Ошибки абстрагирования	20%
Цикломатическая сложность	15%
Предупреждения компилятора	15%
Соответствие стандартам кодирования	10%
Дублирование кода	10%
Связность кода	5%
Мертвый код	5%

1. Покрытие кода тестами

score = min(0.75 * test_coverage + 32.5, 100)

Категория	Критерий	TQI счет
A	>= 76.7%	>= 90%
B	>= 63.3%	>= 80%
C	>= 50%	>= 70%
D	>= 23.3%	>= 50%
E	>= 10%	>= 40%
F	< 10%	< 40%

Пример


Project: infra
Code Quality
FEI
infra
infra_ms
Measured by
TiCS
Higher Quality
A
B
C
D
E
F
C
Lower Quality
TIOBE Quality Indicator
(based on TQI definition 2)
70.21%
Measurement performed: 2015-01-19
Code Coverage A B C D E F
Abstract Interpretation A B C D E F
Cyclomatic Complexity A B C D E F
Compiler Warnings A B C D E F
Coding Standards A B C D E F
Code Duplication A B C D E F
Fan Out A B C D E F
Dead Code A B C D E F
This product has been tested with the utmost care against the TIOBE Quality Indicator definition. The definition can be found at www.tiobe.com.
90
80
70
60
50
40
Nov '14
Treemap of HIE/infra/infra_m
F <40% E ≥40% D ≥
http://acht918.w2k.feico.com:42506/tiobeweb/TIOBE/AnnotatedSource.html#metrics=Annotations(UNIT)&path=HIE://infra/infra_ms/INFRA/compbinding/feiMOT/TreeNode.cpp
Annotated Source
infra_ms/INFRA/compbinding/feiMOT/TreeNode.cpp
43 // increase ref count on construction
44 InternalAddRef();
45 }
47 TreeNode::~TreeNode()
48 {
49 Infra::AutoCriticalSection tsAccess(m_section);
51 if (m_parent)
52 INFRAREPORT(sevInfo, InfraObjects, L"TreeNode", L"Releasing node '%s'", GetName().c_str());
54 // release interface if they still exist
55 ReleaseAll();
57 // release all client notifications
58 for (NOTIFYLIST::iterator client = m_clients.begin(); client != m_clients.end();) {
59 // clear clients sink link to node
60 client->second.unregsink->Clear();
61 client = m_clients.erase(client);
62 }
63 }
65 //
66 // INTERFACE
67 //
69 void TreeNode::Initialize(const std::istring& nodeName, TreeNode* parent)
70 {
71 Infra::AutoCriticalSection tsAccess(m_section);
73 if (nodeUninitialized != m_style) {
74 // already registration in given node, return duplicate
75 throw Unexpected(L"Node initialize called multiple times for node '%s'", GetName().c_str());
76 }
78 // setup node based on passed information
79 m_style = nodeInitialized;
80a if (
80b parent &&
80c parent->m_style >= nodeEmpty) {
81 m_nodeName = parent->GetName() + L'\\' + nodeName;
82 } else {
83 m_nodeName = nodeName;
84 }
Unit Code Coverage

Для чего?

- Тесная связь с TDD и BDD
- Низкое значение (**F**) является следствием:
  - недостатка ресурсов
  - лени
  - непроработанной архитектуры

2. Ошибки абстрагирования

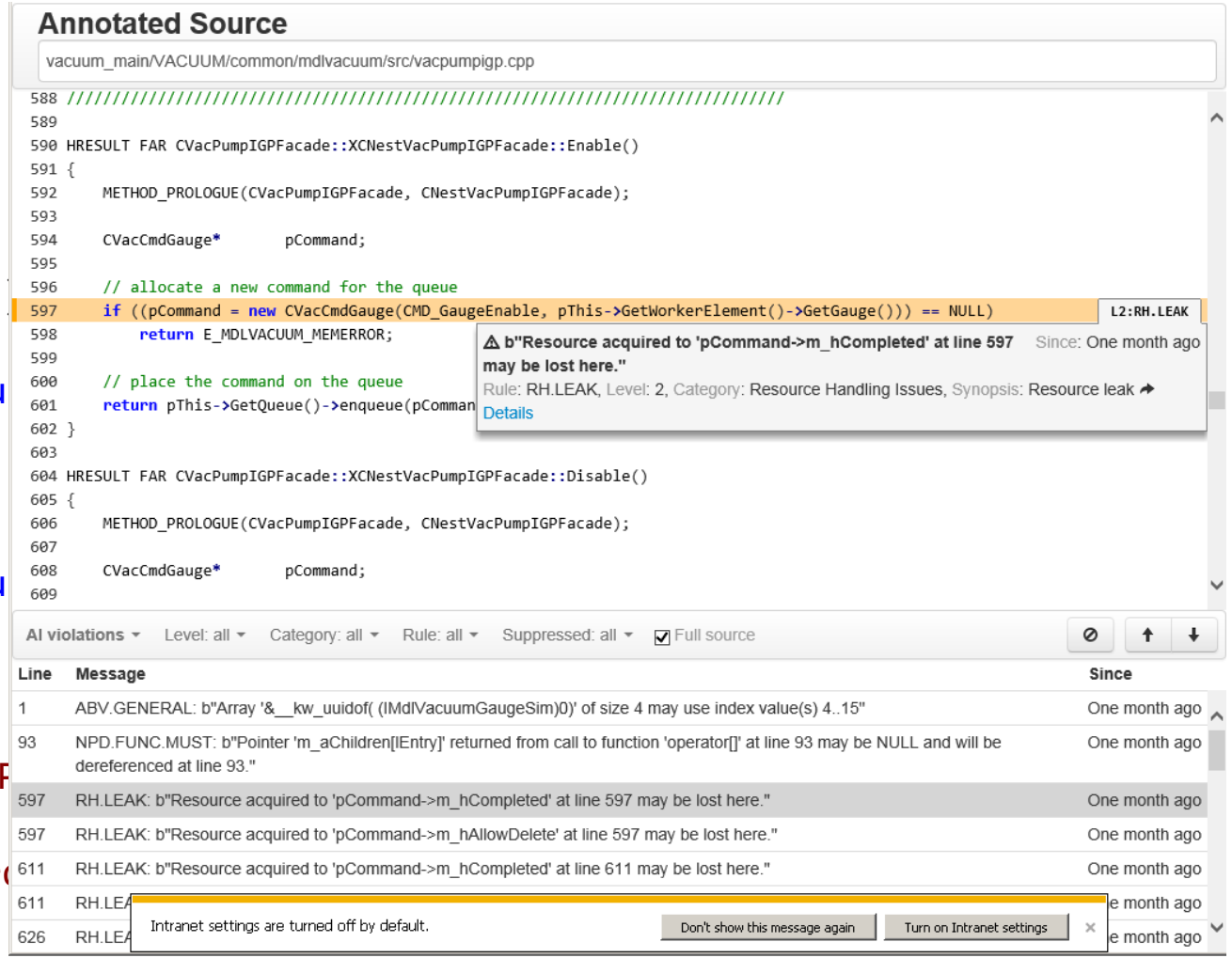
score = max(compliance_factor(abstract_interpretation_viloations)*2 – 100, 0)

Категория	Критерий	TQI счет
A	>= 95%	>= 90%
B	>= 90%	>= 80%
C	>= 85%	>= 70%
D	>= 75%	>= 50%
E	>= 70%	>= 40%
F	< 70%	< 40%

Пример

```
Order getOrder()
{
        if (isValid(
        {
                retu
        }
        else
        {
                retu
        }
}

List<Order> getOrderF
{
        return getOr
}
```



Для чего?

- Низкое значение (**F**) является следствием:
  - небрежности
  - использования опасных конструкций
  - обмане компилятора
  - наличия слишком старого кода

3. Цикломатическая сложност

score = min(max(140 – 20 * cyclomatic_complexity, 0), 100)

Категория	Критерий	TQI счет
A	<= 2.5	>= 90%
B	<= 3	>= 80%
C	<= 3.5	>= 70%
D	<= 4.5	>= 50%
E	<= 5	>= 40%
F	> 5	< 40%

Пример

```
int getValue(int param)
{
        int value = 0;
        if (param == 0)
        {
                value = 4;
        }
        else
        {
                value = 0;
        }
        return value;
}
```

Для чего?

- Низкое значение (**F**) является следствием:
  - наличия огромных GOD объектов
  - проблем с атомизацией функциональности
  - проблем со специальными случаями и требованиями

4. Предупреждения компилятора

score = max(100 – 50 * log10(101 – compliance_factor(compiler_warnings)), 0)

Категория	Критерий	TQI счет
A	>= 99.42%	>= 90%
B	>= 98.49%	>= 80%
C	>= 97.02%	>= 70%
D	>= 91.00%	>= 50%
E	>= 83.22%	>= 40%
F	< 83.22%	< 40%

Пример

```
int func(int i)
{
        if (i = 0)
        {
                return -1;
        }
}
```

Для чего?

- Аналогично ошибкам абстрагирования
- Низкое значение (**F**) является следствием:
  - небрежности
  - использования опасных конструкций
  - попыток обмана компилятора
  - низкой квалификации

5. Соответствие стандартам кодирования

score = compliance_factor(coding_standard_violations)

Категория	Критерий	TQI счет
A	>= 90%	>= 90%
B	>= 80%	>= 80%
C	>= 70%	>= 70%
D	>= 50%	>= 50%
E	>= 40%	>= 40%
F	< 40%	< 40%

Пример

```
int abs(int i)
{
        int result;

        if (i < 0)
        {
                result = -i;
                goto end;
        }
        result = i;
end:
        return result;
}
```

Для чего?

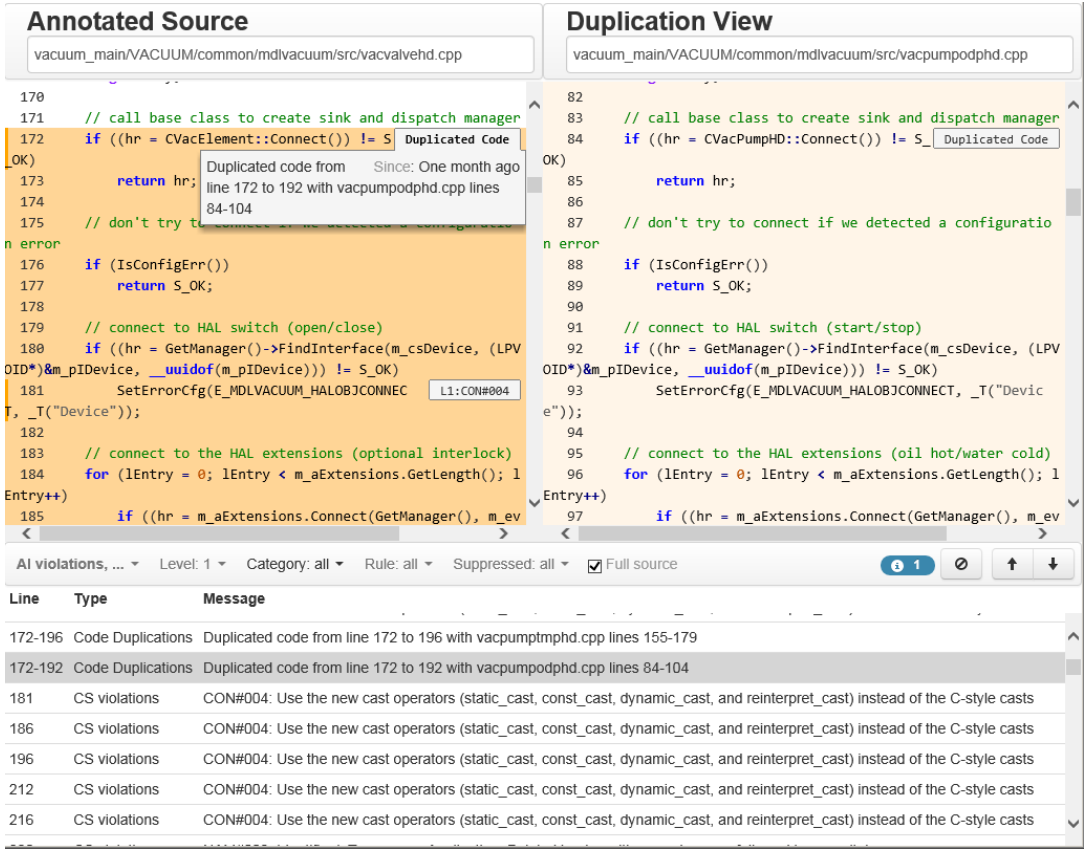
- Аналогично ошибкам абстрагирования и предупреждениям компилятора
- Программисты не смогли договориться друг с другом о единых правилах написания кода

6. Дублирование кода

score = min(-20 * log10(code_duplication) + 60, 100)

Категория	Критерий	TQI счет
A	<= 0.03%	>= 90%
B	<= 0.10%	>= 80%
C	<= 0.32%	>= 70%
D	<= 3.16%	>= 50%
E	<= 10.00%	>= 40%
F	> 10.00%	< 40%

Пример


Annotated Source
vacuum_main/VACUUM/common/mdlvacuum/src/vacvalvehd.cpp
Duplication View
vacuum_main/VACUUM/common/mdlvacuum/src/vacpumpodphd.cpp
Duplicated Code
Duplicated code from line 172 to 192 with vacpumpodphd.cpp lines 84-104
Since: One month ago
L1:CON#004
AI violations, ...
Level: 1
Category: all
Rule: all
Suppressed: all
Full source
Line
Type
Message
172-196 Code Duplications Duplicated code from line 172 to 196 with vacpumptmphd.cpp lines 155-179
172-192 Code Duplications Duplicated code from line 172 to 192 with vacpumpodphd.cpp lines 84-104
181 CS violations CON#004: Use the new cast operators (static_cast, const_cast, dynamic_cast, and reinterpret_cast) instead of the C-style casts
186 CS violations CON#004: Use the new cast operators (static_cast, const_cast, dynamic_cast, and reinterpret_cast) instead of the C-style casts
196 CS violations CON#004: Use the new cast operators (static_cast, const_cast, dynamic_cast, and reinterpret_cast) instead of the C-style casts
212 CS violations CON#004: Use the new cast operators (static_cast, const_cast, dynamic_cast, and reinterpret_cast) instead of the C-style casts
216 CS violations CON#004: Use the new cast operators (static_cast, const_cast, dynamic_cast, and reinterpret_cast) instead of the C-style casts

Для чего?

- Используется метод «copy-paste»
- На пишется новый код, а дублируется старый
- Указывает на будущие проблемы с поддержкой кода

7. Связность

score = min(max(120 – 5 * fan_out, 0), 100)

Категория	Критерий	TQI счет
A	<= 6	>= 90%
B	<= 8	>= 80%
C	<= 10	>= 70%
D	<= 14	>= 50%
E	<= 16	>= 40%
F	> 16	< 40%

Пример

Для чего?

- Наличие GOD-объектов
- Проблемы с архитектурой
- Указывает на будущие проблемы с поддержкой кода

8. Мертвый код

score = max((100 – 2 *dead_code), 0)

Категория	Критерий	TQI счет
A	<= 5%	>= 90%
B	<= 10%	>= 80%
C	<= 15%	>= 70%
D	<= 25%	>= 50%
E	<= 30%	>= 40%
F	> 30%	< 40%

Пример

```
int calculateindex(unsigned int i)
{
        if (i < 0)
        {
                return 0;
        }
        else
        {
                return i;
        }
}
```

Для чего?

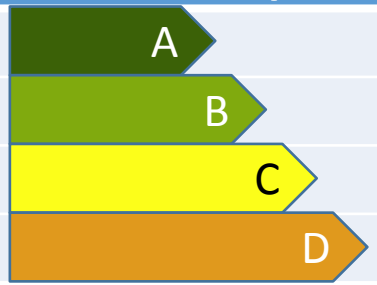
- Проблемы с архитектурой
- Указывает на проблемы с поддержкой кода
- Часто разбирается сложный код, который ничего не делает

Для чего использовать данные метрики?

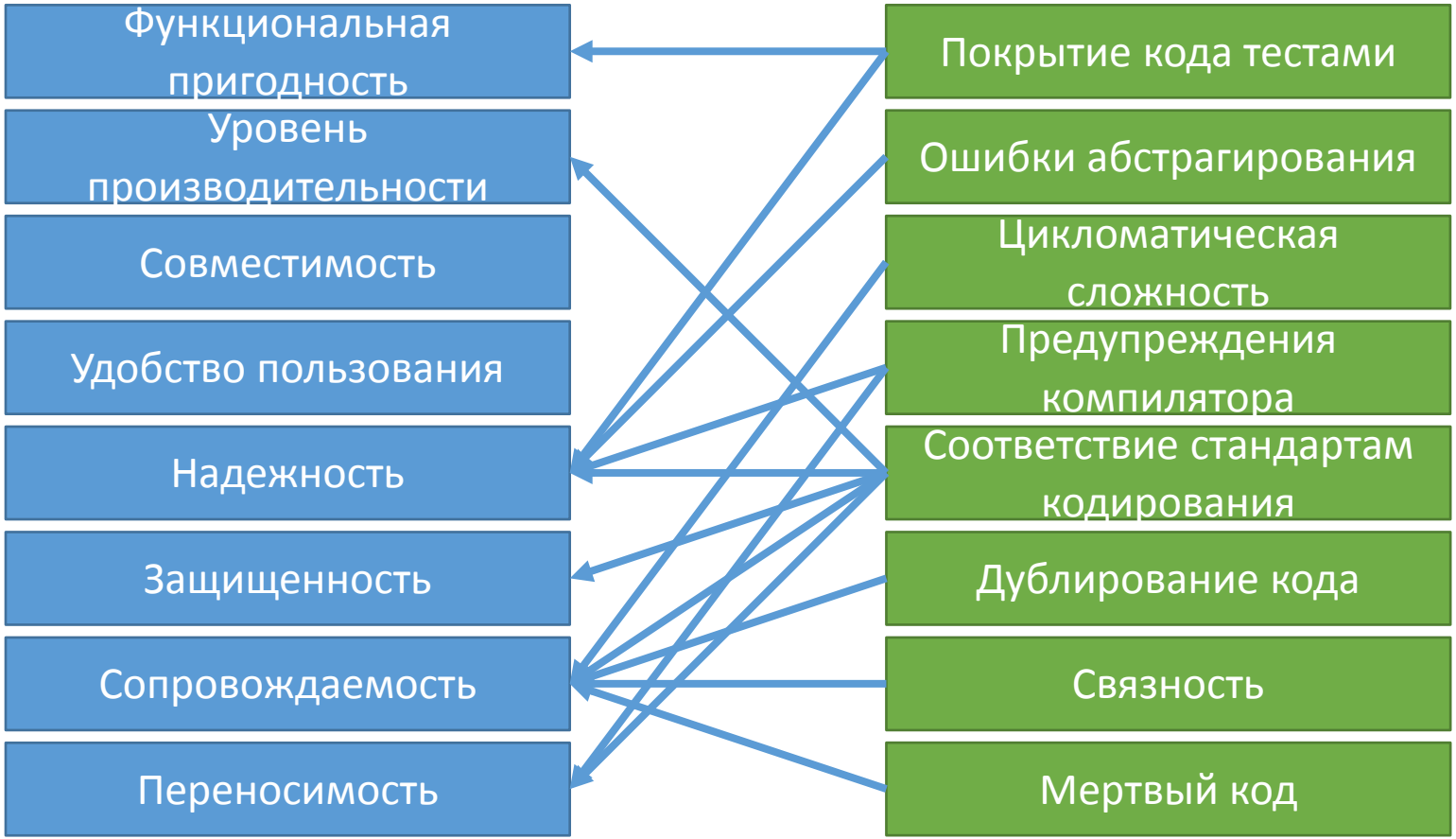
- Для принятия решений (например, при рефакторинге)
- При оценке трудоемкости
- При оценке качества кода по косвенным признакам
- Как элемент KPI
- Для анализа трендов менеджерами, архитекторами, разработчиками

Дополнительные замечания

Рекомендации по отраслям

Категория	Сфера
	Авиация, ВПК
	Космос, медицина, Автомобили
	Полупроводники, биоинформатика
	Бизнес-решения

Связь метрик и характеристик качества



Выводы

- Данный набор метрик (TQI, Tiobe Quality Induicatior) является довольно прагматичным способом получить представление о качестве написанного кода до его релиза и даже до этапа тестирования.
- Данный набор комбинирует наиболее известные метрики качества кода, через определение метрик второго уровня, через то, как они измерены и то, как их следует анализировать при оценке качества кода.
- В результате мы можем дать оценку этому коду по шести бальной системе. И сказать об этом шефу, указав на проблемные места.

Спасибо за внимание

Пример как посчитать

- На 20% сокращается количество ошибок, достигших потребителя
- Считаем затраты на поиск исправление багов традиционным способом + репутационные потери
- Считаем затраты на использование системы

Compliance factor

compliance factor =

процент проверенного года * вес предупреждений

плотность предупреждений определенного типа =

количество предупреждений определенного типа / объем кода

вес предупреждений =

плотность / (4 ^ уровень предупреждений)

Jansen, Paul; Krikhaar, Rene; Dijkstra, Fons, “Towards a Single Software Quality Metric – The Static Confidence Factor
http://www.tiobe.com/content/paperinfo/DefinitionOfConfidenceFactor.html

Поддерживаемые языки

- Ada, C, C++, C#, COBOL, Java, JavaScript, MATLAB, Objective-C, PL/SQL, Python, Scala, VB.NET, XAML

Аналогичные фреймворки

- CodEnforcer
- Cast SoftWare
- McCabe
- RSM

Используемые утилиты (на примере C+++)

Источник http://www.tiobe.com/index.php/content/TICS/FactSheet.html

Метрика	Утилиты
Code Coverage	BullseyeCoverage (Bullseye), Squish Coco (FrogLogic), CTC++ (Testwell), gcov/lcov (SourceForge), PureCoverage (IBM), C++Test (Parasoft), VectorCAST (Vector Software)
Abstract Interpretation	Coverity (Coverity), C++Test/BugDetective (Parasoft), Klocwork (Klocwork), CodeSonar (GrammaTech)
Compiler Warnings	Visual Studio (Microsoft), GCC (GNU project), Keil (Keil), Tasking (Tasking), MULTI (Green Hills Software), CodeComposer (Texas Instruments)
Cyclomatic Complexity	TICSpp (Part of TIOBE's TICS framework)
Coding Standards	TICSc/TICSpp (Part of TIOBE's TICS framework), C++Test (Parasoft), PC-Lint (Gimpel), QA-C++ (Programming Research), cpplint (Google)
Code Duplication	CPD with TIOBE tokenizer (PMD project)
Fan Out	TICS (Part of TIOBE's TICS framework)
Dead Code	TICS (Part of TIOBE's TICS framework), Cppcheck (SourceForge project)

Источники

- DeMarco Tom. Controlling Software Projects: Management, Measurement, and Estimates. Prentice Hall, ISBN 0131717111, 1986.
- Chidamber Shyam R. A Metrics Suite for Object Oriented Design. / F. – New Jersey, Prentice-Hall, Inc, 1994.
- DeMarco Tom. Software Engineering: An Idea Whose Time Has Come and Gone? IEEE Software (ISSN 0740-7459). July/August 2009. P.95-96
- Pressman, Roger S. Ph.D (2001). Software Engineering - A Practitioner's Approach - Fifth Edition. ISBN 0-07-365578-3, 2001.
- ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models
- The TIOBE Quality Indicator a pragmatic way of measuring code quality // http://www.tiobe.com/content/paperinfo/TIOBEQualityIndicator.pdf
- Jansen, Paul; Krikhaar, Rene; Dijkstra, Fons, “Towards a Single Software Quality Metric – The Static Confidence Factor // http://www.tiobe.com/content/paperinfo/DefinitionOfConfidenceFactor.html
- Software source quality and source code analysis approaches realized in www.codenforcer.com // http://www.codenforcer.com/pdf/codEnforcer_book.pdf