



«Code review как средство
обеспечения качества программного
обеспечения»

Надежда Кобозева. КРОК Инкорпорейтед 

Code review. Что же это такое??

Инспекция кода или **рецензия кода** или **ревизия кода** (от англ. *Code review*) — систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки.



Найденные на ранних этапах разработки ошибки «ценнее», чем ошибки, найденные при формальном тестировании!

«Зачем?!»

Цели внедрения процесса:

- унифицировать код;
- обучить новичков;
- ускорить тестирование;
- найди и уничтожить ошибки там, где они живут;
- разъяснение кода гарантирует его понимание;
- уменьшить риск выбиться из графика, потому что качество продукта растет с каждой завершенной фазой проекта;
- выявить синтаксические ошибки, которые компиляторы не могут отследить;

```

int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
              // guaranteed to be random.
}
  
```

Что такое «хорошо» и что такое «плохо»?

Для качественного построения процесса тестирования, правила рецензирования кода следует унифицировать!



Code Review Checklist

Очень индивидуален для каждого проекта, НО! Выделим базовые контрольные точки:

- ✓ Поиск уязвимостей
- ✓ Поиск синтаксических ошибок
- ✓ Поиск утечки ресурсов
- ✓ Контроль структуры
- ✓ Комментарии

Кому доверить процесс?

Каждый проект по-своему подходит к ревизиям: используются различные инструменты, различные процедуры, в приоритет ставятся различные цели.

Возможные схемы процесса:

1. По времени

- До check-in – а
- После check-in –а

2. По ответственному за процесс:

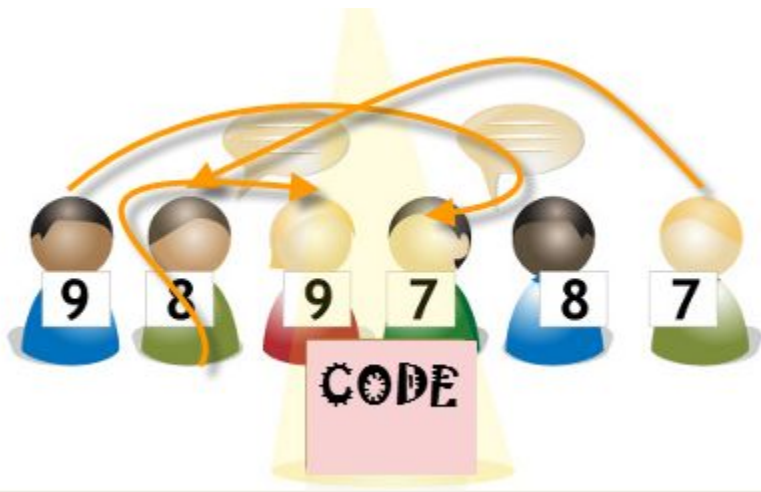
- Ответственный ведущий разработчик
- Распределенная схема рецензирования между разработчиками
- Ответственный ведущий тестировщик

1. Ведущий разработчик ревьюит

- **Преимущества:**
 - Единообразие кода
 - Одинаковые требования
 - Гуру делится опытом
- **Недостатки:**
 - Упущенные возможности
 - Только выборочная проверка
 - Скорее всего off-line проверка
 - Более важные задачи в приоритете, на ревью зачастую не хватает времени



2. Друг за другом



- **Преимущества:**
 - Полная взаимозаменяемость
 - Единообразие кода
 - Обучение
- **Недостатки**
 - «Если мой коллега пишет такой ужасный код, зачем и мне стараться»

Инструменты для организации процесса анализа кода

Сводят к минимуму хлопоты по организации, отслеживанию, подведению итогов и отчетности.

Открытые решения:

- Codestriker (David Sitsky)
- Review Board (Beanbag, Inc.)
- Rietveld (Google)

Платные продукты:

- Code Collaborator (SmartBear Software)
- Crucible (Atlassian)

3. Тестировщики!

- **Преимущества:**

- разработчики стараются писать качественнее;
- знание кода. Проще локализовать ошибку, проще читать логи;
- сокращение времени тестирования;

- **Недостатки:**

- технический уровень тестировщика;
- знание кода может помешать найти ошибки.

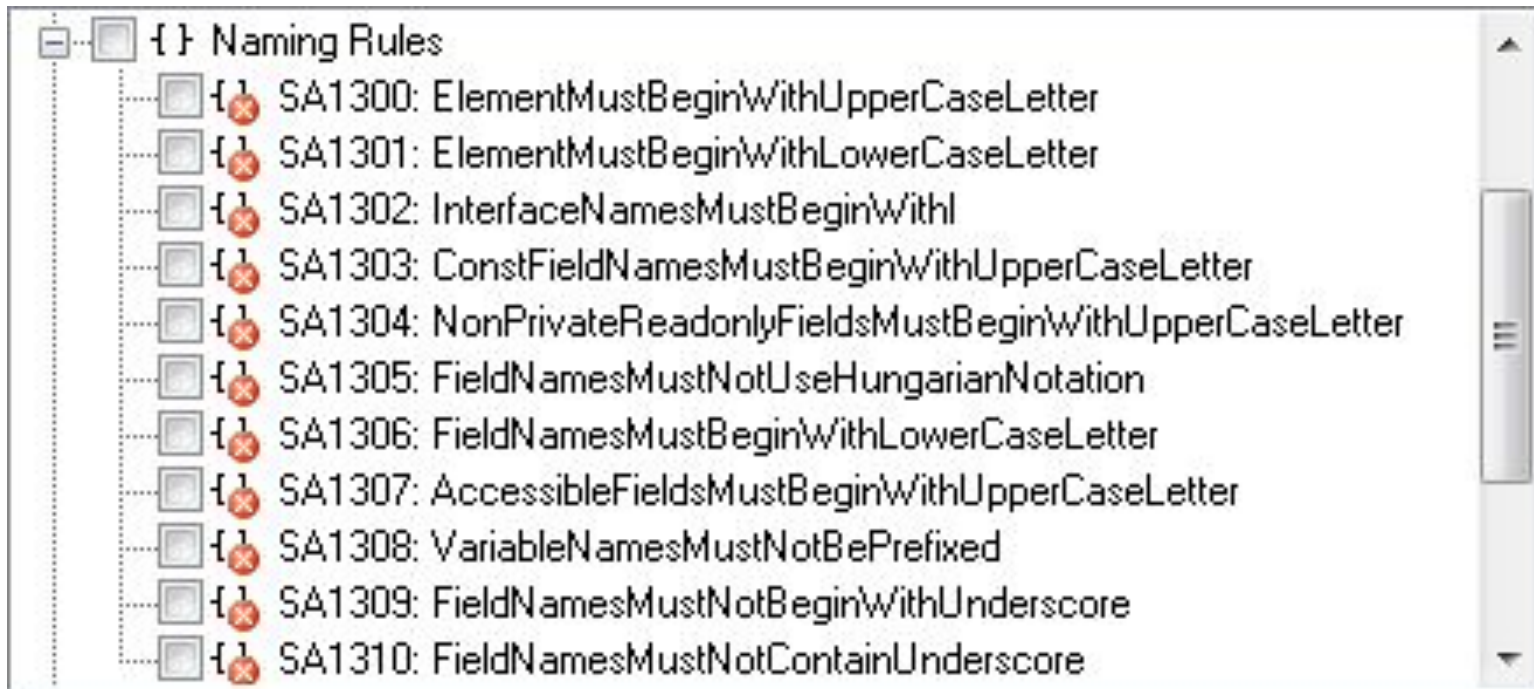


Легко отловим рецензированием



- **ошибки форматирования строк;**
- **состояние гонки;**
- **утечка памяти;**
- **переполнение буфера.**

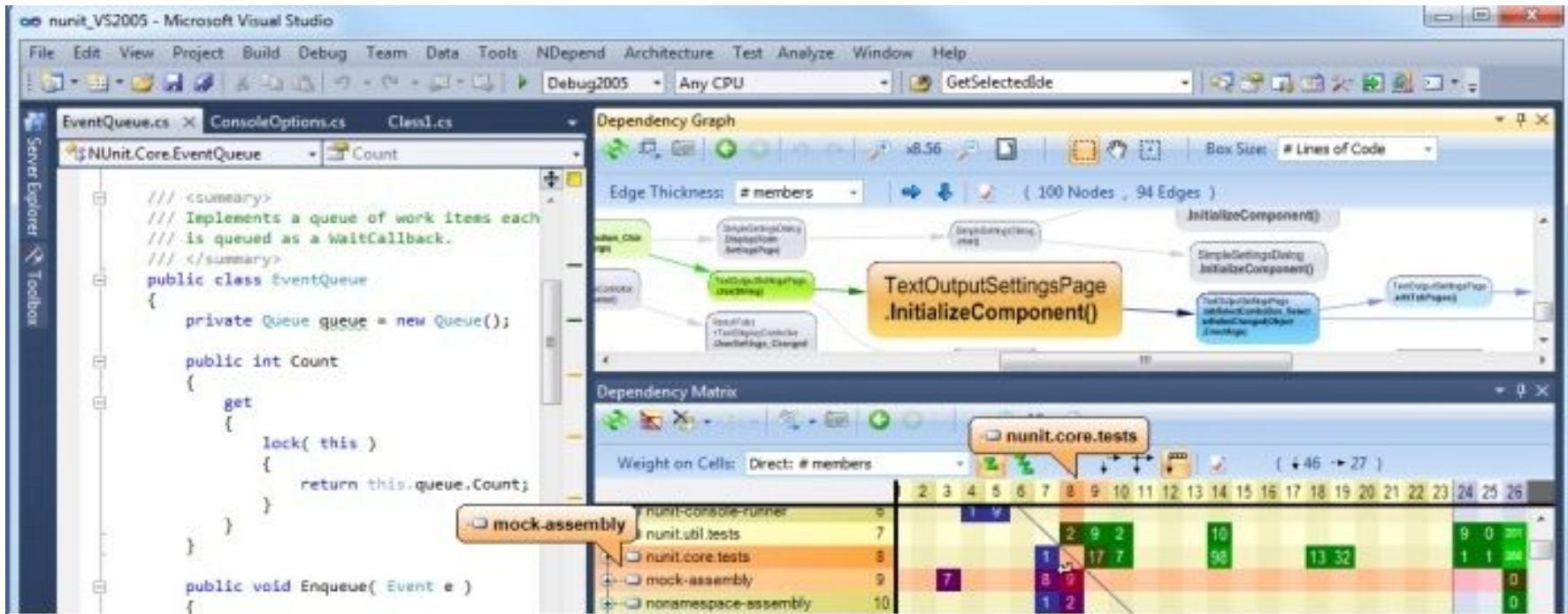
Инструменты для статического анализа кода. StyleCop



StyleCop — статический анализатор C# кода на предмет соответствия стилю.

+ плагин StyleCop Team Foundation Server check-in policy

Инструменты для статического анализа кода. NDependent



The screenshot displays the NDepend tool within Visual Studio. The main window shows a code editor with the following C# code for `EventQueue`:

```

    /// <summary>
    /// Implements a queue of work items each
    /// is queued as a WaitCallback.
    /// </summary>
    public class EventQueue
    {
        private Queue queue = new Queue();

        public int Count
        {
            get
            {
                lock( this )
                {
                    return this.queue.Count;
                }
            }
        }

        public void Enqueue( Event e )
        {
    }
    }
  
```

The **Dependency Graph** window shows a network of nodes representing code elements, with a central node for `TextOutputSettingsPage.InitializeComponent()`. The **Dependency Matrix** window shows a table of dependencies between assemblies:

Assembly	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
nunit-console-runner																										
nunit.util.tests																										
nunit.core.tests							1	2	9	2			10											9	0	
mock-assembly						7		8	9																1	1
nonamespace-assembly							1	2																	0	

NDepend – инструмент для Visual Studio для проведения комплексного анализа .NET кода и обеспечения его высокого качества.

Спасибо за внимание!

Кобозева Надежда

Компания КРОК



E-mail: nkobozeva@croc.ru

***p.s. Не гонитесь за идеалом!!!
Лучшее-враг хорошего!***