JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2005**

Vol. 4, No. 9, November-December 2005

The (Re) Emergence of Agile Languages and Domain-Oriented Programming

Dave Thomas, Bedarra Research Labs

The JAOO conference in Aarhus, Denmark has become one of the most enjoyable and informative developer conferences in the land of OO. Originally a Java conference, it has expanded to cover a wide variety of topics from MS.NET and Java technology to best practices in software engineering. This year's conference followed on the heels of the Microsoft PDC earlier in the month. Both conferences featured tracks and sessions on scripting, dynamic languages and domain specific languages. There was also a Dynamic Languages Symposium (http://decomp.ulb.ac.be:8082/events/dls05/program/) and Ruby conference at OOPSLA. These efforts are all aimed at lowering the barrier to developing applications. Hurrah!

1 YOU CAN DO THAT IN BASIC? (VB ON MONADS!)

Would any OO professional really consider using Basic when she has access to the full power of C#? Can MS really make it possible for disenfranchised Basic developers to have first class language access to XML, Databases and Objects with ease? Many at the PDC and JAOO were in shock as Erik Meijer (http://research.microsoft.com/~emeijer/) and the VB team showed off VB 9 features that combine clever type inference and structural sub-typing to simply and elegantly manipulate squares (relational tuples), circles (objects) and triangles (XML info sets) as simple polymorphic collections. Erik has also abandoned his solemn monadic oath of Haskell and quietly snuck some really interesting functional machinery under the hood of C#3.0 and VB 9 (http://msdn.microsoft.com/vbasic/future/default.aspx?pull=/library/en-us/dnvs05/html/vb9overview.asp).

Erik is of course one of the key contributors to MS Zen and COmega (http://research.microsoft.com/COmega/) research efforts which are now bearing fruit in MS next language products under the name Language Integrated Query (LINQ – http://msdn.microsoft.com/netframework/future/linq/). The LINQ team has really been thinking out of the box on this project and has made brave new efforts with very popular and widely used languages, which is always a great language design and implementation challenge.

2 SCRIPTING THE JVM AND CLR

Both JVM and the CLR have a bad history with the dynamic language community since they are designed to support statically-typed languages. Initial implementations of languages from Scheme (Kawa) to Python (Jython), both for the JVM, have had disappointing performances and have almost always involved language sub-setting in one form or another.

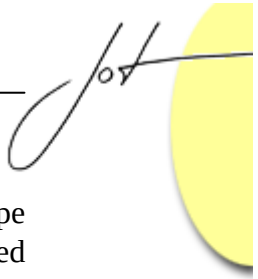
Even the scripting interoperability story is broken for JVM and CLR. This means that some of the better implementations, which typically are written in C, are not available to CLR/JVM developers. In old COM, Microsoft provided a nice pluggable mechanism called the windows scripting host. Unfortunately, this disappeared when .NET arrived. In Linux, Apache provides a feature called modules which are used by ModPerl, ModPHP, etc. to integrate them into the process space.. It appears that recent efforts by Sun through a JSR, and by MS through CLR and LINQ, may improve this situation.

Recently Jim Hugunin, who developed Jython, presented IronPython (http://www.ironpython.com/), a Python implementation which shows that a more sophisticated implementation of Python can leverage the new features of the CLR to provide a good implementation of the Python language. The benchmarks and open source implementation provide an opportunity to use Python as a scripting language for .NET. Hopefully the Python community will provide libraries such as PyNumerics so that Python developers can find a familiar home developing in .NET. There have been similar research efforts for ML and Scheme which use some complex implementation hacks based on closure analysis, type inference, combined with AOP tricks for byte engineering which appear more promising. However, these are cruel and unnatural acts required by the omission of reasonable support for dynamic languages in both popular VMs.

3 TOWARDS A SOFTER, GENTLER JAVA

Sun has recently hosted workshops on scripting, and has supported Groovy (http://groovy.codehaus.org/) in the Sun community process. Groovy is a dynamic scripting language which has a Java friendly syntax, which means that Java developers don't have to learn new syntax. Furthermore, Sun has work in progress to ensure interoperability with ECMAScript, which is very important given the increased interest in JavaScript due to AJAX.

Gilad Bracha, computational theologist, of Strongtalk (http://www.cs.ucsb.edu/projects/strongtalk/pages/index.html) and HotSpot fame, discussed the challenges of building dynamic languages on the JVM. After careful consideration, Gilad has proposed a major improvement which should make life easier for dynamic language implementers – a new `Invoke Dynamic` byte code (http://blogs.sun.com/roller/page/gbracha?entry=invokedynamic). This simple mechanism eases the implementation of languages like Smalltalk and Ruby. It also allows more elaborate features such as multiple inheritance to



be realized using techniques such as Smalltalk's `doesNotUnderstand:`. Let's hope that the Java and dynamic language community support Gilad in this long awaited improvement!

4 RUBY, RUBY, WILL YOU BE MINE?

Ruby (http://www.ruby-lang.org/en/) is burning the blogs of the world, especially with the easy-to-use RubyOnRails (http://www.rubyonrails.org) web applications framework. Ruby is derived primarily from Smalltalk and Perl with some improvements. The language is pure OO, much like Smalltalk, but has a more traditional syntax. Ruby is seeing increasing use in web development and tooling as its Smalltalk lineage appeals to the Agile community because it allows very rapid development of readable programs.

If you have not tried Ruby you should check it out; you may well want to use it in lieu of Perl, PHP or Python. While the support on the JVM and CLR is less than ideal, there are efforts underway. There are rumors that Microsoft Research is supporting a Ruby for the CLR at QUT in Australia.

5 DSLS - LANGUAGES THAT SPEAK YOUR LANGUAGE

Domain Specific Languages (DSLs) are also making a return to the scene. Given the exponential increase of domain knowledge in modern software, DSLs are being used to raise the level of the platform to the domain expert or at least to developers who can work with the domain experts. Simple DSLs are often implemented using a scripting language to configure a framework (frameworks are really just little languages) or to construct the inputs to a large application.

Not all DSLs need to be linear or visual; for example FIT/FitNesse, the popular acceptance testing tool, uses a tabular language to provide a middle ground between customer/analysts and developers so that requirements can be expressed by those familiar with the application domain.

Of course, this is what the XML community has been doing for most of the last five years. They just don't seem to need a syntax or a semantic account. This has been a great source of frustration for those of us who deal with the complexities of XML Schema and XQuery as well as UML and MDA for that matter.

With the development of more sophisticated domain-oriented programming (http://doi.acm.org/10.1145/949344.949346) with domain specific textual or visual syntax and debugging at the level of the abstraction, one needs more powerful language technology to build or extend DSL IDEs. Martin Fowler has recently discussed Language Workbenches in his blog (http://www.martinfowler.com/articles/languageWorkbench.html). JetBrian's MPS, MetaCase's MetaEdit+, Microsoft Software Factories and Intentional Software have tools either coming to or already on the market to support

DSL developer and users. The other proven implementation approach is to embed the DSL in a powerful language substrate such as Scheme, Haskell, CLOS, or Smalltalk.

6 SUMMARY

After a long dry spell it appears that developers may finally get a kinder, gentler path to exploratory programming, test case development and indeed even applications development. Hopefully this will encourage more computational diversity as well as interesting new language research and applications.

About the author

Dave Thomas is cofounder/chairman of Bedarra Research Labs (www.bedarra.com), www.Online-Learning.com and the Open Augment Consortium (www.openaugment.org) and a founding director of the Agile Alliance (www.agilealliance.com). He is an adjunct research professor at Carleton University, Canada and the University of Queensland, Australia. Dave is the founder and past CEO of Object Technology International (www.oti.com) creator of the Eclipse IDE Platform, IBM VisualAge for Smalltalk, for Java, and MicroEdition for embedded systems. Contact him at dave@bedarra.com or www.davethomas.net.