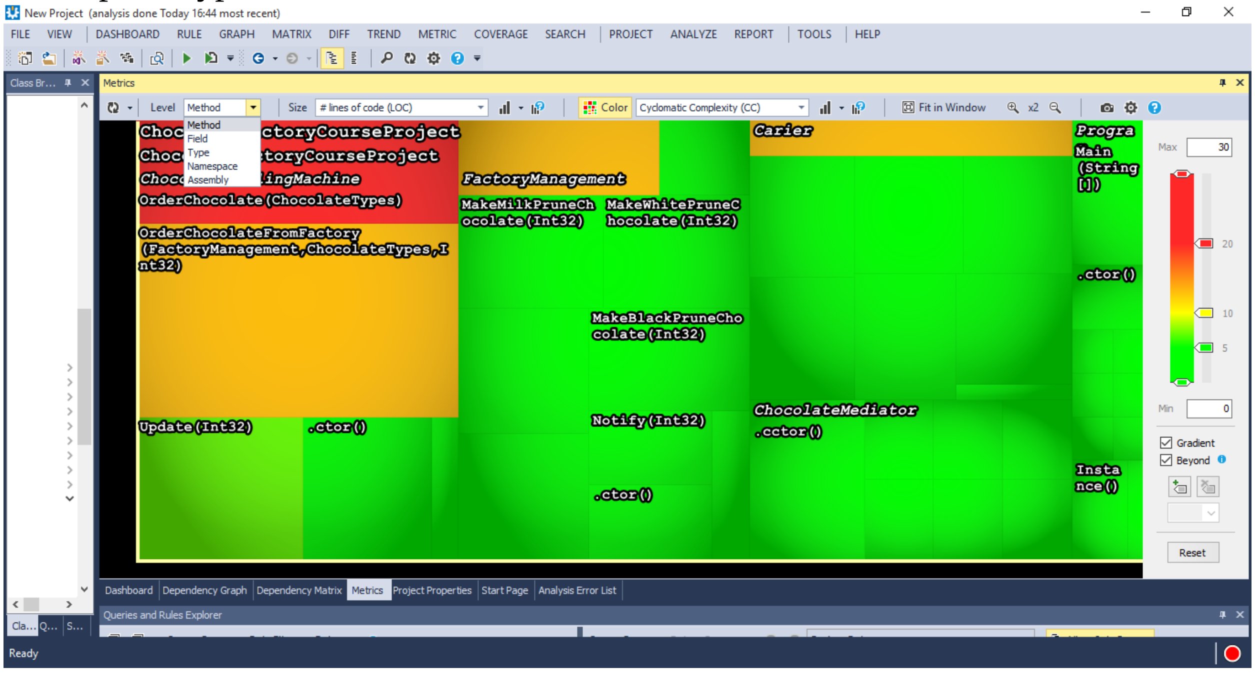
Denys Aleksyeyev

d) Diagrammatic representation (Treemap).

In the **Metric View**, the code base is represented through a **Treemap**. Treemapping is a visualization algorithm for displaying tree-structured data by using a nested rectangles hierarchy. The tree structure used in NDepend treemap is the usual code hierarchy:

- .NET assemblies contain namespaces
- namespaces contain types
- types contains methods and fields

Treemap rectangles represent code elements. The option **level** determines the kind of code element represented by unit rectangles. The option **level** can take the 5 values: assembly, namespace, type, method and field.



Level: Method

Size: # lines of code (LOC)

Color: Cyclomatic Complexity (CC)

Max: 30

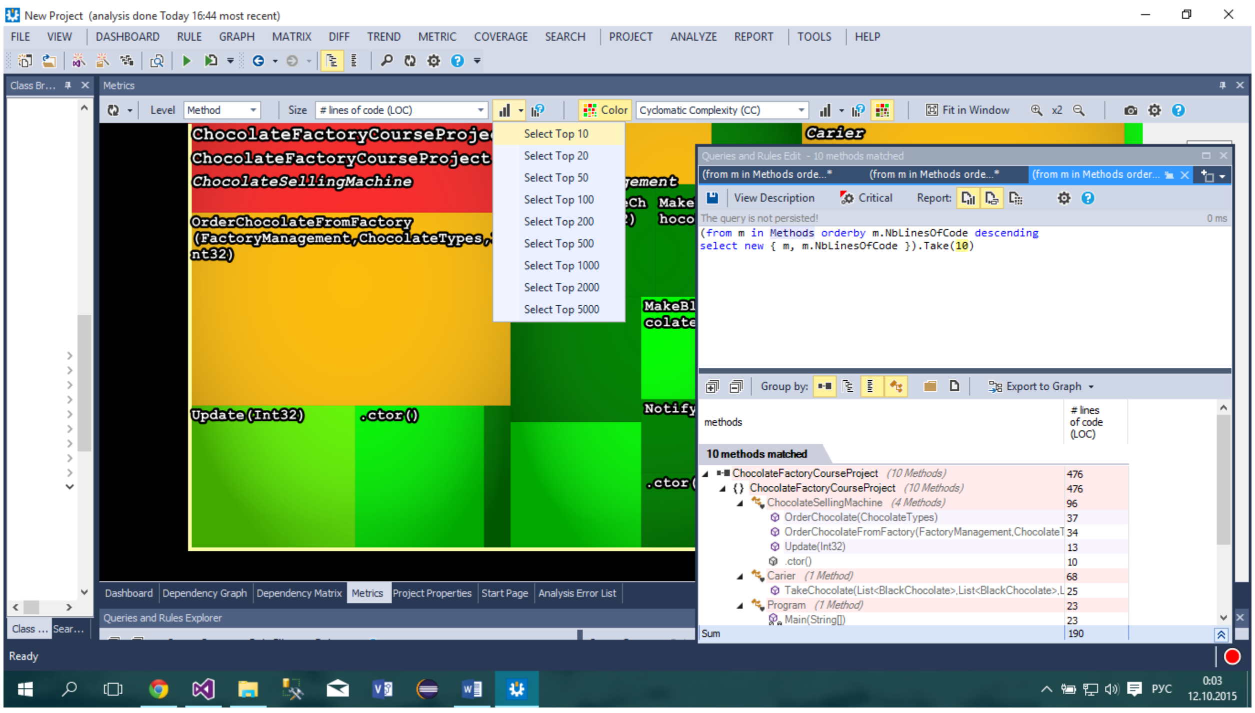
Min: 0

Notice how code elements that belong to the same parent code elements (like methods in a class, or classes in a namespaces) are located near on the treemap.

The option **Size** of the treemap determines the size of rectangles. For example if level is set to *type* and metric is set to *number of Lines of Code*, each unit rectangle represents a type, and the size of a unit rectangle is proportional to the number of Lines of Code of the corresponding type.

For each code element level, several code metrics are proposed.

An option lets generate a CQLinq code query to *Select the Top N* code elements according to the selected Size code metrics.



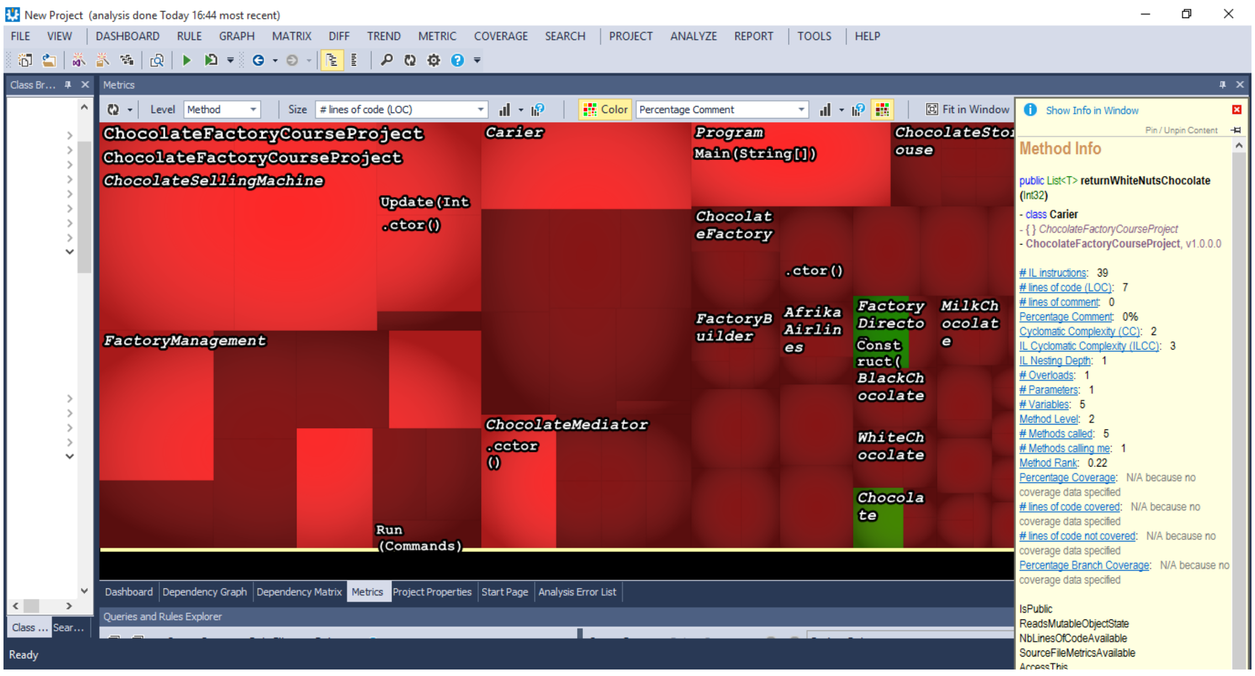
Queries and Rules Edit - 10 methods matched

The query is not persisted!

```
(from m in Methods orderby m.NbLinesOfCode descending
select new { m, m.NbLinesOfCode }).Take(10)
```

methods	# lines of code (LOC)
ChocolateFactoryCourseProject (10 Methods)	476
ChocolateFactoryCourseProject (10 Methods)	476
ChocolateSellingMachine (4 Methods)	96
OrderChocolate(ChocolateTypes)	37
OrderChocolateFromFactory(FactoryManagement,ChocolateT	34
Update(Int32)	13
.ctor()	10
Carier (1 Method)	68
TakeChocolate(List<BlackChocolate>,List<BlackChocolate>,L	25
Program (1 Method)	23
Main(String[])	23
Sum	190

The section above explained how a code metric can be visualized thanks to treemaping. A second code metric can be represented by coloring code elements rectangle. Hence the NDepend metric view makes convenient to correlate 2 code metrics. For example the screenshot below shows code coverage by comments. As we see the project doesn't have comments ☺.



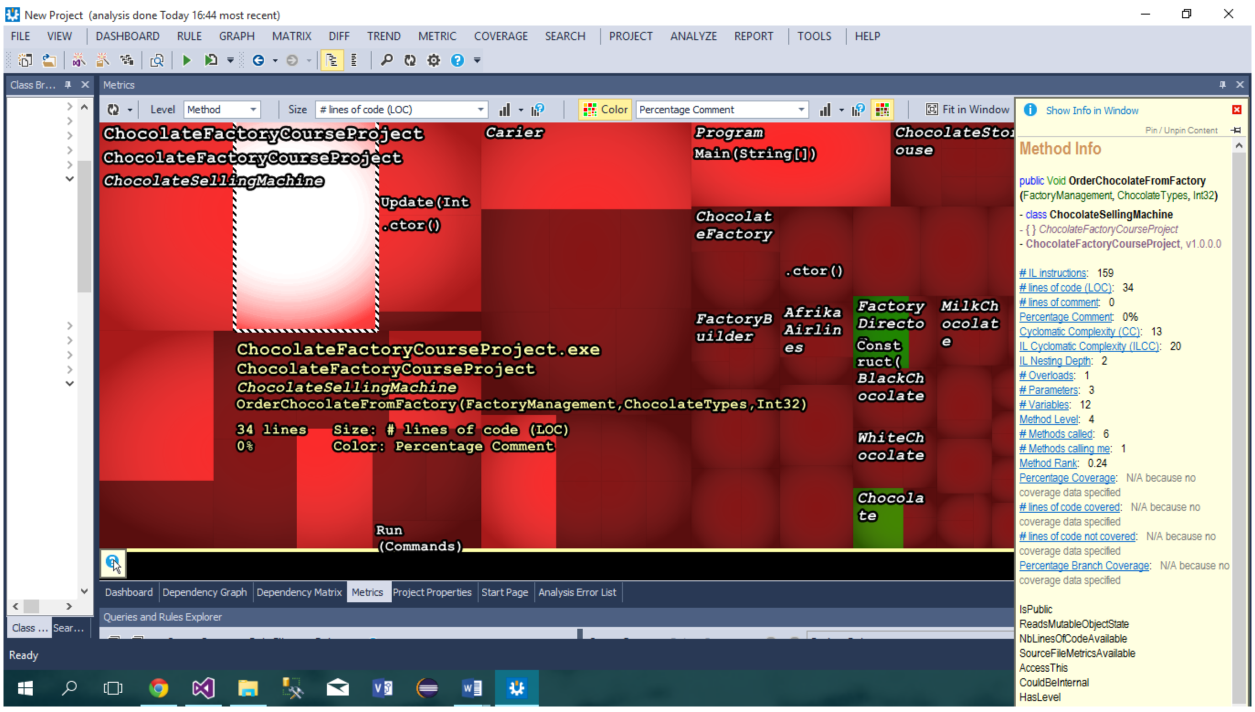
Method Info

public List<T> returnWhiteNutsChocolate (Int32)
- class Carier
- { } ChocolateFactoryCourseProject
- ChocolateFactoryCourseProject, v1.0.0.0

IL instructions: 39
lines of code (LOC): 7
lines of comment: 0
Percentage Comment: 0%
Cyclomatic Complexity (CC): 2
IL Cyclomatic Complexity (ILCC): 3
IL Nesting Depth: 1
Overloads: 1
Parameters: 1
Variables: 5
Method Level: 2
Methods called: 5
Methods calling me: 1
Method Rank: 0.22
Percentage Coverage: N/A because no coverage data specified
# lines of code covered: N/A because no coverage data specified
# lines of code not covered: N/A because no coverage data specified
Percentage Branch Coverage: N/A because no coverage data specified

IsPublic
ReadsMutableObjectState
NbLinesOfCodeAvailable
SourceFileMetricsAvailable
AccessThis

Several ways are proposed to identify code elements shown in the Metric View. When you hover the treemap with the mouse, the pointed code element rectangle gets highlighted, and its name and containing parents' names, are shown. The code element Size and Color metrics values are also shown.


New Project (analysis done Today 16:44 most recent)
FILE VIEW DASHBOARD RULE GRAPH MATRIX DIFF TREND METRIC COVERAGE SEARCH PROJECT ANALYZE REPORT TOOLS HELP
Class Br…
Metrics
Level Method
Size # lines of code (LOC)
Color Percentage Comment
Fit in Window
ChocolateFactoryCourseProject
ChocolateFactoryCourseProject
ChocolateSellingMachine
Carier
Program
Main(String[])
ChocolateStorehouse
Update(Int
.ctor()
ChocolateFactory
.ctor()
FactoryBuilder
AfrikaAirlines
FactoryDirector
Construct(
MilkChocolate
BlackChocolate
WhiteChocolate
Chocolate
ChocolateFactoryCourseProject.exe
ChocolateFactoryCourseProject
ChocolateSellingMachine
OrderChocolateFromFactory(FactoryManagement,ChocolateTypes,Int32)
34 lines
0%
Size: # lines of code (LOC)
Color: Percentage Comment
Run
(Commands)
Dashboard Dependency Graph Dependency Matrix Metrics Project Properties Start Page Analysis Error List
Queries and Rules Explorer
Class … Sear…
Ready
Show Info in Window
Pin / Unpin Content

Method Info

public Void OrderChocolateFromFactory (FactoryManagement, ChocolateTypes, Int32)
- class ChocolateSellingMachine
- { } ChocolateFactoryCourseProject
- ChocolateFactoryCourseProject, v1.0.0.0

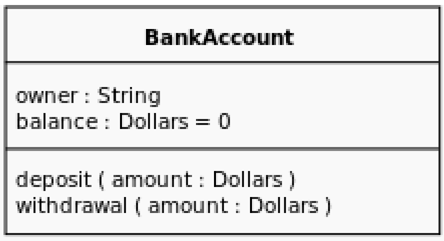
IL instructions: 159
lines of code (LOC): 34
lines of comment: 0
Percentage Comment: 0%
Cyclomatic Complexity (CC): 13
IL Cyclomatic Complexity (ILCC): 20
IL Nesting Depth: 2
Overloads: 1
Parameters: 3
Variables: 12
Method Level: 4
Methods called: 6
Methods calling me: 1
Method Rank: 0.24
Percentage Coverage: N/A because no coverage data specified
# lines of code covered: N/A because no coverage data specified
# lines of code not covered: N/A because no coverage data specified
Percentage Branch Coverage: N/A because no coverage data specified

IsPublic
ReadsMutableObjectState
NbLinesOfCodeAvailable
SourceFileMetricsAvailable
AccessThis
CouldBeInternal
HasLevel

UML Class Diagrams

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models intoprogramming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.



A class with three sections.

In the diagram, classes are represented with boxes which contain three parts:

- The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behaviour of systems, these class diagrams can be complemented by a state diagram or UML state machine.

UML provides mechanisms to represent class members, such as attributes and methods, and additional information about them.

To specify the visibility of a class member (i.e., any attribute or method), these notations must be placed before the member's name:

+	Public
-	Private
#	Protected
/	Derived (can be combined with one of the others)
~	Package

My class diagram is created in Visual Studio. On the top you can see a name of a class, and attributes and operations (methods) below. There are such relations as inheritance, dependency and implementation.

Textual visualization: Sandcastle Help File Builder

This project is composed of two separate parts that work together: the Sandcastle tools and the Sandcastle Help File Builder. The Sandcastle tools are used to create help files for managed class libraries containing both conceptual and API reference topics. API reference topics are created by combining the XML comments that are embedded in your source code with the syntax and structure of the types which is acquired by reflecting against the associated .NET Framework assemblies. Conceptual topics are created by converting XML documents that you author containing Microsoft Assistance Markup Language (MAML).

The Sandcastle tools are command-line based and have no GUI front-end, project management features, or an automated build process.

The Sandcastle Help File Builder was created to fill in the gaps, provide the missing NDoc-like features that are used most often, and provide standalone GUI and command line based tools to build a help file in an automated fashion. A Visual Studio integration package is also available for it so that help projects can be created and managed entirely from within Visual Studio.

Sandcastle was originally created by Microsoft back in 2006. The last official release from Microsoft occurred in June 2010. Until October 2012, it was hosted at the Sandcastle project site on CodePlex. In October 2012, Microsoft officially declared that they were ceasing support and development of Sandcastle. The Sandcastle tools have been merged into the Sandcastle Help File Builder project and all future development and support for them will be handled at this project site. The Sandcastle tools themselves remain separate from and have no dependency on the help file builder. As such, they can be used in a standalone fashion with your own scripts and build tools if that is your preference.

If you are new to Sandcastle and the help file builder, see the topics in the Getting Started section to get familiar with it, set up your projects to produce XML comments, and create a help file project. Below you can see help documentation created by me for one of my laboratory works.

Documentation.ch
m