

# Re: Strategic Functional Migration and Multiple Inheritance

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2006-06/msg00638.html>

---

- *From:* Shawnk <[Shawnk@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:Shawnk@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Mon, 5 Jun 2006 06:29:01 -0700
- 

Nicholas,

Thank you for your response.

The numbers in points [1] and [2] are completely contrived.

We wanted to articulate the difference between (1) understanding and (2) use of II (Implementation Inheritance). We also wanted to establish a metric level for change (by evolution, user migration to other languages, user loyalty (as in C++)) that would account for the lack of MI (Multiple Inheritance) in recent language offerings (java, C#, D++).

The metrics of 'less than 1%' seemed to be a reasonable way to account for the apparent 'level of interest' (lack of interest) in MI/SFM among 'expert' and 'senior' level programmers in the non-C++ programming communities.

As to 'why MI is not supported' – the lack of interest or the response to this post is an indication that verifies (informally) the 'less than 1%' water mark. And yes, ... it appears that the II, MI and SFM are not used/understood/desired in the non-C++ programming communities :-)

We wanted to post this question in several language communities prior to any other actions (such as recommendations, etc). Also, making recommendations would be distracting to an audience/community that has no interest/understanding/desire of the utility of MI/II/SFM.

Your good point about a recommended design implementation for MI in C++ (in my mind) is really hinged/connected to Anders position of 'no MI in C#'. Since all the videos, interviews, etc. (That I have seen) indicate he has NO interest in MI/II/SFM it is reasonable to assume that C# will always be a less expressive (relative to functional orthogonality) and powerful language compared to C++.

Note that C# is an excellent language and wonderfully expressive from a productivity standpoint (type safety, etc). Its current contribution via LINQ is excellent as its historic introspective (reflection) and binding

Re: Strategic Functional Migration and Multiple Inheritance

(delegates,events) mechanisms. As a mechanism for automated pattern detection and refactoring it fails (IMHO) because of the importance of functionally orthogonal mechanisms (such as MI/II/SFM, etc).

The lack of use of .NET in the Microsoft Vista implementation is something of interest (to the MI/II/SFM issues) that we will pursue (as opposed to C# recommendations) to get a feel for MI/II/SFM issues in various programming communities.

Thanks again for your input.

Shawnk

PS. The lack of MI/II/SFM in .NET is a basic, serious and fundamental flaw (IMHO) of the 'operating system' (as in .NET) itself. This, of course, hinges on the validity of point [3] which is the basis for Ander's position on MI in C#.

"Nicholas Paldino [.NET/C# MVP]" wrote:

I would be interested in knowing how you came to the numbers that you have in points 1 and 2.

While I agree that implementation inheritance would be useful, if the numbers that you claim are anywhere near correct, you have already answered why MI is not supported in modern languages (meaning most .NET languages, Java).

In all of this, I didn't see a recommendation on how such functionality might be included in C# (proposed language syntax, rules, etc, etc). Why not show something of that nature?

—  
– Nicholas Paldino [.NET/C# MVP]  
– mvp@xx

"Shawnk" <Shawnk@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message [news:37B632C2-66E7-433A-A449-27FA734BB33D@xxxxxxxxxxxxxxxxxxxxx](mailto:news:37B632C2-66E7-433A-A449-27FA734BB33D@xxxxxxxxxxxxxxxxxxxxx)

Some Sr. colleges and I have had an on going discussion relative to when and if C# will ever support 'true' multiple inheritance.

Relevant to this, I wanted to query the C# community (the 'target' programming community herein) to get some community input and verify (or not) the following

## Re: Strategic Functional Migration and Multiple Inheritance

two statements.

[1] Few programmers (3 to 7%) UNDERSTAND 'Strategic Functional Migration (SFM)' (see PS below).

[2] Of those few, even less (another 3 to 7%) are GOOD at Strategic Functional Migration (or 0.9% to 0.49%).

Do these percentages seem about right? (less than 1% of the target programming community are GOOD at SFM)

Thanks ahead of time for any relevant QUALITY input.

Shawnk

PS.

Strategic Functional Migration (SFM) is described in the post following this one.

PPS. I submit to my fellow colleges that the answer (point spread) determines

[A] Short term mitosis of the few to pioneer a new language incorporating C# productivity with C++ (SFM) powers

[B] Long term community evolution and industry migration away from C# as a 'core competency' solution (subtle point)

Both A/B, in turn, instantiate the 'early adopter' model in the compiler market.

PPPS.

I have a 'core competency' project I want to do that would be a lot easier if I could use SFM with C#.