PROD: Motivations

Tim Menzies

Lane Department of Computer Science, West Virginia University, PO Box 6109, Morgantown, WV, 26506-6109, USA;
http://tim.menzies.us; tim@menzies.us

Wp ref: ~menzies/src/pl/prod/prodabout.tex, January 30, 2003.

Abstract The motivations for the PROD system are explained.

Contents

List of Figures

1 Why?

I wrote PROD because I needed new training material to teach grad students about Prolog programming. Why was that so? Well, to begin with, I think the training given to most programmers emphases one particular *paradigm of computer systems*. I wanted to broaden that training. Also, as a software engineering educator, I needed to *show software systems* in my classes. Finally, despite all its flaws, *Prolog is a very useful language* and I found myself using techniques not found in the standard Prolog texts.

Not convinced? Well, then I see I'll have to explain myself a little more...

1.1 Paradigms of computer systems

Computer development systems can be divided up according to what they are trying to do. Systems based on "C" are usually optimized to *run* fast. Systems based on some normalized database design are optimized to be *maintained* quickly. And systems written in high-level languages are usually the ones where programs can be *written* quickly.

This characterization of systems into "fast to run" or "fast to maintain" or "fast to write" is, of course, only an approximation. Clearly, these kinds of systems can be combined so that (e.g.) systems written in high-level languages can be quickly written then compiled to "C" so they run fast. Nevertheless, the emphasis in most programming courses is on procedural programs in languages such as "C" programming. This has the side-effect that most programmers never see how much of their work is influenced by the "fast to run" approach. As standard desktop machines get faster, it becomes practical to move beyond the "fast to run" paradigm.

System development for systems that are fast to maintain has been discussed extensively elsewhere [2]. Here, we focus on systems written in high-level languages that are quick to write.

1.2 Must show software

Erwin Schroedinger once said *if you cannot- in the long run- tell everyone what you have been doing, your doing has been worthless.* Well, it took me a while to work it out but finally I realized that as a software engineering educator, I should be presenting software to my classes. Every class, lots of software.

I've tried various languages and various formats and this is my latest attempt. In this experiment, the new mix is:

- The programming language is Prolog [1,3,4,6,8]) since, for me anyway, it lets me show the most functionality in the fewest lines.
- The examples are biased towards logic-for- software-engineering...
- ...coded up in a programming style not discussed extensively in the standard Prolog texts ...

Textbooks:

- *The classic*: Clocksin & Mellish, Programming in Prolog 4th ed. Springer-Verlag 1994.
- *The best seller:* Ivan Bratko, Prolog Programming for Artificial, Addison-Wesley. At the time of this writing, the third edition was out.
- *For the theoreticians:* Hogger, C. J., Introduction to Logic Programming Academic Press 1984.
- *Contains lots of cookbook solutions:* Sterling and Shapiro, The Art of Prolog. MIT Press, Cambridge, Mass. 1986.
- *Advanced Prolog programming (and small!)*: Clause and Effect: Prolog Programming for the Working Programmer By William F.Clocksin, Springer Verlag 1997. ISBN: 3540629718
- *Prolog & General AI*: Computational Intelligence: A Logical Approach: D. Poole and A. Mackworth and R. Goebel, Oxford University Press, 1998.

On-line material (there's a lot out there; here is just a sample):

- *Newsgroups*: `news://comp.lang.prolog` is a friendly place to post queries, but be warned. Many students post queries like "HELP: my assignment is due and I don't know how to reverse a list. What can I do?". And, sometimes, the locals at `news://comp.lang.prolog` express (mild) annoyance at this sort of behavior
- *The Prolog FAQ*: `http://flits102-126.flits.rug.nl/~dirk-jan/prolog/faq/html/`
- *Code Libraries*:
  - Dick Botting's Prolog collection: `http://www.csci.csusb.edu/cs320/prolog/index.html`. The Techniques section is especially recommended.
  - David Poole's code library from the Computational Intelligence book: `http://www.cs.ubc.ca/spider/poole/ci/ci_code.html`.
  - The CMU Prolog repository: `http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/lang/prolog/0.html`

Tutorials:

- *Nice*: `http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html`
- *Fun, but a little basic:* `http://burks.bton.ac.uk/burks/language/prolog/amzi/index.htm`
- *Very nice:* `http://ktiml.mff.cuni.cz/~bartak/prolog/index.html`

Fig. 1 Prolog resources.

– … presented in a very hands-on programming style.

Also, I can't convince anyone that Prolog is pretty good without pretty good documentation. PROD is a bunch of LaTeX tricks that let me rapidly generate camera-ready documents from my Prolog code.

Maybe you'll find it useful as a programming or teaching resource.

1.3 But why are you still using Prolog?

Well, it's like this. As long as:

- I want to write state-of-the-art model-based software.
- I can keep up with the next generation of grad student programmers.
- I don't have unlimited programming time.
- There is a large international community of educators, researchers and vendors supplying high quality Prolog materials.

then, I'll stay with Prolog. Here's my reasons:

Prolog endures. It is an old language (1972) yet it keeps being taught, keeps being used industrially, and high-quality textbooks and interpreters are readily available (see Figure 1). Nevertheless, like many folk, I used Prolog in grad school then left it behind for something better. "Prolog?", I used to say, "that was something we used to do as kids".

Only thing was, the "something better" wasn't so much better. After nearly a decade of real-world commercial programming in Prolog, then Lisp, then Smalltalk, I realized that my best logic programming code was smaller than my best functional or OO code. This was a surprise since I thought that (e.g.) OO would allow me to better structure my programs. (Actually, I stopped thinking that when I discovered the wonderfully messy and practical Perl programming language. Heck, my Prolog has to be at least as well structured as the average Perl program.)

Anyway, then I became an academic again and I noticed a couple of things. Firstly, the machines were getting faster. It is now perfectly viable to deliver interpreted systems in a commercial setting (look at all the Perl code out there).

While the machines get faster, I seem to be getting slower. I kept finding that I didn't have time to write huge programs. I had to return to my Prolog in order to get certain jobs done.

Slow as I was, I found that I could still out-program my grad students. See, I'd thrash around for a weekend in Prolog and then give it to them saying "code it in C, make it faster, make it better". Trouble was, months later, they were still fighting their way through memory leaks, pointer problems, etc.

Another reason for staying with Prolog is that, at least in my view, logic programming techniques are being used more and more. For example, the reflection pattern proposed by OO design gurus (and implemented as e.g. Java beans or aspect-oriented programming) is just programmers realizing that things like `clause/2` is astonishingly useful. But this is just a special case of my next point.

Old-fashioned software is like obsessed over-trained athlete sprinting for the finish line. Such software runs fast since it doesn't look where it is going. But, it will stumble and crash at the first hole in the road. Modern model-based software builds and reflects over some model carefully crafted at runtime. Choices are recognized, and processed via an on-going analysis of the internal model. Prolog is just terrific for implementing such reflection- 20 lines of meta-interpretation can give you so very much.

Q.E.D.

2 Who?

PROD emerged after many years of programming. I list here some of the people who, along the way, most influenced my thinking. These credits are listed alphabetically by main handle; e.g. last name or web site title.

Graham Mann: who always challenges me to reach further.

Numerous grad students: who always stared at me funny when I confused them, forcing me to say it again, better, simpler, more clearer. Special mention to:

- Eliza Chaing
- Lindsay Mason

David Poole: David was the first to show me the the trick of adding LaTeX command into a Prolog program. He also "lead from the front" by writing some of the most elegant Prolog I have ever seen [7].

Claude Sammut: who I believed, all those years ago, when he said that Prolog can be used for real world applications. For my masters back in the 1980s, he asked me to write a "good" Prolog expert system shell. Well, Claude, I've nearly got it right so I hope you'll give me a passing grade.

Roland Sammut and Michael Wise: who originally taught me Prolog.

The following people have never meet me, probably don't know my name, or my face. But I always sought to rise to their exceptionally high standards:

Peter Norvig: whose *Paradigms of Artificial Intelligence* [5] is an inspiration to all educators.

Guy L. Steele: The man who wrote two of the greatest technical documents in the world: *Common Lisp, the Language* and *The Hacker's Dictionary*. Any man who writes a PhD thesis where the font sizes change twenty times in a single sentence is my kind of Guy (I've been waiting decades to make that joke...).

Acknowledgements This research was conducted at West Virginia University under NASA contract NCC2-0979. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA IV&V Facility. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

References

1. I. Bratko. *Prolog Programming for Artificial Intelligence. (third edition)*. Addison-Wesley, 2001.
2. C. Date. *An Introduction to Database Systems*, volume 6. Addison-Wesley, 1995.
3. P. Deransart, A. Ed-Dbali, and L. Cervoni. *Prolog: The Standard*. Sprunger, 1996.
4. T. Menzies. Domain-specific knowledge representations. *AI Expert*, Summer 1989.
5. P. Norvig. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann, 1992.
6. R. O'Keefe. *The Craft of Prolog*. MIT Press, 1990.
7. D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: A Logical Approach*. 1998.
8. L. Sterling and E. Shapiro. *The Art of Prolog (second edition)*. MIT Press, 1994.

A License

A.1 nowarranty.txt

A.2 warranty.txt

A.3 conditions.txt

1. *prod1.pl* : “An example of the PROD Prolog delivery and documentation system.”
   Available from `http://tim.menzies.com/pdf/03prod1.pdf`.
2. *prod.pl* : “A PROlog Documentation, and Delivery Tool”.
   Available from `http://tim.menzies.com/pdf/03prod.pdf`.
3. *prod0.pl* : “TITLE”: a bare-bones minimal example of PROD.
   Available from `http://tim.menzies.com/pdf/03prod0.pdf`.
4. *prodabout.pl* : “Motivations”: the why and who of PROD.
   Available from Available from `http://tim.menzies.com/pdf/03prodabout.pdf`.
5. *family.pl* : “A family database”: documentation of a very simple Prolog family database.
   Available from `http://tim.menzies.com/pdf/03family.pdf`.
6. *lib.pl* : “Commonly used predicates”:
   Available from `http://tim.menzies.com/pdf/03lib.pdf`.
7. *cfg.pl* : “Handler for config files and command line options”:
   Available from `http://tim.menzies.com/pdf/03cfg.pdf`.
8. *gpl.pl* : “Including GPL-2 in Prod”:
   Available from `http://tim.menzies.com/pdf/03gpl.pdf`.
9. *omo.pl* : “Software cost estimation”:
   Available from `http://tim.menzies.com/pdf/03gpl.pdf`.

Fig. 2 This document is part of the PROD delivery and documentation tool for Prolog applications. To find out more about PROD, the best place to start is memo #2.

either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term ’modification’.) Each licensee is addressed as ’you’.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then

the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and 'any later version', you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS