

Consumer Demand Drives Continuous Delivery in the Automotive Industry

[Electronic Design](#)

Mark Warren

Fri, 2014-12-19 09:21



The past decade has seen extraordinary change in the automotive embedded-systems world. It's not long ago that voice-control, SatNav, and ABS systems were optional extras and only for high-end cars such as the Mercedes S-Class. Today, they're often standard, even on small or entry-level vehicles. Engine-management systems are now high-power computing environments, delivering fuel-economy figures that once seemed unobtainable in the first fuel-injection systems.

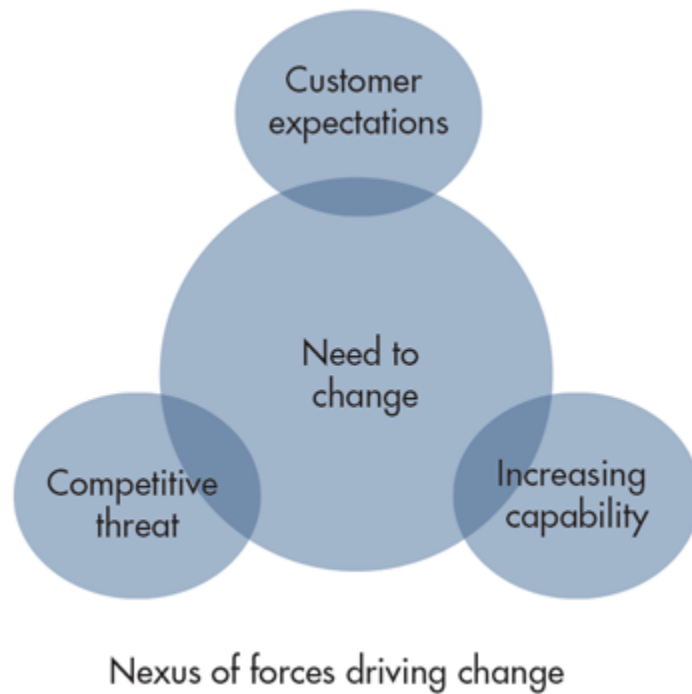
These new complexities are driving (if you'll excuse the pun) more traditional companies to adopt new processes in all aspects of their business. Companies in the automotive industry that fail to adopt certain best practices to meet increased consumer demand for "the next best thing" will soon get left behind.

Change Is All Around

To some degree, the automotive industry has been sheltered from disruptive change due to the high cost of entry. It's a brave person who decides to start a whole new, reasonably high-volume car factory today (Elon Musk may be the only example with Tesla). Cost may be the reason that many processes and systems haven't evolved since the early days of automation and electronic component capabilities, which started to appear at the end of the 20th century.

However, at least for embedded software and devices, those days are probably behind us. Today, more and more components are outsourced to parts suppliers, and the vehicle has increasingly become a "platform" for running applications, not unlike your smartphone. In this environment, the cost of entry drops significantly lower and there's room for innovation.

Such a shift could be a major threat for traditional manufacturers; they will have to surrender some control over what's being done with their product. It also has implications for international standards that aren't yet fully understood.



The status quo is no longer “good enough.” Three major disruptive forces are forcing change (*Fig. 1*):

- *Rising customer expectations*: Customers have always expected more from their suppliers, so it’s not accurate to say that expectations changed with the launch of the first iPhone. However, more than ever, customers expect modern interfaces, rapid innovation, and high quality at reasonable prices.
- *Increasing capabilities of systems*: The power of modern processors is exponentially higher than the early generations of chips. For example, NVIDIA provides chips and systems used in millions of vehicles worldwide. Its latest Tegra K1 architecture furnishes the NVIDIA CUDA system with a parallel-processing capability that, until recently, would have been considered “supercomputing” scale. What’s even more surprising is that delivering this capability comes at a reasonable cost. Moore’s law may be very close to running out of steam, but the potential for powerful automotive systems at affordable costs is staggering.
- *Financial crises have refocused manufacturers*: Since 2008, we’ve seen a major purge of unsuccessful brands, a re-evaluation of archaic processes, and a greater focus on customer satisfaction. Competition has always been strong, but technology in the vehicle has increasingly become the key differentiator. The “toys” in the cabin will become ever-more important to a purchaser in the future when choosing between brands or models.

With this nexus of powerful forces at play, all vendors should evaluate their tools and processes to ensure that they are fit for purpose and able to handle future demands. Those who aren’t equipped properly will be left in the proverbial dust.

Looking into the Crystal Ball

So, what major changes will automotive suppliers and manufacturers need to prepare for? Peter Drucker summed up the prediction well: *“Trying to predict the future is like trying to drive down a country road at night with no lights while looking out the back window.”*

Bearing that definition in mind, some surprises will likely emerge. However, some trends are easy to predict, the three most pressing being:

reducing release cycles

- Enhancing connectedness
- Using open architectures

Reducing Release Cycles

No one ever said, “Let’s release new products less often.” In reality, the drive is to continually shorten the gap between requirement identification and definition to usable output. This trend extends across all industries, even in manufacturing, as more traditional businesses follow the lead of Internet companies such as Google, Facebook, and Amazon to practice continuous delivery, as evidenced in [recent research](#) compiled by Evans Data.

Continuous delivery (CD) usually includes [a number of key elements](#), but two are worth highlighting:

- CD shortens the path from requirement or development to a customer
- CD provides developers with shorter feedback loops

For software projects, these two factors could, as in the Facebook scenario, mean that every time a programmer commits a change to the version control system, it initiates an automated build (continuous integration, or CI) with automated tests. If the changes pass, they’re pushed to live servers. In theory, every change that’s made is releasable to production.

Related

[Managing Stack Size for Automotive Embedded Systems](#)

[M2M Client-Side Challenges Emerge In Mobile Embedded System Updates](#)

[Version Management’s Role In Component-Based Embedded Design](#)

The fast feedback element is exemplified by that CI step—if the build fails, then the programmer is informed very early, enabling a quick and inexpensive fix. For many software developers, this approach mirrors a part of the agile processes. CD is seen as a natural extension of having a demonstrable feature at the end of a sprint. If a feature is demonstrable and passes its tests, then any further testing really looks for a reason why the change should *not* be made live.

Could continuous delivery be applied in the automotive environment? The answer is twofold. First, realize that “releasable” doesn’t necessarily mean actual delivery into customers’ hands, as in the previous examples. Second, in situations where physical prototypes need to be created, the “customer” might be the prototyping shop.

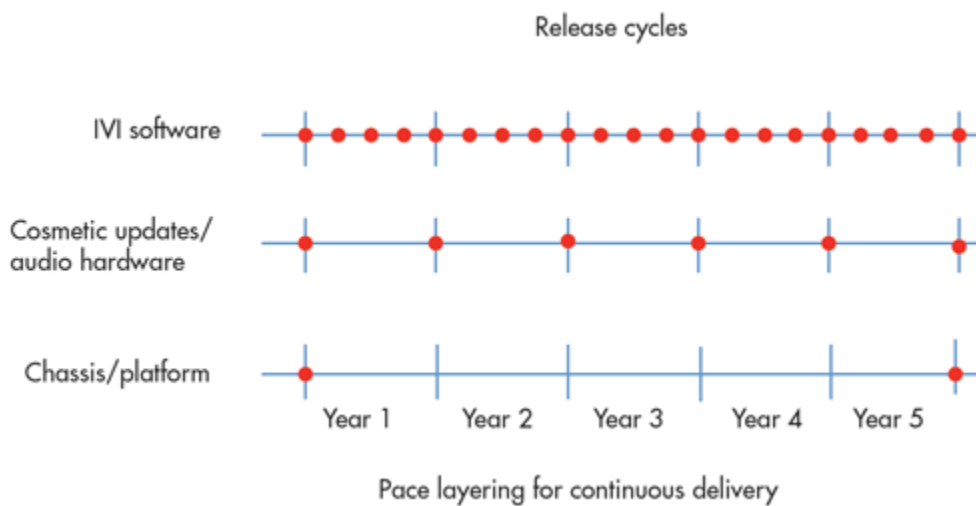
For feedback, code analyzers like those from Parasoft can enforce standards such as MISRA or look for common coding faults. Simulators and test harnesses provide ever-improving environments to develop hardware or software. This makes it possible to find and fix faults or ergonomic issues early on, preferably before creating physical prototypes or even production models.

Customer expectations are driving big changes, thus making the “continuous” part of CD even more relevant for automotive-systems developers. In a world where smartphone users expect updates to their apps on a weekly or daily basis, they also expect similar updates for their in-car systems. In the past, there might be lightweight, usually cosmetic updates each year with more substantial chassis or platform updates every five or 10 years.

ough the latter cycle has been shortened, lead times required for production tooling setup, safety testing, and other elements may always keep cycles in month- or year-long timeframes. Customers won't wait five years to get an update to the Facebook app in their IVI system.

Tesla takes this approach to heart. It regularly [rolls out updates](#) to vehicles over the air, and they're not just inessential or cosmetic changes. In the past, updates have ranged from adding new streaming music services to [avoiding vehicle fires](#).

How does one adopt CD practices? Working out a “pace-layering” architecture chart for vehicles or subsystems might prove effective, which would help pinpoint the right update cycles for each. In a banking environment, the smartphone client (“System of Engagement”) might update weekly; however, the back-end systems (“Systems of Record”) that deal with payments or interest calculations probably need to have less frequent changes for compliancy testing. For a vehicle, the chassis may change rarely, but the SatNav map data updates frequently (*Fig. 2*).



Enhancing Connectedness

Currently one of the hottest topics for many systems developers, the “Internet of Things” (IoT) is often represented by the “smart refrigerator” that tells you when your leftovers will expire, or the “smart toaster” that gets your breakfast ready right when you need it in the morning. In reality, IoT has the potential to transform the way many industries will function.

Various examples of IoT exist in the automotive world; some of them are still concepts while others have been with us for many years (even the term “Internet of Cars” is [almost old now](#)). For example, numerous toll roads or bridges offer some kind of “smart-card” system. Cars carrying a suitable card or device can drive over sensors in the road that automatically charge the toll fee to the driver (although the term “Internet of Things” certainly wasn't used [back in 1959](#)).

In regard to changes on the way, lots of interest surrounds car-to-car communications—for example, to assist with traffic-flow management or smart cruise-control systems. If a car communicates with road signs, lighting, or the road itself, then it's possible to achieve better and safer cruise control, intelligent route planning, or reliable self-driving vehicles. At a more personal level, having the car seat recognize the driver because he or she is carrying a particular phone, and in turn enabling adjustment of itself, mirrors, and the radio station, would be a boon for many drivers.

However, such innovation creates its own challenges. With few standards for machine-to-machine

communications and rapid innovation in the market, it's likely that component suppliers and vehicle manufacturers will need rapid release and update cycles. Customers may expect regular updates when new connections become available—they wouldn't expect to have to replace their car because the latest upgrade enables their SatNav to receive additional traffic warnings. This kind of rapid change suggests another reason why CD will become more critical among developers.

Using Open Architectures

Customization of cars is almost as old as the industry itself. However, future tweaks won't just involve changing the hubcaps or lowering the suspension. Instead, "users" (a broader but perhaps more forward-looking description of people in the vehicle) of the vehicle will want to tailor their driving experience just as they do on their phone or tablet. They'll decide what music services to use, or install apps to automatically order pizza when heading home from work on Friday evening. The endless opportunities make it impossible for manufacturers to predict all of the requirements that may arise during the vehicle's lifetime.

It sounds exciting for users, but incredibly complex and, indeed, risky for manufacturers. It is one thing to let users select their own music service, but what happens if they want to install their own engine-management tweaks? What if they could turn off safety systems such as the electronic stability program (ESP) or airbags? Would the manufacturer be liable? What happens if conflicts arise between the customizations or apps installed? Does the local dealer have to solve the problem?

Various initiatives exist that try to at least supply some kind of management for this increasingly complex environment. However, many of the projects being led by providers of app platforms (e.g., Apple's CarPlay) are largely an extension of their phone or tablet experience, rather than an avenue toward a better driving experience. Though the Open Automotive Alliance is starting from a more open background, manufacturers are being asked to stake some pretty large bets when choosing a platform, because it's unlikely they can support all of the different initiatives. We also probably haven't seen all of the players yet. Another as-yet-unknown platform could become a key player within five years.

Open Source in the Development Process

Stepping back from the in-car experience to look at the design and build process for embedded automotive systems, we see open architectures creating new opportunities.

Most development tools used by systems designers or firmware developers are very large and heavyweight toolsets with high licensing and maintenance costs. These tools have a hard time keeping up with demands for increased complexity and flexibility in the systems under development. As a result, teams haven't always been able to stay current with innovations in other industries—for example, the rise of agile methods and tools such as open-source distributed version control systems (DVCS), in particular Git.

DVCS is important to developers for numerous reasons, not least of which are the systems' very high-performance and rapidly growing community of advocates. Any graduate software engineer is likely to know Git but highly unlikely to be acquainted with IBM's legacy version management tool ClearCase, still used in many shops that haven't prioritized its replacement. These new recruits are starting to bring Git into their development toolset whether their company knows it or not.

This worrisome trend hasn't yet been fully appreciated in some companies. The features that make Git attractive to programmers also make it a real risk for the enterprise. Git's local repositories mean that there's no visibility into all of the changes in progress, no way to protect access to sections of the code base, and no guarantee of backups. The company's valuable IP could reside on a laptop, just waiting to be left on a train seat one day.

Other category of issues surfaces with DVCS tools such as Git: They're very high-performing for source code, can't handle the non-source code digital assets critical for modern systems development. These might include chip or board layouts, graphic elements, even video. Most likely, documentation will be available either for end users or for compliance that needs to be kept in lockstep with the product under development. The right version of the documentation must match the product being delivered.

If the designers, writers, or artists all use different tools, there's no "single source of truth." This is a critical capability for compliance with standards such as ISO 26262. At best, it introduces complexity and cost; at worst, it introduces risk of poor quality or compliance failures.

Solutions are available, though. Chip and systems developers, including Intel, AMD, NVIDIA, Harmon, and many others, use [Perforce](#). Its Git Fusion solution provides an enterprise-friendly master repository for Git users and anyone involved in creating digital assets as part of the product.

Laying the Foundations

All of these changes and the potential they hold for the future are great—but they also look intimidating. Some simple steps can help prepare solid foundations on which to build adaptable processes that can deliver rapid innovation. Here are five recommendations to consider:

- *Evaluate existing tools and processes:* Look at all of the tools in your toolchain to get a clear view of the different contributors to your products—hardware, software, documentations, graphics, and so on. If they haven't had significant change in the past five years, they're unlikely to be flexible enough to support agile processes or CD. If you can't store all your assets in a single repository, it will raise costs, increase quality and compliance risk, and slow innovation.
- *Focus on differentiation:* Pinpoint the parts of your product that are really your "secret sauce." Use a pace-layering approach to understand which systems or products need rapid innovation in order to compete. Start with those projects and teams for early rewards.
- *Automate:* A key element of CD is to reduce dependency on human beings. Although people perform many tasks well, they generally aren't so good at highly repetitive tasks. You also should not rely on a "hero" every time you want to make a release—one day that person won't be available. This issue may not seem like a big deal if you're handling a release cycle of a year or more, but it's absolutely critical if you're rolling out updates every week, day, or hour.
- *Create rapid feedback:* Use automated test and verification tools wherever possible and as early as possible to get feedback to designers and developers. Tools like those by Parasoft provide excellent standards checking (for example, MISRA) as part of the development stage in the CD pipeline. Early feedback is always cheaper to address than feedback later in the release cycle.
- *Version everything:* The repository for your digital assets, as described above, is the fundamental base for the entire tool chain. It's also critical to your compliancy strategy. Although that repository may appear to be the last thing you might want to change if you're worried about risk, it might actually be the simplest and most effective first step to take. It's also the enabler for implementing agile methods and CD pipelines that would be difficult or impossible to achieve with legacy tools.

Driving into the Future

Lots of excitement pervades in today's evolution of automotive systems. Opportunities for change are unlike anything we've seen before, and there's a persistent hunger in the market for innovation.

wever, that kind of change becomes challenging with traditional tools and processes. Learning lessons from er industries and adopting modern development tools and processes, such as agile methods and continuous delivery, could be key enablers for success.

Mark Warren is the solutions director at Perforce, based in the UK. Mark has over 25 years of experience working with development and product management teams in software-development tools and configuration management.

Source URL: <http://electronicdesign.com/embedded/consumer-demand-drives-continuous-delivery-automotive-industry>