

**Static Analysis**  
for the WIN

**What** is Static Analysis?

Inspection of a system without actually running the code.

**Why** use Static Analysis?



Image Credit:

401(K) 2012 @ Flickr

<http://www.flickr.com/photos/68751915@N05/6355811869/>



If you can easily eliminate boring bugs, you get to spend your time in more interesting ways.

Like writing new bugs

Image Credit: SuperFantastic @ Flickr  
<http://www.flickr.com/photos/superfantastic/1547089906/>

Phase in Which a Defect Is Introduced

Requirements

Architecture

Construction

↑  
Cost

Requirements Architecture Construction System Test Post-Release

Phase in Which a Defect Is Detected

Steve McConnell **Code Complete (1993)** Chapter 3

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Informal design reviews	25%	35%	40%
Formal design inspections	45%	55%	65%
Informal code reviews	20%	25%	35%
Formal code inspections	45%	60%	70%
Modeling or prototyping	35%	65%	80%
Personal desk-checking of code	20%	40%	60%
Unit test	15%	30%	50%
New function (component) test	20%	30%	35%
Integration test	25%	35%	40%
Regression test	15%	25%	30%
System test	25%	40%	55%

Steve McConnell **Code Complete (1993)** Chapter 20



There is **good news**



"The upshot is that  
defect-detection methods  
**work better in combination**  
than they do singly."

Steve McConnell **Code Complete (1993)** Chapter 20



Image Credit:  
bayasaa @ Flickr  
<http://www.flickr.com/photos/bayasaa/2693171833/>

Do it **yourself**

all

**DataTransferObjects**

should be

**DataContracts**

```
var assembly = typeof(ClassificationDto).Assembly;
var dtoTypes = assembly.GetExportedTypes()
    .Where(type => type.IsClass)
    .Where(type => IsDataTransferObject(type));
var nonSerializable
    = dtoTypes.Where(type => !IsDataContract(type))
    .OrderBy(t => t.Name).ToList();
Assert.That( nonSerializable.Count(), Is.EqualTo(0);
```

All properties of  
**DataTransferObjects**  
should be  
**DataMembers**

```
var assembly = typeof(ClassificationDto).Assembly;
var contracts = assembly.GetExportedTypes()
    .Where(type => type.IsClass)
    .Where(type => IsDataTransferObject(type))
    .Where(type => IsDataContract(type));
var withMissingProperties
    = contracts.SelectMany(type => GetPublicProperties(type))
    .Where(property => HasDataMember(property) == false)
    .ToList();
Assert.That( withMissingProperties, Has.Count.EqualTo(0);
```

Should never directly create a


**PersistentEntity**

with new



```
var badMethodQuery
= from type in facadeAssembly.GetTypes()
  from method in type.GetMethods(allMethods)
  let inspector = new MethodInspector(method, false)
  from calledMethod in inspector.AllCalledMethods
  where calledMethod.IsConstructor
  where IsPersistentEntity(calledMethod.DeclaringType)
  select new {
    DeclaringType = type,
    Method = method,
    ConstructedType = calledMethod.DeclaringType
  };
```

# **Code Metrics**



DEMO

**COMMANDLINE**

CamelCase.ToDashedName(camelCase)

StringSequenceExtensions.JoinWith(sequence, separator, lastSeparator)

Show how the columns are sortable – but that this is useless

# **Code Digger**

Extensions &gt; Tools &gt; Microsoft Code Digger

## Microsoft Code Digger

DEVLABS

Free

Code Digger analyzes possible execution paths through your .NET code. The result is a table where each row shows a unique behavior of your code. The table helps you understand the behavior of the code, and it may also uncover hidden bugs. Through the new context menu item "Ge...

CREATED BY	RiSE (Research in Software Engineering) (Microsoft Corporation)	LAST UPDATED	7/12/2013
REVIEWS	★★★★ (20) Review	VERSION	0.95.3
SUPPORTS	Visual Studio 2012	LICENSE	View
DOWNLOADS	Download (11,770)	SHARE	
TAGS	Unit Testing, Testing, pes, Data Generation, Symbolic Execution	FAVORITES	Add to favorites

DESCRIPTION REVIEWS (20) Q AND A (13)

## Overview

Code Digger analyzes possible execution paths through your .NET code. The result is a table where each row shows a unique behavior of your code. The table helps you understand the behavior of the code, and it may also uncover hidden bugs.

Through the new context menu item **"Generate Inputs / Outputs Table"** in the Visual Studio editor, you can invoke Code Digger to analyze your code. Code Digger computes and displays input-output pairs. Code Digger systematically hunts for bugs, exceptions, and assertion failures.

Out of the box, Code Digger only works on public .NET code that resides in Portable Class Libraries. Keep reading until the end of the page to learn how to configure Code Digger to explore other .NET projects.

Report abuse to Microsoft



View Large

## More From RiSE (Research in Software Engineering)




Code Contracts  
Editor Extensions  
VS2010  
★★★★ (12)



Code Contracts  
for .NET  
★★★★ (13)



Code Contracts  
Editor Extensions  
VS2012  
★★★★ (4)



DEMO


## **COMMANDLINE**

`CamelCase.ToDashedName(camelCase)`

`StringSequenceExtensions.JoinWith(sequence, separator, lastSeparator)`

Show how the columns are sortable – but that this is useless

# **Code Analysis**



DEMO

**BUILDSCREEN**

Application that I use at work, though with modifications

This version unchanged from the latest version on CodePlex, except to add in Code Analysis

CA1001 – implement IDisposable to ensure the background worker is cleaned up properly

CA1008 – no defined value for '0', so any variables initialised with illegal value

CA1704 – misspelled namespace

CA2001 – a naïve plugin framework is vulnerable, might not load the expected assembly

CA2235 – member isn't serializable, need to ensure it's not included in serialization

Show how to suppress a warning

Can you find a list of suppressed messages?



# Rules

Create your own ruleset  
with the rules you value

Here are some interesting rules ... use the ones of value to you ...

## **CA1006: Do not nest generic types in member signatures**

Dates back to the original introduction of generics, but of little use to teams that are thoroughly comfortable with their use.

Would flag useful things like `Lazy<List<string>>`

## **CA1062 Validate arguments of public methods**

<http://esmithy.net/2011/03/15/suppressing-ca1062/>

<http://geekswithblogs.net/terje/archive/2010/10/14/making-static-code-analysis-and-code-contracts-work-together-or.aspx>

## **CA1704 Identifiers should be spelled correctly**

Detail adding a new dictionary file with new terms

Share it across multiple projects

## **CA2000 Dispose objects before losing scope**

## **CA2210 Assemblies should have valid strong names**

Do you use strong names? If so, this is a useful reminder; if not, disable it.

**Gendarme**

## Gendarme.Roadmap

Gendarme has synchronized it's version numbers with Mono since the 2.0 release. All future releases will share the same schedule. See [Mono Roadmap](#) for the exact release dates planned for each version.

### Gendarme 2.8 (in progress)

- Framework
  - [in progress] Use System.ComponentModel to configure the rules
    - allow to build dynamic configuration UI for rules (e.g. property grid)
  - Complete XML documentation and add some examples
  - More unit tests and move coverage to existing fixtures
  - Engines
    - Finish the incomplete engine from Hack Week 3
- Rules
  - More (of course ;-)
  - More rules that works on the assembly set, i.e. that benefits from the information of all assemblies being analyzed
  - Better visibility checks (e.g. InternalVisibleTo)
  - Moonlight/Silverlight specific rules
- Runners
  - Add GUI configuration to the wizard runner

### Gendarme 3.0+ (in planning)

- General
  - Start building against FX 4.0 and C# 4
  - Update to CecilFlight (latest version)
- Rules
  - More (of course ;-)
- Runners

#### Table of contents

- [Gendarme 2.8 \(in progress\)](#)
- [Gendarme 3.0+ \(in planning\)](#)
- [Ideas](#)
- [Mono Build](#)
- [Runner-related](#)
- [Framework](#)
- [Rules](#)

# **Code Contracts**

# what are **Code Contracts**?

Preconditions that have to hold for the routine to function

PostConditions that will hold afterwards

Invariants that are always true

Code Contracts make those explicit



Invented a programming language – Eiffel – that includes Code Contracts as a first class concept

Image credit:

[Extensions](#) > [Tools](#) > Code Contracts for .NET

## Code Contracts for .NET

DEVLABS

Free

Code Contracts are static library methods used from any .NET program to specify the code's behavior. Runtime checking and static checking tools are both provided for taking advantage of contracts.

[Report abuse to Microsoft](#)**CREATED BY**[RiSE \(Research in Software Engineering\) \(Microsoft\)](#)**LAST UPDATED**

5/3/2013

**REVIEWS**★★★★★ (13) [Review](#)**VERSION**

1.5.60502.11

**SUPPORTS**

Visual Studio 2012, 2010

**SHARE**[D](#) [T](#) [+](#) [+](#)**DOWNLOADS**[Download](#) (21,286)**FAVORITES**[Add to favorites](#)**TAGS**[CodeContracts](#), [contracts](#), [requires](#), [ensures](#), [invariant](#), [assume](#), [assert](#), [static checker](#)**DESCRIPTION**[REVIEWS \(13\)](#)**News**

- We released the contract editor extensions for VS2012. See contracts as you write code in intellisense and when you look at framework method definitions.  
[VS2012 version](#)  
[VS2010 version](#)


**Questions/Answers/Discussion:**

There is no discussion page here on the Gallery.

- Please use the [Code Contracts Forum](#) for all questions, bug reports, and especially compliments!
- Much more information, including videos and papers can be found on our [website](#).

### More From RiSE (Research in Software Engineering)

[Code Contracts Editor Extensions VS2010](#)  
★★★★★ (12)[Microsoft Code Digger](#)  
★★★★★ (20)[Code Contracts Editor Extensions VS2012](#)  
★★★★★ (4)



DEMO

## **GROUPINSPECTOR**

StringExtensions.Before()

Pin error list open

Build Solution (Analysis configuration) – note longer build time, 'tis doing a lot of work

Show how contracts for interfaces



**Unit Tests**  
**Code Metrics**  
**Code Digger**  
**Code Analysis**  
**Gendarme**  
**Code Contracts**

**StyleCop**

[HOME](#) [SOURCE CODE](#) [DOWNLOADS](#) [DOCUMENTATION](#) [DISCUSSIONS](#) [ISSUES](#) [PEOPLE](#) [LICENSE](#)[Page info](#) | [Change History \(all pages\)](#)★ [Follow \(1403\)](#) | [Subscribe](#)Follow [@stylecopdev](#) to keep up to date with all StyleCop announcements.

#### StyleCop Governance

Microsoft has turned over governance and coordination of the StyleCop project to the community. The new Coordinator of this project is responsible for managing contributions. Microsoft is not granting any IP rights to code or other material contributed to this project by third parties. The IP rights to each particular contribution are granted to you by the respective contributor under the Ms-PL license.

#### Project Description

StyleCop analyzes C# source code to enforce a set of style and consistency rules. It can be run from inside of Visual Studio or integrated into an MSBuild project. StyleCop has also been integrated into many third-party development tools.

#### 4.7 is compatible with:

- JetBrains ReSharper 5.1.3 (5.1.3000.112)
- JetBrains ReSharper 6.0 (6.0.2202.688)
- JetBrains ReSharper 6.1 (6.1.37.86)
- JetBrains ReSharper 6.1.1 (6.1.1000.82)
- JetBrains ReSharper 7.0.1 (7.0.1098.2760)
- JetBrains ReSharper 7.1.3 (7.1.3000.2254)
- JetBrains ReSharper 8.0 (8.0.14.856)
  
- Visual Studio 2008
- Visual Studio 2010

Search Wiki &amp; Documentation

download

CURRENT	4.7
DATE	Thu Jan 5, 2012 at 7:00 AM
STATUS	Stable
DOWNLOADS	318,285
RATING	★★★★★ 47 ratings

#### RECENT REVIEWS

★☆☆☆☆ StyleCop make VS2010 to Crash!! I don't know how was old version of this app but I installed StyleCop 4.7.45.0 and each time I run L...  
[\(more\)](#)

★★★★★ Great tool, fast support for ReSharper, good job!

[View all reviews](#)

#### ACTIVITY

**Font: Fredoka One**  
Font: Give You Glory