

Eclipse and the eight embedded software life cycle truths

By Don Dingee



Software development jobs in military embedded systems are filled with life cycle challenges, and they probably are not exactly what we learned in college. I'll list them as the eight embedded software life cycle truths:

1. Embedded code has to be supported forever.
2. It's more fun to develop new stuff, but most of us get paid to maintain existing stuff (at least a non-trivial part of the time).
3. To support code, it's wise to archive both the source and the tools it was built with (like some people who store a VCR with those old tapes of the family vacation).
4. Development tools haven't always played well together, sometimes new versions of toolsets from the same vendor don't get along with previous versions, and it gets more challenging with multiple vendors involved.
5. Just when the tools are working well and we're comfortable, there's newer technology we are compelled to look at.
6. To reduce risk (both technical and personal), we wait to upgrade software development tools as long as possible, preferably when starting a new project and there is a clean break.
7. The people that developed both the code and the tools around today in all likelihood aren't the people who are going to be dealing with them in 10 years.
8. "Forever" is a long time and there is not much anyone can do about it, especially if we are the ones who just took ownership of the code in year 10.

This is admittedly somewhat pessimistic, but I'll bet this pain sounds familiar. Fortunately, there is a solution emerging to help invalidate some of these truths: enter open source and the Eclipse Integrated Development Environment (IDE).

Eclipse is very similar to that toolbox most folks have in their house. The one with screwdrivers, a hammer, a socket set, Channellocks, Vise-Grips, a few Allen wrenches, and whatever other odds and ends end up in there. It started with a few indispensable tools, and new tools are added as needed, but rarely do any get tossed unless they absolutely break or something much better is found.

Eclipse development tools are built with open source on an open-standard framework. By definition, tool developers are strongly incentivized to make things play together. Here's how it should work for developers: choose an Eclipse-based environment with tools for code development, testing, debug, source control, and build control. Now select best-of-breed Eclipse plug-ins from a variety of sources to make it exactly fit the needs. New tool functionality requirements? Get new plug-ins. New microprocessor? Update the processor support files with XML. Found a better tool than the one in use? Switch to it without throwing away all the rest of the tools in the environment.

Embedded technology vendors received the message loud and clear and are getting on board with Eclipse in droves. Here is a list of some of the embedded technologies I've found searching the Internet for Eclipse IDE support.

- **Operating systems:** Linux in several flavors including versions from LynuxWorks, MontaVista, TimeSys, and Wind River is supported. Numerous RTOSs including LynuxWorks LynxOS, Enea OSE, Accelerated Technology Nucleus, QNX Neutrino, and Wind River VxWorks are supported.
- **Processors:** 1750A, ARM, Blackfin, MIPS, Pentium, PowerPC, SPARC, TI MSP430, and Xscale processors are supported.
- **FPGAs and SoCs:** Both Altera and Xilinx FPGA tools are Eclipse based, as are Tensilica and CoWare for SoC development.

- **Languages:** Java and C++ are well supported, along with HTML, Perl, PHP, Python, and XML. Aonix has support for Ada. The University of Illinois and Los Alamos National Laboratories have support for FORTRAN. I even ran across plug-ins for Pascal and COBOL, and it looks like DDC-I is pondering Jovial support.
- **Structured development:** SlickEdit has a new version of their editor for Eclipse, and some folks at MIT have done an Emacs plug-in. Source control tools such as Rational ClearCase and Serena PVCS are supported. Catalyst Systems' Openmake helps to automate and control the build process, and Telelogic SYNERGY automates the change management process. Cadena is a plug-in developed at Kansas State University for modeling and building CORBA components.

My apologies to the vendors I have probably left out, but my point is there is a lot of embedded Eclipse technology out there, either commercially available or posted as a contributed plug-in.

What is driving this proliferation of Eclipse tools for embedded? The answer: life cycle costs. Eclipse tools help the vendor's development teams and their customers deal with the 10-plus year life cycles familiar to most developers in military embedded systems. It's a win-win situation for the vendors doing the work because it provides value for their customers while also reducing their costs.

Defense programs are already developing in Eclipse. Accenture used tools from ILOG to build graphical elements of the US Air Force Effects-Based Operation planning tools. Boeing built parts of the Bold Stroke avionics mission-control system using Cadena. As previously mentioned, Eclipse work is being conducted at Los Alamos National Laboratories and other US government labs.

Let's look at our eight embedded software life cycle truths again. This time with the potential benefits of Eclipse in mind:

1. Supporting code forever: Eclipse won't change this, but maybe Congress will pass a statute of limitations on support for fielded military embedded systems. It could happen.
2. Supporting old stuff: If we have to, it should be in the same environment as the new stuff. Eclipse is impacting this as developers solving this exact problem create more plug-ins to deal with the old stuff in the new environment, such as the team at Los Alamos dealing with their FORTRAN.
3. Archiving tools with code: This need may be greatly reduced with Eclipse. We certainly won't have to archive the environment, but there might be some older plug-ins like compilers and processor descriptions to keep around. When we get them out and dust them off 10 years from now, they won't look totally foreign.
4. Development tools don't always play together: things seem to be playing together nicely, especially in the latest Eclipse 3.0 framework.
5. Darn, we just got the bugs out of these tools: When that new microprocessor shows up, just ask for the new Eclipse plug-ins and XML files that support it. When somebody says we have to move to object-oriented techniques, get the CORBA plug-ins. There won't be as much reason to fear sweeping progress as technology advances.
6. We fear change: The risk of a tools upgrade is greatly reduced, again due to the plug-in strategy. Developers can create two Eclipse perspectives, one with the old plug-ins and one with the new plug-ins.
7. We'll be gone in 10 years: Each of us will be so productive with Eclipse that with any luck we'll advance in our careers and won't have to work on the 10-year-old stuff.
8. "Forever" is a long time: We can do something, choose Eclipse, and the problems will be lessened 10 years from now. And, there's always a chance the person we help will be ourselves.

Eclipse is only the toolbox, solving the life cycle problem really depends on the tools we select to put in it and how skilled we are at using them. What are you seeing

out there with Eclipse tools that work for you? Drop me a line.

Don Dingee is the editorial director of PCI Express Resource Guide. He has more than 23 years' experience in marketing, selling and designing embedded computing products and is the co-founder of Embedify LLC. Don's product marketing experience includes 11 new embedded computing product launches, and he co-authored the EBX specification in 1997. Before co-founding Embedify, Don's career featured leadership positions in marketing and sales at the Motorola Computer Group, and design and new business acquisition roles at General Dynamics. Don holds a Master of Science degree in Electrical Engineering from the University of Southern California and a Bachelor of Science degree in Electrical Engineering from California State Polytechnic University, Pomona.

For further information, contact Don at ddingee@opensystems-publishing.com.