

Decompiler Debate



The following commentary is from the flurry of e-mails that were posted on the .NETDJ Web site (www.sys-con.com/dotnet/) list regarding the article entitled "Decompiler Roundup" (.NETDJ, Vol. 2, issue 8).

This article is bizarre. The author makes a lot of effort to downplay .NET Reflector, which is the tool everybody on the street is using (it was rated one of the top 10 developer tools). Next, he is comparing more expensive tools but only along the criteria that make Decompiler.NET look good. The code output example isn't correct or maybe just outdated (again it makes Decompiler.net look good). The comment "how each could handle pointers, because they are my favorite aspect of programming" makes this way too obvious.

—Mario

I was pleased to see the author's unbiased review confirming his positive experiences using our product. I've done my best to send bug reports regarding code generation issues to each of my competitors regarding their products, including the bugs exposed by the examples the author has chosen in this article. I reported most of these bugs to the authors of the products mentioned here including Reflector 4.0 back as early as 04/2004. The newsgroup discussions in which I had participated regarding code generation accuracy occurred prior to this article being written and I personally made all of the vendors mentioned in the article aware of code generation problems in their products. Each vendor had ample time to address these code generation issues prior to this article being written, and their response time is a good measure of the level of support that they provide for their products.

The author just asks each developer to take the time to evaluate each product on his/her own. His conclu-

sions are still accurate, even with newer versions of each of the products available that he reviewed. Decompiler.NET still generates higher-level and more accurate code than the other tools reviewed by the author in this article, and I'm aware of several additional code generation problems that still exist in current versions of the products mentioned here that cause them to continue to generate code that won't compile or run correctly. The author's conclusions are still 100% accurate and each developer can confirm this by downloading the trial versions of each of the products as the author recommends in his article. Please download a free trial version of our product yourself and confirm that it is still the best alternative for your needs at www.jungle-creatures.com.

—Jonathan Pierce

The author is not unbiased. Just read this:

"I chose to leave these two tools out of this article. Why? Well, for the same reason I don't do my own electrical work."

"They do not offer the needed level of support."

[Three times in a row. Apparently they do, his example runs fine in the current version.]

"Decompiler.NET with obfuscator is available for \$500 per CPU, giving it leverage over its toughest competitor" [Please click here to purchase without thinking]

—Anonymous

Reflector is an excellent tool that I use for casual browsing and code generation comparison. However, there are still many instances where Decompiler.NET generates higher-level and more accurate code, which is important to most professional developers. The current version of Reflector was released after this article was published and addresses these specific bugs over 4 months after they

Continued on page 49

Send us your feedback. Letters may be edited for length and clarity. Please provide full name, location, and, if applicable, title, and company.

▶ dndj@sys-con.com



CLIENT



SERVER



EVERYWHERE

President and CEO

Fuat Kircaali fuat@sys-con.com

President, SYS-CON Events

Grisha Davida grisha@sys-con.com

Group Publisher

Jeremy Geelan jeremy@sys-con.com

ADVERTISING

Senior Vice President, Sales and Marketing

Carmen Gonzalez carmen@sys-con.com

Vice President, Sales and Marketing

Miles Silverman miles@sys-con.com

Advertising Director

Robyn Forma robyn@sys-con.com

Advertising Sales Manager

Megan Mussa megan@sys-con.com

Associate Sales Managers

Kristin Kuhnle kristin@sys-con.com

Beth Jones beth@sys-con.com

Dorothy Gil dorothy@sys-con.com

PRODUCTION

Production Consultant

Jim Morgan jim@sys-con.com

Lead Designer

Richard Silverberg richards@sys-con.com

Art Director

Alex Botero alex@sys-con.com

Associate Art Directors

Louis F. Cuffari louis@sys-con.com

Tami Beatty tami@sys-con.com

Andrea Boden andrea@sys-con.com

WEB SERVICES

Vice President, Information Systems

Robert Diamond robert@sys-con.com

Web Designers

Stephen Kilmurray stephen@sys-con.com

Matthew Pollotta matthew@sys-con.com

ACCOUNTING

Accounts Receivable

Shannon Rymysza shannon@sys-con.com

Financial Analyst

Joan LaRose joan@sys-con.com

Accounts Payable

Betty White betty@sys-con.com

SYS-CON EVENTS

President, SYS-CON Events

Grisha Davida grisha@sys-con.com

Conference Manager

Lin Goetz lin@sys-con.com

SUBSCRIPTIONS

201 802-3012

888 303-5282

subscribe@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinators

Edna Earle Russell edna@sys-con.com

Linda Lipton linda@sys-con.com

JDJ Store Manager

Brunilda Staropoli bruni@sys-con.com



all fields in constructor methods and providing only property getters and/or methods that retrieve data from the object, without any mutator logic whatsoever. Many classes in the .NET Common Type System are immutable: System.String, System.Drawing.Font, etc. In addition, care should be taken that any values returned from property accessors, etc. are immutable as well. Otherwise, this data may be copied to insure the integrity of the object itself. Example 11 shows the performance benefit of immutable objects over synchronization.

Data Copying

This flies in the face of the advice given earlier, to minimize the use of objects. However, it's really the other side of the coin from immutable objects. Object copying allows you to use the data in a non-immutable object, but in a way that still completely avoids synchronization. The more highly multithreaded the environment, the more strategies like this make sense.

Read-Write Locks

Synchronization issues in managed code mirror those in databases. In some situations, optimistic concurrency strategies can be used; in some dirty reads are acceptable, etc. For situations in which a structure is seldom updated and often read, the ReaderWriterLock class can give significant performance benefits over simple synchronization. It allows either multiple read access or single write access at once. Example 12 compares ReaderWriterLock to simple synchronization in a read-heavy scenario.

Minimizing Synchronized Blocks

The use of the [MethodImpl(MethodImplOptions.Synchronized)] attribute should be avoided, as it always locks an entire method and is also non-standard C# usage. Instead, the lock keyword or one of the System.Threading classes should be used. Wherever possible, adjust the start of a synchronized section forward and the end backward.

Do whatever you can to decrease the number of synchronized operations.

Suggested Reading

- Lidin, Serge. (2002). *Inside Microsoft .NET IL Assembler*. Microsoft Press.
- Archer, Tom, and Whitechapel, Andrew. (2002). *Inside C#, Second Edition*. Microsoft Press.
- Rico Mariani's Weblog: <http://blogs.msdn.com/ricom>
- *Garbage Collector Basics and Performance Hints*: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/dotnetgcbasics.asp>
- *Improving .NET Application Performance and Scalability*: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenet.asp>
- *An Introduction to C# Generics*: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/csharp_generics.asp
- *WikiWikiWeb*: <http://c2.com/cgi/wiki?UniformlySlowCode> ☉

Continued from page 8

were reported to the author. There are many other code generation bugs in the current versions of the tools mentioned here that were not covered by the article.

The author also made the point that Decompiler.NET was the only tool that generated code that had correct compile and runtime behavior for all of the test cases that he attempted including his Hypersonic example. This is not the case for any of the tools that he evaluated, including the current release of Reflector 4.0.18. I've sent several bug reports to Reflector's author that still exist in his current release regarding many code generation issues not covered by this article.

~Jonathan Pierce

Did the author of this article report those bugs to the tool authors or did you tell him what to write about?

~Anonymous

The author of this article contacted each of the vendors on his own and requested permission to include their product in his evaluation. The article was written entirely by the author including his code examples chosen by him based on his own experiences with each of the products he tested for his own needs. He was trying to identify unusual test cases to measure the robustness of all of the products included in his evaluation. From his conclusions, it appears that he was unable to identify any cases where Decompiler.NET

didn't generate correct code to meet his needs. There are many other cases where the other products fail, so I'm not surprised that the author was able to identify some common cases that affected him personally.

I have always encouraged developers to try each of the products themselves and form their own opinions about their accuracy. This article seems to point them in the correct direction so that they can confirm its conclusions themselves before purchasing any of the products mentioned.

~Jonathan Pierce

Okay, let's leave it that way. The article has too many coincidences and you are too well informed on what the author did and didn't do to make it believable. All the tools mentioned are very nice and no harm done if everybody is doing his own evaluation...

~Anonymous

The author isn't unbiased. The scoring card and the careful phrasing to ditch Spices and Salamander shows this. Also, he is leaving half the tools (Anakrino, Reflector, LSW) out, only giving a "doing his own electrical work" argument as a rationale so it isn't a good overview in the first place.

~Anonymous

Do you have an opinion about this article? If you want to put in your two cents, please visit www.sys-con.com/dotnet/.