Should my C# .NET team migrate to Windows Presentation Foundation? [closed]

Asked 16 years, 3 months ago Modified 12 years, 8 months ago

Viewed 2k times

15

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 12 years ago.

We make infrastructure services (data retrieval and storage) and small smart client applications (fancy reporting mostly) for a commercial bank. Our team is large, 40 odd contractual employees that are C# .NET programmers. We support 50 odd applications and systems that we have developed.

A few members of the team began making WPF, WF and WCF based applications. Given that they are the first, most members do not understand these technologies.

What benefits do they convey that would overcome the cost of retraining the team?

.net wpf

Share

Improve this question

Follow

edited Apr 5, 2012 at 13:07

casperOne

74.5k ● 19 ● 188 ● 259

asked Sep 16, 2008 at 20:21

Robert

9 Answers

Sorted by: Highest score (default)

23

We are just wrapping up a project in which myself and 4 others developed a rather successful, distributed enterprise application. We started using Win32 and then switched to WPF after the first iteration to meet the demands of our usability expert. Here is my experience.

WPF has some really, really great features. In general, it makes the really hard things trivial (such as creating listboxes that show rich presentation data, such as images mixed with tables, copy, etc.), but in turn can make the "this used to be so easy in Win32" painfully frustrating. I've been working in WPF for 6 months now, and I still find databinding a combobox to an XML dataprovider a dreaded experience.

As I eluded to above, WPF has some great and not-so-great binding. I love how you can bind to an XML document or inline-fragment using XPath, but I hate how you can only use the built-in binding validations if your binding is two-way (and I doubly hate how you can't force the built-in binding validations to pass user input back to the object, even if the data falls outside the range of some business rule).

WPF has a huge learning curve. It's not even a curve - it's a wall. It's a rough go. It's a completely different way of working with Windows presentation, and, for me anyways, it required a lot of reading and playing before I started to feel somewhat comfortable. It's not the easiest thing in the world, but it allows you to do some incredibly powerful stuff (e.g. In our project I created a form engine that creates full fledged XAML forms from XML using about 300 lines of XSLT - complete with full binding and validation).

Overall, I'm extremely satisfied that we chose XAML, despite the learning curve, the somewhat buggy nature of it all, and some of the deep frustrations. The positives have far outweighed the negatives and it allowed us to do things I didn't think were possible without an enormously heavy hit to performance.

If you decide to go the route of WPF, I would highly recommend these 2 books:

- Windows Presentation Foundation Unleashed, by Adam Nathan is a great intro, with full colour! It reads

like a blog and gives you a great great intro - http://www.amazon.ca/Windows-Presentation-Foundation-Unleashed-WPF/dp/0672328917/ref=pd_ys_iyr3

- Programming WPF: Building Windows Ui with Windows Presentation Foundation, by Chris Sells. More detail and a great book to accompany the WPF Unleashed - http://www.amazon.co.uk/Programming-WPF-Building-Presentation-Foundation/dp/0596510373

Good luck!

Share Improve this answer

Follow

edited Jan 31, 2010 at 13:34

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 16, 2008 at 20:42

cranley

623 ● 1 ● 5 ● 10

1 I completely agree with the learning "wall." However, I disagree with the comments that certain things are inherently difficult in WPF; the items you feel that way about are just things you're still on the "wrong side of the wall" with.
– Sam Harwell Jan 31, 2010 at 13:29

15

WPF UI's are easier to design implement and maintain than the current C# alternatives, so if a lot of your codebase is responsible for handling UI, migrating may serve beneficial-- as in, you'll find your team will save

time in dealing with their UI layer. If most of your code is business logic, it won't help all that much.

Share Improve this answer Follow

edited Feb 5, 2010 at 4:48

answered Sep 16, 2008 at 20:24

Loren Segal

3,272 ● 1 ● 30 ● 29

5 I don't think that's the case at all. It seems easier to design and implement *simple* UIs in WinForms than WPF, and I say this with several months of WPF experience. The exception is that when you need something WinForms doesn't do out-of-the-box, you'll do a lot more work to get it. – PeterAllenWebb Oct 21, 2008 at 14:48

8 @PeterAllenWebb - This isn't true for me and VS2010. I'd had years of WinForms experience and once over the WPF learning hump, I find that I can knock out a WPF UI much faster, regardless of the complexity. It's mostly due to the ease of writing UI code declaratively in XAML and binding it to a C# view model, something which the years of ASP.Net made me fast at doing. – codekaizen Jan 27, 2010 at 18:44

12

WPF enables you to do some amazing things, and I LOVE it... but I always feel obligated to qualify my recommendations, whenever developers ask me whether I think they should be moving to the new technology.

Are your developers willing (preferably, EAGER) to spend the time it takes to learn to use WPF effectively? I never

would have thought to say this about MFC, or Windows Forms, or even unmanaged DirectX, but you probably do NOT want a team trying to "pick up" WPF over the course of a normal dev. cycle for a shipping product!

Do at least one or two of your developers have some design sensibilities, and do individuals with final design authority have a decent understanding of development issues, so you can leverage WPF capabilities to create something which is actually BETTER, instead of just more "colorful", featuring gratuitous animation?

Does some percentage of your target customer base run on integrated graphics chip sets that might not support the features you were planning -- or are they still running Windows 2000, which would eliminate them as customers altogether? Some people would also ask whether your customers actually CARE about enhanced visuals but, having lived through internal company "Our business customers don't care about colors and pictures" debates in the early 1990s, I know that well-designed solutions from your competitors will MAKE them care, and the real question is whether the conditions are right, to enable you to offer something that will make them care NOW.

Does the project involve grounds-up development, at least for the presentation layer, to avoid the additional complexity of trying to hook into incompatible legacy scaffolding (Interoperability with Windows Forms is NOT seamless)?

Can your manager accept (or be distracted from noticing) a significant DROP in developer productivity for four to six months?

This last issue is due to what I like to think of as the "FizzBin" nature of WPF, with ten different ways to implement any task, and no apparent reason to prefer one approach to another, and little guidance available to help you make a choice. Not only will the shortcomings of whatever choice you make become clear only much later in the project, but you are virtually guaranteed to have every developer on your project adopting a different approach, resulting in a major maintenance headache. Most frustrating of all are the inconsistencies that constantly trip you up, as you try to learn the framework.

You can find more in-depth WPF-related information in an entry on my blog:

http://missedmemo.com/blog/2008/09/13/WPFTheFizzBinAPI.aspx

Share Improve this answer

Follow

edited Jan 31, 2010 at 12:53

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 17, 2008 at 2:05

AndyL

1,106 ● 9 ● 5

WPF:

4

- Is all about graphics!
- Is a resolution independent framework (meaning - WPF has fully adopted the concept of vector graphics - and has also made the scaling of bitmap graphics a thoughtless process)
- Is hardware accelerated!!!! WPF graphics are hardware accelerated where possible through Direct3D - it is NOT GDI based!
- Has no `Paint()` function - WPF is based on a retained graphics mode / tree based drawing system. Finally!
- Is very graphically dynamic - everything can be animated - and animation is built into the framework. Remember.... there is no `Paint()`!
- Is extremely customizable - though getting into the nitty-gritty of `ControlTemplates` is where the complication begins. You simply add objects to the display tree and let WPF worry about updates.
- Is very rich in text rendering features.
- Is hopefully improving the designer/coder's workflow with the use of a declarative language (XAML) for graphic definitions and complex GUI design software (Expression Blend). Though it is important to realize that anything that can be done in a declarative way can also be accomplished in code. And it is also debatable that the complexity of WPF has scared away many designers - but it has put a powerful framework in the hands of coders.

WPF:

- Is NOT Windows Forms++ - it's just a totally different concept all-together
- Is NOT Silverlight - Silverlight is a subset of WPF. Quite a light subset.
- Is NOT MFC - OK, this should be obvious
- Is not easily distributable with Windows XP - this is a shame and maybe one of its biggest failures
- Is not XAML. This distinction needs to be understood. XAML is an **optional** declarative language which can be used in the development process of WPF applications. It is absolutely not a necessary component, though once understood, it definitely improves the workflow, design, and refactoring of complex graphical frameworks.

Share Improve this answer

Follow

edited Jan 31, 2010 at 12:39

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Feb 10, 2009 at 5:29

helifreak

1,303 ● 2 ● 11 ● 16

2

WPF is radically different from Windows Forms. This means a lot of training for your team.

Share Improve this answer

edited Jan 31, 2010 at 13:31

Follow

Sam Harwell
99.7k ● 22 ● 214 ● 282

answered Sep 17, 2008 at 10:54

Hallgrim
15.5k ● 11 ● 48 ● 54

0

I think the key word from your original question is "fancy". If your customers really expect a lot of glitter in the deliverable, then you probably do have something to gain from switching to WPF.

Share Improve this answer

Follow

answered Sep 16, 2008 at 20:28

Joel Coehoorn
415k ● 114 ● 577 ● 813

0

I was not sure at first, most applications seemed to be pretty laggy (granted, WinForms isn't lightning fast either). This seems to be fixed with .NET 3.5 SP1 where they integrated hardware acceleration for a number of techniques.

The integrated animation / storyboard / vector capabilities are very nice and a step into the right direction. If you get a grip of Expression Blend, you will be able to prototype apps pretty quickly. These are clear benefits in my opinion.

In the long run, I don't think WinForms and older techniques are a sustainable choice.

There's also Adobe Flex, Adobe/Macromedia have experience in more powerful and 'exciting' GUI solutions because of their experience with Flash.

I just hope we don't end up with 10 different VM's installed on a desktop pc just to run all those different frameworks...

re:

> fancy reporting

fanciness is probably one of WPF's strengths...

Share Improve this answer

Follow

answered Sep 16, 2008 at 20:35

kitsune

11.6k ● 15 ● 59 ● 78

0

WPF is a very fresh approach to designing UIs. The only problem is that it introduces a large amount of concepts, some of those only to hide the verboseness of XAML (XML). It also suffers a little from an architecture astronaut approach to the design but overall I'm pretty happy with it. It makes things that you before would have said no way to, into something that is manageable to do.

Share Improve this answer

Follow

edited Jan 31, 2010 at 12:57

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 16, 2008 at 21:45

Anders Rune Jensen
3,786 ● 2 ● 44 ● 54

0

WPF is the current "state of the art" in UI methodologies. Had it been available as people were learning to write UIs (instead of GDI, Win32, and later WinForms which is relatively similar), it wouldn't take so long to learn it. You can probably think of it like switching to a Dvorak keyboard - the most difficult part is changing your thinking about the parts of UI design you think you know well.

That said, you should at least be encouraging members of your team to experiment with WPF in their spare time. Make resources available from the very beginning, maybe by the following:

- Have links to pages that tell about what you need to have installed to work with WPF - if it doesn't mention Blend then I wouldn't trust it.
- Look on here for "getting started" questions, since they'll probably have good answers from experienced individuals.
- Buy at least a few good books and let people borrow them if they want.

Share Improve this answer

Follow

answered Jan 31, 2010 at 13:39

Sam Harwell
99.7k ● 22 ● 214 ● 282