Coping with rapidly changing technology (in particular Microsoft) [closed]

Asked 16 years ago Modified 12 years, 8 months ago Viewed 3k times

38

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.

Today as I booted up Visual Studio 2008 to do my work, I saw on the Start page a new blog post about Visual Studio 2010 and .NET 4.0.

I can't help but feel a little overwhelmed. I am just starting to come to grip with all the stuff Microsoft added in .NET 3.0 & 3.5 (LINQ, WPF, lambda functions, etc, etc).... I know these things are no longer new, especially for the folks here, but I've been too busy solving real problems for customers... albeit with "older" technologies like .NET 2.0, and haven't had time to continually keep up.

Sorry my question is non-technical but I think its very relevant for programmers, both employees and independent consultants, as well as software businesses (which I aspire to start one day): How to you cope with such rapid change, and still stay relevant? Do you ignore it until you need it or do you try to stay ahead? I'd love to stay ahead and stay relevant (read: keep my house and food on table), and not stuck working on legacy stuff, but it just seems like an endless treadmill. Especially with MS... it seems like they have so much money and people that they just keep pumping out new stuff... and worse, rendering previous technologies obsolete and unsupported (like VB6... something from my previous life).

.net

Share

Improve this question

Follow

edited Apr 5, 2012 at 13:41

casperOne

74.5k ● 19 ● 188 ● 259

asked Dec 18, 2008 at 18:19

Hamster

389 ● 3 ● 3

1 My advice is this: Change your perspective. You should be viewing Microsoft as a technology vendor, not as "the definitive source of what it means to be a good programmer". You should sample their API stack at whatever interval

makes most sense for you. ... – Scott Wisniewski Dec 19, 2008 at 0:18

2 ...If someone won't hire you because you sample the Microsoft API stack too infrequently, then they aren't basing their hiring decisions on the right criteria. You don't want to work there. A good dev can learn an API when he or she needs to use it. – Scott Wisniewski Dec 19, 2008 at 0:21

26 Answers

Sorted by: Highest score (default)

21

You don't need to be constantly updating your production work to use the latest and greatest technology. But you should stay knowledgable about what is out there and what is being created. There will come a time when it is appropriate to start using the new technology, but that all heavily depends on your projects and timelines.

But at some point you will need to move on. Certain technologies become unsupported. You should be aware of when this will happen, and plan the future of your production work to match.

Share Improve this answer Follow

answered Dec 18, 2008 at 18:24

Joseph Daigle
48.4k ● 10 ● 50 ● 73

+1, but the problem with MS is their older techs go by the wayside and fixes to problems are only available in NEWER versions that your production code most likely can't use... – user7116 Dec 18, 2008 at 18:30

19

<rant>
Personally, I wish Microsoft would **fix bugs first** before releasing any more new widgets, frameworks, or whizbang thingamajigs. There are known bugs in .NET still around from 1.1 and early 2.0 - bugs that are *five years old or older* that MS shows no inclination to fix. C'mon, MS, just take a year and fix things, then bring on the new stuff.
</rant>

On the upside, if you wait a year or two (or SP2, which ever is longer) before using MS's newest stuff, it tends to be pretty good.

EDIT: in the interest of actually providing an "answer" to the question, I tend to learn what I need as I go (Google is your friend!), and don't get too excited about the latest shiny widget from Microsoft or anyone else. Far too often it is just the same old stuff in a brand new package. When the rare gem appears its' hard to miss (buzz buzz goes the Internet!).

Share Improve this answer Follow

edited Dec 19, 2008 at 0:13

answered Dec 18, 2008 at 18:25

Steven A. Lowe
61.1k ● 19 ● 135 ● 204

if only MS would fix their bugs as fast as drive-by downvoters express their lack of a sense of humor... – Steven A. Lowe

Dec 18, 2008 at 18:29

+1, please make what I have now work well. If they didn't drop support so fast I'd be cool with .Net 4.0 etc. – user7116 Dec 18, 2008 at 18:30

1 -1. The question is not asking what MS should do, but what HE should do. – Micah Dec 18, 2008 at 18:37

Unfortunately, new features sell (and earn job promotions), talk about bugs doesn't. – dkretz Dec 18, 2008 at 18:39

I've had the same feeling for a while until I read a few of the MS developer blogs. They really do want to fix a lot of the bugs but most of the less severe bugs with workarounds cannot be fixed without breaking backwards compatibility. – joshperry Dec 18, 2008 at 18:41

13

Don't measure your relevance by what cool whizbang feature of the 10.8 release of your favorite framework that you were able to stuff into your code.

Your customers will measure your relevance by whether you solve their problems or not. They don't know if you used a ForEach instead of a Linq query, or a delegate instead of a lambda expression.

Share Improve this answer Follow

answered Dec 18, 2008 at 18:30

joshperry
42.2k ● 16 ● 92 ● 104

There is a broader picture, though. With the layoffs that are currently happening around the industry, if you make no room for self-improvement in this area, you find it even harder to get another job in the event you end up in this position. It is in everyone's interest to try and stay current. – Joseph Ferris Dec 18, 2008 at 18:36

I agree wholeheartedly and I have the same problem as the OP, I just wanted to provide a perspective I wasn't sure that everyone else was going to post. Also, when you are your own boss (as he aspires) you don't need to worry about the hiring interview going well. :) – joshperry Dec 18, 2008 at 22:33

10

As a manager, I actively solicit opportunities for my developers to use new technology. We are a .NET shop, but "keeping up with the Jones'" is a paradigm-shift from some of the older policies that we used to have. Until last year, we were still on the 1.1 Framework for everything. It was very painful, both in being limited by an older version of the Framework and by having to use Visual Studio 2003.

We started to align business requests with the functionality available in the Framework. After a few "it will take ten developer days to do with our current infrastructure, or you can invest five days in allowing us to address our Framework restrictions - then these kind of requests will take two days", we were not only encouraged to upgrade the Framework, but also to stay current and adopt features and functionality as it became available.

While Microsoft is not perfect, they do offer great availability into CTPs of their products now, allowing people to stay ahead of the game, given the time can be made within the organization to allow this. These kinds of products are great for inclusion in proof of concept projects, as well.

With the advances that are being made, it is very difficult to assimilate them all, regardless of the immediate time investment you can make. You need to determine what features, functionality, and tooling can provide the largest positive impact up front, and focus on that. You will be surprised how fast the other pieces start to just fall in place as you expand your usage. Lambda expressions are a great example of that. Working with LINQ provided a logical progression into them without it being the focus of what we were trying to implement.

Share Improve this answer Follow

answered Dec 18, 2008 at 18:33

Joseph Ferris
12.7k ● 3 ● 48 ● 73

+1. This is pretty much what we went through where I work, and when Vista came, we got burned for sticking with old tech (read Visual Studio 6.0 and InstallShield 5.5) for so long. Now, we're using Wise Install Studio to build our installers, and VS 2008. (We've actually upgraded both twice so far.) – RobH Dec 18, 2008 at 20:18

"Visual Studio 6.0 and InstallShield 5.5" Ouch, painful – Nifle Dec 18, 2008 at 21:17

I know how it is, I've got a lot of stuff in VB 6 and .NET 1.1. [sigh] – Booji Boy Dec 19, 2008 at 3:59

We still have some legacy COM+ in VB6 that we are trying to get rid of. It is painful, but not like it used to be.
– Joseph Ferris Dec 19, 2008 at 11:18

9

After nearly twenty years I feel pretty confident in saying it is an endless treadmill. Well, almost ...

The first thing you notice is that some technologies are more permanent than others. Generally that is the better designed stuff, but it's not always that consistent, sometimes the crap wins, at least for a while. Obviously it's a wise choice to learn the good stuff, and ignore the rest. If I tried to learn every technology that came along over the years my head would have burst, and so much of it wasn't worth learning.

The second thing you notice is that much of this stuff is recycled. It's just the same crap over and over again, only this time with a different name, and a newer sound bite. Ultimately, it turns out that there aren't that many ways to do things, and although it gets reintroduced every five years or so, the same basic concepts float to the top. I'd go into them, but it's not really worth it right now, I think virtually all of the leading technologies right now are really just spins on earlier versions that have existed over the last twenty years. The industry hasn't seen any real innovation in decades, it's just been turning out more stuff at a faster rate, that's all (OK, it's a little flashier too, but not much).

The third and most important point, as was told to me long ago: "None of this stuff really works!". And that is probably the best all around time saver. So much of what is out there might be cool and have potential, but the sad truth is that it doesn't do enough of what's advertised to make it usable for some real application. We're an industry of fakes. That's probably driven by the realization that it is really easy to produce a slick looking demo, but really hard to produce something that works correctly. And as we have littered the waters with more and more crap, our foundation has become weaker and weaker. Thus, it doesn't make sense to learn how to use something outside of its sweet spot. Most technologies do one thing well, and the rest is just hype. Figure out what that is, and remember it, but don't bother trying to use the technology to do something it can't do.

Over the years I've had to jump from different languages and different operating systems. My editing environment has de-evolved, and the tools that I use to automate the projects have gotten fancier and less trustworthy. The technologies change, but as we've already passed over some threshold, while they have more features they're no longer dependable, so in a weird way it has gotten slower, and easier. It used to be that we really had to understand how things worked, now you just slap the code together and then blame someone else when it acts weird.

Pick a few good technolgoies and ride the wave most of the way back to the beach. Every five years hop off, swim back out and do it again. With luck, you'll be in the right spots at the right time. With no luck, you'll end up switching to some other career mid-stream like most other ex-programmers.

Paul.

Share Improve this answer Follow

answered Dec 19, 2008 at 3:29

Paul W Homer
2,740 ● 1 ● 19 ● 25

8

Spike it! Regularly take a chunk of time and write a solution using the new technology. It doesn't have to be fancy or useful - just enough for you to learn the basics and when to use it in a real project.

Care about it. I disagree with some of the answers- it is very important to stay up on this stuff. If you are not interested in learning new technologies and constantly improving your skillset, maybe there's another line of work you *would* care more about. I don't mean that to be ugly, but by continuing in older technologies, you are not taking advantage of the fact that the problems you face have been solved. You're ignoring the work of thousands of people and the community. For example, if you write a new service using WebServices instead of WCF, you are not taking advantage of improvements in security- at best, you are wasting time and money by solving those problems again yourself. At worst, you're doing it wrong and are a risk to your company/client.

Know when to say no. There is a balancing act when it comes to updating old code. You can't just rewrite everything every time a new technology comes out. It will help though if you focus on writing quality, "future proof" code. For example, if you wrote a well structured WinForms app, it should be possible to re-use your logic in WPF. If you wrote a decent data layer, you should be able to drop in a linq data layer. That level of code is difficult to master, but refactoring tools, unit testing, and static analysis will help.

Share Improve this answer Follow

answered Dec 18, 2008 at 19:48
Daniel
1,516 ● 1 ● 13 ● 24

I was half way through saying the same thing! Friday afternoons after lunch are the best spike time there is.
– Jennifer Dec 18, 2008 at 19:52

7

I feel pretty much the same way. A lot of times I get worried that the technology is changing faster than I can master the current tool set. I always try to come up with some sort of project that I can apply the technology to. For instance, I've really wanted to get up-to-speed on MVC so I decided to write an app thats a cross between digg, stackoverflow, and dotnetkicks using the new framework. I've only been into it for 3 days and I've already learned a TON.

The difficult part about our job is that it's so hard to stay relevant. yet it's that challenge that makes it so exciting.

Share Improve this answer Follow

edited Dec 18, 2008 at 18:31

answered Dec 18, 2008 at 18:25

Micah
116k ● 87 ● 237 ● 331

4

On the personal side.

By learning good practices, design patterns, and principles that apply across all programming languages. It helps to have a through understanding of algorithm and various aspects of math

Be active in the programming community of your platform so you let them have it when they made gratuitous and breaking changes (hopefully BEFORE they actually implement them)

On the software side.

By having a good design for your software so that problems from switching platforms are minimized.

By having good tests that translate across platforms and languages so you can detect changes in behavior when you port.

Minimize the use of third party software if you don't have the source code if your application is meant to be supported over decades.

Share Improve this answer Follow

answered Dec 18, 2008 at 18:25

RS Conley
7,206 ● 2 ● 22 ● 37

3

I share your sentiment about Microsoft's output. It has its benefits, true, but their constant stream of releases is hard to keep track of and often leads to a cluttered (and sometimes nonfunctioning) computing environment. And their products often overlap in functionality, as if different teams are attacking the same problem from different ends without talking to each other. And time spent learning their stuff early is wasted if the product turns out to be a dud.

Anyway, I say as a rule, **learn what interests you, and ignore the rest**. It's the best (only?) way to stay motivated and to keep learning. This effort should be all about you, not Microsoft.

If you're like me and have more interests than time to pursue them, then you need to find ways to be more efficient: reduce your "time overhead" (time spent on uninteresting things), use time you're not using now, and learn faster.

I find that the techniques for efficiency in the real world are often similar to what you'd do when writing software.

Think about your "life performance" as if it were a program, and find ways to optimize it - maybe by trading off one resource for another, caching, etc.

More specifically, here's what I like to do:

1) Don't feel like you need to know everything about a topic you're interested in. Learn it in a level of detail appropriate to your level of interest.

2) In programming, look for an interesting problem and create a spike solution for it. If the spike is actually usable, all the better. For me, the best way to learn a new language or technology is to get my grubby hands on it and make something.

3) Minimize multitasking. What I do is try to always have exactly two things to be working on - when one is held up, I work on the other. It's rare that both will be held up, and trying to spread my attention across more than two things both causes the quality of my attention to suffer, and also introduces too much context switching cost.

4) Maximize time spent on interesting, constructive things; i.e., listen to podcasts so as to take advantage of time otherwise not spent learning. Don't waste time on non-constructive things like TV and games. Physical exercise is a must though - to paraphrase some old martial arts teacher, "learn to control your mind by controlling your body".

5) Keep your motivation level high by associating with other people who are interested in the same things you are. One way is to make the work you've done on your spike solution available to the public.

6) Have a handful of different things you're learning at any given time, and switch between them as soon as you start getting bored with the one you're on. But go back to it later as soon as your interest builds back up. I find this helps increase the amount of overall time I spend on learning, and also avoids my burning out on any one of them.

answered Dec 19, 2008 at 5:16

slipjig
131 ● 8

3

Technology is just a **tool** that helps to **solve problems**.

Software development is a **profession of solving business problems** with the tools at hand.

So I simply **concentrate on solving business problems** without worrying too much about the tools. However in most cases it eventually turns out that the most efficient tools are the latest ones. So it feels natural to switch to a new one that is slightly better than the previous, although all the major principles are still the same.

answered Dec 19, 2008 at 11:15

Rinat Abdullin
23.6k ● 8 ● 60 ● 80

2

I don't think there's any reason to feel bad for not using the latest and greatest technology. Learning a new language or technology is analogous to retooling a factory: if you spend all your time retooling, you won't spend any time actually producing anything.

Note: this should not be taken as a reason to keep using Visual Basic 6.

Share Improve this answer Follow

answered Dec 18, 2008 at 18:41

MusiGenesis
75.3k ● 41 ● 197 ● 338

Which is how it goes a lot of places. See the discussion about the average life span of an enterprise application is 3 years. – dkretz Dec 18, 2008 at 18:43

2

It makes sense to leapfrog some releases, maybe adopt every other release. If you feel the need to keep up with every release of every technology, you're not going to spend as much time doing real work or improving your fluency with the tools you're already using.

We've gotten into this mindset that newer is always better. Even when newer *is* better, it's not always so much better that it's worth the hassle.

Share Improve this answer Follow

answered Dec 18, 2008 at 18:44

John D. Cook
30k ● 10 ● 69 ● 94

2

I just read and stay aware of Microsoft technologies but I do not rush to use them or implement them just because they are out. Microsoft have so many new technologies coming out all the time, some even conflicting with their own technlogies. Like Linq to Sql and Entity Framework are two new technologies in the same area, it is confusing what Microsoft is trying to do there.

It is upto you to use what works for you and stick with until something new comes out that really provides benefits that you cannot live without anymore. Otherwise enjoy the plethora of new Microsoft technologies and try to get a giggle out of their confusion and mess they create most of the time until they get their act together and release something worth your time.

Share Improve this answer Follow

answered Dec 18, 2008 at 19:41

Ahmad
1,055 ● 7 ● 14

Regarding linq to SQL and Entities. LinqToSQL was never meant to be permanent. It was a sort of stop gap until Entities was released to show what Linq would be able to do. Unfortunately MS didn't make that clear and many developers didn't get that point. – dtc Dec 19, 2008 at 2:01

2

I work for a large utilty on the corporate side, so I'm not a vendor that needs to by conversant on all technologies-- I can focus on the problems we have at hand.

I'm finding that now that we've got our shop current and in sync with MS's releases that it's not that difficult to keep current. It was only overwhelming to get current with .NET. Getting to 3.0 was difficult: I hired into the company right after 2.0 released which was the first .NET version the company used to write a non-trivial .NET app... then 3.0 quickly came out. that felt overwhelming to try to get the existing developers to understand 2.0 while 3.0 was in the mix. But we really wanted WCF (we had a project using WSE which was painful: WCF looked and turned out to be a great solution to using WS-*). So when 3.0 came out we quickly adopted it and started down the WCF route. WCF is a really great stack IMHO but was (and remains) difficult to grock. At the time of the 3.0 release, we didn't really have need to do anything fancy on the presentation tier & workflow wasn't making much sense to me so we avoided those parts of the 3.0 framework.

That made the 3.0 release semi-managable since we were only really focusing on the WCF side of things given our business needs (i.e. the pain we were feeling w/o having WCF).

3.5 was also easy for us to get our heads around--we used LINQ as it made sense. It's not as difficult to adopt as say WCF because you can use as little or as much as you want. I personally started by just eliminating some of my foreach statements to find a collection member and grew my understanding of LINQ from there. We don't use it currently data-direct (i.e. Linq to Sql) since we are an Oracle shop. The Lambda way of C# is slowly penetrating our codebase--it's quite a switch but I love it. Again, since it's totally optional it's been easy to adopt as needed.

Since we feel comfortable with what we really needed with 3.0 and 3.5 we have had some time to start looking at other parts of those releases to see how they might improve our productivity. I've been spending some time recently with Workflow which I'm really starting to see the power of. Seeing the PDC demo's of Workflow in 4.0 makes me really want that to release... instead of feeling overwhelmed by the new technology, I'm actually very much looking forward to it's release... same too for baked in task parallel library.

So I guess I'd say that I don't feel overwhelmed at this point by the flood of new. I think that focusing on what the new releases are fixing as far as my pain points and ignoring the rest until I feel ready for them helps. It also has helped to quickly adopt the new frameworks. 3.0 and 3.5 were easy to adopt since they sit on top of 2.0 and don't require a wholsale core code upgrade. 4.0 is a year away at best... that's plenty of time for us to start looking at WPF/Silverlight as have the clockcycles to do so. Typically we just give the guy with the most interest the little bit of free time we can spare to go become the lead scout and blaze the trail... that helps too. We don't all need to be experts in the whole stack.

It also has helped us to adopt the new frameworks soon (i.e in about a month) as they are released. The 4.0 release may be more difficult to do that with... Keeping our core code separate from individual projects via version control branching has helped isolate this shared core code from 'real' development efforts, allowing us to move core forward without breaking an individual project. The individual project just merges the changes when it has the time to absorb them.

Share Improve this answer Follow

answered Dec 18, 2008 at 20:00

Kevin

2

Recent readings on this:

http://www.pseale.com/blog/LearningLahar.aspx

http://www.secretgeek.net/3way.asp

The executive summary is: Don't learn it unless you need to!

Share Improve this answer Follow

answered Dec 19, 2008 at 1:54

Stobor

45.1k ● 6 ● 68 ● 69

I wish it was possible to give all of today votes to this :)
– dr. evil Dec 19, 2008 at 9:32

1

I think the trick is to learn timeless and portable skills. Knowing how to program in more than one language and framework, is a lot more important than knowing how to program in the latest language or framework. Similarly, knowing about security principles is more important than knowing the latest buffer overflow in some web server.

Unfortunately if you are changing jobs you may come up against recruiters who don't know that, and whose job function is essentially *grep buzzword cv*. However if you are able to get past that and be interviewed by somebody who understands the job, that shouldn't matter. And if it does matter, you would probably hate working in that place anyway.

Besides, most of this 'change' is just pointless churn, and a lot of new 'technology' is just a program or a library somebody wrote...it's no more 'new technology' than the last program or library you wrote is new technology. You don't need to know most of it any more than the rest of the world has to study everything you produce.

You certainly don't need to know all of it, any more than you need to install and learn every email program ever written in order to be able to send an email.

It's a bit like trying to keep up with everything in your RSS feeds or read absolutely everything on a mailing list. Let it go. Mark it all as read. If it's important, you'll hear about it again.

Share Improve this answer

Follow

answered Dec 18, 2008 at 19:01

frankodwyer

14k ● 9 ● 52 ● 70

I completely agree. Just learn how things *work*. Everything after that is just syntax. – Mick Dec 19, 2008 at 5:32

1

I don't think you're the only one feeling that C# has grown really quickly. Fortunately the changes announced for C# 4.0 and .NET 4.0 are by no means as overwhelming as the new features of C# 3. They are very nice features, and I'm especially looking forward to IronPython and IronRuby as well as new TDD features of VS2010.

Share Improve this answer

Follow

answered Dec 18, 2008 at 19:07

Brian Rasmussen

116k ● 34 ● 224 ● 323

1

Yes, technology moves fast, actually every industry moves fast. It moves as fast as people are willing to create things. I think the comment that comes to mind is specialize. Find something you enjoy and specialize. You will never insulate yourself from change, the key is to make yourself marketable and pick the things you want to learn, not the things someone else tells you are hot. As we've seen, there are many companies out there who do not even deploy the .NET Framework on their machines, because with the apps they currently use, they have no need. Those companies will start to upgrade, but many of those upgrades are decade long migrations.

Share Improve this answer Follow

answered Dec 18, 2008 at 20:13

Chris
6,750 ● 8 ● 46 ● 61

1

There is an article from one of our co-hosts (Joel, from JoelOnSoftware) that deals at some point with this rapid changing technologies, which Joel compares to cover fire. I found it quite interesting when I read it long ago. The related part is somewhere down below, after the picture of a young paratrooper.

Share Improve this answer Follow

answered Dec 18, 2008 at 23:45

David Rodríguez - dribeas
208k ● 23 ● 302 ● 494

0

Study study study. Read read read. Participate participate participate. Code code code.

There are 24 hours in a day. Keeping current is what keeps us relevant and needed.

Share Improve this answer

answered Dec 18, 2008 at 18:29

Follow

Tad Donaghe
6,588 ● 1 ● 31 ● 64

It's hard to do these things when you have deadlines to meet and a life outside of work to tend to. – Rex Morgan Dec 18, 2008 at 18:33

Yes it is. It's also necessary. – Tad Donaghe Dec 18, 2008 at 19:34

This is a solution for a kid still in school.. not a realistic one for someone with other responsibilities outside of work. – Kon Dec 18, 2008 at 20:07

+1 for fallen888. Studying/reading/participating/coding 24 hours a day is not a helpful answer. It's a recipe for burnout. – Dhaust Dec 18, 2008 at 23:03

I didn't say 24 hours a day. I'm 39 years old - I have a family and a life outside work. I also seem to find a lot of hours each week to devote to learning and coding at home. It's all a matter of priorities. – Tad Donaghe Dec 18, 2008 at 23:56

0

Usually by doing a combination of keeping up with local user groups, trying out some things where I work at times to see what experiments produce what results, and being aware that my employer can play a role in what I am using and will continue to use. Thus, I got into the .Net 3.5 framework because someone else needed to use Linq in a project as well as a new CMS system that uses it and so now I get to explore that somewhat.

Doing little experiments isn't quite either staying ahead or ignoring it until I need it, but rather keeping up the babysteps regularly. For example, in the past decade I have gone from ISAPI extensions to ASP to ASP.Net and its various versions from 1.0 to 3.5 including the wonderful idea of the 3.0 including the 2.0 and adding a few things rather than call it WinFx which was a name it had at one time.

If one's tools are coming from an employer, then there is the question of how do they view new technology and handle shifting a developer along with new hardware or software from time to time, e.g. going from a P4 machine to a Core 2 Duo machine was nice as well as upping the amount of RAM but I did lose some software that I used on my old box like previous versions of Visual Studio.

Share Improve this answer Follow

answered Dec 18, 2008 at 19:01

JB King
11.9k ● 4 ● 40 ● 49

0

For me (in my early 40's) the need, hell even the want, to stay current wanes. The real jobs, with real lasting benefits, are the ones where you're taking care of some older technologies. And in my mind, older technologies would be .NET 2.0.

IMHO, the sign of a real trend setter is the one that can know enough about new technologies to realize if they fit into your support of older technologies or not and do they make business sense? For instance, does Linq to Sql make sense in your older environment? Research that one thing, decide whether it makes sense or not and move forward with that one newer technology.

Don't implement new stuff for the sake of implementing new stuff. Only time will bear out whether a new technology has given any business a substantial ROI.

Share Improve this answer Follow

answered Dec 18, 2008 at 19:09

GregD
7,000 ● 5 ● 36 ● 61

0

> Do you ignore it until you need it ...

This is a sure-fire way to keep being overwhelmed by new technology.

A slow learning pace is fine as long as it's steady.

Share Improve this answer Follow

answered Dec 18, 2008 at 19:25

orip
75.3k ● 21 ● 119 ● 149

0

Learn every other release. That's how I keep up. Most books will usually give you insight about the previous framework and tell you what was added.

Just my 2 cents.

Share Improve this answer

answered Dec 18, 2008 at 19:43

Follow

Smart Alec

0

Things I try to do to keep up with technology:

- Try to incorporate new stuff at work. That way I am basically getting paid to learn.
- Side project or work on someone else's open source project. Helps get someone else's perspective on new technologies too.
- Listen to podcasts about technology. In particular I listen to Java Pose and .NET Rocks. If i'm not actively coding, at least I can hear about new stuff!

I heard someone say that you should learn a new programming language every 2 years. Personally I find it extremely difficult to have that much time outside work. At least not if I want to stay married! :)

Share Improve this answer

Follow

answered Dec 19, 2008 at 2:48

CodingWithSpike

43.7k ● 18 ● 105 ● 139

0

I find myself in the same dilemma often and try to make myself learn a new technology or a new language. But most of the times, unless your employer demands it, you are not going to be doing much with that piece of technology or anything useful with that language. You will end up writing a bunch of little programs useful to learn the features of the language and try out some cool advertised bits. And more often than not you are trying to convince yourself that the new language is trying to do what you did with your old language in a better way. And in trying to adopt this functionality I forget to use the language for what it is good for.

Eventually, two things happen -- The new way is better than my old way, because it saves on a few lines of code. But I am still doing the old things with the new packaging. OR The new features suck, I'll stick to the old ways and flame all propaganda creators. :)

But that learning was essential in some cases. And sooner or later you begin to gain an intuition about what will probably take off and what won't once you've done this drill a couple of times. So it does help to stay updated, at least there will be a lot of commentary on the internet to help you out in this regard.

But always remember, learning technology is easier than learning to program.

Share Improve this answer Follow

answered Dec 19, 2008 at 9:22

tsps
1,200 ● 3 ● 13 ● 25