

Communications middleware – building versus buying to attain platform manageability and high availability

By *Jim Lawrence*

This article examines the build versus buy decision facing Telecommunications Equipment Manufacturers (TEMs) for software components that support availability, management, and serviceability in 5- and 6-nines business critical communications applications.

The task of building telecommunications systems has experienced multiple paradigm shifts in the last decade and continues to morph in response to both changing carrier requirements and available platform technology. Until the late 1990s, TEMs relied on in-house resources to design and build highly customized hardware and the entire software stack that ran on that hardware, bottom to top. The communications industry slowdown that began in 1999 and culminated in the dot-com bubble bursting in 2000-2001 left TEMs with vastly reduced human resources and shelved legacy designs that did not meet emerging converged market requirements.

To meet this new reality, equipment manufacturers have moved from switched to packet data and from segregated voice and data to converged systems. They have also opted for Commercial Off-the-Shelf (COTS) systems, CompactPCI, and AdvancedTCA, and migrated from *roll-your-own* Operating Systems (OSs) to commercial RTOSs and Linux. Most recently the move has been from proprietary software development to Open Source.

These trends are reflective of an ever-rising *value line* that separates commodity from differentiating technology. Today, that value line for TEMs and other OEMs encompasses systems software and is already impacting choice of applications-enabling middleware. Despite this rising tide, communications systems manufacturers continue to invest valuable resources in middleware, notably in the area of availability and fault resilience.

Consolidation through modularity

Traditional TEMs' engineering practices emphasized autonomy, for better or worse.

Different groups designed and built line-cards and blades, shelf controllers and appliances, and systems controllers and back-end systems. Through the vagaries of product line evolution, product end-of-life, and mergers and acquisitions, equipment manufacturers frequently developed and deployed highly divergent mixes of hardware platforms, embedded OSs, enabling software, and applications software components.

Despite mandates for technology reuse, TEMs often found themselves facing licensing, training, development, and support burdens for half a dozen or more proprietary and commercial (RT)OSs programmed in a mix of C, C++, Java, assembly, and in-house languages, running on a mix of little- and big-endian COTS and custom CPUs. The daunting task of integrating value-added applications and enabling software integration primarily occurred in-house, regardless of cost, due to the complex interaction of software and hardware components.

Old habits die hard. Even with the advent of COTS hardware (AdvancedTCA) and COTS software (Carrier Grade Linux, high availability software, and ecosystem software components), many TEMs continue to invest *below the value line* by maintaining legacy and/or building new enabling software, which may be mission-critical, but does not provide differentiating functionality.

Scope of OEM value-add

Software content in intelligent devices of all types is doubling annually and long ago outstripped hardware as the nexus of communications product value-add. Legacy communications systems embodied millions of lines of code, with no particular attention to the locus of OEM value-add. Today's converged voice and data systems can embody tens of millions of Lines of Code (LoC). The Linux kernel can include up to 4 million lines and the rest of a carrier grade OS up to 5 million more. Development tools and

"In today's marketplace of standards-based blades and intelligent chassis, implementing standard interfaces and correctly driving standards-compliant hardware is both nonoptional and nontrivial."

IDEs consume 5-6 million more LoC. When user interface, networking, and utilities are accounted for the code base quickly adds up to an integration and support burden that can top 30 million LoC – incompatible with the resource realities of today's OEM.

The scope and size of TEMs' value add, conversely, has remained fairly constant (approximately 500,000 – 1,000,000 LoC), but it has moved higher and higher in the software stack. TEMs, rather than seeking to take on increasing software loads, look to streamline software investments for improved ROI and faster time to market.

Standards compliance

In systems built from specialized, proprietary hardware, or even integrations of in-house and COTS hardware, it was easy to justify investments in custom software for manageability and availability. It was also the norm to *add value* by enhancing standard interfaces and protocols for optimal performance (and customer lock-in).

In today's marketplace of standards-based blades and intelligent chassis, implementing standard interfaces and correctly driving standards-compliant hardware is both nonoptional and nontrivial. Certainly, in-house code can be built or modified to implement or interface to systems based on AdvancedTCA, BladeCenter-T, IPMI, Service Availability Forum (SAF) AIS/HPI, SNMP, and other industry standards and specifications. However, there is a large and important difference between building a "one-off" that implements standard interfaces and creating a truly standards-compliant platform that enables interoperability and ecosystem partnering. In particular, standards compliance involves:

- Acquiring/licensing complete standards documentation and validation code
- Exhaustive implementation and testing of standard interfaces and protocols
- Assuming risk of warranting standards conformance for customers and partners
- Ongoing support and maintenance of evolving standards across product revisions

These activities, for example, building and applying a regression test base for ongoing standards compliance and conformance, can be costly and time-consuming. In the past, when TEMs' value emphasized vertical integration, those vendors could charge a premium for such infrastructure work. In today's distributed COTS-centric marketplace, standards compliance represents a core requirement, but not a differentiating one. As such, the mechanics of standardization are better left to "experts" so that equipment manufacturers can concentrate

on their core value-add – video, voice, and data applications that enable next-generation services.

Open Source multiplier

Open Source and standardization go hand in hand. The rapid advance of Linux, Apache, MySQL, and other IT infrastructure projects builds on key standards like POSIX, TCP/IP, HTTP, HTML, and SQL, while extending those standards and creating new ones and new channels of interoperability (for example, LSB, Carrier Grade Linux, PAM, Python, and PHP). This same symbiotic relationship applies to enabling technologies in telecommunications: synergies among Carrier Grade Linux and the standards it embodies – LSB, SAF, PICMG – have not only enabled platform providers and TEMs, but spawned new initiatives and ecosystems including LTP, OpenPOSIX Test Project, SCOPE Alliance, and Communications Platforms Trade Association (CP-TA). Until recently however, TEMs had to choose from a limited, mostly proprietary menu of middleware for availability and fault resilience, as Table 1 shows.

Proprietary options merely substitute commercial problems of lock-in and vendor viability in place of challenges surrounding internal resource allocation. The emergence of 100 percent open source options, while promising, has not removed the burden of integration, test, and maintenance from TEMs' budgets. Moreover, fragmented building blocks, however open their source code may be, do not engender the needed community-based implement-deploy-enhance cycle that has driven success in other areas of open source. The optimal combination – prepackaged and integrated open source COTS middleware for availability and reliability – simply did not exist.

Build versus buy

Even after three decades of embedded software development, the build versus buy decision still appears complex and remains emotionally charged. However, what was once "rocket science" is being reduced to mere computer science and economics. Volume deployment of software in the enterprise results in constant incremental gains in reliability of base platforms like Linux. The "many eyes make all bugs shallow" axiom now also applies to the realm of embedded software and even to the rarified domain of high availability and fault resilience. Today, while hardware building blocks boast MTBF ratings that exceed human lifetimes (a mere 650,000 hours), reality dictates most faults still emanate from hardware – hard disks, RAM, network links, bus connectors, and power supplies.

The need to manage those points of failure, as well as actual software faults, dictates the ongoing and indeed increasing need for high availability middleware. However, the same economics that drive COTS hardware and commodity software adoption also impact the remainder of the software platform, all the way up to TEMs' own differentiators of carrier service enablement and brand. In a world now dominated by outsourcing and considerations of cost and time to market, the debate is no longer "why buy?" but "how to buy" with lowest risk and vendor lock-in. Increasingly, the answer to that question is: *commercial open source*.

OpenClovis offers a mature code base in two licensed flavors: 100 percent royalty-free GPL version 2 source code for noncommercial use, and a commercially licensed version of the same code base, accompanied by support, training, warranty, and the other values traditionally only associated with proprietary

Option	Challenge
In-house legacy middleware	Marginal investment; low value-add
Middleware bundled with hardware platforms	Mixed quality and hardware lock-in
Third-party proprietary middleware	ISV lock-in and vendor financial viability
Nascent open source components	Immaturity and lack of component integration

Table 1

offerings. Figure 1 shows the OpenClovis application service platform architecture.

OpenClovis source code is just that – open. With no commercial commitment, OEMs can examine, evaluate, and even prototype not only the open APIs and interfaces but also the underlying implementation of key functions like message passing, heartbeat, chassis management, and failover. The open code base gives OEMs, other ISVs, platform manufacturers, and even individual developers the ability to create patches and contribute to the evolution of the platform in ways impossible with proprietary code. For rapid advancement, nothing beats an open community. 🌐



Jim Lawrence has been a leader in the communications industry for more than 25 years. Prior to OpenClovis, he developed organizational and product strategies that have contributed to the growth and market leadership of several organizations and products within Intel and Motorola. Jim is an active member of the Service Availability Forum Board of Directors and Open Communications Architecture Forum. Jim has a Masters in Program Management from George Washington University and a BS from the University of Maryland.

To learn more, contact Jim at:

OpenClovis
 1310 Redwood Way, Suite B
 Petaluma, CA 94954
 707-285-2852
jim.lawrence@openclovis.com
www.openclovis.com

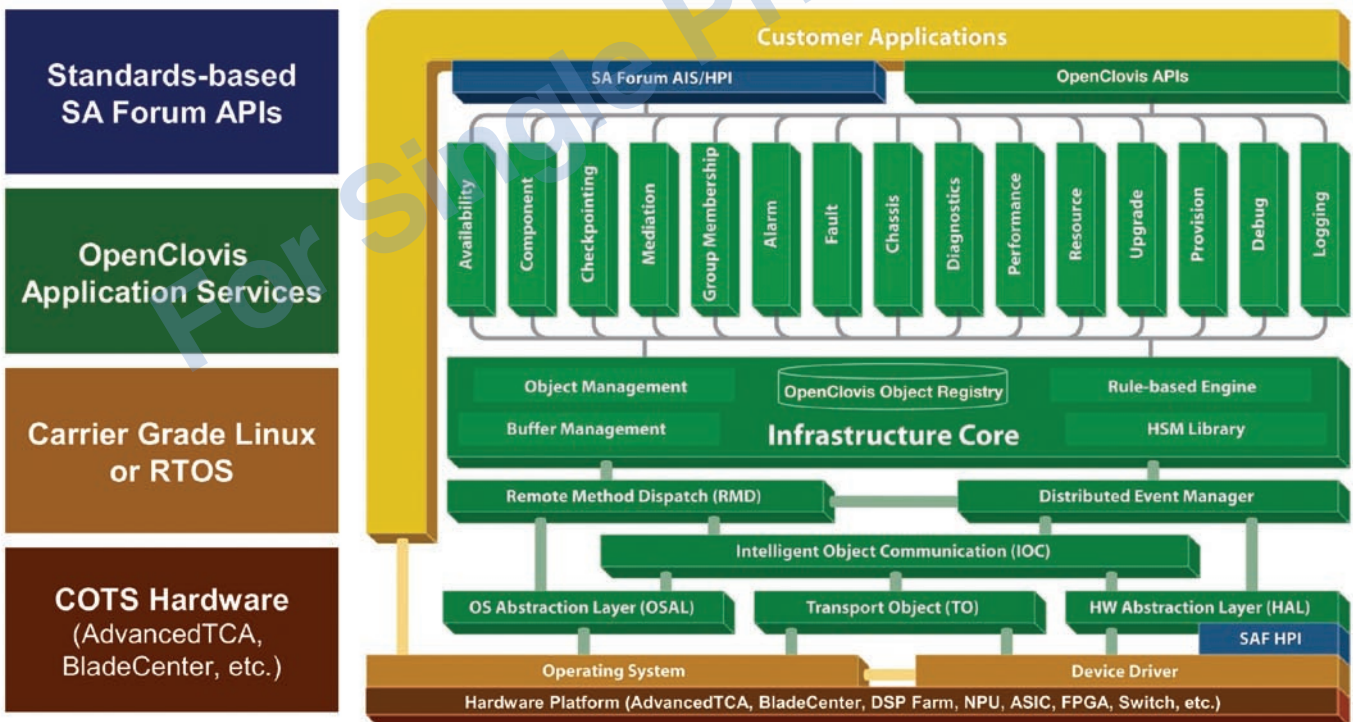


Figure 1