What programming language paradigm fits which job?

Asked 16 years, 2 months ago Modified 16 years, 1 month ago

Viewed 2k times

12

As far as I know (not much I'll admit), the currently popular programming paradigms are Object Oriented (Java, C#, Ruby) vs functional (F#). As someone who is mostly familiar with the first paradigm, I have several questions:

- Can a programmer simply stick with one paradigm all of his/her life? Or in other words, can all problems be reduced to nails for one hammer?
- If not, which tool is suited for which type of task? For instance: web-based vs desktop, creating beautiful and responsive interfaces, able to crunch data quickly, etc.
- Have people ever needed to learn a new paradigm? For my past two jobs, my workplaces required Java and C#. Are there workplaces that specifically use non-OO languages?

Obviously, there are no "best" languages, but I'm wondering whether it's worth the investment of time and energy to learn a new paradigm. Thanks in advance!

paradigms

Share

Improve this question

Follow

edited Nov 8, 2008 at 7:58

John Millikin

201k ● 41 ● 215 ● 227

asked Oct 9, 2008 at 19:55

echoblaze

11.6k ● 14 ● 46 ● 49

Ruby (and Python) are BOTH object-oriented and dynamic. The object-oriented vs. procedural vs. functional is one dimension (and that's debatable). Dynamic vs. Static is another dimensions. Web vs. Desktop -- not mentioned. This question is confusing. – S.Lott Oct 9, 2008 at 20:13

Sorry about the mess-up on the paradigms, I'm still learning about them thus the question. Besides that though, what part of my question was confusing? – echoblaze Oct 9, 2008 at 20:21

@echoblaze: Since the paradigms made no sense, there was no point in trying to parse anything else. That's why I specifically listed the paradigm problem. It was a show-stopper. Perhaps you could simplify the opening paragraph. – S.Lott Oct 9, 2008 at 20:51

8 Answers

Sorted by: Highest score (default)

"Or in other words, can all problems be reduced to nails for one hammer?" Yes. Period. Any programming language you are likely to run into will be as complete as

all others. There's actually a formal definition of "completeness" for a programming language.

"Have people ever needed to learn a new paradigm?" Always.

There's actually a trick to following the ups and downs of the "paradigm shifts". Over the last 30 years of my career, I've seen that programming has grown from a relatively simplistic imperative/procedural model to a number of much richer models that include a better balance between process and data.

I've noticed the following...

Part of the driving force is the artificial intelligence community. Many of these "new models" started as AI knowledge representation schemes. They got traction there, then they trickled into more mainstream applications.

The Entity Relationship model was originally for knowledge representation, not business transactions. The Object model, similarly, was for knowledge representation. Then the simulation folks found it. Now the rest of us have it.

Here's my conclusion.

Software is Knowledge Representation.

Your choice of Paradigm or Model or Approach or Style is based on the answer to the following question:

"How Can I Best Represent This Problem?"

If the problem has objects and relationships, OO. If the problem has algorithms and transformations, maps, filters and reduces, Functional. If the problem is dynamic, changing and flexible, Dynamic. If the problem is static and will scale up rapidly, Static.

Share Improve this answer

Follow

answered Oct 10, 2008 at 13:09

S.Lott

391k ● 82 ● 517 ● 788

> total agreement, good summary. The history of the development of programming languages is characterized by the increase in the knowledge embedded in the language.
> – Steven A. Lowe Oct 30, 2008 at 19:06

10

It's worth learning alternate paradigms (OO, functional, procedural, dynamic, etc) because it will help you think about problems in different ways.

For example, think about the difference in solving a tree traversal in a linear fashion (the first way I ever did it) versus using recursion. Or Google's combination of Map and Reduce to help them index the internet.

New ways to think applied to old problems can help break some of the hardest issues.

answered Oct 9, 2008 at 19:59

Share Improve this answer

Follow

warren

33.4k ● 23 ● 89 ● 128

7

The paradigm is independent of a language. You can develop in OO style in C (take a look at GTK). When I program in Java, I use mainly functional style.

It's worth knowing as many paradigms as possible. Some problems are trivial to solve in one paradigm, and require delicate crafting in another.

As a (trivial) example, compare quicksort implementations in Java and Ocaml, or better still, Haskell, on this page: http://www.rosettacode.org/rosettacode/w/index.php?title=Quicksort

(That doesn't mean functional is better. There are problems better solved with OO).

Share Improve this answer

Follow

answered Oct 9, 2008 at 20:10

Tomo

3,511 ● 5 ● 27 ● 29

Thanks, that makes things a bit clearer! What sort of problem would be easier solved with OO? – echoblaze Oct 9, 2008 at 20:17

> Can all problems be reduced to nails for one

4

> hammer?

Err yes. You can solve problems with just one hammer. Its just sawing that door in half takes much longer with it.

> Have people ever needed to learn a new paradigm? For my past two jobs, my workplaces required Java and C#. Are there workplaces that specifically use non-OO languages?

Developers have to do this every 15-20 years or so.

There are definitely a whole industry of small companies with Access based systems written with procedural VBA. (and I think I've worked for most of them). Classic ASP developers are having to learn ASP.NET. Perl developers are learning Python. Batch driven development gave way to event driven development.

Share Improve this answer

Follow

answered Oct 9, 2008 at 20:40

Johnno Nolan

29.6k ● 19 ● 113 ● 160

1 Sawing that door in half with a hammer would not only take longer but make it look really crappy. :)
– JUST MY correct OPINION Jan 16, 2011 at 16:22

I think that you will find answers all over the board. The more I work, the more I find that it is "helpful" to know some of the others. as a C#/VB/SQL Server developer I

2

find it more helpful for me to learn a bit about F# and some of the other languages out there to just get a broad exposure, to really figure out what tool is the right...

Share Improve this answer

Follow

answered Oct 9, 2008 at 19:57

Mitchel Sellers

63.1k ● 15 ● 114 ● 174

1

The dynamic stuff scares the crap out of me, but Ruby on Rails is by far the best development system I've seen for web stuff. I don't feel comfortable with using it for a really big, heavy maintenance project though because it's too easy to alter the meaning of existing, compiled, finished code. Also too easy for one person's coding style to make it into a new language.

Dynamic/scripting is also good to know for system admins and anyone who runs a Linux system. Writing a quick script in BASH or Ruby beats the HELL out of trying to implement the same functionality in Java, or C++.

OO makes it much easier to understand large amounts of code. If you have a large team or multiple large teams and need to give an overview quickly, OO makes it much easier to describe and to isolate a given piece of functionality. I should say CORRECTLY CODED OO!

I understand functional is good for multi-threaded programming since everything tends to be immutable.

Share Improve this answer

answered Oct 9, 2008 at 21:52

Follow

Bill K

62.8k • 18 • 112 • 158

0

Developing the design and architecting skills with OOP in mind is the most desirable skillset for a great language agnostic career.

The benifit when you code in Oops is both for the other team members out there and for the organization as a whole. Because the code will be understandable to all and if the developers quit the job, company doesnt need to worry much. In the other case if you follow functional style it will be real hard for others to understand what you have done.

Share Improve this answer

Follow

edited Oct 9, 2008 at 20:19

answered Oct 9, 2008 at 20:04

Jobi Joy

50k • 20 • 110 • 120

0

As most others have said, you can generally use any language to solve any problem and you can usually write in the style of one paradigm in another.

If you take the time to learn to use the different paradigms as they're intended then you do learn different things about knowledge representation and problem solving

which can be helpful in whatever paradigm you're using in the future.

While there is some alignment between paradigms and domains it is usually best to choose a language based on the specifics of the environment your software needs to operate in.

- Does it need to run on multiple desktop platforms?
- If it's a desktop application does it need to have a native look and feel?
- Is rapid iteration of designs important
- How is it to be maintained?
- What 3rd party systems does it need to operate with?
- Existing programmer knowledge / skills / preferences.

Share Improve this answer

Follow

answered Oct 30, 2008 at 19:01

Tom

44.7k ● 4 ● 43 ● 61