

THE COMPELLING CASE FOR OPEN SOURCE EMBEDDED TOOLS

James B. Calvin, Jr., CEO
Steven L. Rodgers, CTO
Microcross, Inc.

This paper presents a case that open source embedded software development tools, popularly know as GNU tools, are ready to be widely adopted as a business strategy and a solution to software tools obsolescence, vendor lock-in, and vanishing manufacturing sources as applied to embedded systems in complex, long lifecycle computer systems that are typical of many commercial Original Equipment Manufacturer (OEM) and Department of Defense (DoD) systems.

Evolution of Open Systems

The role of embedded computer systems continues to become more dominant in all electronic systems, devices, and appliances. Nowhere is this truer than in the complex world of software intensive commercial and military systems. There are many examples of OEM and military systems that within their lifespan have encompassed as many as four generations of computer technology. During this period, software development tools and languages have experienced similar evolutionary changes. An interesting contrast is, that while developers and maintainers of long lifecycle systems are seeking more reliability, maturity, and stability in software support systems, the Commercial Off-The-Shelf (COTS) software technology providers are driving products to ever-shorter lifecycles. Moreover, software tools obsolescence has long been a major problem and due to increased systems lifetimes and lifecycle extensions, the problem is only getting worse.

On the hardware systems side, OEMs and the DoD have long recognized problems with using proprietary systems, and the DoD has responded with policy guidance promoting 'Open Systems' in acquisition programs as a method of improving weapon systems management and lowering costs. Open Systems are systems that can be supported by the market place, rather than by a single (or limited) set of suppliers due to the unique or proprietary aspects of the design. Current DoD policy for Open Systems is expressed in DoD 5000.2R and the DoD Joint Technical Architecture (JTA). The JTA lists the minimum set of non-governmental standards needed to maximize interoperability and affordability within DoD. In sum, the JTA is a single unifying technical architecture for integrating all DoD system components.

Software development tools are one of the DoD system components, and open source software tools, popularly known as GNU tools, are an evolutionary extension to the Open Systems concept in that an entire software design is exposed to open review, collaboration, and extension rather than being limited to standards for functionality and interfaces. Widespread adoption of open source software development tools as an extension of Open Systems management, acquisition, and development methodology would bring supportability, interoperability, and reuse for large software intensive systems to the next level of maturity, making tools obsolescence a problem of the past.

What is GNU?

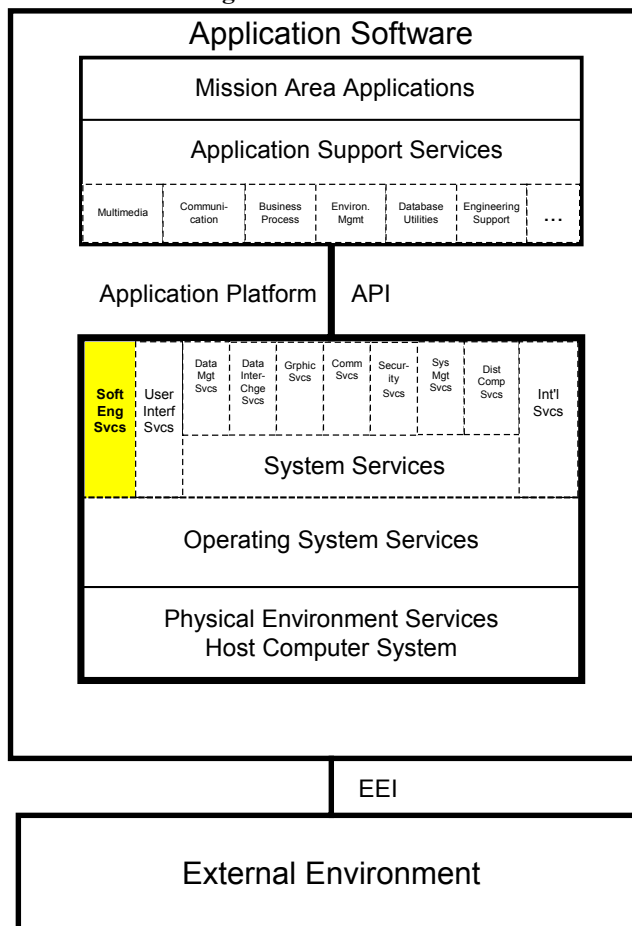
According to the Free Software Foundation (FSF), which was started in 1985 by Mr. Richard Stallman, GNU stands for 'Gnu's Not Unix' (a recursive acronym). GNU is pronounced 'guh-NEW.' The FSF was chartered to support the development of non-proprietary, freely distributable software based on collaborative development. One of the organizational goals of the FSF was to develop a free Unix implementation, which was later enhanced by Mr. Linus Torvalds and is now called Linux or GNU Linux. To meet the goal of creating a free Unix, Mr. Stallman recognized that a good, highly portable C compiler was truly a prerequisite to the success of this effort, so he set out to develop one. As it turns out, it was so well thought out, eloquent, and portable in design that over the years the GNU Compiler Collection (GCC) has been ported by contributors to almost every commercially viable computer system as a native compiler and later added other popular programming languages to the GCC, such as C++, Fortran, Pascal, Ada, and Java. Attendant to this evolution was the growth of related tools, including assembler, linker, librarian, make utility, and a debugger. Today the GNU toolchain contains every component that is expected

of a professional software development system, and it has reached a high level of reliability and maturity rivaling the best proprietary commercial toolchains. Over the past four years, GNU tools have become very appealing to large corporations like Cisco Systems, IBM, Airbus, Lucent, and Lockheed Martin to name a few. Organizations in Europe have created entire programs to promote the evolution of GNU development tools. For example, the European Space Agency (ESA) created an organization called FRESKO (Free Software for ERC32 Systems COoperation) to support the GNU tools for the ERC32 Sparc architecture. ESA's goal is to ensure the support, evolution, versioning, quality and obsolescence abatement.

Standardization and Policy

The DoD Open Systems design strategy mandates standards that have been adopted and approved by a sponsored standards organization that created the JTA. Implementation of the JTA, or the use of applicable JTA mandated standards, is required for all emerging or changes to an existing capability that produces, uses, or exchanges information in any form electronically; crosses a functional or DoD Component boundary; and gives the warfighter or DoD decision maker an operational capability. It includes a Technical Reference Model (TRM) that describes, from the foundation up a services view of the architecture. It includes physical environment services (host computer system), the operating system services, and the system services such as software engineering services, user interface services, and data management services. Figure 1 illustrates the services view of the TRM. It is the software engineering services portion of this model that is of interest to us (shown with yellow background). It provides system developers with the tools that are appropriate to the development and maintenance of computer applications. Although it has been identified as a major component of the TRM, there are no mandated standards for this services area, and a viable approach is needed. The operating system API is the interface between these services and the host platform operating system. The currently mandated operating system API standards are 1) ISO/IEC-9945-1 (POSIX), which is the Unix operating system, and 2) the Microsoft Win32 API, which is not an open standard, but must be listed due to its current pervasive use.

Figure 1. TRM Model



The ideal software engineering services component of the DoD TRM would consist of a rehostable, retargetable, API compliant software development system supporting both native and cross development and would include the following attributes:

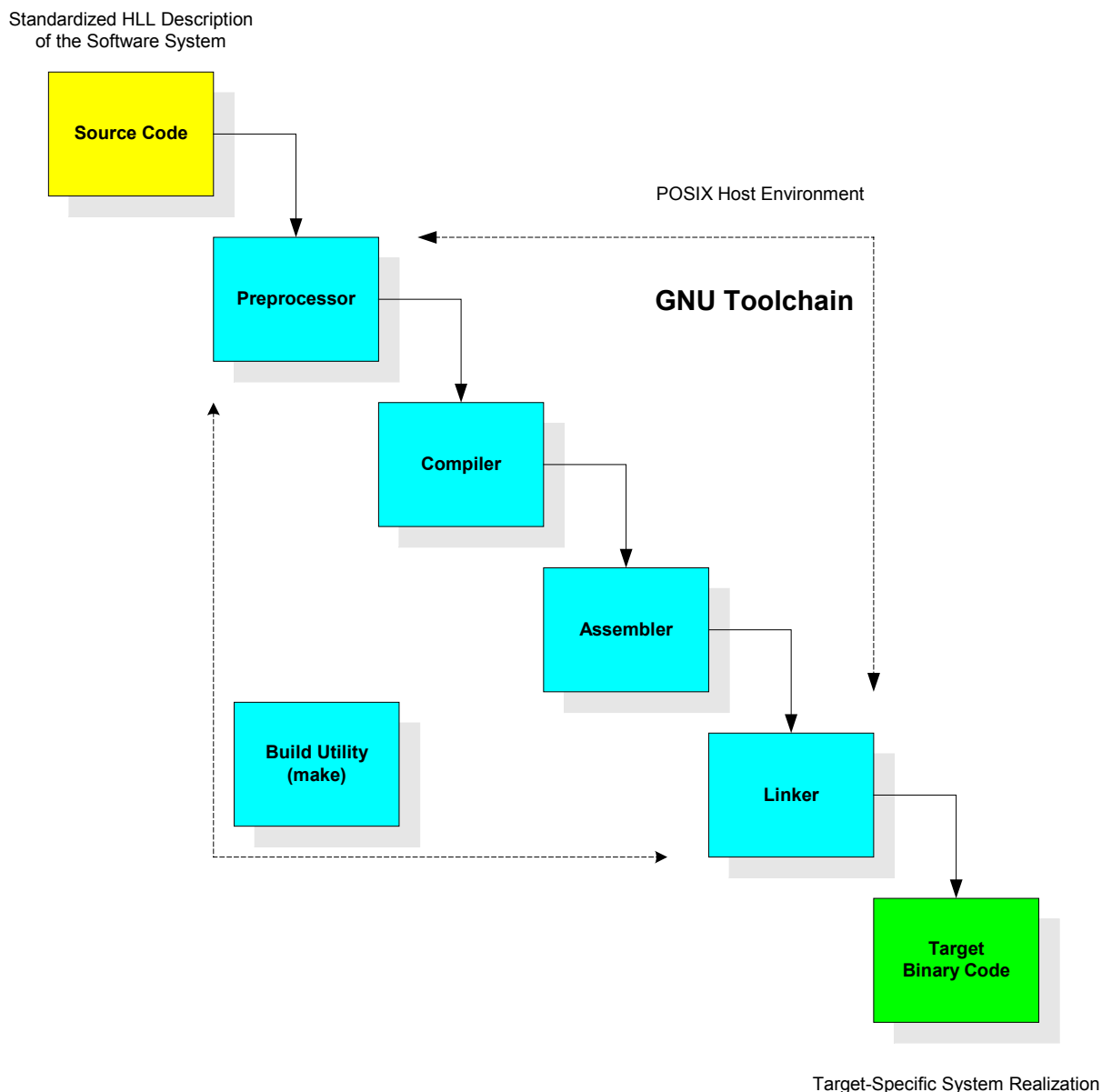
1. Include support for all widely-used and standardized compiler languages.
2. Provide all related utilities and productivity tools comprising a complete software development system.
3. Provide JTA compliant and uniform operator interfaces and commands.
4. Possess uniform assembler language and linker syntax.
5. Demonstrate ease of portability to multiple host platforms.
6. Support a wide range of target CPUs and be retargetable to new CPUs.
7. In its entirety be unrestricted open source software.
8. Be supportable over an indefinite period through both organic and COTS sources.
9. Have a long-term record of stability, reliability, and supportability.
10. Have a low Total Cost of Ownership (TCO).

A software development system built around a standardized toolset exhibiting the above attributes should become the preferred solution for software developers and maintainers of long lifecycle embedded systems who now view selection, sourcing, and sustaining the software development toolset as one of major management challenges of the system lifecycle. Over the past 10 years, the GNU compiler and related tools have quietly evolved to the extent that these criteria are now being met. Supporting all popular computer languages, the GNU toolchain is the only universal toolset that can achieve ubiquity.

GNU Tools as a TRM Component

The GNU toolchain structure is typical, consisting of preprocessor, compiler, assembler, and linker -- all under the control of a Unix-style Make utility. Figure 2 shows how the toolchain sequentially processes the input High Level Language (HLL) to produce a loadable system image for execution on a target CPU. The GNU toolchain design is atypical in that it was planned to be open and expandable by future contributors to the FSF. The toolchain design is modular, and its table-driven features facilitate porting to new target CPUs and host computers with relative ease. Because of the elegant design approach and open source nature of the GNU toolchain, it exhibits all of the ideal attributes of a POSIX compliant toolset; thus, it meets all requirements in the DoD 5000.2R and the JTA and should be adopted in the TRM as part of its Software Engineering Services Component. The JTA's adoption of GNU tools would have a profound impact in awakening embedded software developers and managers to a mature, standardized toolchain that has more benefits than any competing alternative.

Figure 2. GNU Toolchain



Adoption Benefits

The GNU toolchain is completely open, and it is a development system having no restrictions on use or modification of binaries and sources – freedom to do whatever the user wishes. This attribute is the only true insurance against software tool obsolescence now hindering many long lifecycle software intensive embedded systems. Because of this, multiple sources for organic and commercial sources for support and maintenance can exist.

Developed within and as part of the Unix (POSIX) software domain, GNU tools have a straight-line maturity path nearly 18 years long. When host platform portability and interoperability are desired, there are no better alternatives, and POSIX compliance is already mandated as part of the DoD JTA. Because GNU tools are POSIX compliant, they are directly compatible with the DoD JTA.

Because the GNU Toolchain is open source software, it is the most understood and widely supported compilation system available today. Its freely available sources and design are studied and improved upon by thousands of software engineers worldwide. The GNU tools are the most peer-reviewed software in the world, which is why there is such high quality in their design and bug-free operation. Both free and commercial sources of support for

GNU tools exist. This promotes the existence of a large pool of subject matter expertise that seldom exists with proprietary software systems.

In spite of this freedom and accessibility, the evolution of the ‘officially maintained’ sources is tightly controlled, and integrity of the system may be easily assured when required by commercial support organizations. At least a dozen companies worldwide commercially support the GNU tools and other OSS products.

The unity of design and long-term stability of GNU tools drastically reduces training costs and preserves career investments in GNU toolset proficiency. The cost of retraining a large software team is usually a major cost impact and sometimes unplanned.

Another great benefit is cross-native duality; it is a valuable test and validation attribute of GNU tools that is offered with few, if any proprietary development systems. Because the GNU tools are normally hosted in native form on the development host platform, development source code may be built and tested on the host platform in native mode, as well as on the cross target with the same GCC compilation system with no changes to the code under test. Moreover, on most cross-compiler versions of the GNU tools, the toolset includes a target CPU simulator that supports yet a third execution mode (targeted simulation) in which cross compiled code may be executed on the host target simulator. These three independent code, test, and validation features provide an unprecedented capability unavailable from proprietary tool vendors.

The GCC was originally designed for command shell based operations, but now GNU tools are highly integrated with several open source, as well as proprietary, Integrated Development Environments (popularly called IDEs). A couple of these environments are particularly well suited because they use of the GNU Make as the build tool and manage the project by manipulating the makefile. This approach provides the unique ability to execute a project build from within the IDE or from a standalone shell, or better yet from another makefile. These batch operations are often needed to support maintenance and production builds of large systems that are not supported by many COTS IDEs.

Most Quoted Objection

Open Source Software cannot be a high quality product. The GNU tools have been evolving on a straight maturation path since 1985. The requirements have not changed significantly, nor have they been ‘re-invented’ every few years at the whims of marketers. The GNU source maintainers utilize an effective bug reporting, peer review process, and correction methodology that is also proven and mature. The GNU compiler and associated tools have been benchmarked and evaluated by independent studies and organizations, and though recognized not as always the best performer for a given target CPU, it always performs respectably and does a superior job at insuring code correctness. In the authors’ experience in code porting, the GNU compiler often identifies coding errors and incorrect constructs that went undetected by ‘high-quality’ proprietary COTS compilers. A final testament to the robustness and quality is the fact that some of the most robust, tested, and stressed software systems in use today are compiled using the GNU tools, including several Unix operating systems (including Linux), DARPA TCP/IP protocols, Apache Web Server, and the European Space Agency’s embedded spacecraft software systems to name a few.

Business Case for GNU Tools

Acquisition cost of the GNU tools is by far lower than competing proprietary tool vendors. Cost can be as low as free (if the developer is willing to invest significant time to build, test, and validate the tools) to as much as \$500 for a limited support, shrink-wrapped COTS product, and as high as \$2,000 for a fully supported product. These prices compare favorably to proprietary tools cost, which ranges from \$3,000 to \$75,000 for a comparable functioning product. Moreover, the proprietary COTS tools vendor locks the user into a product that may have an uncertain future in regards to long-term support and configuration control; also, no source code will be available to counter the obsolescence threat and enable developers to fix defects or make necessary modification for a unique system requirement.

Although multiple sources were once a problem in the past, GNU tools are now available through multiple vendors, including the Free Software Foundation and several commercial companies, including Microcross, Red Hat, and Wind River. On the other hand, proprietary software products are essentially single sourced, and if that source vanishes, so does the support and ability to fix defects – back to the problem of obsolescence.

Supportability over the long-term (5-20 years) is of major concern for OEMs and the DoD on modern embedded computer systems. The proprietary COTS tools vendor is market driven due to economics, and long-term supportability is not always high on the priority list as evidenced by the obsolescence problems confronting the OEMs and DoD today. The best assurance possible to ensuring long-term supportability and preventing tools obsolescence is using the GNU tools.

Security is an issue in many software intensive systems. When the developer/maintainer is vendor-locked by proprietary systems, there are few, if any, alternatives to solving a security issue (bug). If the developer/maintainer uses the GNU tools, then by nature of having the source code under control is assurance that security bugs can be resolved quickly and inexpensively by using internal tool maintainers or support purchased from a reputable open source company where there are multiple sources to select from. If, however, the developer/maintainer is totally dependent on a proprietary tools vendor and this vendor vanishes or decides against making fixes, the security problem will become virtually unsolvable.

Scalability is an attribute desired in all software intensive embedded systems, and it is a key feature of the GNU tools. From small, deeply embedded applications to large-scale systems-on-chip solutions, the GNU tools are widely accepted and used. For example, on the low-end the GNU tools are being used to build 16-bit embedded computer applications, such as control systems, medical devices, computer peripherals, and consumer electronics. On the high end, the GNU tools are being used to build 32/64-bit native and embedded applications that include the Unix and Linux operating systems, Internet routers and switches, telecommunication soft switches, aerospace navigation systems, cockpit controls and displays, electronic warfare systems, and general transportation control systems.

Conclusion

The authors have briefly presented a technical and business perspective on what the GNU toolchain is, its potential role as a standardized embedded development platform, and why it makes logical sense to adopt their use in long lifecycle systems. Widespread adoption of the GNU tools can have a coalescing effect on the entire science of embedded software engineering, without the attendant commercial domination of a single vendor and the disruption of continual product reinvention. This standardization would bring commonality to training, creating a sustaining labor force of embedded developers with skills right out of college; it would also bring down the cost of system development, providing lower cost goods and services that use embedded computer technologies. Colleges and universities are already adopting the GNU tools for both the benefit of commercial industry and the student. The GNU tools are the only viable solution to prevent vendor lock-in, vanishing manufacturing resources, and tools obsolescence. The authors highly recommend that the DoD independently analyze the compelling benefits and adopt open source as part of the Open Systems architecture and mandate and promote the GNU tools as part of the JTA TRM.

REFERENCES

1. *Open Systems and the Systems Engineering Process*, Michael Hanratty, Robert H. Lightsey, & Arvid G. Larson, Acquisition Review Quarterly Published by the Defense Systems Management College, 1999, <http://www.dsmc.dsm.mil/pubs/arq/99arq/hanratty.pdf>
2. *A Business Case Study of Open Source Software*, Carolyn A. Kenwood, Mitre Corporation Publication MP01B0000048, 2000
http://www.mitre.org/support/papers/tech_papers_01/kenwood_software/
3. *Open Source Initiative (open case for business)*, 1999-2002, <http://www.opensource.org/advocacy>
4. *Guidelines for Successful Acquisition and Management of Software Intensive Systems*, Department of the Air Force Software Technology Support Center, Feb 1995 – 2001,
<http://www.stsc.hill.af.mil/>
5. *Embedded.com Buyer's Guide*, 2000, <http://www.embedded.com>
6. *DoD Joint Technical Architecture (JTA)*, Version 4.0, 2002, <http://www.disa.mil>
7. *DoD 5000.2R, Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs*, 2002,
<http://www.acq.osd.mil/ar/library.htm>
8. *FRee Software for ERC32 Systems COoperation (FRESCO)*, 2002,
<http://www.estec.esa.nl/wmwww/EME/compiler/Fresco/>

ABOUT THE AUTHORS

James B. Calvin, Jr.

Mr. James Calvin is co-founder and Chief Executive Officer at Microcross, Incorporated. He is a former military officer and engineer in the USAF. He has over 22 years of experience in software development and program management supporting aircraft operational flight programs, system integration and flight testing. Mr. Calvin is championing the cause of free and open source software tools as a solution to tools obsolescence in long lifecycle, software intensive support programs that are typical of military and aerospace systems. Mr. Calvin is a native of Jackson, California and holds a BEE from the Georgia Institute of Technology, an MSEE from the Air Force Institute of Technology, and an MSA from Central Michigan University.

Microcross, Inc.
151 Osigian Blvd., Suite 154
Warner Robins, GA 31088
Phone: (478) 953-1907
E-mail: james@microcross.com

Steven L. Rodgers

Mr. Rodgers is co-founder and Chief Technology Officer at Microcross, Incorporated. He has over 30 years of engineering experience in multi-disciplined environments having hardware and software development projects as complementary activities. Previously, Mr. Rodgers served as chief engineer at the TRW Aerospace and Defense Electronics Division in Georgia. Mr. Rodgers has been embedding computers since 1979 (pre-PC era) and is most noted for work as a pioneer in using the PC (circa 1985) as an embedded computer in aerospace and DoD / Government applications. Mr. Rodgers is a native of Macon, Georgia and holds a BEE from the Georgia Institute of Technology and several industry certifications in voice/data communications and network engineering.

Microcross, Inc.
151 Osigian Blvd., Suite 154
Warner Robins, GA 31088
Phone: (478) 953-1907
E-mail: stever@microcross.com