

Одиннадцатая независимая научно-практическая конференция «Разработка ПО 2015»

22 - 24 октября, Москва



Опыт работы с метриками для обеспечения качества ПО

Колесников Александр

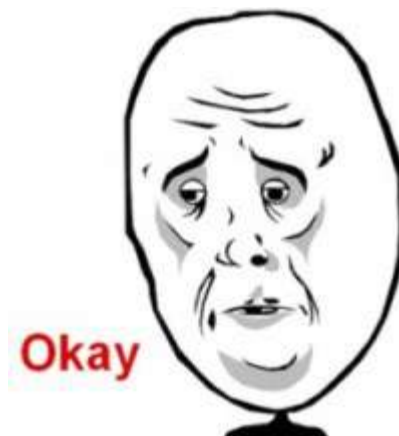


Допущения

- 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

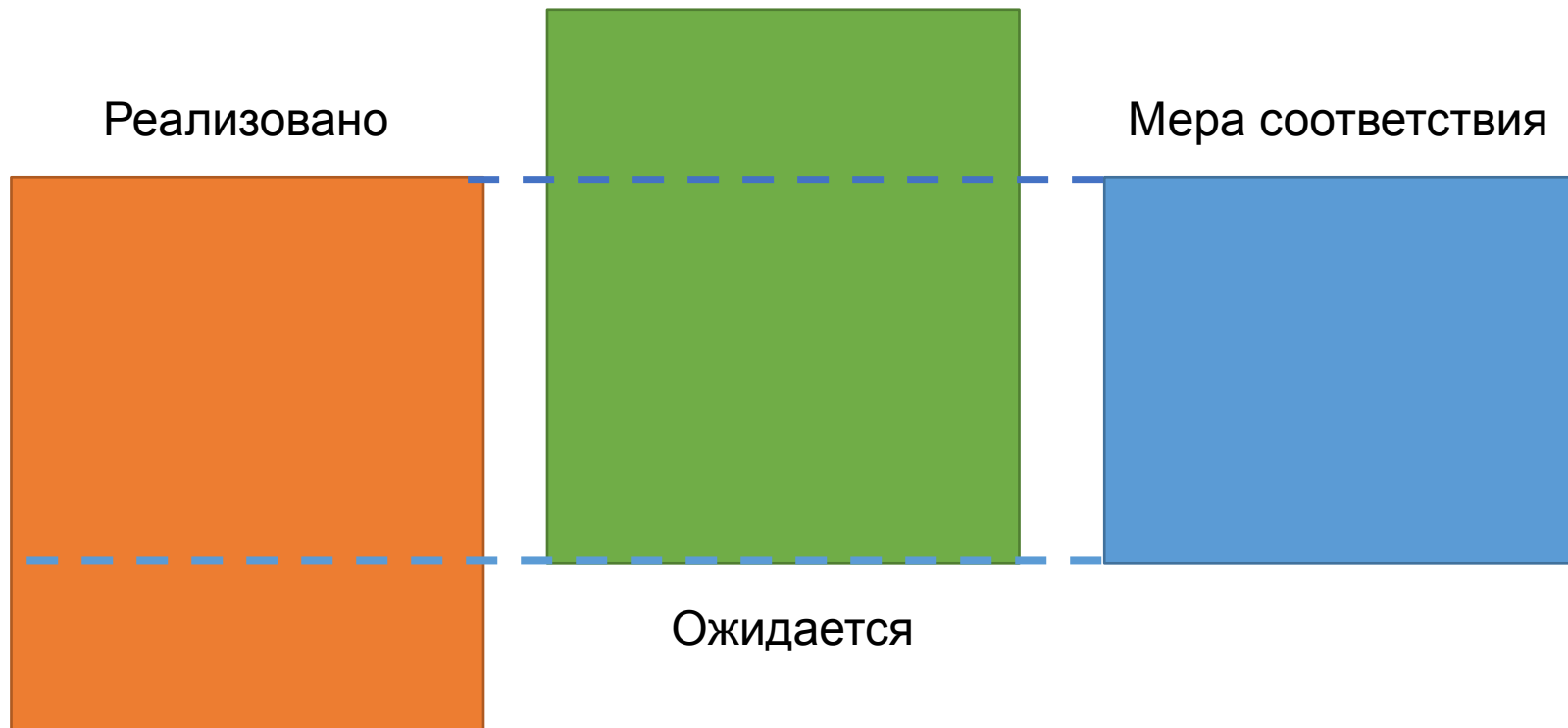
- Поделиться практическим опытом использования инструментария для обеспечения и контроля качества, построенного на базе Tiobe TICS Framework
- Я расскажу про:
 - Метрики качества
 - Факторы качества
 - Их место в процессе разработке ПО
 - Приведу кейсы

А что такое качество?



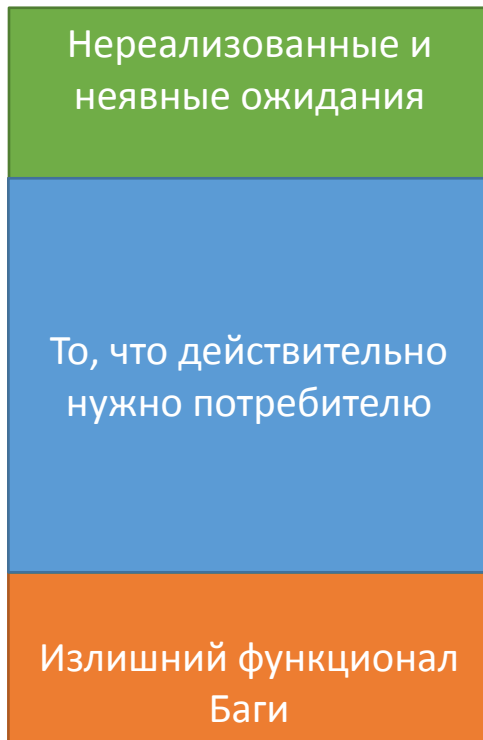
А что такое качество?

ISO 25010, ISO 9000...



А что такое качество?

ISO 25010, ISO 9000...

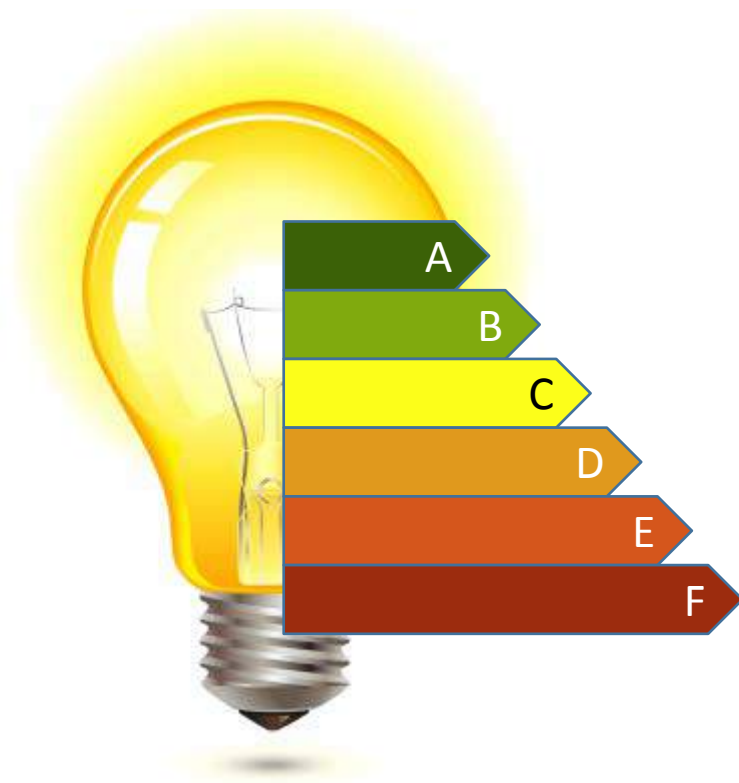


Характеристики качества

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
 A	Отлично	$\geq 90\%$
 B	Хорошо	$\geq 80\%$
 C	Неплохо	$\geq 70\%$
 D	Умеренно	$\geq 50\%$
 E	Слабо	$\geq 40\%$
 F	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

$$\text{score} = \min(0.75 * \text{test_coverage} + 32.5, 100)$$

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

Пример

Project: infra

Code Quality

FEI

Measured by **TiCS**

Higher Quality

A B C D E F

Lower Quality

70.21%

TIOBE Quality Indicator (based on TQI definition 2)	70.21%
Measurement performed: 2015-01-10	
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.

Annotated Source

infra_ms/INFRA/combinding/feiMOT/TreeNode.cpp

```
42
43 // increase ref count on construction
44 InternalAddRef();
45 )
46
V 47 TreeNode::~TreeNode()
48 {
49     Infra::AutoCriticalSection tsAccess(m_section);
50
->F 51 if ( m_parent )
52     INFRAREPORT(sevInfo, InfraObjects, L"TreeNode", L"Releasing node '%s'", GetName().c_str());
53
54 // release interface if they still exist
55 ReleaseAll();
56
->F 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 }
64
65 ////////////////////////////////////////
66 // INTERFACE
67 ////////////////////////////////////////
68
V 69 void TreeNode::Initialize(const std::wstring& nodeName, TreeNode* parent)
70 {
71     Infra::AutoCriticalSection tsAccess(m_section);
72
->F 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 }
77
78 // setup node based on passed information
79 m_style = nodeInitialized;
80a if (
tf 80b     parent &&
80c     parent->m_style >= nodeEmpty ) {
81     m_nodeName = parent->GetName() + L'\\' + nodeName;
82 } else {
83     m_nodeName = nodeName;
84 }
```

Treemap of HIE/infra/infra_r

F <40% E ≥40% D ≥

Unit Code Coverage -

Для чего?

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

2. Ошибки абстрагирования

$\text{score} = \max(\text{compliance_factor}(\text{abstract_interpretation_violations}) * 2 - 100, 0)$

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

Пример

```
Order getOrder()
```

```
{
```

```
    if (isValid()
```

```
    {
```

```
        return
```

```
    }
```

```
    else
```

```
    {
```

```
        return
```

```
    }
```

```
}
```

```
List<Order> getOrderF
```

```
{
```

```
    return getOrder
```

```
}
```

Annotated Source

vacuum_main/VACUUM/common/mdlvacuum/src/vacpumpigp.cpp

```
588 ///////////////////////////////////////////////////////////////////
589
590 HRESULT FAR CVacPumpIGPFacade::XCNestVacPumpIGPFacade::Enable()
591 {
592     METHOD_PROLOGUE(CVacPumpIGPFacade, CNestVacPumpIGPFacade);
593
594     CVacCmdGauge*    pCommand;
595
596     // allocate a new command for the queue
597     if ((pCommand = new CVacCmdGauge(CMD_GaugeEnable, pThis->GetWorkerElement()->GetGauge())) == NULL)
598         return E_MDLVACUUM_MEMERROR;
599
600     // place the command on the queue
601     return pThis->GetQueue()->enqueue(pCommand);
602 }
603
604 HRESULT FAR CVacPumpIGPFacade::XCNestVacPumpIGPFacade::Disable()
605 {
606     METHOD_PROLOGUE(CVacPumpIGPFacade, CNestVacPumpIGPFacade);
607
608     CVacCmdGauge*    pCommand;
609
```

L2:RH.LEAK

▲ b"Resource acquired to 'pCommand->m_hCompleted' at line 597 may be lost here." Since: One month ago

Rule: RH.LEAK, Level: 2, Category: Resource Handling Issues, Synopsis: Resource leak →
[Details](#)

All violations Level: all Category: all Rule: all Suppressed: all Full source

Line	Message	Since
1	ABV.GENERAL: b"Array '&_kw_uuidof((IMdlVacuumGaugeSim)0)' of size 4 may use index value(s) 4..15"	One month ago
93	NPD.FUNC.MUST: b"Pointer 'm_aChildren[Entry]' returned from call to function 'operator[]' at line 93 may be NULL and will be dereferenced at line 93."	One month ago
597	RH.LEAK: b"Resource acquired to 'pCommand->m_hCompleted' at line 597 may be lost here."	One month ago
597	RH.LEAK: b"Resource acquired to 'pCommand->m_hAllowDelete' at line 597 may be lost here."	One month ago
611	RH.LEAK: b"Resource acquired to 'pCommand->m_hCompleted' at line 611 may be lost here."	One month ago
611	RH.LEAK	One month ago
626	RH.LEAK	One month ago

Intranet settings are turned off by default.

Don't show this message again

Turn on Intranet settings

×

Для чего?

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

3. Цикломатическая сложность

$$\text{score} = \min(\max(140 - 20 * \text{cyclomatic_complexity}, 0), 100)$$

Категория	Критерий	TQI счет
A	≤ 2.5	$\geq 90\%$
B	≤ 3	$\geq 80\%$
C	≤ 3.5	$\geq 70\%$
D	≤ 4.5	$\geq 50\%$
E	≤ 5	$\geq 40\%$
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

$\text{score} = \max(100 - 50 * \log_{10}(101 - \text{compliance_factor}(\text{compiler_warnings})), 0)$

Категория	Критерий	TQI счет
A	$\geq 99.42\%$	$\geq 90\%$
B	$\geq 98.49\%$	$\geq 80\%$
C	$\geq 97.02\%$	$\geq 70\%$
D	$\geq 91.00\%$	$\geq 50\%$
E	$\geq 83.22\%$	$\geq 40\%$
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	$\geq 90\%$	$\geq 90\%$
B	$\geq 80\%$	$\geq 80\%$
C	$\geq 70\%$	$\geq 70\%$
D	$\geq 50\%$	$\geq 50\%$
E	$\geq 40\%$	$\geq 40\%$
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

$$\text{score} = \min(-20 * \log_{10}(\text{code_duplication}) + 60, 100)$$

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

Пример

Annotated Source

vacuum_main/VACUUM/common/mdlvacuum/src/vacvalvehd.cpp

```
170
171 // call base class to create sink and dispatch manager
172 if ((hr = CVacElement::Connect()) != S_OK) Duplicated Code
173     return hr;
174
175 // don't try to connect if we detected a configuration error
176 if (IsConfigErr())
177     return S_OK;
178
179 // connect to HAL switch (open/close)
180 if ((hr = GetManager()->FindInterface(m_csDevice, (LPV
181     SetErrorCfg(E_MDLVACUUM_HALOBJCONNECT, L1:CON#004
182     T, _T("Device"));
183
184 // connect to the HAL extensions (optional interlock)
185 for (lEntry = 0; lEntry < m_aExtensions.GetLength(); l
186     Entry++)
187     if ((hr = m_aExtensions.Connect(GetManager(), m_ev
```

82
83 // call base class to create sink and dispatch manager
84 if ((hr = CVacPumpHD::Connect()) != S_OK) Duplicated Code
85 return hr;
86
87 // don't try to connect if we detected a configuration
88 if (IsConfigErr())
89 return S_OK;
90
91 // connect to HAL switch (start/stop)
92 if ((hr = GetManager()->FindInterface(m_csDevice, (LPV
93 SetErrorCfg(E_MDLVACUUM_HALOBJCONNECT, _T("Devic
94 e"));
95
96 // connect to the HAL extensions (oil hot/water cold)
97 for (lEntry = 0; lEntry < m_aExtensions.GetLength(); l
98 Entry++)
99 if ((hr = m_aExtensions.Connect(GetManager(), m_ev

Duplication View

vacuum_main/VACUUM/common/mdlvacuum/src/vacpumpdphd.cpp

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 vacpumpdphd.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

$$\text{score} = \min(\max(120 - 5 * \text{fan_out}, 0), 100)$$

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

Пример

Для чего?

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

8. МЕРТВЫЙ КОД

$$\text{score} = \max((100 - 2 * \text{dead_code}), 0)$$

Категория	Критерий	TQI счет
A	$\leq 5\%$	$\geq 90\%$
B	$\leq 10\%$	$\geq 80\%$
C	$\leq 15\%$	$\geq 70\%$
D	$\leq 25\%$	$\geq 50\%$
E	$\leq 30\%$	$\geq 40\%$
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?

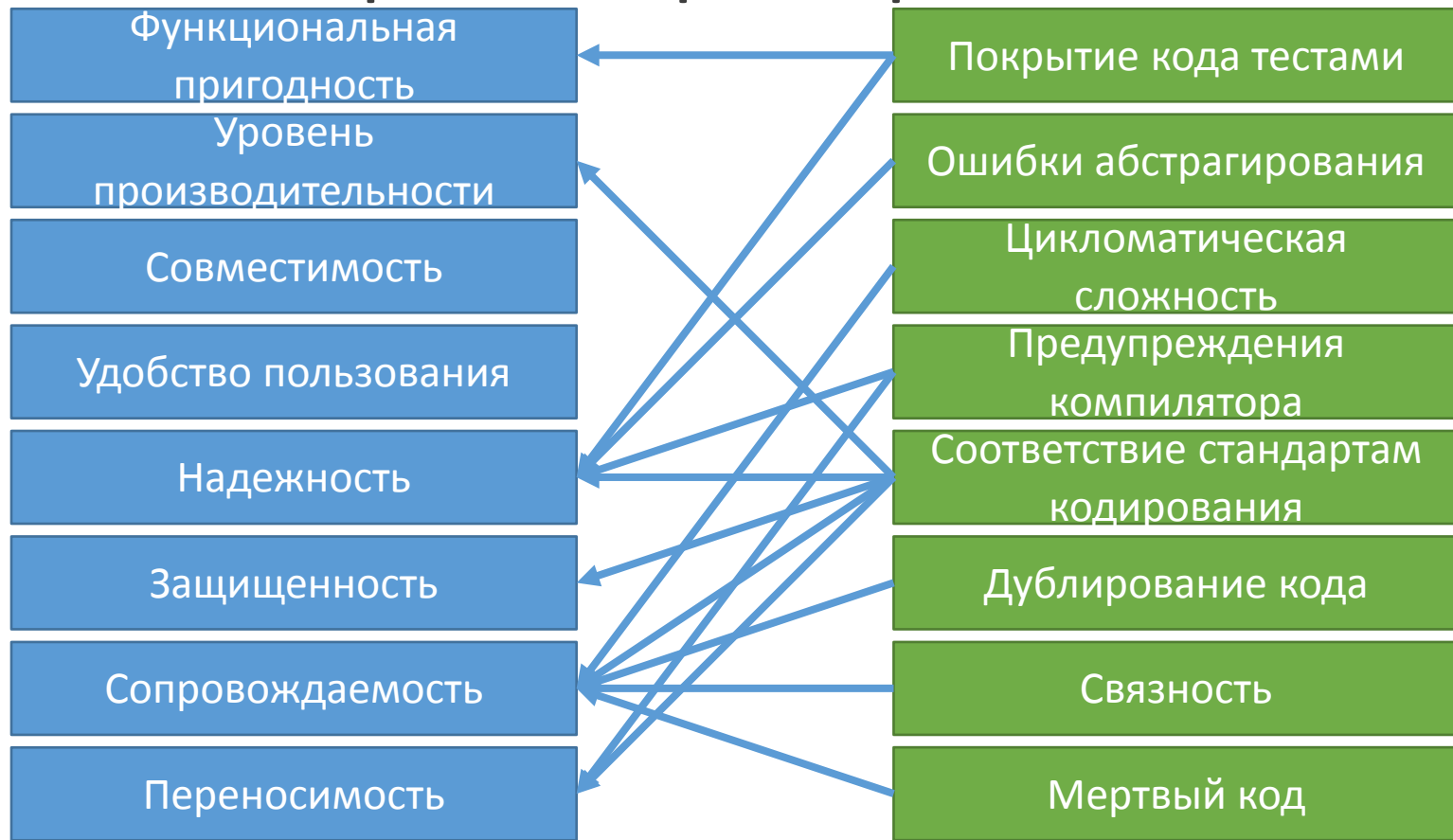
- Для принятия решений (например, при рефакторинге)
- При оценке трудоемкости
- При оценке качества кода по косвенным признакам
- Как элемент KPI
- Для анализа трендов менеджерами, архитекторами, разработчиками

Дополнительные замечания

Рекомендации по отраслям

Категория	Сфера
A	Авиация, ВПК
B	Космос, медицина, Автомобили
C	Полупроводники, биоинформатика
D	Бизнес-решения

Связь метрик и характеристик качества



Выводы

- Данный набор метрик (TQI, Tiobe Quality Indicator) 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)

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 (GrammarTech)
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	TICSsc/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