JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2006**

Vol. 5, No. 2, March-April 2006

On The Next Move in Programming[1]

Dave Thomas, Bedarra Research Labs

IN SEARCH OF THE NEW NEW PROGRAMMING THING

We are now emerging from the dark period earlier in the decade and many in the community are bemoaning the good old days when new languages appeared with great frequency and new paradigms abounded.

Unfortunately, we find ourselves in a world which increasingly ignores the experiences of past successes and failures to claim inventor rights are the new new thing. Simple techno sound bites are easy to communicate using the web and academics have become as adept as major marketers in spin and mindshare capture. Rather than evolving from the best of what we know to the better, each new thing in our field seems to need to kill off those that came before it.

A recent PhD student remarked that it was very upsetting to have to read old papers published in the mid to late 70s that contain immature versions of their new ideas which they must now read and cite. It sometimes appears that scholarly work in our field is no longer a badge of honor but a burden forced on graduate students who have grumpy old supervisors.

We seem to be in denial that our industry is maturing, we are losing our elders, some of us don't code any more and many of our new things are really good ideas from 20 or more years ago. Some good old ideas reappear because the technology is now available to commercially exploit them; others become popular because they provide a better approach to solving a particular problem. Reinvention and improvement of good old ideas is both scholarly and sound engineering provided the prior art is cited and explained as opposed to hyped or even patented. This means we need to look for the important small improvements and not just the big bang!

We need to find the balance which builds from our best past work but also doesn't send a discouraging message that everything has been done before, or if it was that it was necessarily done correctly! We need to benefit from the wisdom of our elders without being constrained by them. We've argued for the increased promotion of computational

[1] This column was inspired by JOT discussions including an article titled "The Next Move In Programming: A Conversation with Sun's Victoria Livschitz, http://java.sun.com/developer/technicalArticles/Interviews/livschitz_qa.html " which inspired the title.

diversity (http://www.jot.fm/issues/issue_2003_05/column1) rather than a narrow focus on using the right current industrial platform or language.

OBJECTS NEXT?

In the OO community there seems to be a concerted search to either find what is "Beyond Objects?" or to argue that "Objects Are It!". To their credit OOPSLA (http://www.oopsla.org/2006/) and ECOOP (http://www.emn.fr/x-info/ecoop2006/) have been openly searching for new ideas in their calls for papers and workshops. The OOPSLA Onward! track deserves special mention in this regard as it provides a platform for new ideas.

The popular research area of AOP is one new next contender which has rapidly emerged from our own community and now has its own successful conference (http://aosd.net/2005/). Recently, there have been papers/proposals for Feature, Language, Law, DSL/Domain, and Context Oriented Programming etc. There is also good work in integrated objects and transactions, unifying objects, info sets and tuples [1]. Closely related to this is the interesting work on making relationships found in ER and UML models first class types in a programming language [2]. Continued work on Actors [3] and more recently Chords [4] seeks to unify concurrency with object orientation. Object Ownership has always proved a challenging problem especially when VM implementations refuse to allow objects to have an associated owner; however, considerable progress has been made [5, 6]. There is also a resurgence in research on dynamic languages including prototype based languages and multi-paradigm languages. Finally, work on using objects to build next generation multi-modal UI toolkits is very interesting [7].

All of this work seeks to provide tools or language extensions which allow the OO model to address specific issues and/or broaden the applicability of OO. While they don't promise a New Oriented Programming they provide the critical and difficult research to build better OO languages, tools and systems. It is clearly a challenge to integrate all of these powerful concepts into a language/platform which is useable by those who are not advanced software professionals.

PEOPLE ORIENTED PROGRAMMING

One of the more exciting changes in our field is the rapid and energetic emergence of Agile Software development initiated by Scrum and XP. Unlike traditional process artifact centered approaches to development, Agile emphasizes human and business values. Large, unfortunately unpublished internal studies have concluded that defects and timely delivery of software are most high correlated with the individuals and teams of individuals rather than the programming environments, languages or platforms.

We have for too long focused on the technical dimension of our field without considering the importance of the social aspects and especially the critical need to effectively communicate with our customers and between ourselves. Even small progress in this area promises huge benefits.

OPEN SOURCE SOFTWARE

Open Source has provided the impetus for small global teams of open source project developers, some of whom have never met in person, to collaborate to build software together. Other developers and users have full view of the source code they create and in many cases the interactions of team members. Simple Open Source tools such as Wiki, JUnit and Fit have a major impact on allowing teams to build better software. Eclipse has enabled many to contribute to the open source tooling. Recently MS and Eclipse have initiated efforts to enable open process definitions which may lead to integration of processes and tools. The availability of open tooling has allowed customers to innovate in their tools and processes, often accelerating the transition to things like Agile development.

The large open source code base provides a rich set of research data for social researchers as well as software engineers who can now finally look at large programs as they evolve over time. The combination of sophisticated program analysis, text understanding and simple heuristics promises new tools for helping us understand how we work.

SOFTWARE WITH POLISH

It is well accepted that it takes time to polish a good essay, research paper or a program into something that we believe is correct and readable [8]. Modern languages lack publication syntax and IDEs lack a nice way to publish/present the code using multiple fonts and indentations[2] so that it is readable [9] by others. We need to make the technical and business case that clean, well-written software is a better investment and that "send it back, refactor it and make it better" is a necessity not a luxury.

SAYING MORE WITH LESS – THE NEED FOR A VHSL

Despite the great progress in tools and languages there is still far too much stuff that gets in the way. The latent and accidental complexity of current languages and platforms makes a 4GL of the past look like a dream. Java and C# are just barely above the machine code of the VM. VMs have lifted the level of the platform but we are still programming just above them. There is too much code to be developed and maintained to keep using

[2] JavaDoc and Doxygen provide basic capabilities for publishing.

low-level languages even if they are supported with fancy visual tools and generators. Even in old functional programming languages such as Lisp and APL all developers knew map reduce as a programming idiom and didn't need a separate course in design patterns.

Perhaps we should take a tip from our colleagues in hardware design and think about languages which are simple, expressive and allow us to describe major applications in a small number of pages of code. If electrical engineers can move to HDL and VHDL we should be prepared to move to at least a similar level of abstraction with a VHSL.

We need to move beyond the complexity, limitations and weaknesses of Smalltalk and Lisp but seek a language with at least as simple a syntax and more expressiveness. We have the benefit of considerable research on advanced type systems which can used to work with the developer as opposed to against him. Our hardware colleagues can help by using some of those transistors to support robust and efficient dispatch, true sandboxing and type and ownership information for every instance. Small investments in hardware software co-design can have a huge impact on the robustness and adaptability of software.

REFERENCES

[1] Gavin Bierman, Erik Meijer, and Wolfram Schulte, The essence of data access in Cω, ECOOP 2005, http://research.microsoft.com/Users/gmb/Papers/ecoop-corrected.pdf

[2] Gavin Bierman, Alisdair Wren, First-Class relationships in an objected-oriented language, ECOOP 2005, http://www.cl.cam.ac.uk/~aw345/talks/ecoop05.pdf

[3] Denis Caromel, Ludovic Henrio, Bernard Serpette, Asynchronous and Deterministic Objects, POPL'04.

[4] Nick Benton, Luca Cardelli, Cedric Fournet, Modern Concurrency Abstractions for C#, http://research.microsoft.com/Users/luca/Papers/Polyphony%20(TOPLAS).pdf)

[5] Chandrasekhar Boyapati, Alexandru Salcianu, William Beebee, Martin Rinard, PLDI 2003.

[6] David G. Clarke, James Noble, John M. Potter, Simple Ownership Types for Object Containment, ECOOP 2001.

[7] Eric Lecolinet, http://www.infres.enst.fr/~elc/ and Stéphane Huot, Cédric Dumas, http://dastuf.free.fr/projects.html

[8] Richard Gabriel, The Poetry of Programming, http://java.sun.com/features/2002/11/gabriel_qa.html

[9] Ronald M. Baecker, Human Factors and Typography for More Readable Programs, Addison-Wesley, January 1, 1990

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.