ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Кодирование

Кодирование: Метрики разработки ПО

Мера и Метрика

- *Мера* - количественный показатель степени, количества, или размеров некоторых атрибутов продукта или процесса.
  - Например, количество ошибок
- *Метрика* - количественная мера позволяющая оценить, в какой степени система, компоненты или процесс обладают заданным атрибутом. “Предположение о значении данного атрибута.”
  - Например, количество обнаруженных ошибок на затраченный человеко-час

Зачем измерять ПО?

- Определение качества существующего продукта или процесса
- Прогнозирование качества продукта / процесса
- Улучшение качества продукта / процесса

Мотивация для метрик

- Оценка стоимости и графика будущих проектов
- Оценка производительности применения новых средств и методов
- Определение тенденций производительности с течением времени
- Улучшение качества программного обеспечения
- Прогноз будущих потребностей в персонале
- Предвидеть и сокращения будущих потребностей в техническом обслуживании

Метрики сложности программного кода

Метрики сложности программ принято разделять на три основные группы:

1. метрики размера программ;
2. метрики сложности потока управления программ;
3. метрики сложности потока данных программ.

Размерно-ориентированные метрики

Размерно-ориентированные метрики

LOC-оценка (Lines Of Code) может включать в себя:

- общие трудозатраты (в человеко-месяцах, человеко-часах);
- объем программы (в тысячах строках исходного кода - LOC);
- стоимость разработки;
- объем документации;
- ошибки, обнаруженные в течение года эксплуатации;
- количество людей, работавших над изделием;
- срок разработки.

Физические и логические строки кода

- «**Физические**» строки кода – SLOC (используемые аббревиатуры **LOC, SLOC, KLOC, KSLOC, DSLOC**) – определяется как общее число строк исходного кода, включая комментарии и пустые строки.
- «**Логические**» строки кода – SLOC (используемые аббревиатуры **LSI, DSI, KDSI, где «SI»** - source instructions) – определяется как количество команд и зависит от используемого языка программирования.

Недостатки SLOC

- Неправильно сводить оценку работы человека к нескольким числовым параметрам. Менеджер может назначить наиболее талантливых программистов на сложнейший участок работы.
- Метрика не учитывает опыт сотрудников и их другие качества.
- Искажение: процесс измерения может быть искажён за счёт того, что сотрудники знают об измеряемых показателях и стремятся оптимизировать эти показатели, а не свою работу.
- Неточность: нет метрик, которые были бы одновременно и значимы и достаточно точны. Количество строк кода — это просто количество строк, этот показатель не даёт представления о сложности решаемой проблемы.

Метрики сложности

Метрики сложности

- **Объектно-ориентированные**: оценка сложности объектно-ориентированных проектов
- **Метрики Холстеда**: вычисляются на основании анализа числа строк и синтаксических элементов исходного кода программы
- **Цикломатическая сложность** : один из наиболее распространенных показателей оценки сложности программных проектов на основе графа управляющей логики
- **Метрика Чепина**: оценка информационной прочности

Объектно-ориентированные метрики

Метрика	Описание
Взвешенная насыщенность класса 1 (Weighted Methods Per Class (WMC)	Отражает относительную меру сложности класса на основе цикломатической сложности каждого его метода. Класс с более сложными методами и большим количеством методов считается более сложным. При вычислении метрики родительские классы не учитываются.
Взвешенная насыщенность класса 2 (Weighted Methods Per Class (WMC2))	Мера сложности класса, основанная на том, что класс с большим числом методов, является более сложным, и что метод с большим количеством параметров также является более сложным. При вычислении метрики родительские классы не учитываются.
Глубина дерева наследования (Depth of inheritance tree)	Длина самого длинного пути наследования, заканчивающегося на данном модуле. Чем глубже дерево наследования модуля, тем может оказаться сложнее предсказать его поведение. С другой стороны, увеличение глубины даёт больший потенциал повторного использования данным модулем поведения, определённого для классов-предков.
Количество детей (Number of children)	Число модулей, непосредственно наследующих данный модуль. Большие значения этой метрики указывают на широкие возможности повторного использования; при этом слишком большое значение может свидетельствовать о плохо выбранной абстракции.
Связность объектов (Coupling between objects)	Количество модулей, связанных с данным модулем в роли клиента или поставщика. Чрезмерная связность говорит о слабости модульной инкапсуляции и может препятствовать повторному использованию кода.
Отклик на класс (Response For Class)	Количество методов, которые могут вызываться экземплярами класса; вычисляется как сумма количества локальных методов, так и количества удаленных методов

Метрики Холстеда

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

- **NUOprtr** (Number of Unique Operators) — число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- **NUOprnd** (Number of Unique Operands) — число уникальных операндов программы (словарь операндов);
- **Noprtr** (Number of Operators) — общее число операторов в программе;
- **Noprnd** (Number of Operands) — общее число операндов в программе.

Оценки на основе Метрики Холстеда

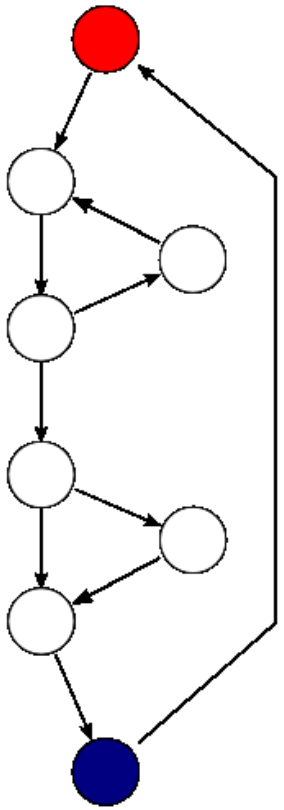
- **Словарь программы** (Halstead Program Vocabulary):
  - HPVoc = NUOprtr + NUOprnd;
- **Длина программы** (Halstead Program Length):
  - HPLen = Noprtr + Noprnd;
- **Объем программы** (Halstead Program Volume):
  - HPVol = HPLen log2 HPVoc;
- **Сложность программы** (Halstead Difficulty, HDiff):
  - HDiff = (NUOprtr/2) × (NOprnd / NUOprnd);
- На основе показателя HDiff предлагается оценивать **усилия программиста** при разработке при помощи показателя HEff (Halstead Effort):
  - HEff = HDiff × HPVol.

Цикломатическая сложность

- Цикломатическая сложность части программного кода — счётное число линейно независимых маршрутов через программный код.
- Если исходный код не содержит никаких точек решений, таких, как указания **IF** или циклы **FOR**, то сложность равна единице, поскольку, есть только единственный маршрут через код.

ЦС: Граф потока управления программы

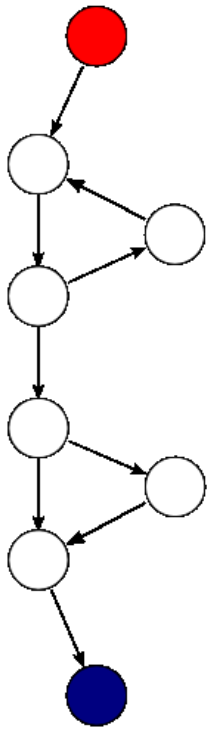
- Графом потока управления программы называется ориентированный граф, в узлах которого находятся неделимые группы команд, а ребрами соединяются такие блоки, которые могут быть выполнены сразу друг за другом.



(2) Цикломатическая сложность

- Показатель цикломатической сложности позволяет не только **произвести оценку** трудоемкости реализации отдельных элементов программного проекта и скорректировать общие показатели оценки длительности и стоимости проекта, но и **оценить связанные риски** и принять необходимые управленческие решения.
- **C = e – n + 2p**, где e – число ребер, n – число узлов, а p – число компонентов связности на графе управляющей логики.

Пример цикломатической сложности



Программа начинает выполняться с красного узла,
затем идут циклы (после красного узла идут две группы по три узла).
Выход из цикла осуществляется через условный оператор (нижняя группа узлов)
и конечный выход из программы в синем узле. Для этого графа E = 9, N = 8 и P = 1,
цикломатическая сложность программы равна 3.

Метрики Чепина

Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы.

- **Множество «P»** – вводимые переменные для расчетов и для обеспечения вывода. Примером может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы, то есть сама переменная не модифицируется, а только содержит исходную информацию.

- **Множество «M»** – модифицируемые или создаваемые внутри программы переменные.

- **Множество «C»** – переменные, участвующие в управлении работой программного модуля (управляющие переменные).

- **Множество «T»** – не используемые в программе (“паразитные”) переменные.

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Вычисление метрики Чепина

В общем виде

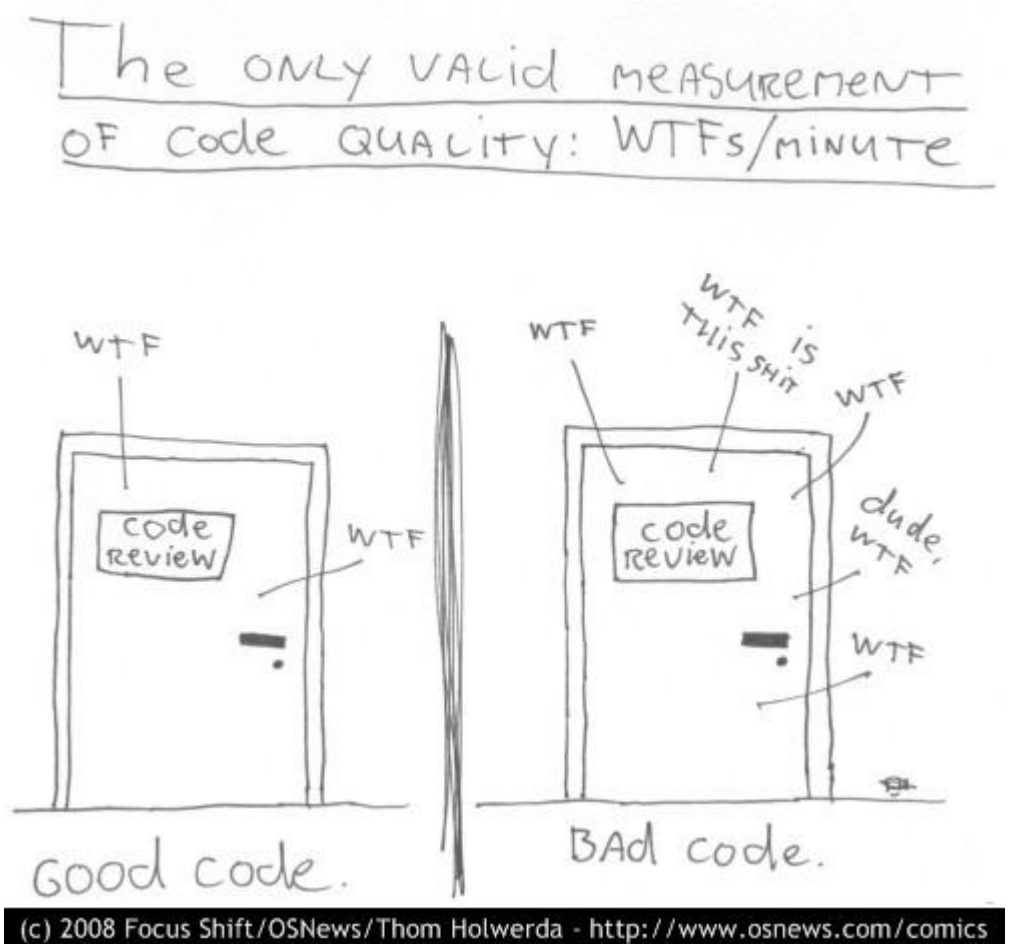
- $Q = a1P + a2M + a3C + a4T$,

где a1, a2, a3, a4 – весовые коэффициенты.

Автор предлагает следующие значения коэффициентов:

- **$Q = P + 2M + 3C + 0.5T$**

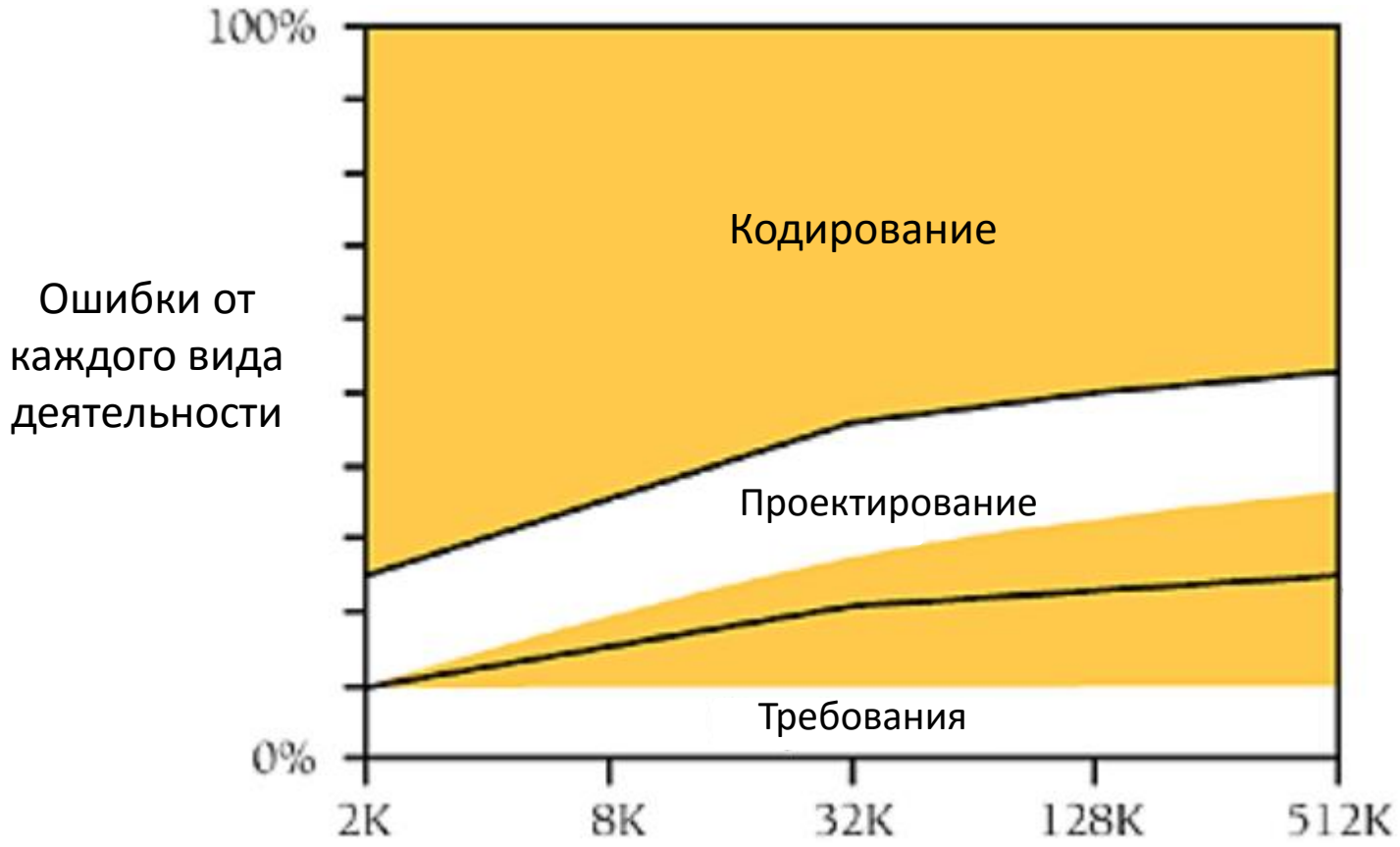
Единственно-правильная метрика качества программного кода



Использование метрик: размер команды разработчиков

Размер команды (человек)	Примерная доля проектов (%)
1 – 3	25%
4 – 10	30%
11 – 25	20%
26-50	15%
50+	10%

Использование метрик: зависимость типа ошибок от размера проекта



Использование метрик: проекта и плотность ошибок

Размер проекта (строк кода)	Плотность ошибок
Менее 2К	0-25 ошибок на 1К строк кода (KLOC)
2К – 16К	0-40 ошибок на 1 KLOC
16К – 64К	0,5 – 50 ошибок на 1 KLOC
64К – 512К	2 – 70 ошибок на 1 KLOC
512К+	4 – 100 ошибок на 1 KLOK

Использование метрик: проекта и плотность ошибок

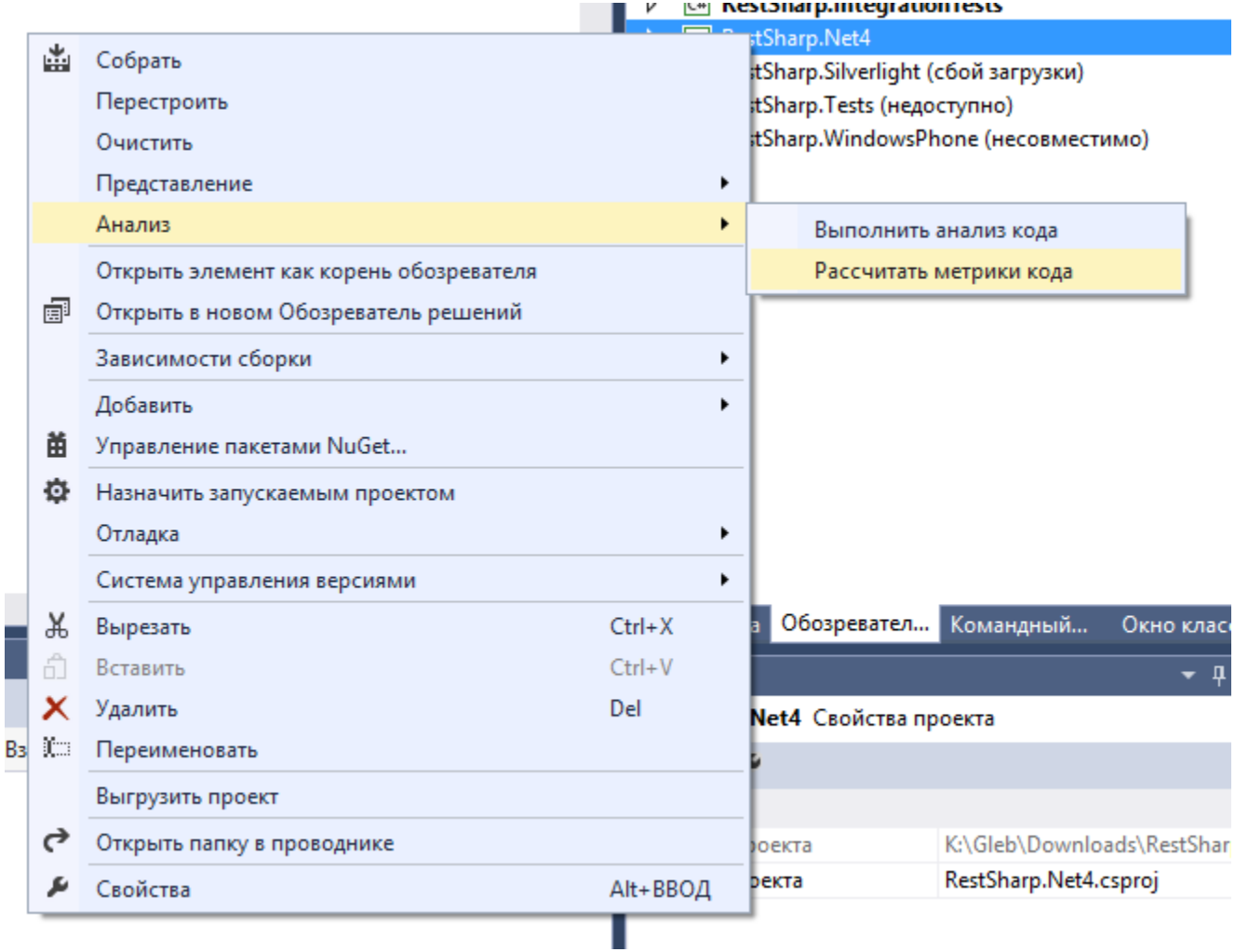
Размер проекта (строк кода)	Строк кода на человека в год
1K	2500 – 25 000 (4 000)
10K	2000 – 25 000 (3 200)
100K	1000 – 20 000 (2 600)
1 000K	700 – 10 000 (2 000)
10 000K	300 – 5 000 (1 600)

Программная поддержка

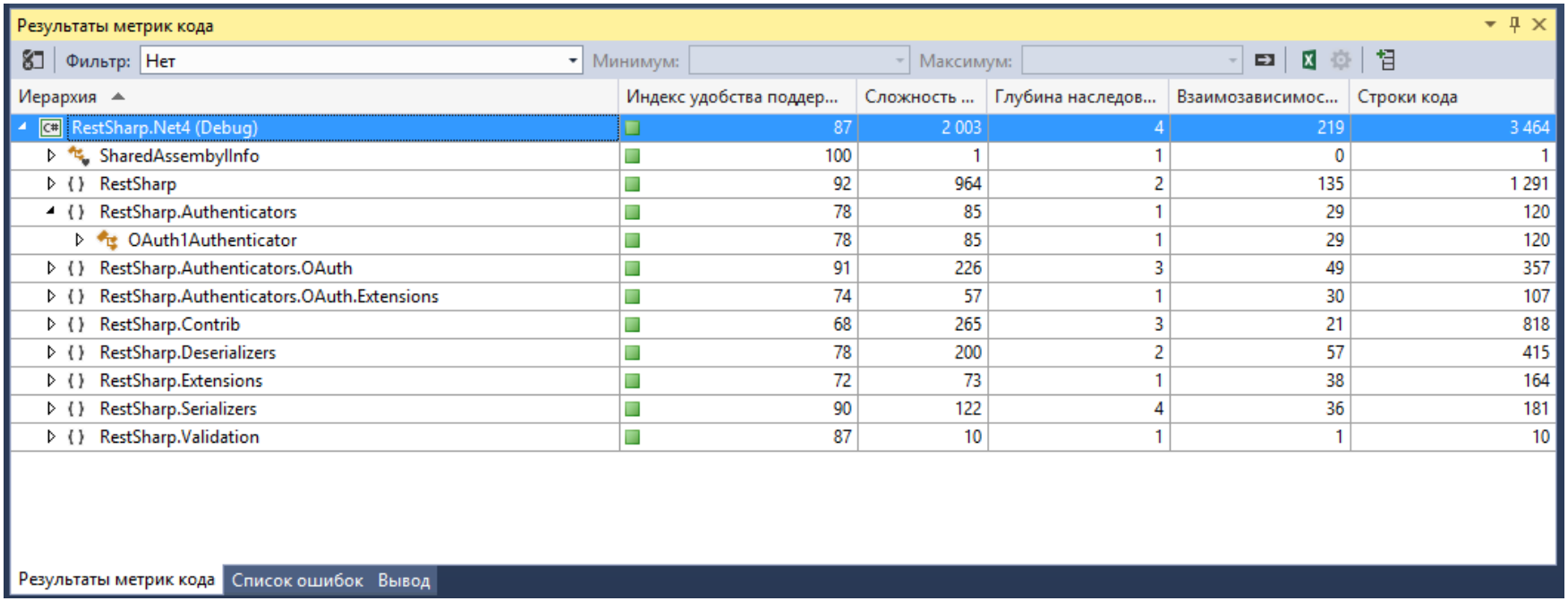
ПО для оценки метрик программного кода

- Практически на всех платформах для разработки ПО присутствуют инструменты для оценки метрик программного кода. Обычно, они совмещаются с системами статистического анализа кода.
- Пример – анализ кода проекта RestSharp в Visual Studio 2013

Visual Studio 2013 Pro – метрики кода



Visual Studio 2013 Pro – метрики кода



Иерархия	Индекс удобства поддер...	Сложность ...	Глубина наследов...	Взаимозависимос...	Строки кода
RestSharp.Net4 (Debug)	87	2 003	4	219	3 464
SharedAssemblyInfo	100	1	1	0	1
RestSharp	92	964	2	135	1 291
RestSharp.Authenticators	78	85	1	29	120
OAuth1Authenticator	78	85	1	29	120
RestSharp.Authenticators.OAuth	91	226	3	49	357
RestSharp.Authenticators.OAuth.Extensions	74	57	1	30	107
RestSharp.Contrib	68	265	3	21	818
RestSharp.Deserializers	78	200	2	57	415
RestSharp.Extensions	72	73	1	38	164
RestSharp.Serializers	90	122	4	36	181
RestSharp.Validation	87	10	1	1	10

Visual Studio 2013 - метрики

- Индекс удобства поддержки:
  - В столбце “Индекс удобства поддержки” помимо числового результата также содержится соответствующий значок.
  - Зеленый значок обозначает относительно высокую степень сопровождаемости. Желтый значок обозначает среднюю степень сопровождаемости. А красный значок обозначает низкую степень сопровождаемости и потенциально проблемный участок.
- *Индекс удобства поддержки* вычисляется на основе трех метрик:
  - сложность циклов,
  - число строк кода и
  - вычислительная сложность.

Visual Studio 2013 - метрики

- **Сложность организации циклов** (цикломатическая сложность): определяет число ветвей циклов в модуле; чем меньше – тем лучше;
- **Глубина наследования**: определяется числом уровней наследования в модуле; чем меньше – тем лучше;
- **Взаимозависимость классов** – определяет число классов, на которые есть ссылки; чем меньше – тем лучше;
- **Строки кода** – приблизительная оценка строк кода.