|  |
| --- |
| Clone Detective for Visual Studio |
| Scenarios |
| How Clone Detective can be used |

|  |
| --- |
| Immo Landwerth  5/3/2008 |

Contents

[Persona 3](#_Toc198129160)

[Installing Clone Detective 3](#_Toc198129161)

[Running the Clone Detective 3](#_Toc198129162)

[Importing a Clone Detective Report 5](#_Toc198129163)

[Finding Clones While Editing 7](#_Toc198129164)

[Looking for Clone Hot Spots 9](#_Toc198129165)

[Customizing Clone Detection 11](#_Toc198129166)

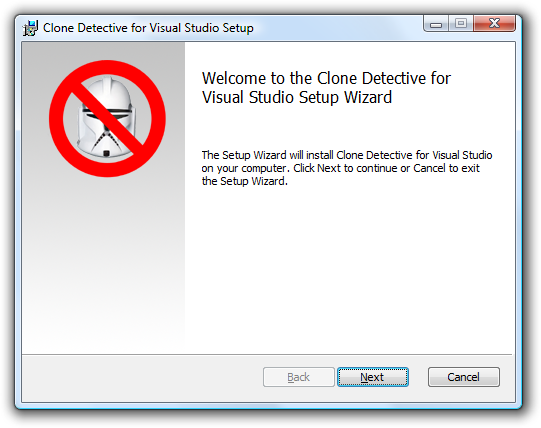
# Persona

Dave is the lead developer on the open source project “NQuery”. NQuery is a relational query engine written in C#. He is working for several years on this project and recently other developers joined the team. They will help him to add other features to the project, such as code completion for the query editor. Dave loves to write code that is correct, efficient, clear and elegant. He wonders whether the other team members will eventually share his goal of writing excellent code.

Because Dave is continuously looking for good development tools he recently found ConQAT and Clone Detective for Visual Studio on the web. He hopes that these tools help him to keep the quality of NQuery as high as possible.

# Installing Clone Detective

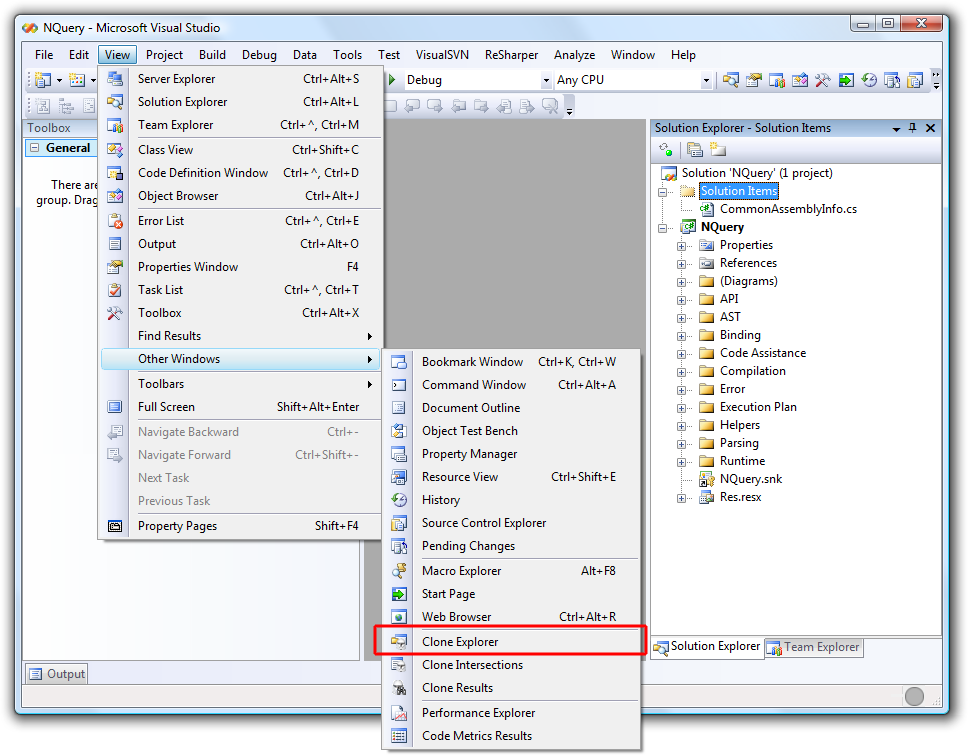
Dave wants to give Clone Detective for Visual Studio just a quick try so he downloads the latest setup from <http://www.codeplex.com/CloneDetectiveVS>. After double clicking the file CloneDetectiveVS.msi he sees a typical Windows setup so he quickly clicks trough it.



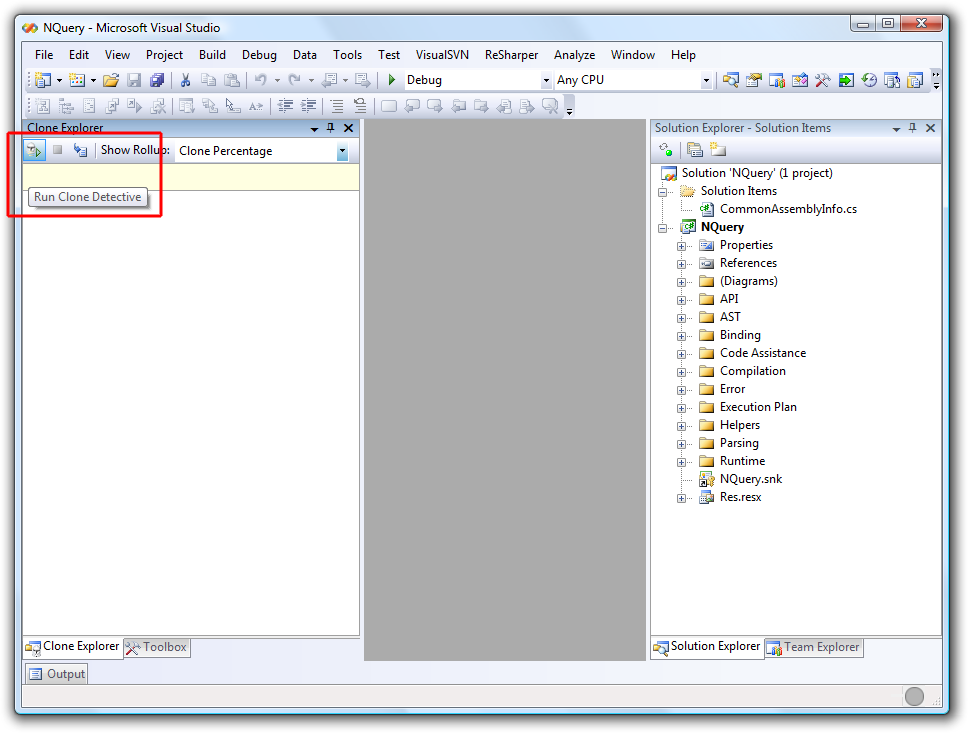
After the installation is complete Dave immediately starts Visual Studio 2008 to see Clone Detective in action.

# Running the Clone Detective

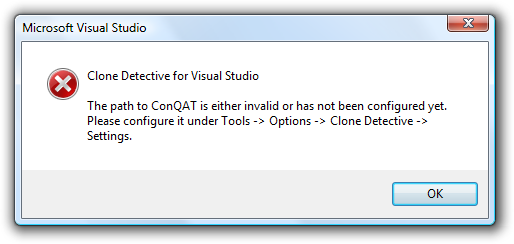
Dave wants to know how many lines of code are cloned in his project. To achieve this, he opens the Clone Explorer using the menu command View – Other Windows – Clone Explorer.



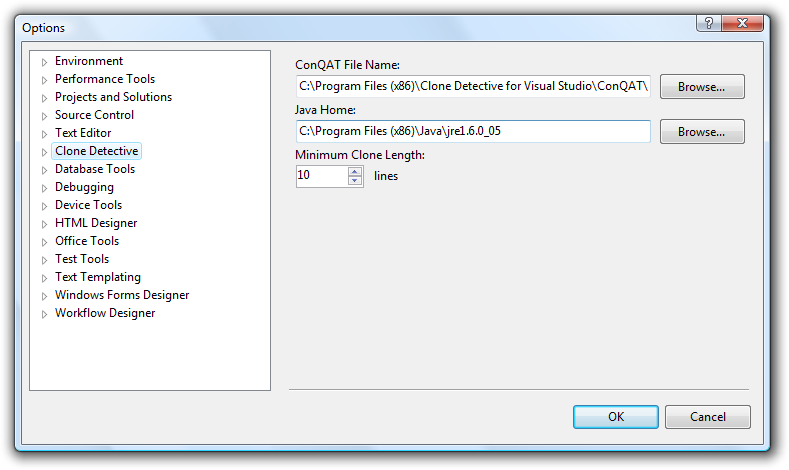
Since he just wants to quickly analyze his project he directly starts the Clone Detective using the toolbar button.



Then he is a bit annoyed by the now popping up message box:



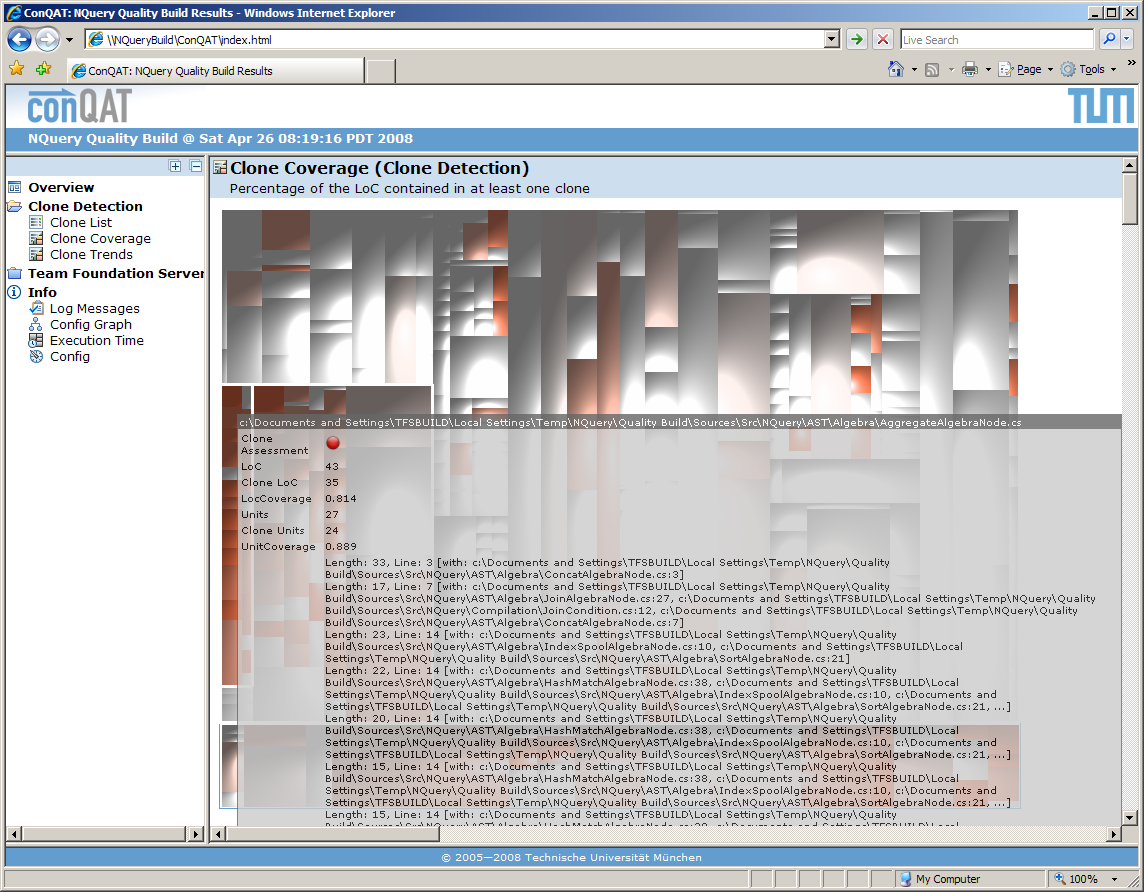
“OK,” he thinks. “Must be one of those silly first-time-setup-things. Anyway, let’s configure it now.” He opens up the settings of Clone Detective for Visual Studio under Tools – Options and enters the path to the batch file ConQAT.bat.



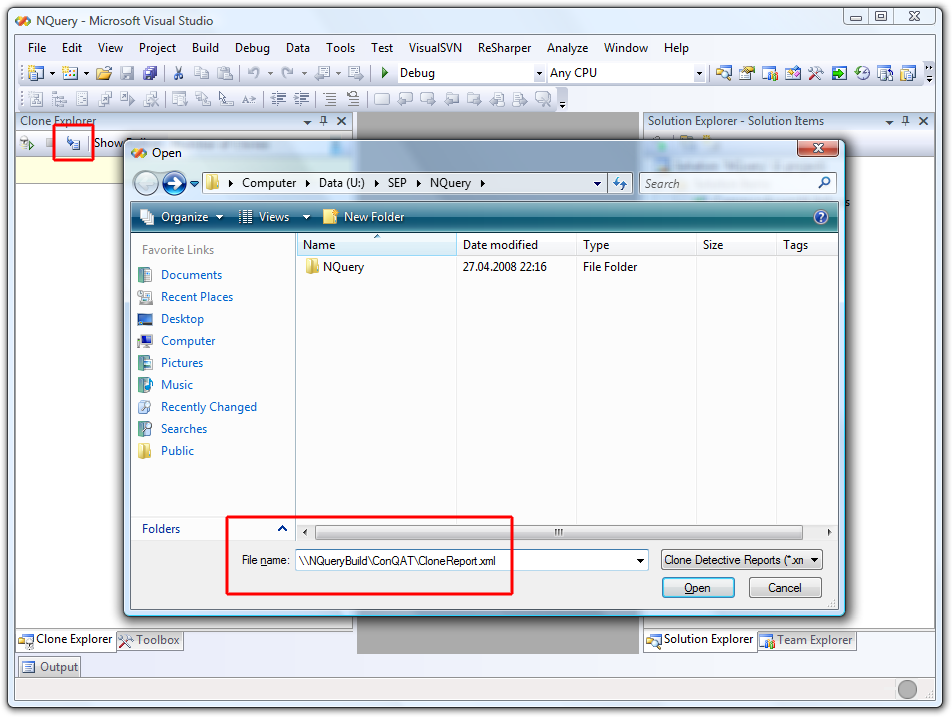
He tries again to start the Clone Detective and now it works as expected. “Wow,” he mumbles. “It even runs in a background thread so it does not block the UI. Nice!”

# Importing a Clone Detective Report

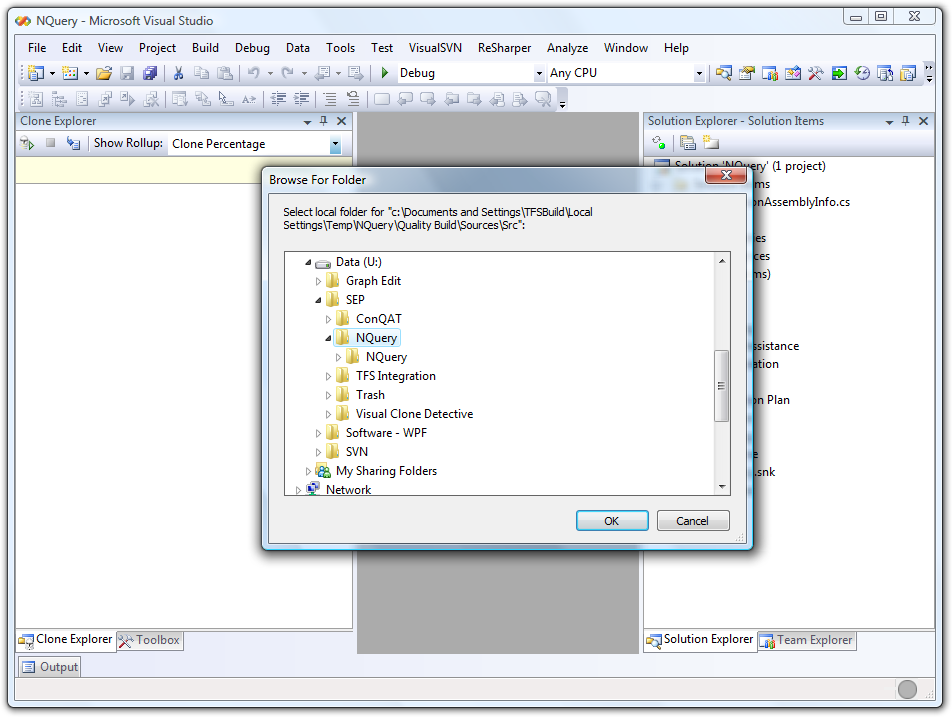
Recently, Dave had setup a build server for his team. Every night the build server performs an automatic build also running some quality assessment tools like FxCop and ConQAT. This morning he looks up the latest news on the ConQAT web site.



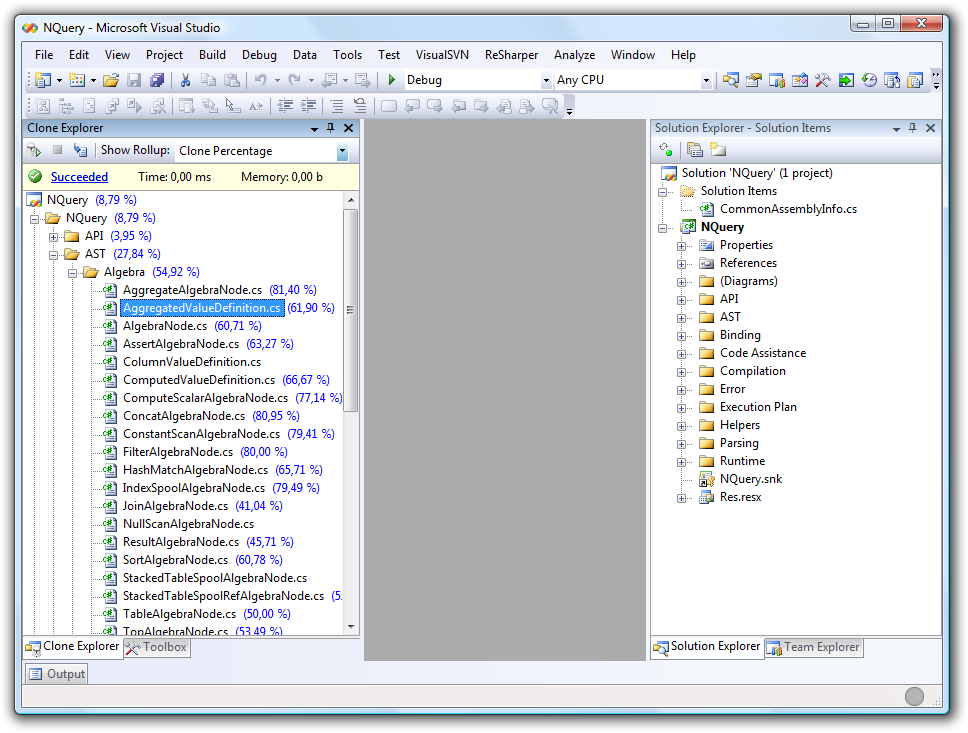
Since the file names on the build server are very long Dave cannot see the forest for the trees. He decides to browse the clone results on his dev machine using Clone Detective for Visual Studio. To do this he first opens the NQuery solution from his local disk and then he imports the clone report from the build server’s drop folder.



Since the paths in the clone report refer to files on the build server Clone Detective for Visual Studio asks Dave to specify the corresponding root path on his dev box.

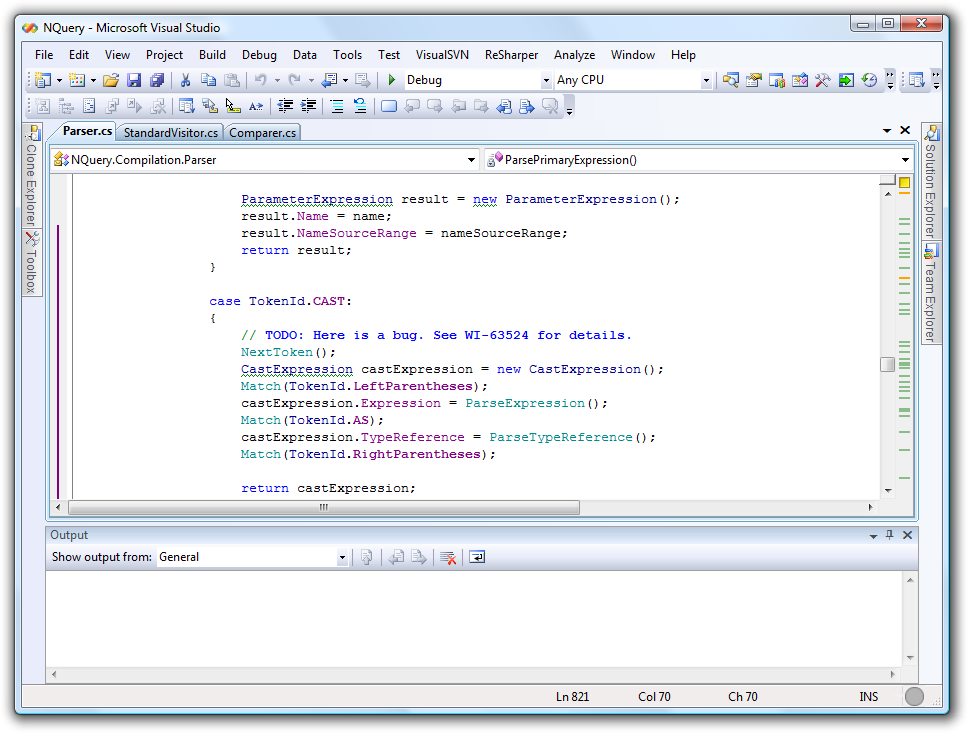


Now Dave can conveniently analyze the clone report in Visual Studio.

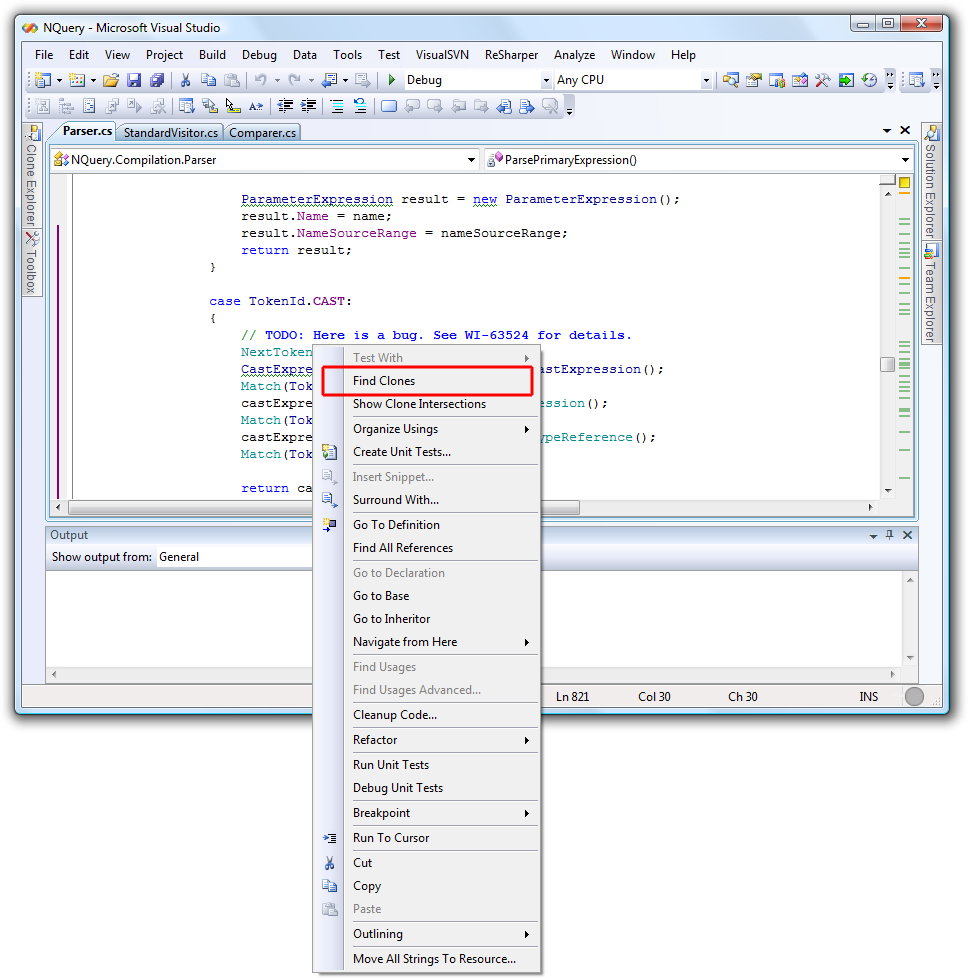


# Finding Clones While Editing

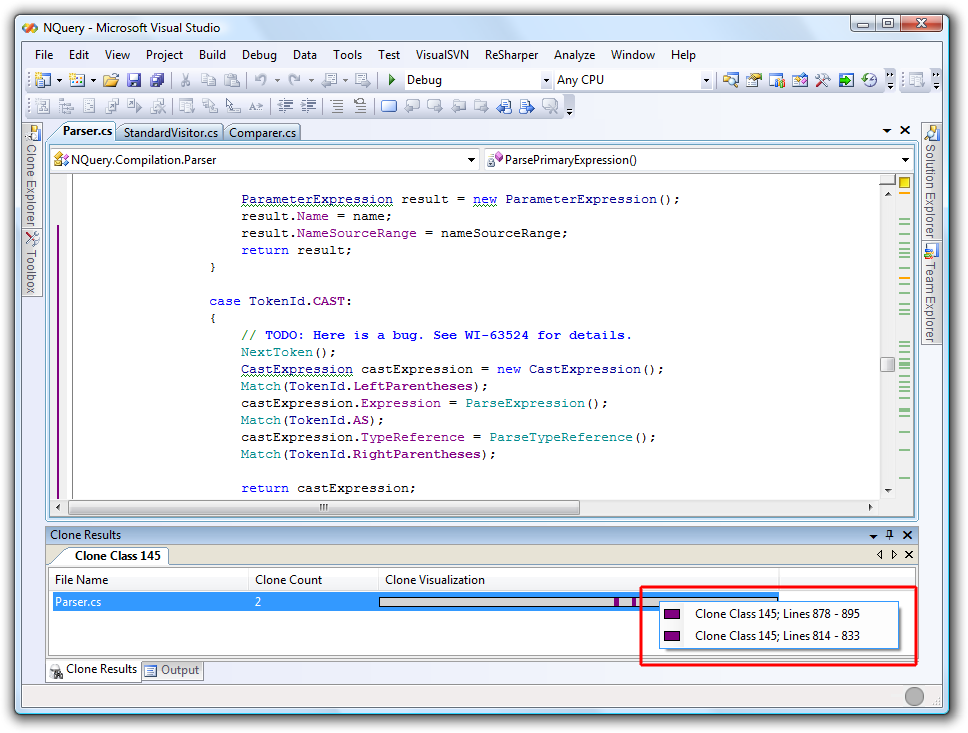
Dave is currently hunting down an ugly bug in the parsing component of NQuery. He has isolated the offending piece of code. Since he likes working in a structured way he first writes a comment. After leaning back for a while and thinking about how he could fix this issue he notices the purple colored vertical bar in the margin of the text editor. As he remembers this indicates a piece of code that is cloned somewhere else.



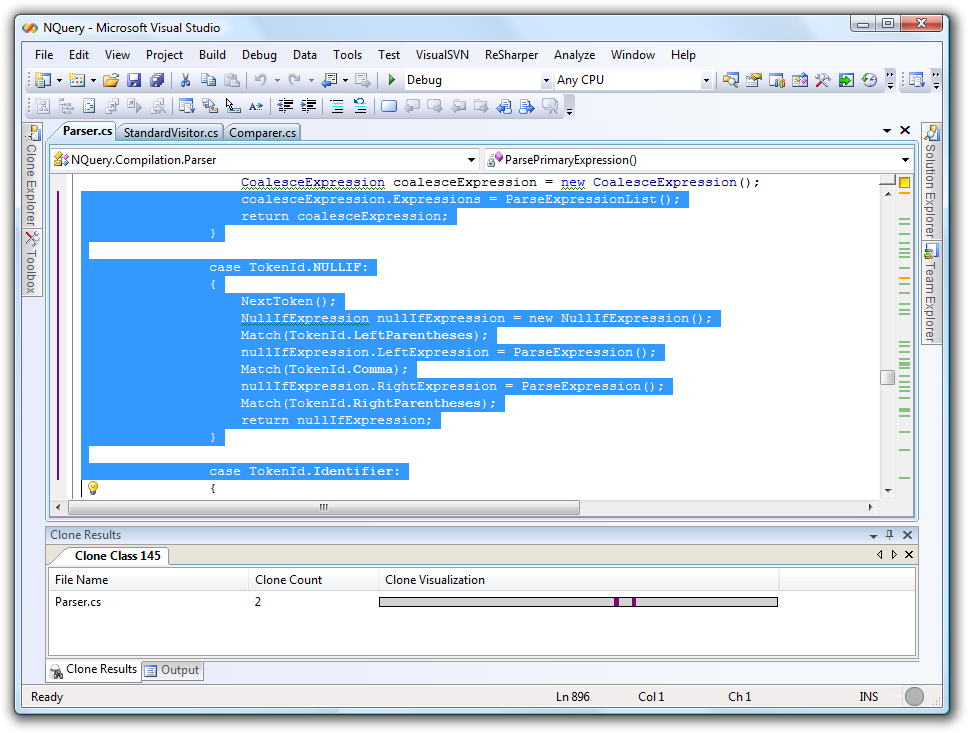
“Hm,” he wonders. “Where else did I wrote code similar to this?”. Using the Find Clones command from the context menu he quickly finds the other occurrence.



This time the clone was even in the same file. Using the context menu he is able to navigate directly to the other occurrence.

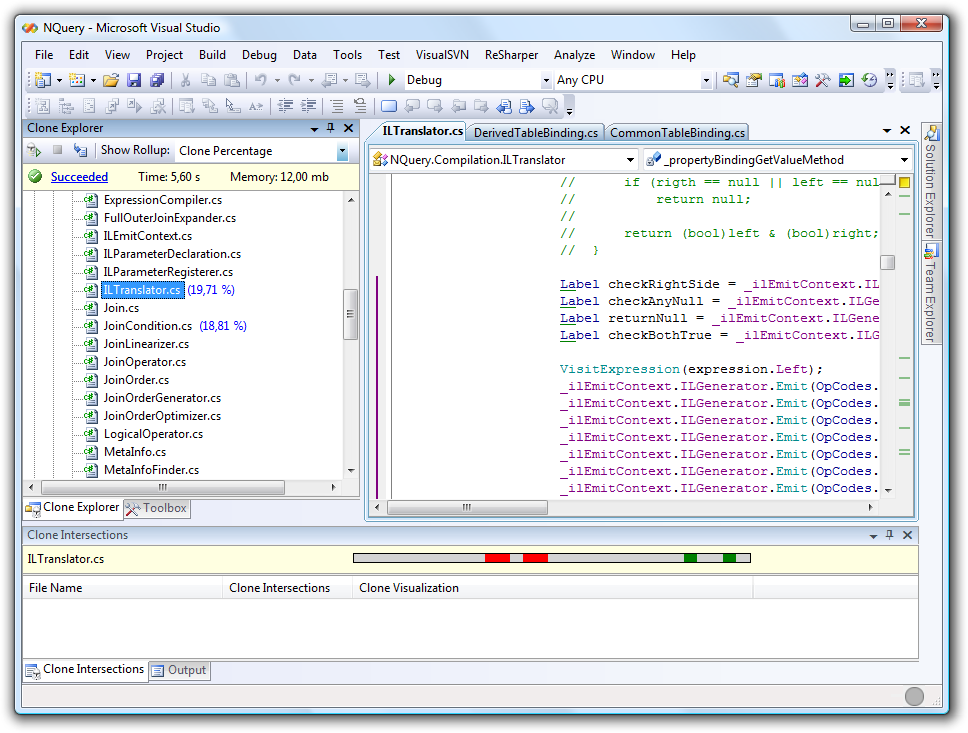


After a quick investigation of the other occurrence Dave realizes that the clone is only syntactically similar – but the code performs something completely different. Dave breathes a sigh of relief and continues thinking about how he could fix the bug.



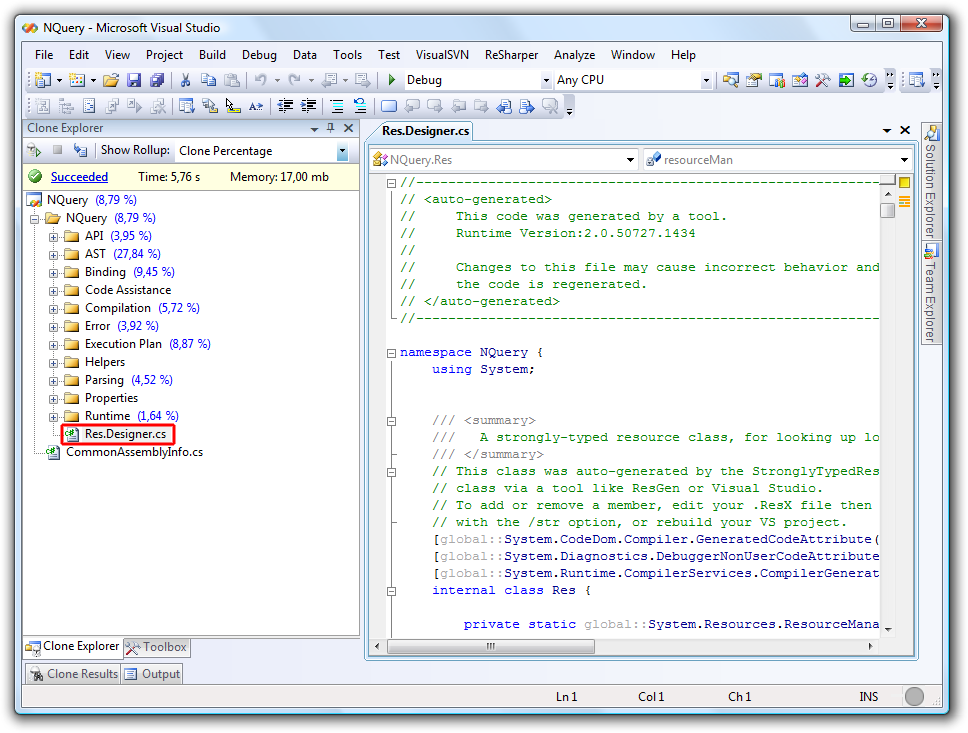
# Looking for Clone Hot Spots

Dave has just finished a complex feature in NQuery. Expressions are now compiled to Microsoft Intermediate Language instead of being interpreted. For this feature he had to do some work from the ground up as well as heavily refactoring some of the existing code. Since the unit tests are all green (finally!) he now wonders if he could improve the quality of his code. He starts the clone detective and uses the “Clone Explorer” and “Clone Intersections” tool windows to see whether he has introduced some clones. He quickly finds out that one of the newly created files ILTranslator.cs has a pretty high percentage of clones and that they are only caused by two clone classes within the same file. A double click on the clone visualization reveals that the clones are large pieces of code emitting IL. Now Dave decides to check whether he can solve the problem by introducing some helper methods.



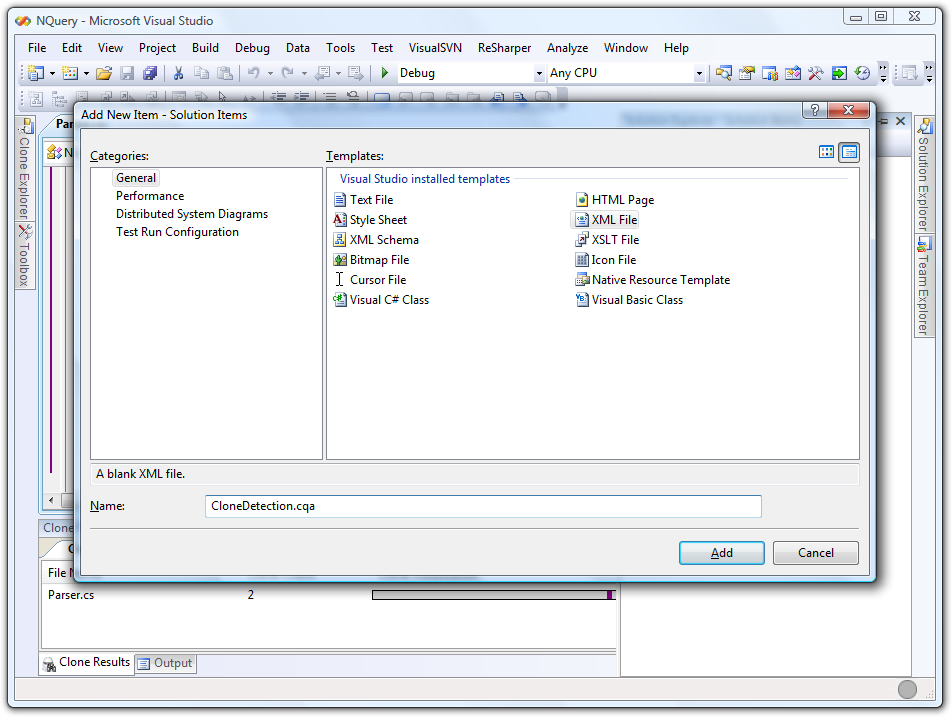
# Customizing Clone Detection

While browsing the clone report using Clone Explorer Dave discovers that designer generated files are included in the clone report. Dave would like to exclude these files.

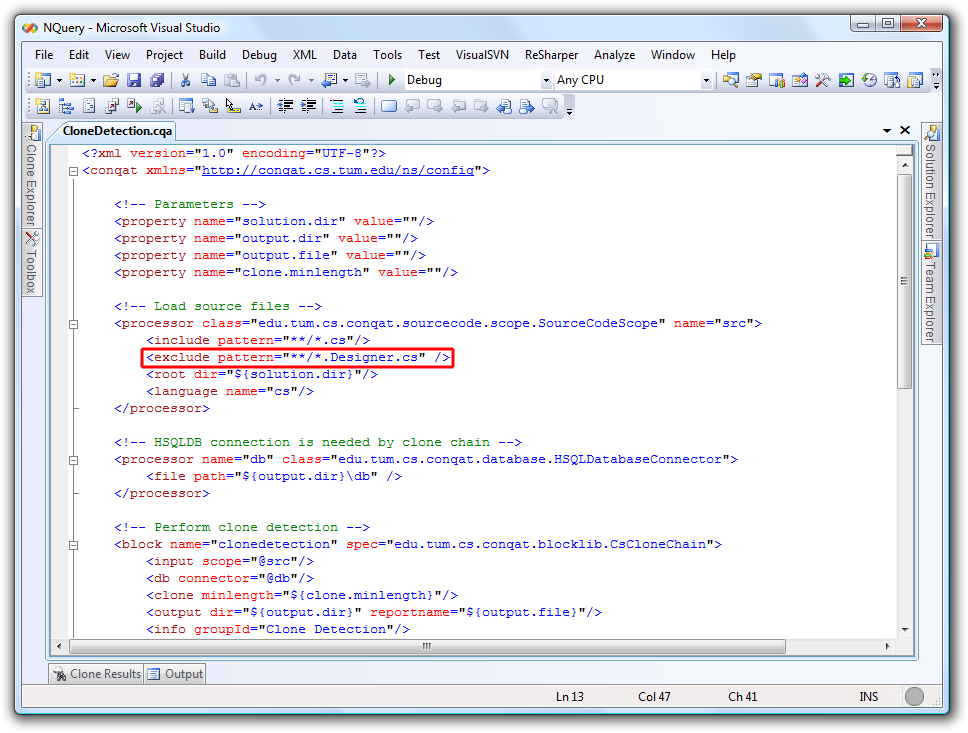


Dave knows that he can tweak the clone detection analysis a bit by using the Tools – Options – Clone Detective – Settings options page. However, the only thing that he can change there is the minimal clone length expressed as lines of code. More advanced changes such as excluding files must be done by using a custom clone analysis. Therefore he has to copy the default clone detection file to his solution and make some changes to it.

To do this he creates a new XML file by right clicking the NQuery solution and selecting Add – New Item – XML File. He names the file CloneDetection.cqa as this is the required name. Then he opens the default clone detection analysis by using File – Open – File and selects the file C:\Program Files\ Clone Detective for Visual Studio\DefaultCloneDetection.cqa. He copies the contents to the clipboard and pastes it in the newly added file CloneDetection.cqa.



He adds an exclude to the SourceCodeScope so that all designer generated files are ignored by Clone Detective.



After running Clone Detective again using the toolbar in Clone Explorer Dave is pleased to see that the file Res.Designer.cs is now missing in Clone Explorer, indicating Clone Detective has in fact ignored it.

