Multiparadigm Programming in Leda

Timothy A. Budd
Oregon State University
Corvallis, Oregon
USA

August 1, 1994

Contents

Preface

Several years ago I was interviewed by a senior writer from the trade journal Datamation, who asked several questions regarding my opinions of the then-emerging concept of object-oriented programming. I had recently authored a book on Smalltalk [Budd 87], and I expect he assumed I would offer the popular opinion that OOP was the wave of the future and would quickly and easily solve all problems associated with software development. Instead, I surprised him by indicating that yes, important revolutions were taking place in the area of object-oriented programming, but that we needed to also be aware of equally important innovations in other areas of programming languages, such as new ideas being developed in the functional programming community, logic programming, and so on.

I was somewhat dissatisfied with the interview when it appeared in print,[1] because the author left the impression I was more-or-less unimpressed with object-oriented concepts. Indeed, nothing could be further from the truth. I was very excited about OOP (as evidenced by the fact that I went on to write two further books about the topic), but I was also equally excited by innovations occurring simultaneously elsewhere in the programming languages world. I knew at that time that the real revolution would arrive when we finally understood how to bring these diverse points of view together into one framework. At that point I did not know exactly how this would happen or what form this framework would take, but I knew it was a topic that needed to be pursued.

In subsequent years I was able to make a small amount of progress in this area. I was able to start to design a small, simple yet powerful language, Leda, that allowed programmers to work in a multi-paradigm fashion. Various students working under my direction developed compilers for this language, which evolved as our experience with the concept of multiparadigm programming increased. Other researchers also became interested in this work. A group at the university of Nice, in France, independently developed a compiler for my language. Their efforts exposed several subtle semantic issues, which led to further refinements of my ideas.

In general, however, the development of Leda progressed much more slowly then I would have liked, due in large part to the number of other constraints on my time. Eventually I was granted a sabbatical leave from my duties at Oregon State University for the academic

[1] "The OOPS Revolution," by John W. Verity, Datamation, Vol. 33(9):73-78, May 1987.

year 1993-1994, and I determined I would spend my time investigating in detail the ideas of multiparadigm programming that had been simmering in the back of my mind for so long. In preparation for my leave, in the Spring of 1993 I taught a graduate seminar on multiparadigm programming at OSU, during which time I started to put my ideas down on paper in a form that eventually evolved into the present work.

I was invited by Ole Lehrmann Madsen to spend the first part of my sabbatical year teaching a course on multiparadigm programming at Aarhus University, in Denmark, and Olivier Lecarme invited me to spend the second half of the year teaching the same course at the Université de Nice–Sophia Antipolis in France. Since teaching and research were one and the same in this instance, and the pressures of teaching would ensure I would devote my time to organizing my ideas into a hopefully coherent set of notes, I gladly accepted these offers. Subsequently Henri Bal also invited me to present a short two-week course at the Vrije Universiteit in Amsterdam, which had the effect of forcing me to once more review and organize my ideas. It was over the course of this year that the text took on its present form.

Multiparadigm Programming as a Teaching Tool

While the present text is not designed as an introduction to programming (it assumes the reader is familiar with, at the minimum, at least one conventional imperative language), the use of multiparadigm programming languages, such as Leda, for teaching introductory programming skills is an attractive idea. Research results from the psychology of programming [Petre 89] indicate that expertise in programming is far more strongly related to the number of different programming styles understood by an individual than it is to the number of years experience in programming. It is, therefore, important to give students familiarity with procedural, object-oriented, functional and logic based programming, as well as other techniques. As to whether this can best be done using a single language, such as Leda, or a combination of a number of different languages is an interesting question, and as yet an unresolved one.

What's in a Name?

One of the hardest tasks with any major software endeavor is to try to come up with an appropriate and evocative name. The name Leda has been associated with my ideas on multiparadigm programming from the start, and my exact reason for selecting that particular name is now lost to my memory. It is most likely due to the fact that one of the first example programs I developed in this new style was an early version of the family tree database described here in in Chapter 19. A more ribald suggestion is that the project took form while I was being considered for tenure at Oregon State, and the name is somehow related to the rather compromising position with regards to the "gods of academia" one can sometimes feel placed in during that process. (In any event, the gods looked favorably upon my case and I was eventually granted tenure.) At times my students and I have tried to, after the fact, come up with a reasonable acronym to explain the name, but without notable success. Most likely the name was originally selected simply because it appeared to be ephoneous and unique (I was at that point unaware of a project in Germany involving data structures in C++ that used the same name). At the present time the name, more than anything else, is perhaps an expression of a hope; namely, that the languages that will come after Leda, like the children of the ancient Leda, will be possessed of a grace and a beauty far above the average.

Acknowledgements

A project of this magnitude cannot come to fruition without a great deal of assistance from a large number of people. Most fundamentally, Leda embodies my own peculiar ideas of programming languages. These ideas have been shaped by two decades of learning about languages from a number of master teachers. In particular, I am grateful to the late Alan Perlis for teaching me APL, to Ralph Griswold for exposing me to Snobol and Icon, to Lambert Meertens and Steven Pemberton for showing me ABC (a language that deserves to be much more widely recognized), to Gary Levin for developing the wonderful implementation of SETL called ISETL, and to Ole Lehrmann Madsen for trying to show me the Beta way to program. Features of all of these languages can be found in Leda, and I hope that none will be offended by the way I have borrowed ideas (perhaps stolen is a more descriptive word).

I have already mentioned that various students at Oregon State University developed implementations for the language as it slowly evolved. John Placer, now at Northern Arizona University, designed and implemented an earlier multiparadigm language [Placer 91a], which helped direct some of my initial ideas on the topic. A trio of students, Vinoo Cherian, Wolfgang Pesch, and Jim Shur, developed the first full implementation of the Leda language [Cherian 91, Pesch 91a, Pesch 91b, Shur 91]. Masami Takikawa subsequently developed another compiler for the second generation of the language [Takikawa 93]. OSU students Rajeev Pandey, Tim Justice, and Nabil Zamil also made important contributions to this work [Pandey 93].

At the same time my students at OSU were creating implementations, Carine Fédèle and Olivier Lecarme at the University of Nice were using Leda as a test language for their compiler writing system (see [Fedele 92]), and I am grateful for the many penetrating and insightful questions they passed on to my group at OSU. These had the effect of driving myself and my students to consider various issues in more detail than we might otherwise have done, and ultimately the language was better for it.

In addition to those named above, a number of students at various institutions have made suggestions or comments which ultimately led to a improvement in the text or, sometimes, in a change in the language itself. Notable among these are: Dick Grune, Vrije University, Tim Justice, Oregon State University, Jakub Těšínský, the University of Prague, Tommy

Thorn, Aarhus University, and Rebecca Walpole, Oregon State University.

I am, of course, indebted to all the students from Oregon State University, Aarhus University, the University of Nice, and the Vrije University who suffered through the course upon which this book is based, bearing with equanimity a new programming language and an incomplete text, both subject to frequent revisions.

Finally, I wish to acknowledge the work of the editorial, production and marketing staff at Addison-Wesley. They make the experience of writing a book a pleasure. My editor has been Lynne Doran Cote, assisted by Maite Suarez-Rivas and Deborah Lafferty. I also wish to thank Helen Wythe, (names to be added).

Obtaining the Code

All the example programs presented in this book have been compiled and executed using the latest version of Leda. Leda is an evolving language, and has changed over time. Compare the example programs in [Budd 91a] or [Budd 92] to those in this text, for example. It is likely that it will continue to evolve further in minor respects.

The current system, as well as all the example programs used in the text, can be obtained through anonymous ftp. Connect to machine cs.orst.edu, then access the directory pub/budd/leda. Alternatively, a mail server will provide access through e-mail. Send a message to almanac@cs.orst.edu and place the text *send leda catalog* in the body of your message. The author can be reached at budd@cs.orst.edu.