



# КАЧЕСТВО КОД. АНАЛИЗ КОДА С NDEPEND

# План

- Качество
- Какие способы достичь качества
- Качество кода в аспекте проектирования
- Принципы ООП/ООД
- Метрики кода
- NDepend

# Зачем качество?



# Холиворная тема

데쓰스타의  
소스콘드를 공개하라!



KSSH~ What?



# “Железный треугольник”

Железный треугольник, или треугольник менеджмента. Его смысл в том, что ограничения на объём работ, сроки и бюджет должны быть разумными и нужно ими управлять (балансировать)



Где же  
качество?

# “Железный треугольник”



Качественное ПО получается в результате баланса между объемом работ, сроками и бюджетом

<http://www.intuit.ru/department/se/msd/4/3.html>

# К чему эти треугольники?



ILLUSTRATION BY BRYAN LEISTER

# QUALITY CONTROL

# Ценности качественного кода

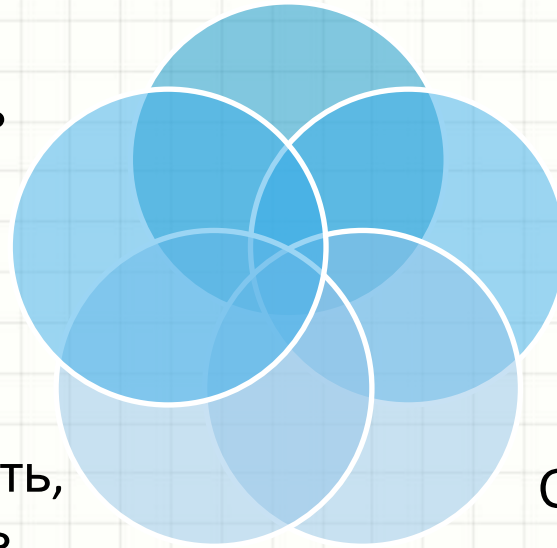
Расширяемость,  
гибкость (extensibility,  
agility)

Тестируемость  
(testability)

Простота  
(simplicity)

Читабельность,  
понятность  
(readability, clarity)

Сопровождаемость  
(maintainability)





# Какие есть способы?

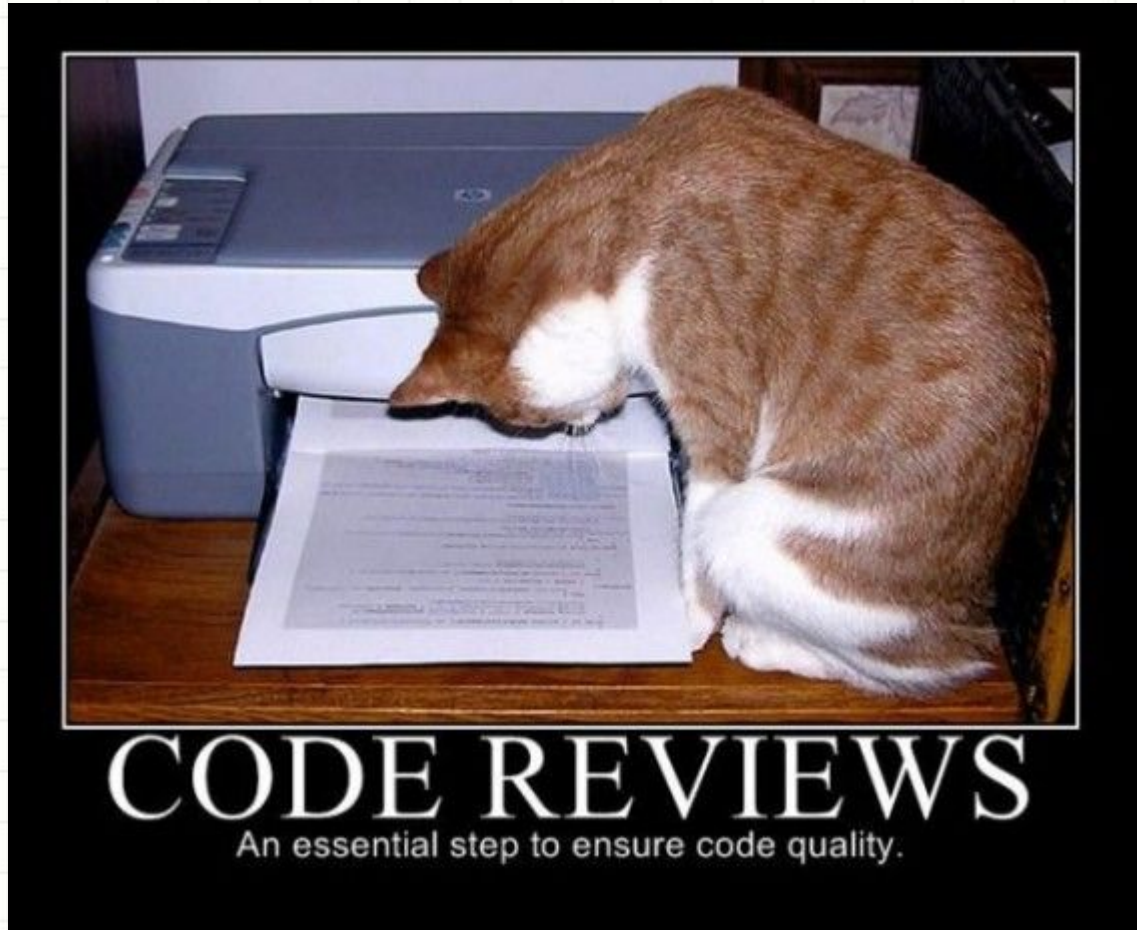
- **Организационные**
  - XP (eXtreme Programming)
  - Code review
  - Project management, methodology,
  - Utilities: StyleCop, FxCop, Code Analysis
  - Requirements...
- **Технические**
  - Юнит-тесты
  - TDD, Defensive programming style
  - OOP/OOD, principles
- **Обучение**



# Какие есть способы?

- **Внешние** – программистские практики
  - Парное программирование
  - Статический анализ кода
  - Code review ,
  - Unit-tests, TDD/BDD
- **Внутренние** - правильное проектирование и рефакторинг кода как способ превращения плохого кода в более хороший
  - OOP/OOD, principles,
  - Programming style

# Code Review



# Статический анализ кода

- StyleCop, FxCop, Code Analysis, Ndepend
- Цель автоматизировать review кода и обратить внимание на распространённые ошибки и скользкие участки.
- В идеале внедрить стат. анализ в CI (build process)



# Методологии и управление

- Управление проектом напрямую влияет на результаты и удовлетворенность от работы
  - **Хаотическое управление = низкое качество** (e.g. Cowboy coding)



# Extreme programming

- Парное программирование
- Всесторонний code review
- Юнит-тесты на весь код (TDD ,
- YAGNI, не пишем того, что не нужно
- Изменчивые требования
- Частая коммуникация с заказчиком и в команде



# Обучение

- Никакие утилиты стат. анализа не заменят людей
- Позволяет писать качественные код
- Повышает коммуникации
- Улучшает команду
- Парное программирование способствует



Обучение

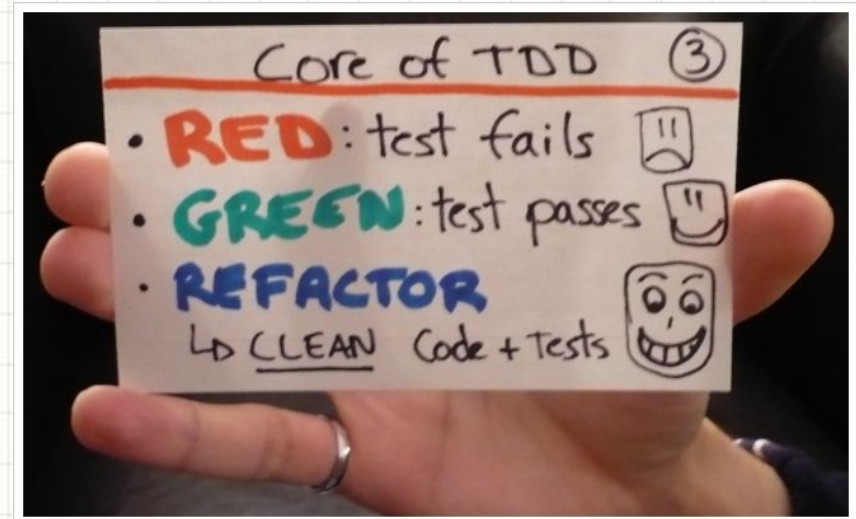
Профессиональная команда

Качественный код

# Юнит-тесты, TDD

- Юнит-тесты – позволяют контролировать соответствие кода задуманному поведению.
- TDD – подход к написанию кода начиная с тестов.

«Тесты вперед»





# Рефакторинг



GORDON FREEMAN

# Защитное программирование

## Defensive programming

- Использование ассертов (asserts)
- Использование контрактов кода (code contracts)

Ассерты или контракты как мини юнит-тесты если что то идет не так



# Технический долг

К его появлению приводит быстрая и бездумная разработка

Когда мы понимаем, что можем написать дизайн лучше, но в силу причин не делаем этого – мы откладываем долг.

Выплачивать придется рефакторингом

# Ценности, принципы и паттерны



# Принципы ООП/ООД

- Принципы SOLID
- Принципы GRASP
- KISS = Keep it simple
- DRY = Don't repeat yourself
- YAGNI = You ain't gonna need it

## OO Principles

Encapsulate what varies.

Favor composition over inheritance.

Program to interfaces, not implementations.

Strive for loosely coupled designs between objects that interact.

Classes should be open for extension but closed for modification.

Depend on abstractions. Do not depend on concrete classes.

Only talk to your friends.

Don't call us, we'll call you.

A class should have only one reason to change.

## OO Basics

Abstraction

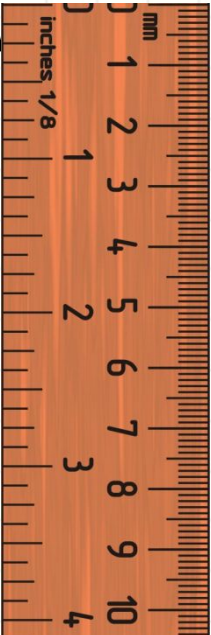
Encapsulation

Polymorphism

Inheritance

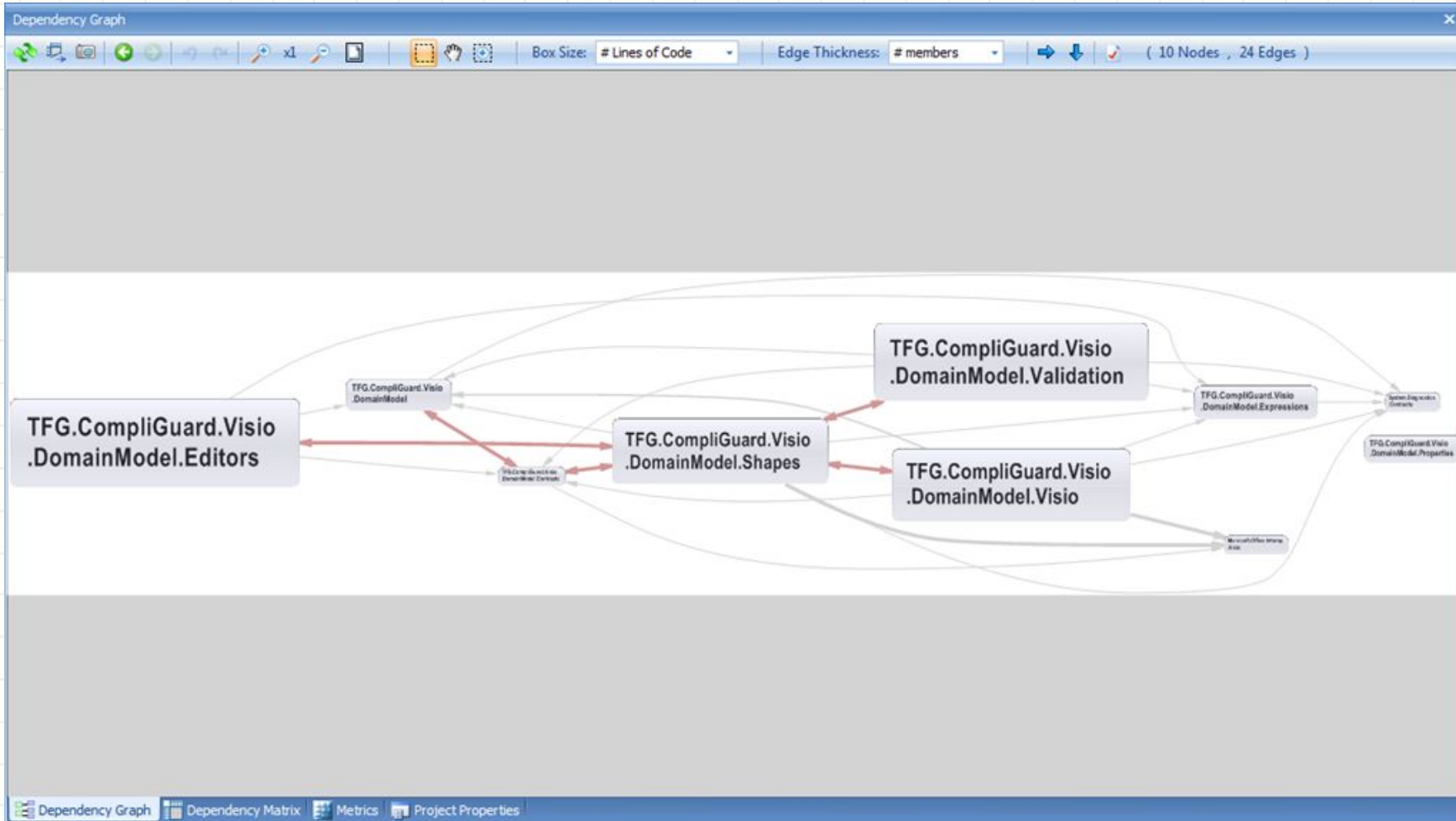
# Метрики кода

- Это количественные показатели, которые можно измерить и которые могут дать представление о качестве кода

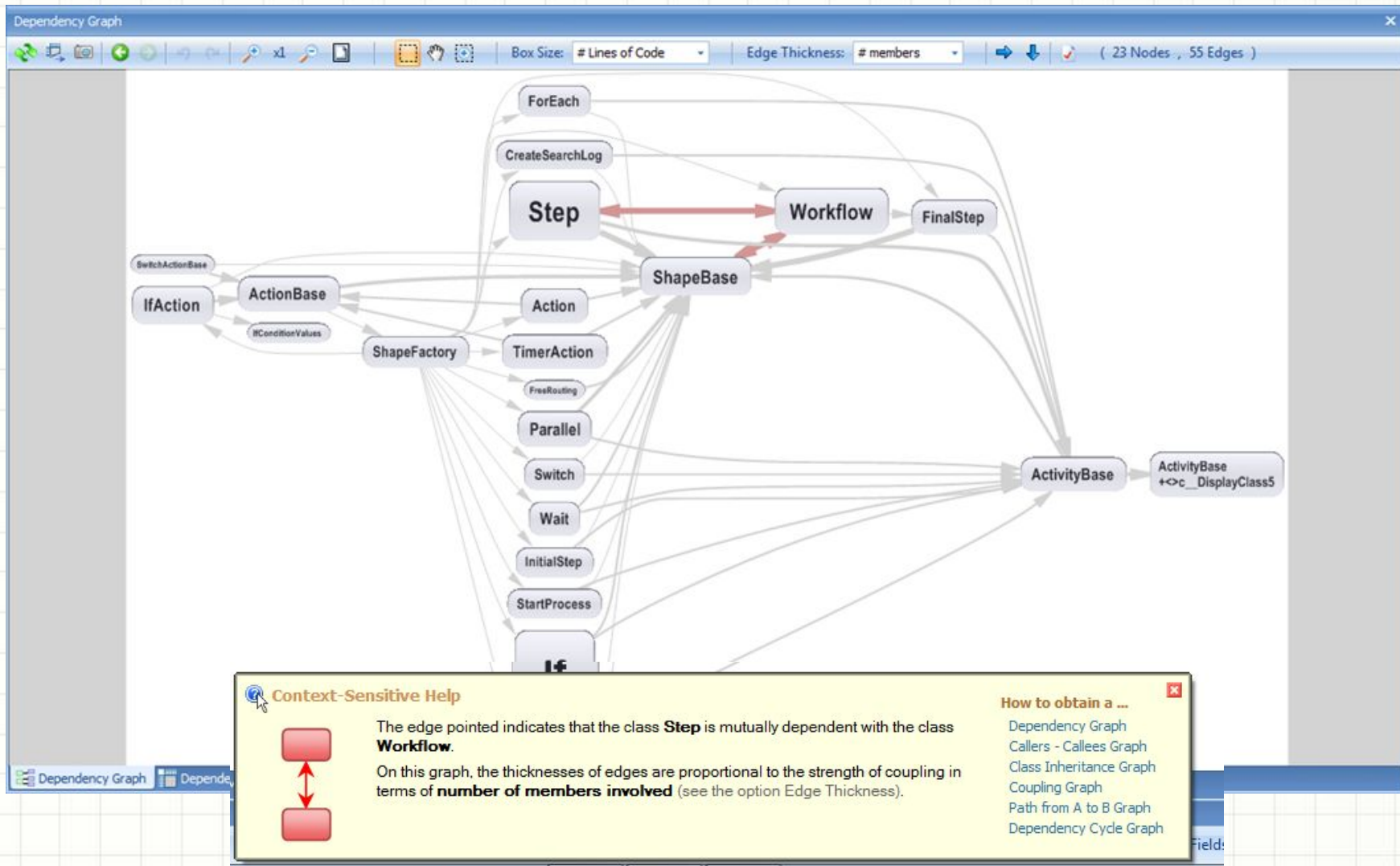
-  можно померять:

- » Lines of code
- » Number of classes
- » Inheritance depth
- » Maintainability index
- » Cyclomatic index

# NDepend. Dependency graph

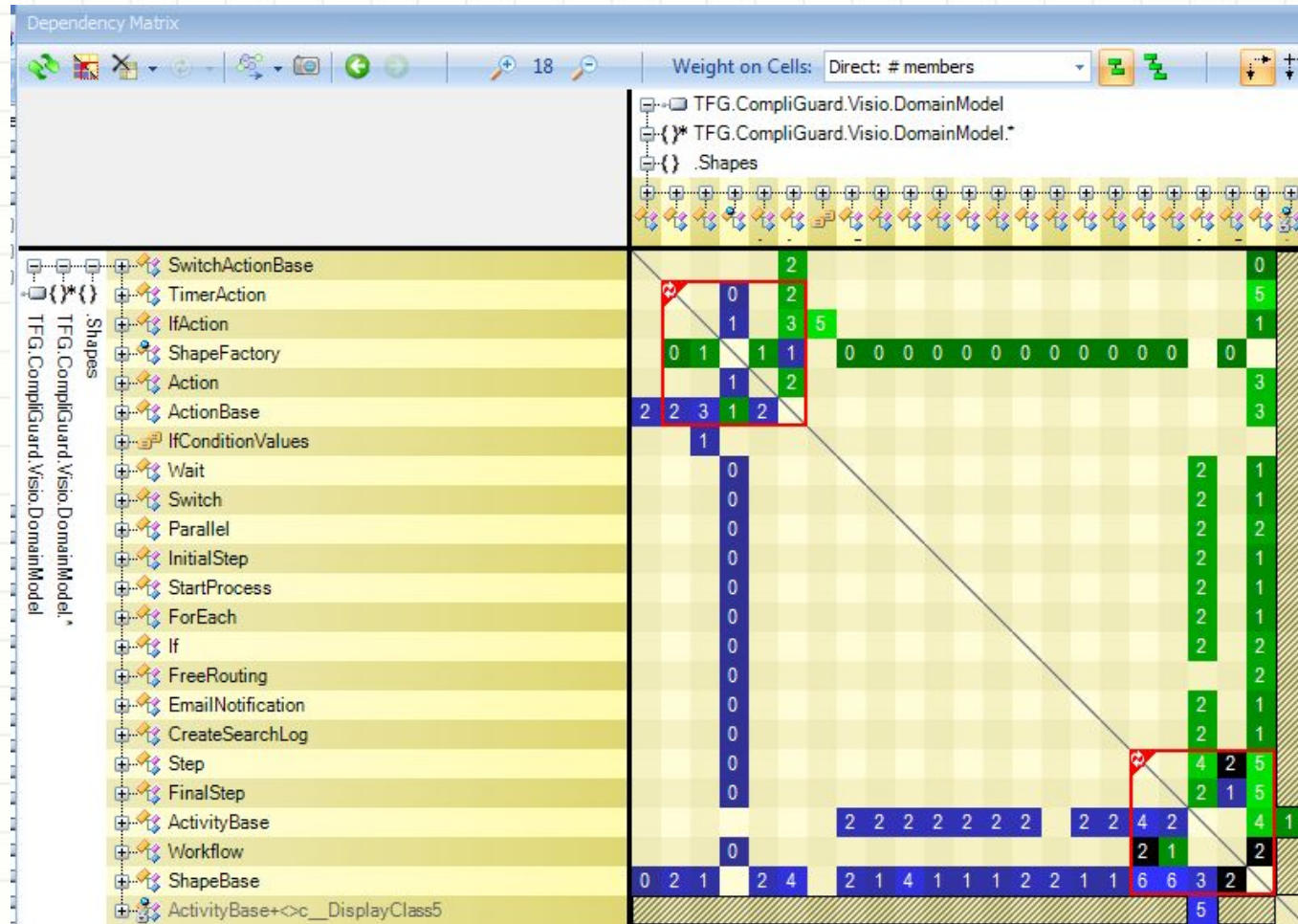


# NDepend. Dependency graph

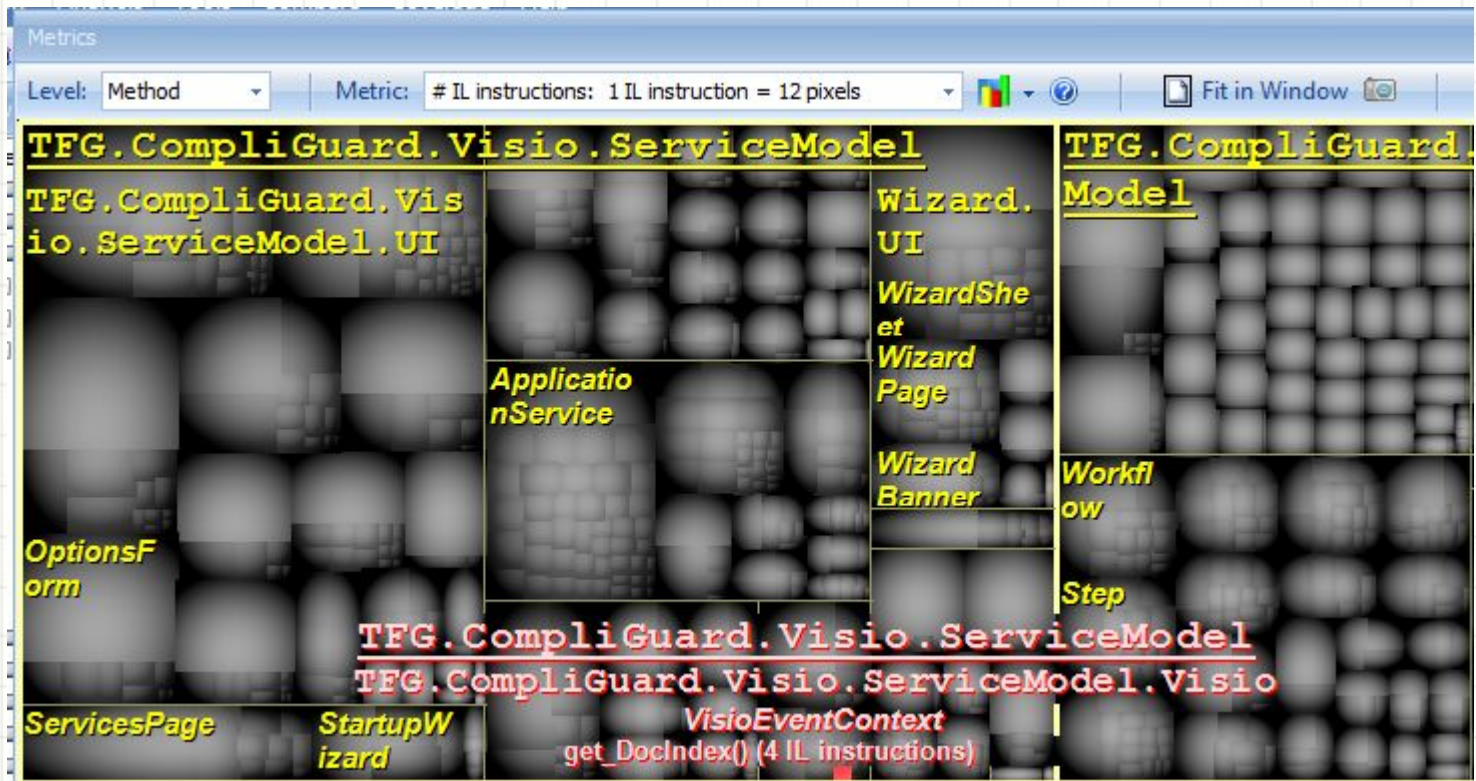




# Ndepend. Dependency matrix



# Ndepend. Metrics view





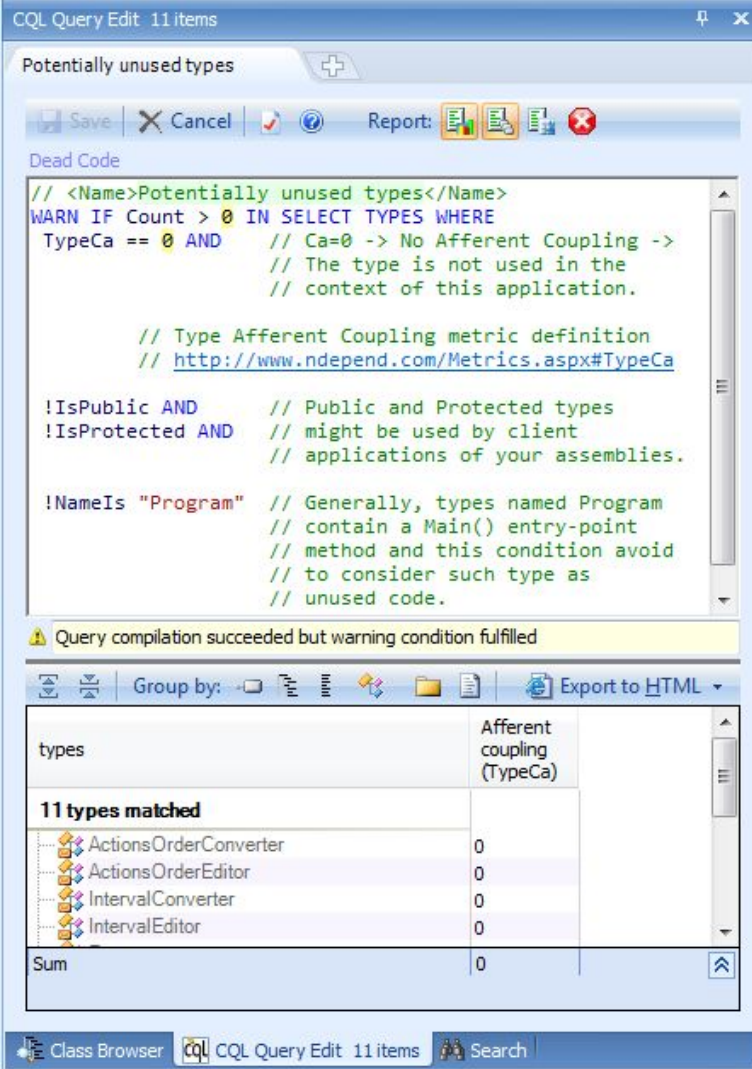
# NDepend. CQL

- SQL подобный синтакс
- Запросы к базе кода (code base), чтобы получить метрики

SELECT TOP 10 METHODS  
ORDER BY NbLinesOfCode DESC

SELECT METHODS  
WHERE NbLinesOfCode > 10

SELECT FIELDS WHERE HasAttribute  
"System.ThreadStaticAttribute"



The screenshot shows the CQL Query Editor interface. The query text is as follows:

```
// <Name>Potentially unused types</Name>
WARN IF Count > 0 IN SELECT TYPES WHERE
TypeCa == 0 AND // Ca=0 -> No Afferent Coupling ->
// The type is not used in the
// context of this application.

// Type Afferent Coupling metric definition
// http://www.ndepend.com/Metrics.aspx#TypeCa

!IsPublic AND // Public and Protected types
!IsProtected AND // might be used by client
// applications of your assemblies.

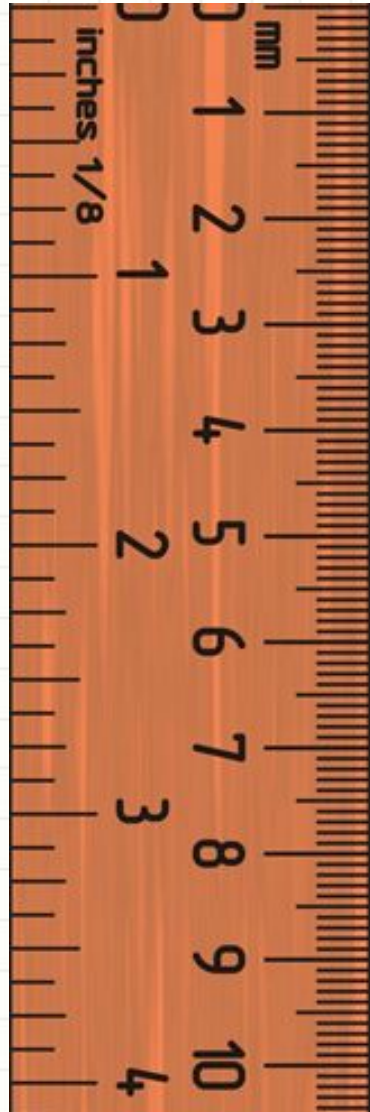
!NameIs "Program" // Generally, types named Program
// contain a Main() entry-point
// method and this condition avoid
// to consider such type as
// unused code.
```

Below the query, a status bar indicates: "Query compilation succeeded but warning condition fulfilled".

The results table is titled "types" and shows the following data:

types	Afferent coupling (TypeCa)
<b>11 types matched</b>	
ActionsOrderConverter	0
ActionsOrderEditor	0
IntervalConverter	0
IntervalEditor	0
Sum	0

# Метрики



## metrics

### cardinality

- Lines of Code (LOC) <sup>1</sup>
- Lines of Comments <sup>2,3</sup>
- Percentage Comment <sup>2</sup>
- Number of IL Instructions <sup>4</sup>
- Number of Assemblies
- Number of Namespaces <sup>5</sup>
- Number of Types
- Number of Fields
- Number of Methods

### relational

- Number of Parameters
- Number of Variables
- Afferent Coupling (Ca)
- Efferent Coupling (Ce)
- Instability (I)
- Relational Cohesion (H)
- Abstractness (A)
- Distance from main sequence (D)
- Level
- Rank
- Cyclomatic Complexity (CC)
- IL Cyclomatic Complexity (ILCC)
- Lack of Cohesion of Methods (LCOM)
- Instance Size
- Association Between Classes (ABC)
- Number of Children (NOC)
- Depth of Inheritance Tree (DIT)
- Response for a Type (RFT)

	Application	Assembly	Namespace	Type	Method	Field
Lines of Code (LOC) <sup>1</sup>	■	■	■	■	■	■
Lines of Comments <sup>2,3</sup>	■	■	■	■	■	■
Percentage Comment <sup>2</sup>	■	■	■	■	■	■
Number of IL Instructions <sup>4</sup>	■	■	■	■	■	■
Number of Assemblies	■					
Number of Namespaces <sup>5</sup>	■	■				
Number of Types	■	■	■			
Number of Fields	■	■	■	■		
Number of Methods	■	■	■	■	■	
Number of Parameters					■	
Number of Variables					■	
Afferent Coupling (Ca)		■	■	■	■	■
Efferent Coupling (Ce)		■	■	■	■	
Instability (I)		■				
Relational Cohesion (H)		■				
Abstractness (A)		■				
Distance from main sequence (D)		■				
Level		■	■	■	■	
Rank				■	■	■
Cyclomatic Complexity (CC)				■	■	■
IL Cyclomatic Complexity (ILCC)				■	■	
Lack of Cohesion of Methods (LCOM)				■	■	
Instance Size				■		■
Association Between Classes (ABC)				■		
Number of Children (NOC)				■		
Depth of Inheritance Tree (DIT)				■		
Response for a Type (RFT)				■		

# Coupling

- **Efferent coupling (Ce):** внутренняя связанность, число типов внутри сборки, которые зависят от типов из вне сборки
- **Afferent coupling (Ca):** внешняя связанность, число типов вне сборки которые зависят от типов в внутри сборки

# Instability (I)

**Instability (I):** отношение внутренней связанности( $C_e$ ) к общей связанности, индикатор устойчивости к изменениям.

$$I = C_e / (C_e + C_a)$$

$I=0$  – полностью стабильная сборка, сложная для модификации.

$I=1$  – нестабильная сборка, внутри слабая связанность

# Abstractness (A)

- Abstractness (A): абстрактность, отношение числа внутренних абстрактных типов к числу внутренних типов.

A=0 – полностью «конкретная» сборка

A=1 – полностью абстрактная сборка



# Relational Cohesion (H)

**Relational Cohesion (H):** относительная сцепленность, среднее число внутренних отношений на тип:

$$H = (R + 1) / N,$$

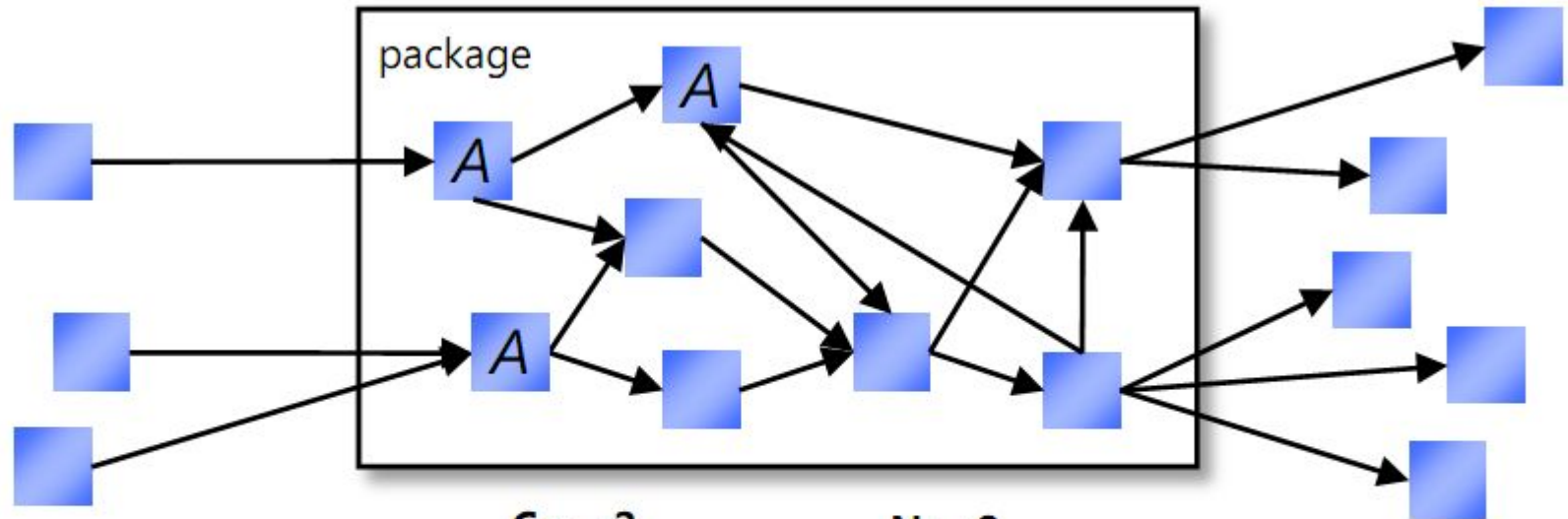
где R = число отношений внутри сборки,

N = число типов сборки

Классы внутри сборки должны сильно соотноситься друг с другом, и сцепленность должна быть высокой.

С другой стороны, слишком большие значения могут означать излишнюю связанность. Хорошие значения сцепленности  $1.5 \leq H \leq 4.0$

# Coupling, Cohesion, Abstractness and Instability metrics on example



$$C_e = 2$$

$$N = 8$$

$$C_a = 5$$

$$R = 12$$

$$I = 2/7 = 0.29$$

$$A = 3/8 = 0.375$$

$$H = 13/8 = 1.625$$

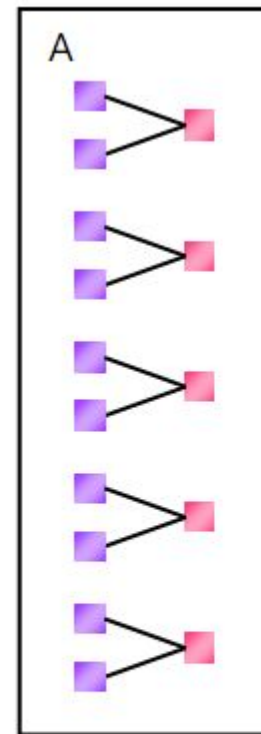
$C_e$  = внутренняя связанность,  $C_a$  – внешняя,  $I$  – стабильность,  $N$  – число типов сборки,  $A$  – абстрактность,  $H$  – относительная сцепленность

# Lack of cohesion (LCOM)

Принцип SRP утверждает, что класс должен иметь не более чем одну причину для изменения. Такой класс сцепленный (cohesive).

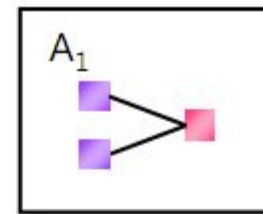
Высокое значение означает плохо сцеплений класс.

Типы, у которых  $LCOM > 0.8$  могут быть проблемными. Тем не менее, очень сложно избежать таких не-сцепленных типов

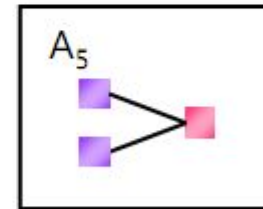


**LCOM = 0.8**

One class with five fields, each with a getter and setter.



**LCOM = 0**



**LCOM = 0**

Five classes, each with one field and a getter and setter.

# Cyclomatic Complexity (CC)

Число следующих выражений в методе:

*if, while, for, foreach, case, default, continue, goto, &&, ||, catch, ? : (ternary operator), ?? (nonnull operator)*

Эти выражения не учитываются:

*else, do, switch, try, using, throw, finally, return, object creation, method call, field access*

CC > 15 сложно понимать, CC > 30 очень сложные и должны быть разбиты на более мелкие методы (кроме сгенерированного кода)

# Distance from main sequence: zone of pain and zone of uselessness

Main sequence в терминах NDepend,

$$A + I = 1,$$

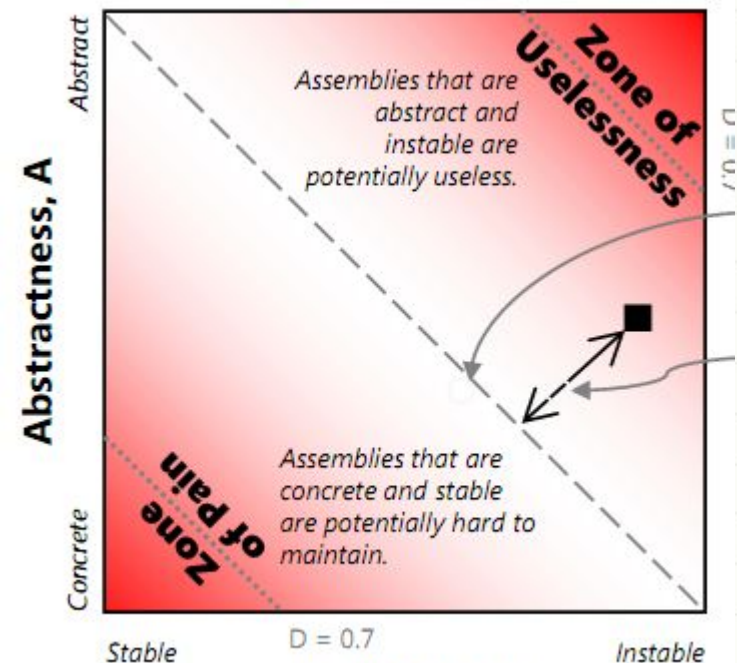
Представляет оптимальный баланс между абстрактностью и стабильностью

D – это нормализованное расстояние от main sequence,  $0 \leq D \leq 1$

Сборки с  $D > 0.7$  - проблематичны

Но, в реальной жизни, сложно избежать таких сборок.

Просто необходимо удерживать число таких сборок минимальным



# ССЫЛКИ И ИСТОЧНИКИ

## > Msug

> <http://msug.vn.ua/Posts/Details/4199> - NDepend

> <http://msug.vn.ua/Posts/Details/4221> - GRASP

## > NDepend

> <http://www.ndepend.com/>

> <http://www.ndepend.com/GettingStarted.aspx>

## > Метрики NDepend

> <http://www.hanselman.com/blog/content/binary/NDepend%20metrics%20placemats%201.1.pdf>

> <http://ndepend.com/Metrics.aspx>

## > О качестве

> <http://merle-amber.blogspot.com/>

> <http://blog.byndyu.ru>

>

Спасибо за  
внимание:)

