Subject: Static OOP (C++...) vs Dynamic OOP (CLOS...)
Posted by fudadmin on Sun, 01 Jan 2006 02:44:26 GMT
View Forum Message <> Reply to Message

just to have in mind...
maybe more use of ESC interpreter in U++?...

old but still good material.
from http://www.algo.be/cl/doop.htm
Quote:
...
Dynamic Object-Oriented Programming

Dynamic Object-Oriented programming is a software development technology that enables applications to be tailored during development and after deployment without access to source code. Made practical by the continuing hardware evolution predicted by Moore's Law, Dynamic OOP languages are much more effective than static OOP languages for managing complexity and adapting to changing needs.

With Dynamic OOP languages, the amount of work necessary to make a change is proportional to the degree of change, not the size of the application. New objects, new classes and new behavior can be added on the fly, and unlike static OOP languages, Dynamic OOP applications do not have to be rewritten to accommodate any change.

Dynamic OOP is the enabling technology for user-evolved software. Developers can incrementally test working prototypes with users...

also interesting http://www.norvig.com/java-lisp.html
Quote:
...The conclusions showed that Java was 3 or 4 times slower than C or C++, but that the variance between programmers was larger than the variance between languages, suggesting that one might want to spend more time on training programmers rather than arguing over language choice...

Subject: Re: Static OOP (C++...) vs Dynamic OOP (CLOS...)
Posted by mirek on Sun, 01 Jan 2006 07:27:22 GMT
View Forum Message <> Reply to Message

Well, those "CLOS" gyus are totally off as all dynamic programming advocates. Even reading their material:

Quote:
C++ requires any redesign in the classes and methods to be completely consistent and correct across the entire application source code before a design modification can be compiled, let alone tested.

...and they think this is a disadvantage?!!!!

Is it so hard to understand that using static type checking, compiler catches many bugs for you, makeing development and refactoring of very large applications easy and safe?

Damn, sometimes I refactor code using C++ by invoking compiler on unfinished change just so he tells me what to fix (via errors indicating type/signature inconsistency).

Subject: Re: Static OOP (C++...) vs Dynamic OOP (CLOS...)
Posted by gprentice on Sun, 01 Jan 2006 12:04:54 GMT
View Forum Message <> Reply to Message

Well I think he's correct when he says this.

Quote:But C++ is hard to use, and requires an inordinate investment in time to avoid memory leaks (that result in poor performance and random crashes) and to tackle exception handling problems.

though it seems use of Garbage collection is becoming more common in C++ and people use new/delete much less than they used to, even without GC. There's still a large amount of discussion goes on in C++ newsgroups about exception safety. I wonder why other langauges don't have complex "resource release" and exception safety problems like C++.

Slava Pestov (wrote JEdit) has created a Forth like language called Factor
http://factor-language.blogspot.com/

He claims the ability to change one little piece of code and carry on execution immediately is a huge advantage over having to recompile the whole thing before trying out the change. I think this is what the CLOS article is talking about regarding improved development times.

BTW - Factor runs on AMD64 and PowerPC (as well as Windows/Linux) and it's GPLd open source (and it has garbage collection). (However I developed a dislike of Forth a long time ago ...)

Graeme

Subject: Re: Static OOP (C++...) vs Dynamic OOP (CLOS...)
Posted by gprentice on Sun, 01 Jan 2006 12:17:15 GMT
View Forum Message <> Reply to Message

BTW - that CLOS article appears to be ten years old !!
That company's website is all about LISP now - which also has the "incremental development" capability.

I don't think dynamic languages have taken over from C++ at all since that article was written.

Graeme

Subject: Re: Static OOP (C++...) vs Dynamic OOP (CLOS...)
Posted by mirek on Sun, 01 Jan 2006 13:34:37 GMT
View Forum Message <> Reply to Message

gprentice wrote on Sun, 01 January 2006 07:04
Well I think he's correct when he says this.

Quote:But C++ is hard to use, and requires an inordinate investment in time to avoid memory leaks (that result in poor performance and random crashes) and to tackle exception handling problems.

though it seems use of Garbage collection is becoming more common in C++ and people use new/delete much less than they used to, even without GC. There's still a large amount of discussion goes on in C++ newsgroups about exception safety.

Actually, both problems are very aggressively addressed in U++.

Resource management is resolved by following "everything belongs to some scope" principle. Together with U++ containers, there are no more resource management problems.

Exception handling problem is pragmatically resolved by ignoring out-of-memory exceptions;) If you go out-of-memory, app simply stops and that is all. I believe that most applications that put effort into solving this are unable to deal with out-of-memory anyway, as there is no reasonable way how to debug such condition. Plus, any application with recursion is prone to similar "out of stack" problem that cannot be solved using exceptions.
