GENIVI: moving an industry to open source

By **Michael Kerrisk of LWN.net**

This article was originally published on 8 August 2012, at the following location: https://lwn.net/Articles/510112/.

Given this editor's recent history (3 years working there), an article on the GENIVI Alliance was perhaps inevitable, and perhaps it's better done sooner while the experience is still fresh. However, GENIVI is more than just a matter of personal interest and experience: the development of GENIVI has some interesting lessons on the adoption of free and open source software, and the results of the consortium's work are soon likely to be directly visible to many readers of this article on a daily basis.

Goals and history

The first question of course is: what is GENIVI? The brief answer is that it is a consortium of companies whose goal is to define a standardized common software platform for developing in-vehicle infotainment (IVI) systems and to nurture a development ecosystem around that platform. IVI systems, known in the trade as *head units*, are the computer systems commonly found in high-end cars and commercial vehicles—and increasingly in mid-range cars and other vehicles—that provide a range of entertainment and other functions.

The name GENIVI is an acronym based on the name of the Swiss city, *Geneva*, and "IVI". Geneva holds no special significance to the consortium beyond the fact that it hosted an industry meeting where some early discussions about formation of the consortium took place; the consortium itself is incorporated in the United States as a 501(c) (6) nonprofit organization.

Typical IVI functions include control of the media system (e.g., music player, radio, rear-seat video), navigation assistance and location-based services, and display from the rear-view camera on vehicles that provide one. Input to a modern IVI system is via physical wheel devices and buttons or a touch-screen interface, and, commonly, voice recognition. Many modern IVI systems provide integration with consumer-electronics devices via technologies such as Bluetooth and USB. The most interesting such devices are of course smart phones, where integration with the IVI system allows functionality such as playback of media hosted on the phone and hands-free, voice-activated calls conducted via the head-unit audio system. This type of integration also allows conveniences such as automatically turning the volume of the radio down when an incoming caller rings the phone.

The formation of the consortium was announced at the start of 2009, although there is some prehistory to its foundation that we'll briefly return to in a moment. The founding membership consisted of 8 companies that typify several categories of what is by now a much larger membership: automotive manufacturers (BMW Group, PSA Peugeot Citroën, General Motors),

tier 1 automotive suppliers (Delphi, Magneti-Marelli, Visteon), Silicon vendors (Intel), and operating system vendors (Wind River Systems, then an independent company, now a subsidiary of Intel). During the subsequent three years, membership in each of the aforementioned categories has swelled (notably, a number of ARM Silicon vendors now balance out the heavyweight Intel among the Silicon vendors). In addition, ISVs (independent software vendors), middleware vendors, and software services companies with an interest in the automotive sector have also joined the consortium, with the result that the GENIVI membership has now grown to over 150 companies spread across Europe, America, and South and East Asia.

Above, I said GENIVI's goal is to define a standardized common software platform. That platform is *not* a complete IVI system. Rather, it is a packaging of operating system and middleware components that implement a range of non-differentiating functionalities that all IVI systems require. (Bluetooth connectivity is an example of what automotive manufacturers might consider non-differentiating functionality: manufacturers want a working implementation, but don't market their IVI systems to customers based on the Bluetooth implementation.) In effect, one of GENIVI's goals is to decrease the cost of developing the base system, so that developer resources can be devoted to innovating at higher levels in the software stack, such as the human-machine interface.

Linux and open source software were chosen as for the GENIVI software platform during an evaluation project that predated the foundation of the consortium. That project (conducted by BMW, Magneti Marelli, Intel, and Wind River) was motivated by the desire to balance two opposing requirements. On one side stand ever-increasing demands on the development and scope of IVI systems: to drivers, an IVI system starts to look more and more like other consumer electronics devices, and drivers expect to see similar levels of functionality and rapid development cycles. Furthermore, there is a market pressure to see IVI systems in all vehicle categories, rather than just the high end. On the other side, the costs of IVI system development have grown astronomical—a figure of $100 million to bring a solution from the drawing board to dashboard is not unheard of, and such costs are becoming intolerable for all but the largest automotive manufacturers.

In the evaluation phase, a number of platform alternatives were considered, including proprietary systems such as Windows CE and QNX. However, it quickly became clear that a platform based on Linux and free software had the most to offer, based on factors such as the economies available from reuse of software components and, importantly, the realization that free software would allow the greatest degree of control of the content and development of the platform. On that basis, the evaluation project embarked (successfully) on a proof-of-concept implementation of a prototype head-unit system based on Linux and free software components.

GENIVI outputs

In addition to code projects worked on by members, the consortium produces two primary outputs: a compliance specification and a baseline software release.

The goal of the compliance specification is to ensure that compliant GENIVI products *ease integration* of third-party software components, rather than guaranteeing full API or ABI

compatibility across implementations. (In other words, GENIVI doesn't set out to be a standardization body.) Compliance is currently based on self-certification, but in time the plan is move to a more test-driven form of certification. The compliance specification is currently a members-only document.

The GENIVI baseline software release is essentially an internal proof-of-concept for the compliance specification. It is a packaging of the components required by a specific release of the compliance specification on top of a Linux distribution. The baseline isn't directly available outside the GENIVI membership, but is indirectly available via a number of GENIVI respins created by incorporating the baseline component set into an upstream distribution. These respins are created by GENIVI members and available to anyone for download. GENIVI respins are currently created for Ubuntu, Tizen, and Yocto.

What is the problem that GENIVI is trying to solve?

Technically speaking, implementing partial or complete IVI systems on Linux isn't fundamentally different or more difficult than using the platforms traditionally employed in the automotive industry. The pre-GENIVI proof-of-concept work, the recent Cadillac CUE system, and a range of demonstrator systems that appear at the twice-yearly GENIVI member meetings provide ample evidence of that fact. This raises the questions: why don't we already have (more) Linux-based IVI systems on the road and why is an alliance like GENIVI even necessary?

To answer those questions requires understanding that GENIVI's objective is not to solve a software *technical* problem, but rather to solve a software *business* problem. Escalating software costs mean that automotive manufacturers need to escape their traditional cycle of constantly re-implementing individually developed, tailor-made solutions for their IVI systems. The name of the game is to *share development costs* by collaborating on the development of a common, reusable software platform. The challenge then becomes: how does a diverse group of companies transform their traditional business and software development practices, stepping toward a new model to collaboratively define a common platform and bring it to reality? In practice that has proved to be quite a challenge.

The rocky road from prototype to product

To see why the path forward for GENIVI has been difficult requires some understanding of the traditional software development model in the automotive industry.

The traditional approach to IVI software development is rather waterfall in style: the automotive manufacturer develops a set of requirements and then, armed with a large checkbook, enters into a contract with a tier 1 supplier who does all of the software development to fulfill those requirements (in fact, the tier 1 supplier is often tasked with delivering the whole package of IVI hardware and software). Once development is completed, the manufacturer black-box tests the resulting software, and then ships it in vehicle head units. (In this traditional approach, it's worth noting that the manufacturer typically has few software engineers, and does little software development.)

Given their historical role as holders of the checkbooks, it's perhaps unsurprising that automotive manufacturers at first tried to remake GENIVI in the mold that was familiar to them. Thus, in its initial incarnation, although GENIVI's stated goal was to create a (largely) open source platform, the proposed development process was rather waterfall style, driven from the top down by the automotive manufacturers. The proposed process consisted of traditional phases: gathering of requirements, discovering an architecture, mapping the architecture to software components, and then selecting (existing) open source software components, and implementing new components to fill the gaps. Waterfall-style development is prone to be complex and time consuming; what made it even worse in GENIVI's case was trying to the scale the development process to handle multiple participating companies.

For many readers, it is probably no surprise that the results of trying to employ such a model to select and create open source software were not as successful as hoped: internal teams got bogged down in trying to define the process, and the alliance found it too unwieldy to implement in practice. Further complicating the problem was the fact that information was not open equally to all members of the alliance (there were restrictions on access to information such as draft specifications and other in-progress work according to the paid-for level of membership). The consequence of that differential access to information was to further impede participation in the work of the consortium.

What happened in response to the early low participation levels is something of a textbook lesson for any company, or, more particularly, any industry group trying to move to open source. Recognizing the problem, the consortium's Board of Directors implemented some simple but radical steps: membership-based restrictions to information inside the consortium were abolished and the top-down waterfall model described above was replaced by requirements gathering and implementation driven from the bottom, via domain-specific "Expert Groups" that any interested member-company was free to participate in. The results of these changes became apparent quite rapidly: the level of mailing-list traffic, wiki activity, scheduled face-to-face meetings, and code contribution all increased dramatically.

Engaging with the open source community

Having read this far, the question you may be left wondering is: is GENIVI open?

From a process perspective, the answer is no. Access to various internal resources such as the wiki, issue tracker, and mailing lists is limited to the (paying) membership. Similarly, attendance at face-to-face meetings is limited to the membership. However, the boundary between members and nonmembers is already somewhat permeable. For example, a number of open source developers with relevant expertise have been invited to GENIVI meetings and provided valuable input—among them Kay Sievers and Lennart Poettering (systemd), Marcel Holtmann (BlueZ, ConnMan, oFono), Samuel Ortiz (ConnMan), and Kristian Hoegsberg (Wayland). In time, it can be expected that the boundary between members and nonmembers may become even more permeable; it's an ongoing process.

From a code perspective, GENIVI is not fully open source, but it's getting steadily closer. As noted above, the GENIVI baseline respins are publicly available, but the repositories of

GENIVI-developed code are not (even though the code in those repositories is all under OSI-approved licenses). However, that situation is likely to change quite soon, as moves are afoot to open GENIVI work more widely to the outside world (so that individual GENIVI code projects have open code repositories, bug trackers, and mailing lists). At that point, it's likely that activity on GENIVI will notch up yet further, as outside developers start to take a closer interest in pieces of the code. (It should be noted GENIVI's goal is, as far as possible, to reuse open source software components; new components are developed by GENIVI members only in cases where no suitable free software component can be found. Thus, there are to date relatively few GENIVI code projects, for example, an automotive specific audio manager and a graphics layer management system; the vast majority of the components in the GENIVI respins are direct from the open source ecosystem.)

Looking in the other direction, GENIVI is increasingly participating in upstream projects, with members getting involved via code or conversations in a number of open source projects, such as systemd and ConnMan. In recent times, GENIVI has even been getting involved with kernel development, sponsoring development of kernel patches to improve D-Bus performance. (As noted in an earlier LWN article, the attempt to upstream this work has not so far proved successful. However, D-Bus is viewed as a key component of GENIVI, and it's likely that further work will be done to come up with a kernel solution to improving D-Bus performance that may be acceptable to the maintainer of the Linux networking stack.)

Challenges

There are a number of ongoing challenges for GENIVI, and one or two that remain unresolved. Some of the challenges can be easily guessed at, or can be deduced with a little reflection. For example, as with most open source projects, more contributors would always speed up development.

The process of adapting from closed software development in competition with peers to a model of collaboratively working and sharing ideas (so far, mainly within the membership) is ongoing. For companies that have not previously done so (which includes much of the GENIVI membership), contributing code under open source licenses involves educating both developers and company lawyers. But considering the heavily proprietary origins of automotive software, the progress has already been considerable.

A notable challenge for automotive manufacturers is that, by virtue of being distributors of open source software in their head units, they now need to ensure their engineers and lawyers are well educated about free software licenses. Furthermore, their code management processes need to be adapted to satisfy the obligations of those licenses, in particular, of course, the source code redistribution requirements of the GPL. By and large, the manufacturers seem to understand the challenge and are rising to it.

The GNU GPLv3 remains a so-far unresolved challenge for GENIVI. Already, a small but significant percentage of free software projects use this license, and over time more can be expected to do so. However, automotive manufacturers feel that they can't use software under this license in IVI systems. The problem hinges on the so-called anti-Tivoization clause of the

GPLv3. In essence, this clause says that if GPLv3-licensed object code is placed on a computer system, then, either the system must prevent updates to that code by all users (i.e., *no one*, including the manufacturer, can perform updates) or, if the system does allow updates to the GPLv3-licensed software (e.g., so that the manufacturer can make updates), then the software recipient (i.e., the car driver) must likewise be permitted to update the software. The automotive manufacturers' position is that they need to be able to update the software in an IVI system, but they can't let the driver do so.

The issues for the manufacturers are driver safety, and manufacturer liability and reputation. Even if head-unit systems where fully isolated from the in-vehicle networks that control safety-critical functions such as the braking system (and in most automotive architectures they are not fully isolated), there are features of IVI systems that can be considered safety-impacting. It's easy to see that accidents could result if the navigation system directs the driver in the wrong direction up a one-way street or the picture from the rear-vision camera is delayed by 2 seconds. Consequently, the manufacturers' stance is that the only software that they can permit on the head unit is software that they've tested. Since the GPLv3 would in effect require manufacturers to allow drivers to perform software updates on the head unit, GPLv3-licensed software is currently considered a no-go area. (An oft-proposed resolution to the manufacturers' conundrum is the "solution" that the manufacturer should simply void the warranty if the driver wants to make software updates to the head unit. However, that's not a palatable option: such disclaimers may or may not hold up in court, and they don't protect reputations in the face of newspaper headlines along the lines of "4 killed following navigation failure in *well-known manufacturer's car*".)

The future

With respect to IVI systems, the future for GENIVI in particular and Linux in general, looks bright. A number of manufacturers plan to have GENIVI-based head units in production within the next two years. In addition, at least one other Linux-based product (the Cadillac CUE) is already in production, and other Linux-based systems are rumored to be under development. Overall, a substantial body of automotive interest seems to be coalescing around Linux, so it's perhaps no surprise that the Linux Foundation is organizing the *second* Automotive Linux Summit next month in England. It seems likely that in a few years, we'll be living in a world where Linux in automotive IVI systems is as common as it is in today's consumer electronic devices.