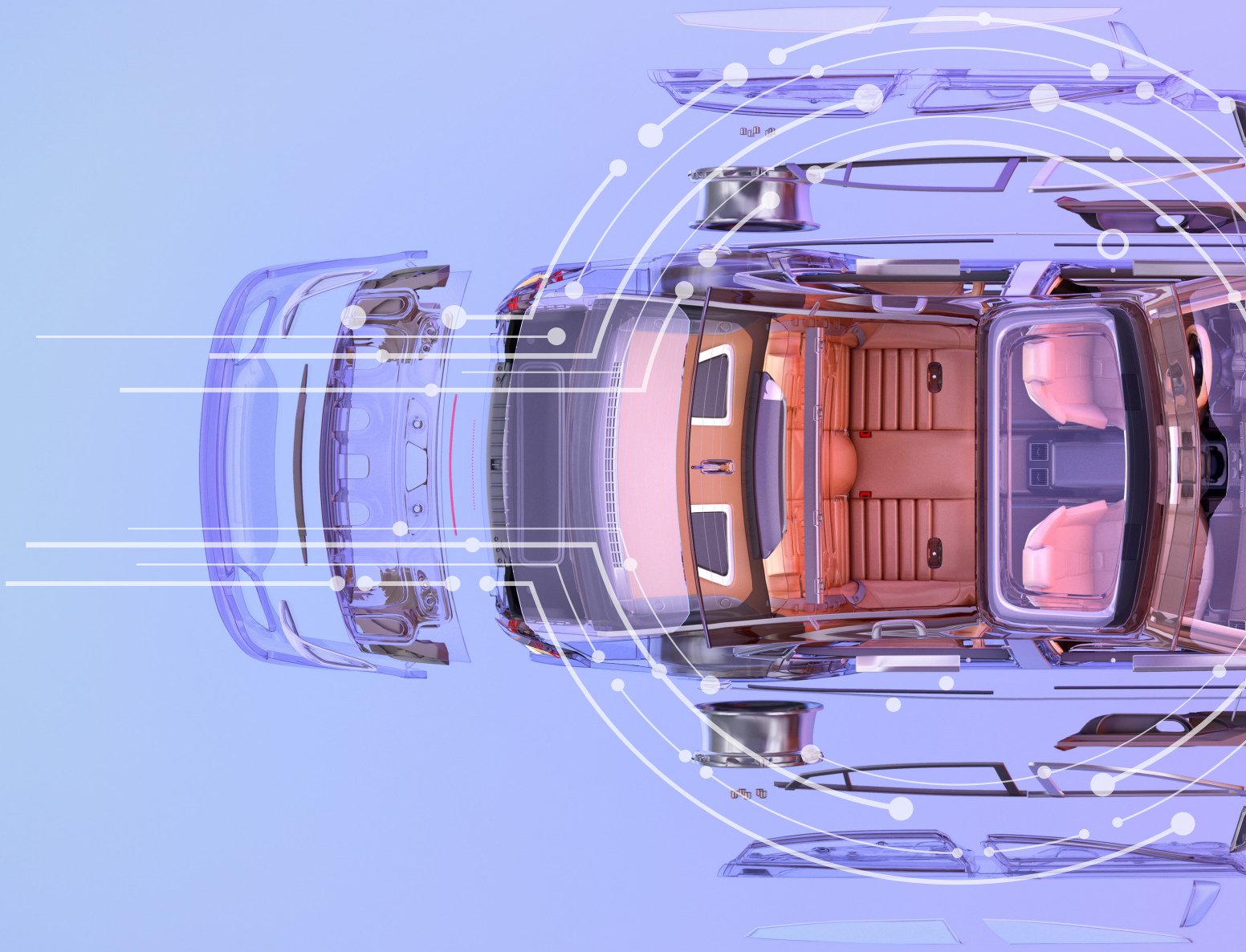
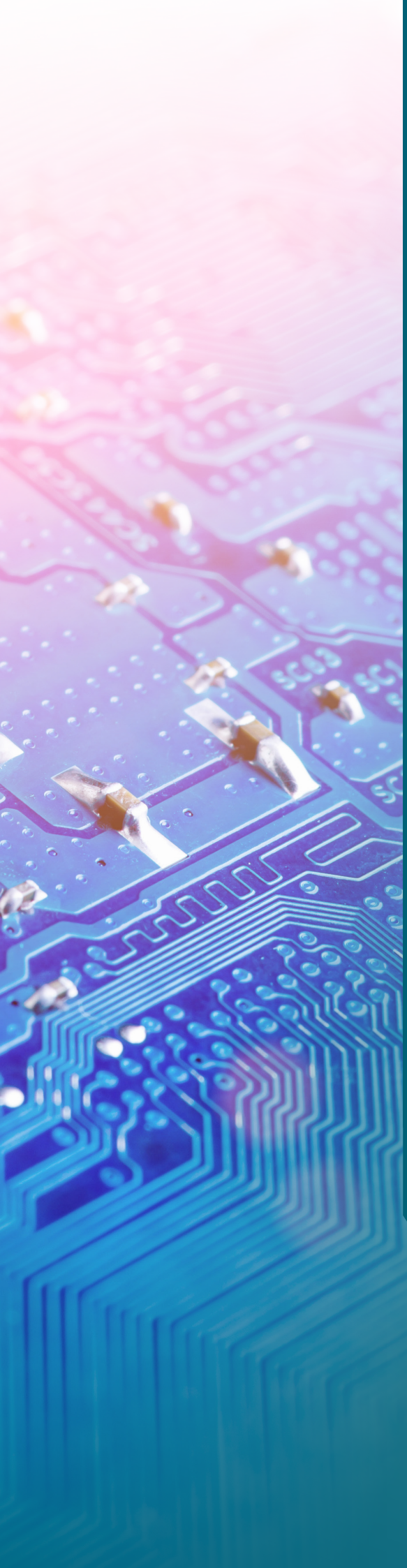
AlixPartners | beryls by AlixPartners

Software Launch Control

How OEMs master the software defects that prevention cannot stop

November 2026





Executive summary

In 2024, software defects drove the recall of 13.4 million vehicles in the US, 48 percent of all recalled vehicles and up from 18 percent a year earlier.[1] In 2025 the share remained close to that record, while the share of software-related recall campaigns rose further.[2] Software has become the gating factor. It decides whether a vehicle can be shipped, and whether it stays on the road without costly remediation. Yet the industry still responds to software defects the way it always has, with prevention, and where prevention fails, with firefighting.

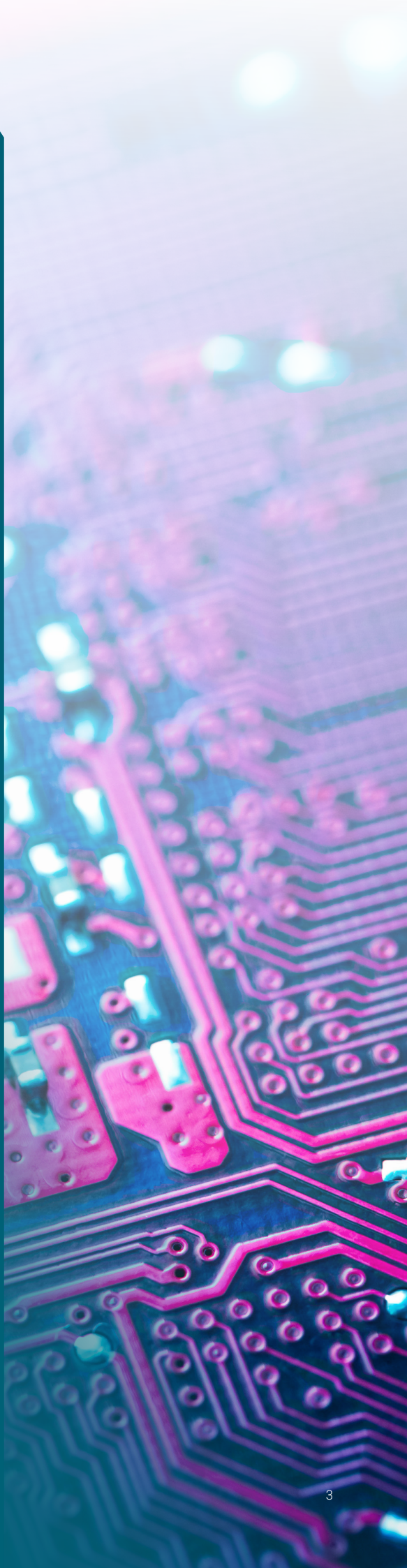
Prevention remains essential, and our Future Launch Management insight sets out how to strengthen it across hardware, supply chain and organization.[3] But prevention has a structural limit, and the capability to execute it consistently is still missing in most organizations. A share of defects will escape into the vehicle, and that is already known before the program starts. The problem is not the limit of prevention itself. The problem is planning as if it did not exist. What is missing is a second discipline alongside prevention. Software Launch Control resolves the defects that escape in a structured way, acting on their impact rather than on their probability. Firefighting is not a third option. It is defect resolution without preparation or structure. The same work gets done, only later, at higher cost and under greater pressure.

Executive summary

The urgency is growing. Rising software content is reshaping the automotive value chain. Established players are taking on responsibilities that are new to them, while technology companies and start-ups enter the ecosystem with a different way of working. Defects now have origins that the conventional launch system was not designed to handle.

At the same time, the measure of success must change. Customer readiness is decided per feature against real exposure, not by a defect count. The criterion is not zero defects but convergence, meaning that more defects are closed than generated. Software-defined companies already work this way. They ship fast and fix fast, delivering a customer-ready product early and resolving remaining defects quickly. At most established OEMs, that capability is still underdeveloped.

Software Launch Control is based on two pillars. The discipline of a project management office (PMO) sets process, cadence and decision rights. Technical depth analyzes and solves defects down to their actual origin rather than merely tracking them. It includes AI-based prediction that places a new defect in a criticality class before it has field data of its own. Both pillars require leaders who go into the content of the critical defects instead of steering by status reports. This paper sets out how that works. A Quick Strike captures and evaluates the organization’s status quo. Six building blocks then establish the required capabilities. One customer feature illustrates the method end to end. The payoff reaches beyond the launch. The capability built to secure the first start of production (SOP) is the one that every feature SOP after it depends on.



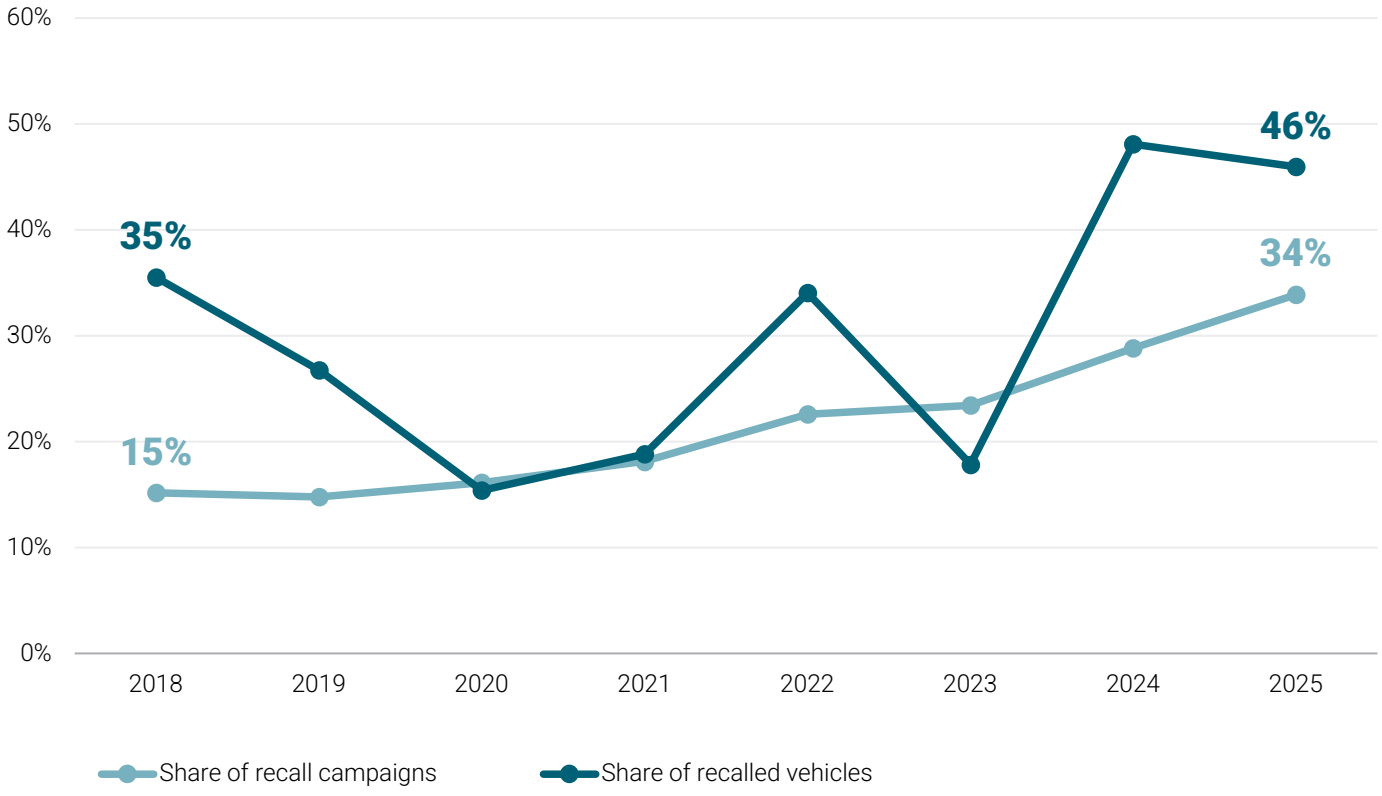
1 Software is on the critical path. From exception to pattern.

Software has moved onto the critical path of vehicle launches.

Within a single year, the number of US vehicles recalled for software defects more than doubled.[4] Volvo postponed the start of production of its EX90 flagship by five to six months to complete software development.[5] General Motors halted sales of the new Chevrolet Blazer EV over software quality issues and resumed them less than three months later with price cuts of up to $6,500.[6] Volkswagen's software unit CARIAD posted operating losses of around €6.9 billion between 2022 and 2024, and software delays pushed back the Porsche Macan Electric and the Audi Q6 e-tron by more than a year.[7] In early 2026, Ford recalled around 4.3 million vehicles over a single software defect.[8] These are no longer isolated incidents. They hit companies already under strain from falling volumes, electrification investments and new competition, and the pattern has become too regular to treat as an exception.

The cost is substantial. The cases above show its scale: heavy losses, price cuts to win back buyers and weeks without sales.

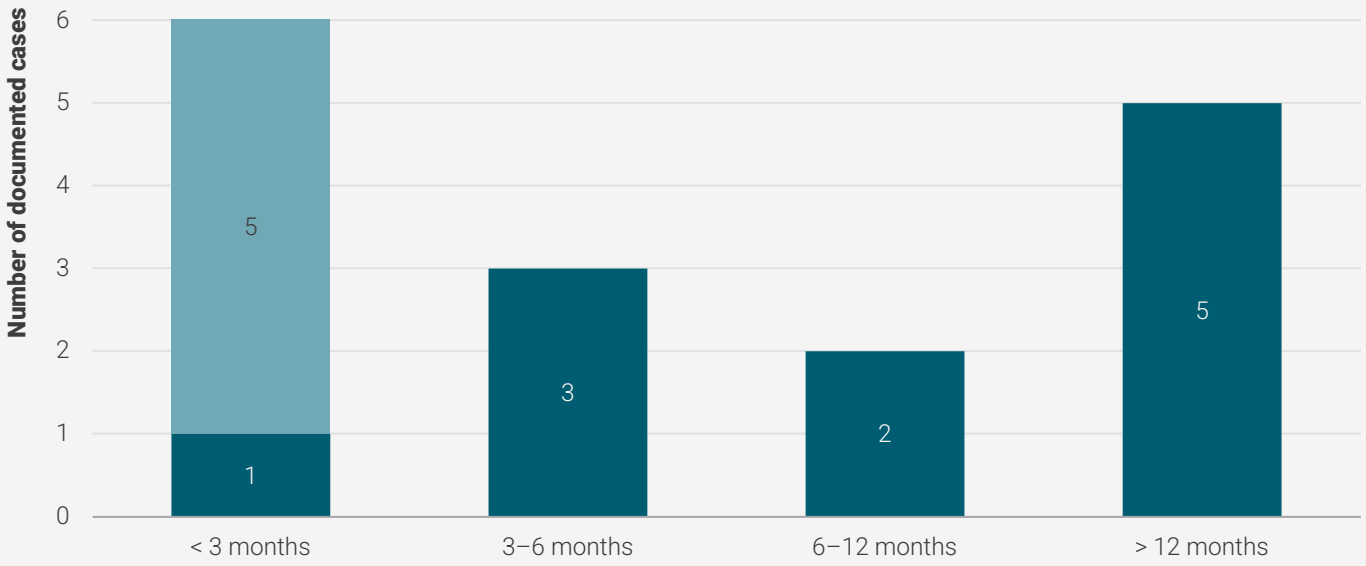
Software-related share of US vehicle recalls, 2018–2025



Campaigns in which the defect or the remedy involves software (keyword classification of NHTSA recall texts), 20 major light-vehicle OEM groups. The vehicle share is volatile because single large campaigns dominate individual years.

Source: NHTSA recall data; Berylls by AlixPartners analysis (October 2026)

Software-driven launch postponements and stop-sales by duration, 2020–2025



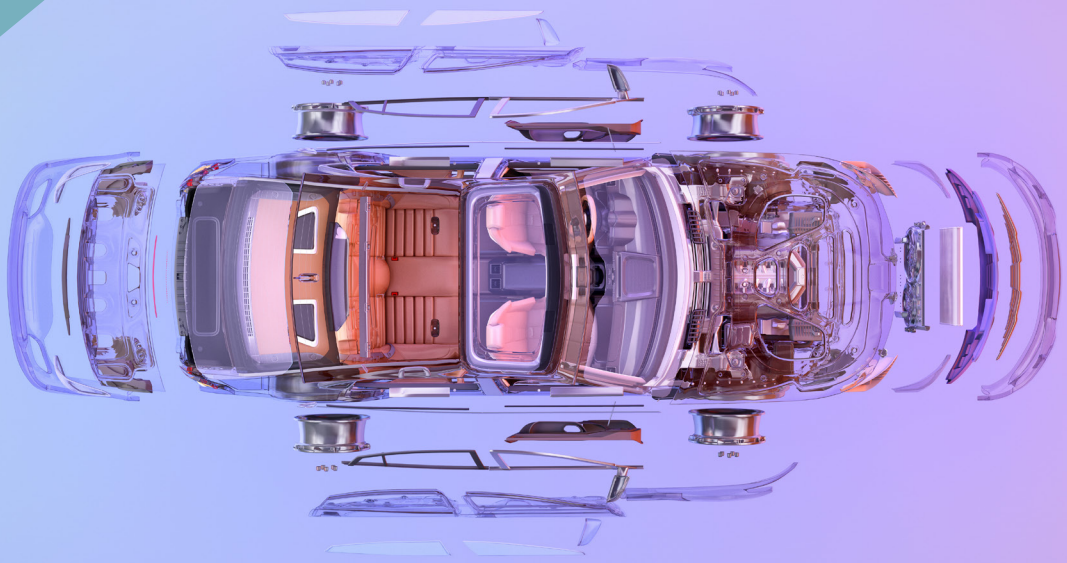
Number of documented cases, 16 in total. Launch postponements are measured against the first publicly communicated start of production or market launch; stop-sales are measured as the duration of the sales or delivery stop. Postponements of more than twelve months form the largest group, while stop-sales are resolved within weeks.

Source: company statements and press reports; Berylls by AlixPartners analysis

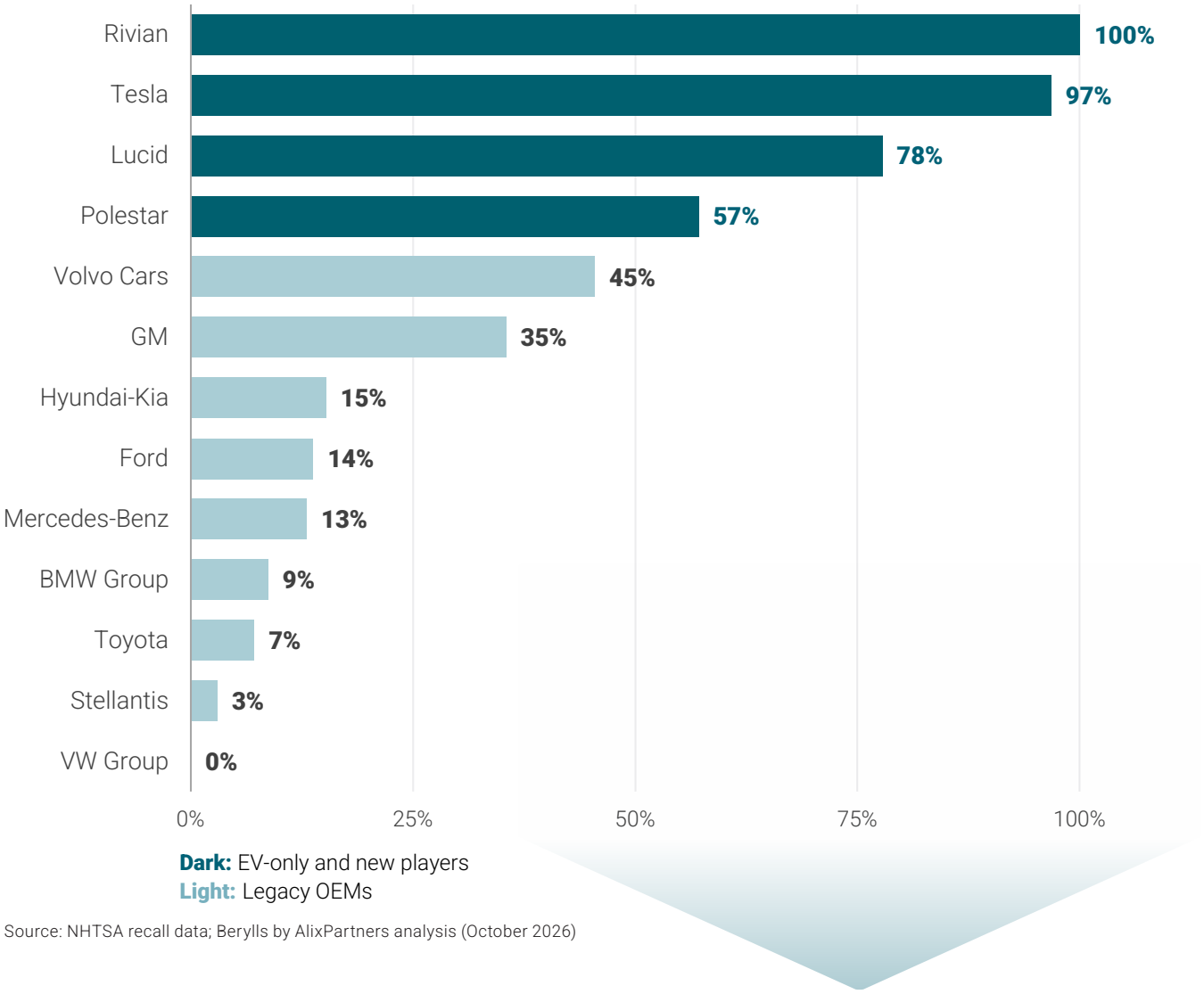
Behind this lies a mindset the industry has yet to overcome. The established logic is that everything must be 100 percent finished at SOP. Software-defined companies work differently. They ship fast and fix fast. The product reaches the customer as soon as it is customer-ready, and what remains is resolved in the field, often over the air (OTA).

Three conditions decide whether OTA pays off and whether vehicles need a workshop visit after the fix. First, the architecture must be designed so that every software defect can be fixed remotely, not just selected ones. Second, the entire process chain must be OTA-capable, from development through production to aftersales. Dealer processes, diagnostic tools and warranty handling must be designed for remote fixes. In US software recalls of legacy OEMs, around 70 percent of affected vehicles could only be remedied in a workshop, and almost every OTA remedy still came with a workshop option.[9] Third, the OTA pipeline needs sufficient capacity. In our benchmarking of global OEMs, fixes for software defects, rather than new functionality, accounted for around 90 percent of all OTA releases at one incumbent, compared with around 60 percent in earlier years. The pipeline is therefore almost fully consumed by remediation, leaving next to no room for the feature updates customers value.[10]

Speed in shipping and in fixing is the capability, but it has two firm boundaries. The product must be customer-ready at launch, and safety, homologation and cybersecurity are never traded for speed.

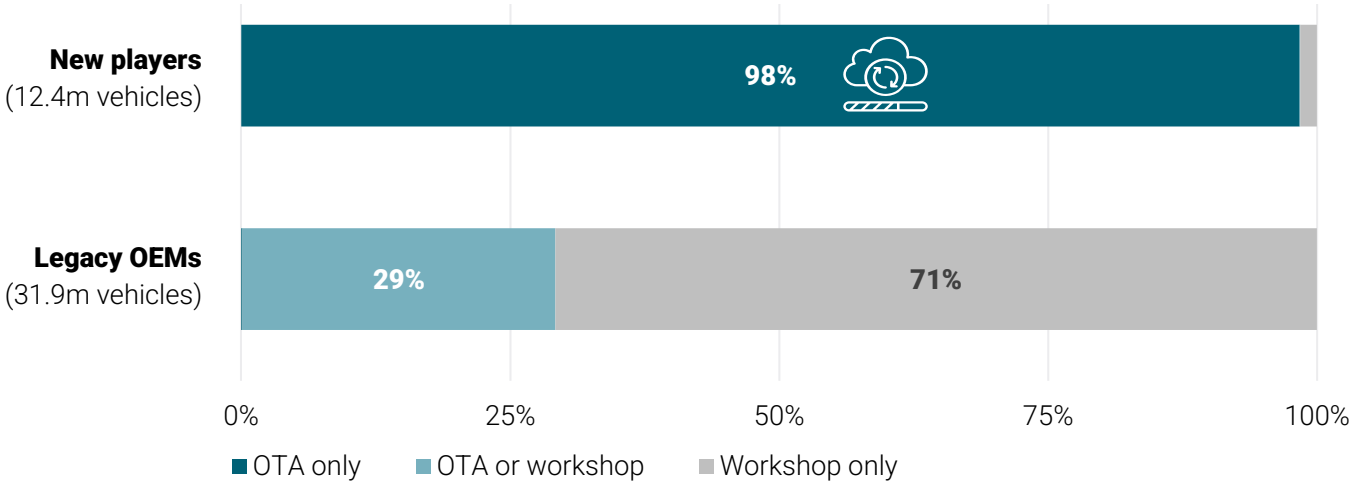


Share of software recall campaigns remedied via OTA, US 2021–2025



Source: NHTSA recall data; Berylls by AlixPartners analysis (October 2026)

Remedy path for vehicles in software recalls, US 2021–2025



Share of vehicles in software-related recall campaigns by the remedy path stated in the recall: OTA only, OTA or at a workshop, or workshop only. 13 US-selling OEM groups, a subset of the 20 groups in the recall analysis; new players are EV-only manufacturers. Actual workshop visits after an OTA update are not published.

Source: NHTSA recall data; Berylls by AlixPartners analysis (October 2026)

This paper complements our Future Launch Management insight[11] and our insight on software procurement, which describes how OEMs must source and steer software suppliers.[12] It goes one level deeper into the part that increasingly determines the launch date, namely software and specifically the defects that reach the vehicle despite everything the program does to prevent them. Understanding where those defects originate is the starting point.

2 Where defects come from. Seven origins in a shifting value chain.

The automotive value chain is being redrawn.

Technology companies such as Google, Amazon and NVIDIA, as well as start-ups that did not exist a few years ago, now supply core parts of the software stack, while OEMs and Tier-1 suppliers redistribute integration work among themselves. Most OEMs still run a launch system built for a different division of labor, and defects now have origins the system was never designed to address.

One central driver of software defects is product complexity. Every architecture, every control unit, every configuration option, every market variant and every supplier interface multiplies the number of software paths that must be built, integrated and tested. Complexity therefore shapes each of the seven origins described below, and our benchmarking shows that the OEMs carrying the highest complexity are also the ones with the weakest software quality.[13]



Requirements and interfaces. Complex software stacks are harder to specify than individual components, which often leaves requirements and interfaces underspecified. In the launch programs we have supported, this origin is consistently among the largest sources of defects.



Capabilities. Software and integration expertise is often still being built up at both OEMs and suppliers. Functional expertise, for example in perception, does not automatically translate into software integration capability, and the gap shows up at the interfaces rather than inside the domain.



Responsibilities. Supplier contracts are often still written per component, such as a control unit, a camera or a software module. A defect that arises between two components therefore has no clear owner. It is unclear who must find it, who must fix it and who pays for it. The gap often starts in procurement, where software is still priced into the unit cost of a hardware component and shared responsibility across components is rarely defined.[14]



Supplier landscape. A vehicle relies on hundreds of suppliers, several of them on a single component family. In our benchmarking, the most fragmented case was an OEM sourcing a different supplier for each seat row. Each supplier brings its own release cycle, tool chain and defect reporting. Unless these are integrated properly into the OEM's development process and tool chain, this fragmentation significantly complicates defect resolution. Analysis, fix and verification must be coordinated across several organizations before a single defect is closed.



Supplier governance. Large platform suppliers serve many OEMs from a single product roadmap, so one OEM's fix requests compete with those of other customers. Because drivers expect the same platforms and services across car brands, the bargaining power has shifted toward these suppliers.[15] OEMs therefore cannot always enforce their quality standards and timelines, and defects that remain open on the supplier's side reach the vehicle.



Time pressure. Late changes, parallel development and skipped verification increase defect risk. Unlike most other origins, time pressure follows directly from management decisions on scope and timing.



Tooling. Fragmented tool chains, ticketing systems and release processes make it difficult to maintain a shared view of defects and of release readiness. Requirements management is hit hardest unless it is standardized through shared tools and aligned processes across OEM and suppliers.[16]

Many of these risks can be reduced through clearer ownership, better coordination and aligned processes. None of them can be eliminated. Even an organization that improves on all seven still faces a defect inflow it cannot stop. That inflow is the subject of the next chapter.

3 The limit of prevention. Why more of it will not close the gap.

The industry relies on prevention and, where prevention fails, on firefighting.

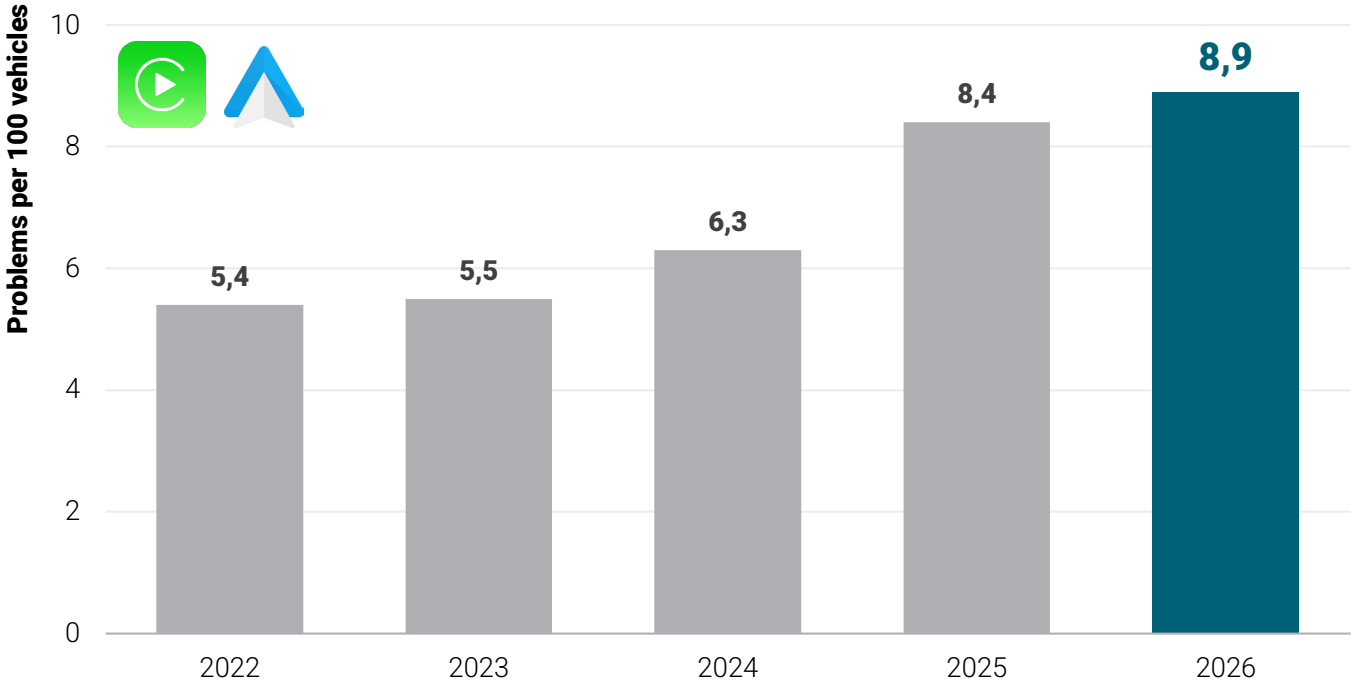
Prevention lowers the probability that a defect occurs, and today it is not even executed to that limit. The intent is rarely the problem; the capability to deliver it consistently is. But even done well, prevention cannot lower the probability to zero, and three sources of inflow sit structurally outside its reach.

The first source is ongoing development itself. Every feature release, every change and every fix creates new defects, including after SOP through OTA updates. J.D. Power reports that vehicles that received OTA updates showed nearly 14 percent more problems in 2026, and only 27 percent of owners who received an update noticed an improvement.[17] A stack that keeps evolving keeps generating defects.

The second is the diversity of the real world. Fleet size, network and backend conditions, time-dependent states and usage sequences produce combinations that no specification anticipates and no test plan fully covers.

The third is code from outside the OEM. App ecosystems, mobile networks, backend services and supplier stacks sit inside the customer feature but beyond the OEM's control.

Apple CarPlay and Android Auto problems in three-year-old vehicles, US

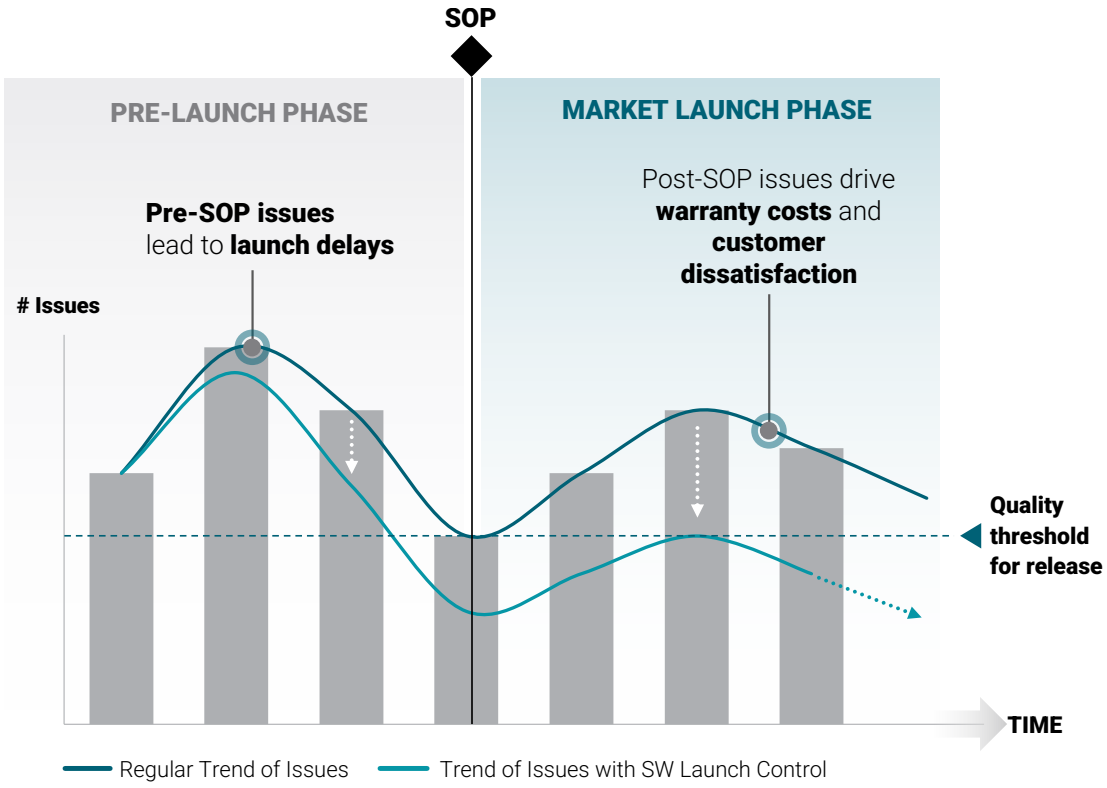


Problems per 100 vehicles (PP100). Phone integration has been the most reported single problem in the study since 2024; infotainment overall reached 56.7 PP100 in 2026, the most problematic of all categories.

Source: J.D. Power U.S. Vehicle Dependability Study 2022–2026

Earlier integration helps, but it addresses detection, not inflow. Test setups such as HIL and G-HIL[18] improve detection. They find more of what is already there and find it earlier. They do not reduce what the stack keeps generating. The inflow therefore continues, and it produces two cost peaks: delay before launch, then warranty cost and lost customer trust after it.

Defect inflow across the program timeline, with and without Software Launch Control



Source: Berylls by AlixPartners

The industry's answer is firefighting in crisis mode, which is expensive and destabilizes everything around it. Firefighting arrives unplanned, and it meets the same seven weaknesses in program after program:

1. **Timeline:** limited transparency on timing and content, internally and externally
2. **Ownership:** no clear ownership of defects, so responsibility is passed back and forth
3. **Defect volume:** a high volume from testing, with insufficient prioritization
4. **Root cause analysis:** slow analysis and a slow path to a solution
5. **Interdependencies:** insufficient understanding of how domains affect each other
6. **Maturity:** limited visibility into how mature a feature is
7. **Cooperation:** weak cooperation between internal and external stakeholders

None of these is solved by more prevention, because they describe what happens after a defect has escaped.

The industry lacks the capability to execute prevention consistently, and a plan for the defects prevention will not catch.

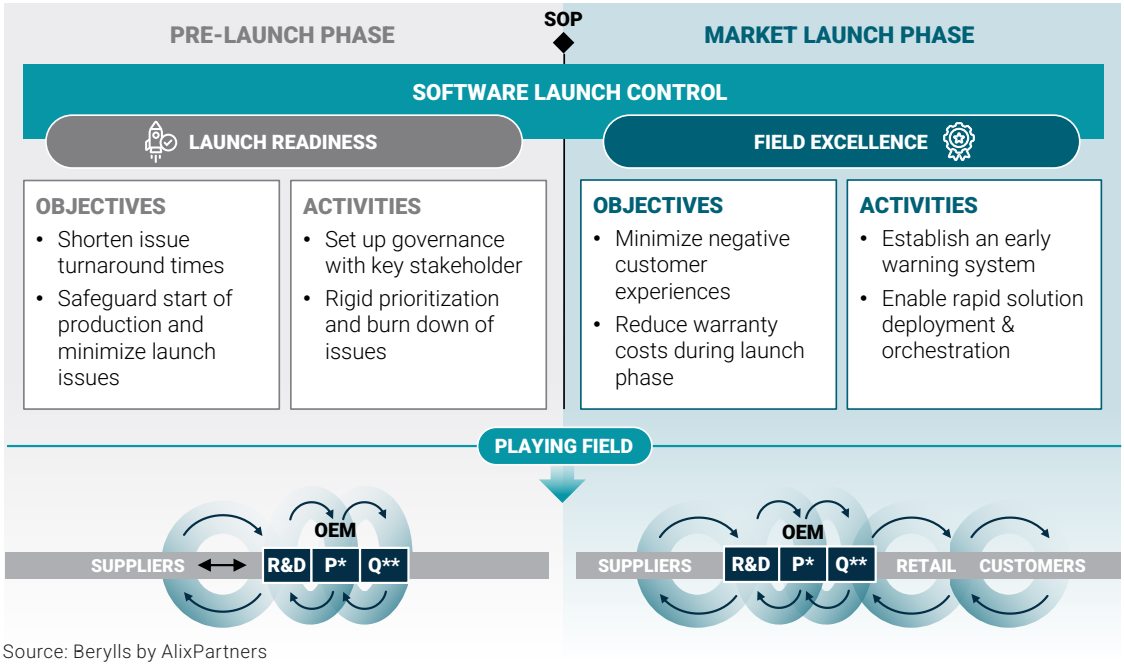
4 Software Launch Control as the second discipline. From firefighting to structured defect resolution.

The answer is two disciplines instead of one discipline plus improvisation.

Prevention lowers probability, Software Launch Control lowers impact, and a vehicle launch needs both. Firefighting is not a third discipline. It is what defect resolution looks like when it is unstructured and unprepared.

Software Launch Control combines governance discipline with technical depth. A strong PMO sets the process, the cadence and the decision rights. Leaders engage with the content of the critical defects, not only with their status. Neither pillar works alone. A PMO without technical depth maintains a defect list it cannot judge, and technical depth without a PMO produces good analysis that nobody converts into a decision.

Software Launch Control across both sides of SOP: launch readiness and field excellence



Source: Berylls by AlixPartners

The discipline covers both sides of SOP. Before SOP it ensures release readiness of the SOP-critical features, shorter defect turnaround and a safeguarded start of production with fewer launch issues. After SOP it has two jobs. For released features, it improves customer experience and lowers warranty cost, as field signals feed back and the organization moves from reaction to anticipation. For new features, it ensures release readiness on the planned date.

Most launch models still treat that second half as outside their scope. On a continuously evolving software stack, every new feature has its own SOP. Software Launch Control is therefore not a phase that ends with the first vehicle, but a permanent discipline.

The first SOP is where the capability is built.
Every feature SOP after it is where it pays off.

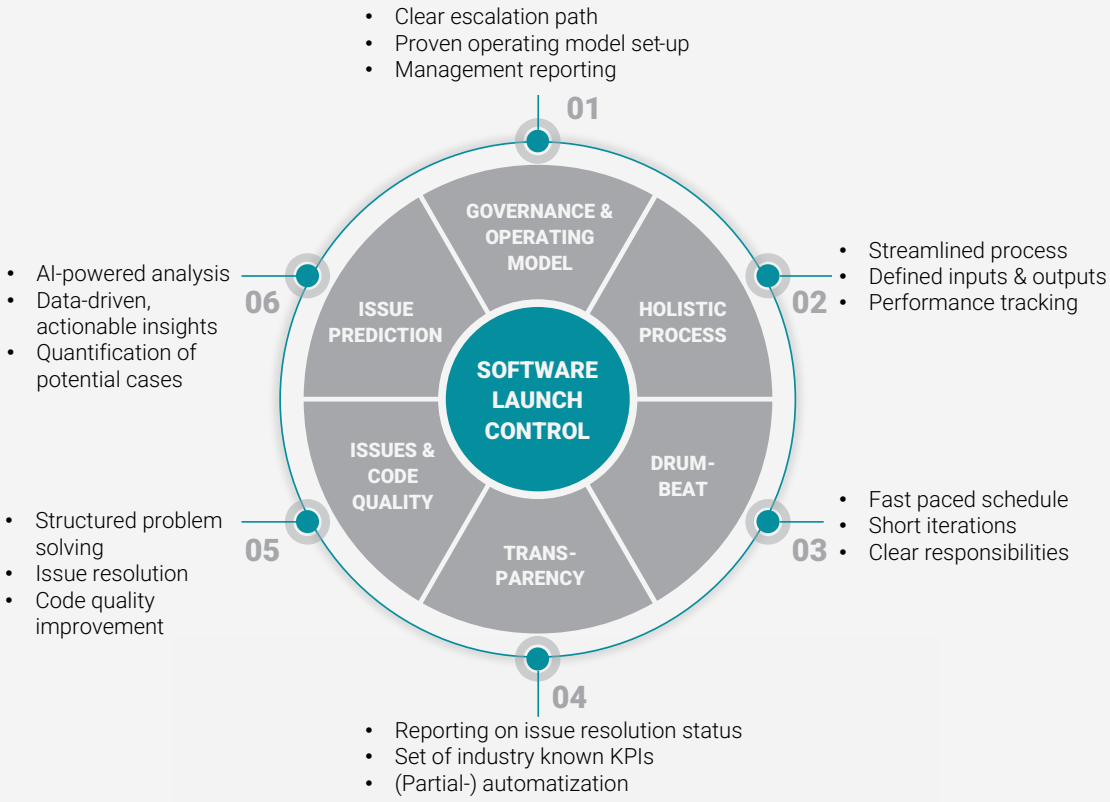
5 How Software Launch Control works. One Quick Strike, six building blocks, one feature.

The approach shifts the unit of steering from defects to features.

What must be ready is not a defect count but a feature. A feature is what the customer experiences; a defect count is an internal artifact that says nothing about whether a customer can pair a phone or use driver assistance. Software Launch Control therefore steers the set of SOP-critical features and evaluates readiness per feature.

The method has seven parts: a Quick Strike that captures the organization's status quo, and six building blocks. Governance and Operating Model, Holistic Process and Drumbeat carry the conventional PMO work of governance, process and cadence. Transparency, Issues and Code Quality, and Issue Prediction add the technical depth, from in-house experts or specialist partners. Together, the six building blocks address the seven weaknesses that recur in every firefighting program.

The six building blocks of Software Launch Control



Remark: Building blocks will be applied across both phases with varying intensity depending on situation (e.g., focus on code maturity before SOP and issue prediction after SOP)

Source: Berylls by AlixPartners

The method produces two outputs. The first is a secured SOP for the program at hand. The second is a documented picture of the defects and their causes, which becomes the input for every SOP that follows. One rule runs through all of it. Every defect ends in one of four decisions: fix now, fix later, mitigate or accept. For safety, homologation and cybersecurity defects the decision is always fix now. One distinction matters. The primary lever is impact, not the number of escaping defects.

Applied consistently, the six blocks deliver lean, fast-acting governance, faster defect resolution, a more robust and stable system, transparency on program status, better code quality and a forecast of upcoming defects and warranty cases.

To show how the method works in practice, the rest of this chapter follows one feature through all six blocks: smartphone integration via Apple CarPlay and Android Auto, wired and wireless, so that a customer can pair a phone and use its apps on the vehicle display. It is no arbitrary choice. Phone integration has been the most reported single problem in J.D. Power's U.S. Vehicle Dependability Study three years running (see chapter 3). It is SOP-critical, owned by the infotainment domain, and has no safety, homologation or cybersecurity relevance, so all four decisions are available. It carries four defects, one per criticality level.

Quick Strike

As a starting point, the Quick Strike establishes the organization's current maturity on each of the six building blocks. It asks one question per building block. Are decision rights, escalation paths and counterparts at the OEM and at the suppliers clearly defined? Is there one path from detection to deployment, with agreed service levels and one criticality scale from P0 (most critical) to P3 across all domains? Does the cadence follow the software build and release plan? Is there one view of defects per feature, or are defects spread across tools and ticket systems? How structured is root cause analysis, how long do fixes take, and which causes recur? Is defect and claim data from earlier programs available and usable to predict criticality?

Each block is rated on five maturity levels, from firefighting through reactive, structured and in control to predictive. The rating is based on interviews, a review of tools and data, and one sample feature carried through end to end. The result is a maturity profile across the six blocks that shows the gaps and sets the priorities for the setup. In our experience, that profile is rarely even. An organization can be close to predictive in its reporting and still be firefighting in root cause analysis.

Governance and Operating Model

Governance runs on four working levels: defects escalate upward, priorities cascade downward. A working team per domain owns root cause analysis and the fix through to release, and a Field Excellence Officer triages incoming defects into the responsible domain. Operational management tracks across domains and enforces the service levels between the meetings. The Launch Control Board decides on every defect, clears roadblocks and challenges the timeline. Software leadership sets targets and owns the regulatory and warranty risk.

The setup is anchored in the existing organization rather than built beside it. Work packages run along domains and value chain steps, each with a named lead from the line organization, so that the structure strengthens it rather than duplicating it. Suppliers are part of this structure as partners with shared responsibility. OEM task forces that take over a supplier's operations when a project turns critical rarely work for software, where the OEM seldom has the expertise to run the supplier's stack itself.[19] The result is a clear escalation path, a proven operating model and reliable management reporting.

For the smartphone integration feature, ownership sits with the infotainment domain. The working team owns root cause analysis, the Board aligns the decision for each of the four defects, and software leadership sees everything that blocks the launch and everything that is accepted. One rule makes governance real rather than nominal.

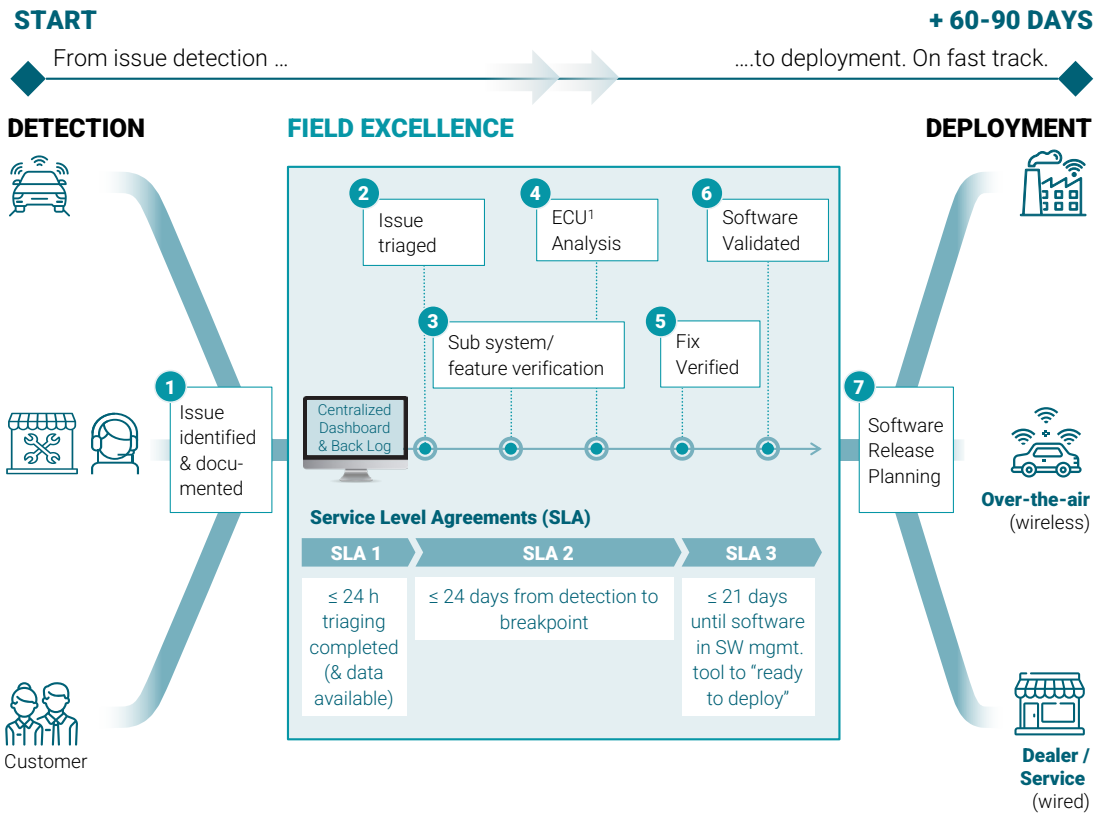
> An accepted defect without board alignment is not accepted. It is ignored.

Holistic Process

Structured defect resolution requires one path from detection to deployment, with agreed service levels for every stage: triage within 24 hours, then 24 days from detection to breakpoint (fix verified), a further 21 days until the software is ready to deploy, and 60 to 90 days in total from detection to the customer, whether through the dealer or OTA. By comparison, around 80 percent of US recalls have a remedy within 60 days, but counted from filing, not from detection.[20] Every stage has a defined input, a defined output and a named owner, and all of it sits on one central backlog.

Triage runs from P0 to P3 and leads into the four-way decision. The decision is driven by the delivery path, not by the severity label alone, with staged rollout available as a mitigation lever. Criticality comes from evidence where evidence exists, and from the prediction model where it does not yet exist. The process is holistic, not software-only. It is embedded in development, production, quality and aftersales. The Field Excellence Officer runs triage and routes each defect to the area that must act.

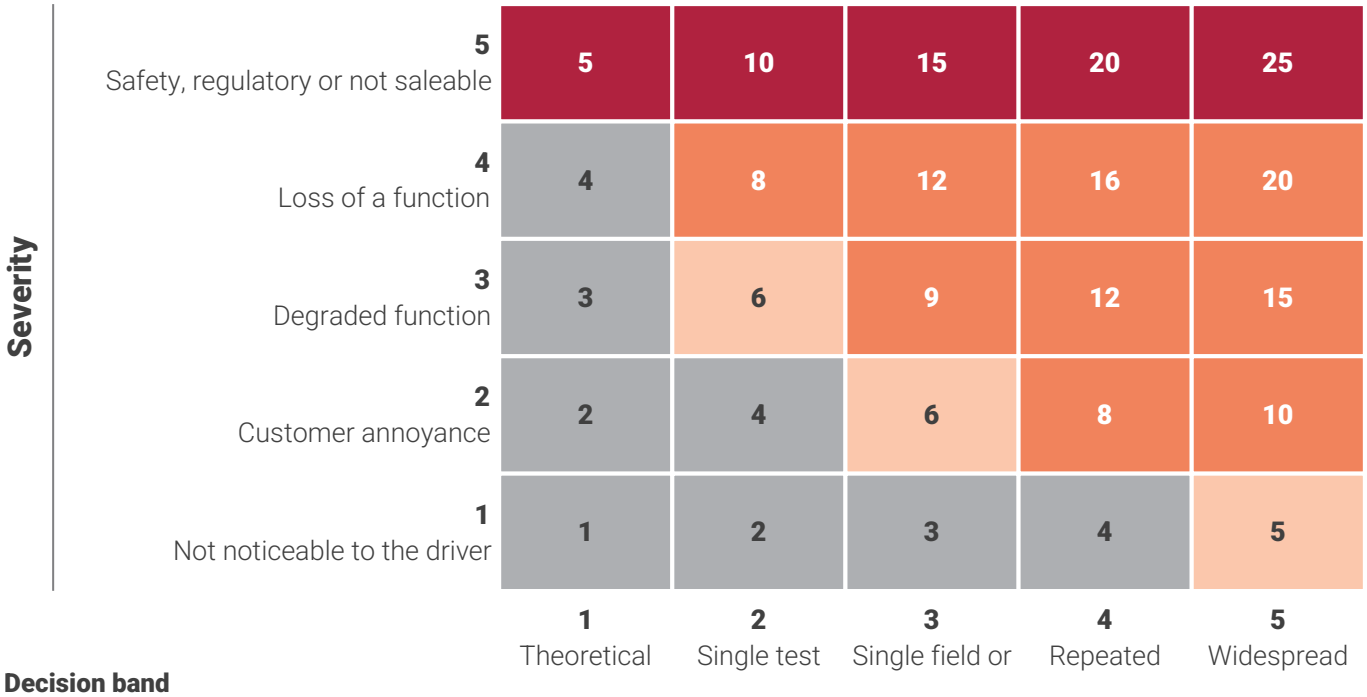
From detection to deployment: end-to-end process and service levels



ECU = Electronic Control Unit
Source: Berylls by AlixPartners

The P0 to P3 class is derived from a Criticality Priority Number (CPN), so that severity and occurrence are assessed separately instead of being merged into a single label. Severity, from safety and regulatory impact down to defects the driver never notices, is multiplied by occurrence, each rated on a five-point scale. After SOP, occurrence comes from field repair rates; before SOP, when volumes are too low for statistics, it comes from the nature of the evidence, from a theoretical failure mode to repeated reports in test fleets. The bands of the resulting five-by-five matrix translate the CPN into P0 to P3, and each class leads into its default decision. Safety or regulatory relevance always overrides the score.[21]

Criticality matrix: CPN as severity times occurrence, banded into P0 to P3



Decision band

- Safety or regulatory: always fix now
- High criticality: fix now, before release
- Medium criticality: fix later (next release or OTA) or mitigate
- Low criticality: accept and monitor

Illustrative matrix. Thresholds are set per program; safety or regulatory relevance overrides the score.
Source: Berylls by AlixPartners, based on established risk-matrix practice

The four defects on the smartphone feature show what the decision looks like in each class. The P0 is a wireless connection that fails on the current Apple iOS version, which would leave the feature dead for a large share of customers: fix now, with no mitigation available. The P1 is a connection that drops after about 20 minutes on some phone models, with a root cause that sits partly in a third-party operating system and cannot be closed before SOP: mitigate, with a background reconnect, a known-issue entry and a staged rollout. The P2 is album artwork appearing in low resolution on the main display, visible to every media user: fix later, OTA in the first post-SOP wave. The P3 is a phone name that loses its special characters in the device list on one model: accept, documented with an owner and a review trigger.

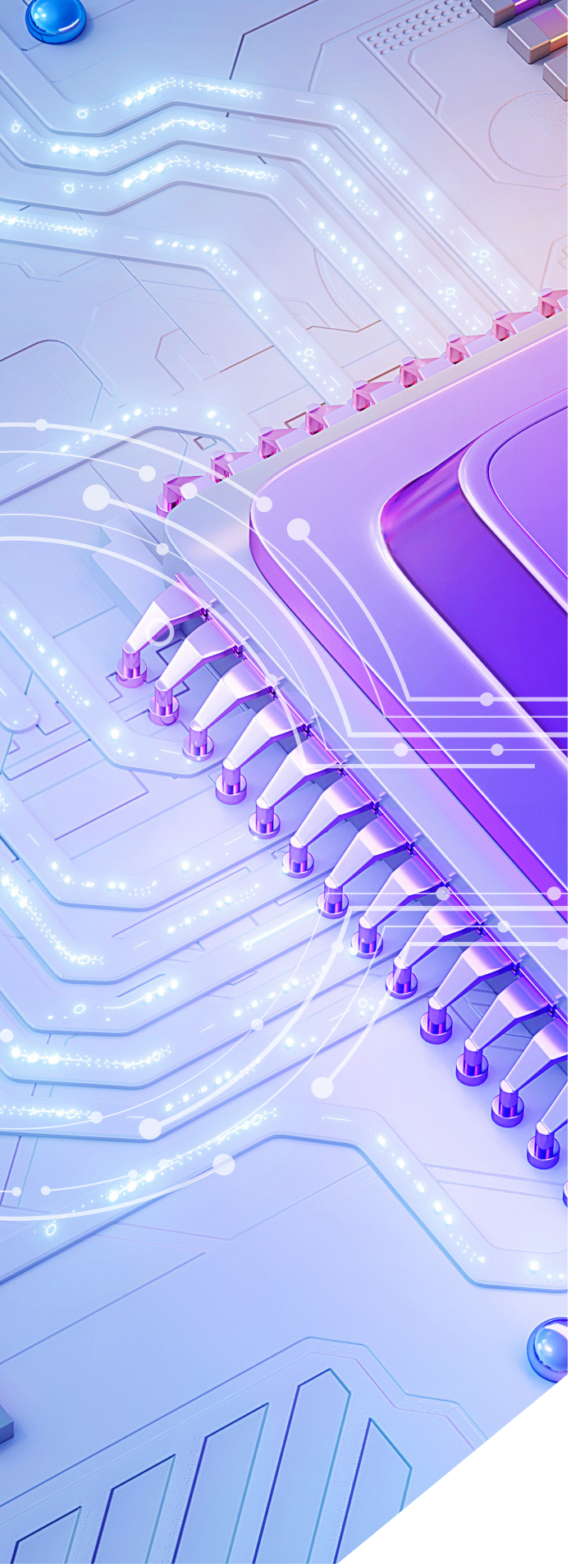
P2 and P3 are similar in nature; what separates them is the number of customers potentially affected. Both could ship OTA, but only P2 is worth a release.

Drumbeat

The drumbeat runs on three of the four governance levels, at three frequencies: daily per domain, the Launch Control Board twice a week on a standard agenda, and software leadership weekly. Anchored to the release plan, the cadence puts decisions before the freeze. Meetings are staffed so that the technically sound decision can be made in the room, with the supplier joining on demand. Members are given dedicated time for the work, and reporting to other committees is streamlined so that the cadence does not add to the existing workload.

The drumbeat must be tailored to each organization. Governance levels, staffing and frequency follow the release plan, the supplier setup and what the teams can sustain for the length of the program. A cadence abandoned after a month is worth less than a slower one that holds.

On the smartphone feature this means a daily check-in for the infotainment domain while the P0 root cause is open, with the phone platform partner on demand. The Board meets twice weekly, and all four decisions are aligned two weeks before the software freeze. The P3 item is decided once and closed, so it does not return to the agenda every week.



Transparency

Transparency has two layers. The first is reporting on the defects of the SOP-critical features: top defects, criticality per week, resolution status, heatmap, burndown and mean time to resolution. The second is the measurement system that answers the launch question rather than the ticket question: feature maturity against target, defects per feature, the number of releasable features, exposure as the denominator, coverage against exposure, and convergence instead of zero defects, meaning that more defects are closed than generated.

Reporting by feature rather than by ticket makes Software Launch Control customer-centric. It delivers transparency on defect resolution status and on the SOP readiness of features, a set of industry-standard key performance indicators (KPIs) and partial automation of the reporting itself.

Reporting is only as reliable as the classification beneath it, and so are the handling and resolution that follow. We therefore run an AI-based scan across the backlog. It checks whether each ticket is assigned to the right feature and rated in the right class from P0 to P3 and returns the incorrectly classified and duplicate entries to triage.

For the smartphone feature, exposure means how often and for how long smartphones connect to the vehicle's infotainment system: connection attempts, total connected hours and fleet-months of use. Coverage means the tested combinations of phone model, operating system version and wired versus wireless connection. The two are not the same, and confusing them is a frequent error in launch reporting. Even 50,000 connection attempts across five phone models may leave relevant configurations untested.

The feature appears in the report as one line with four defects and their criticality, not as four tickets with no connection to the feature. At the freeze the defect rate per 1,000 connection attempts is declining across three test blocks, the closure rate is above one, the criticality-weighted backlog is falling and no P0 is open. The verdict is releasable, with three open defects, each carrying an aligned decision.

Issues and Code Quality

This is where Software Launch Control moves from tracking defects to solving them. Root cause analysis runs in the domain working team, cross-functional and supported by software and integration experts. It combines proven automotive problem-solving methods such as 8D and Ishikawa with software-native practices, among them log and trace analysis, regression analysis across builds, static code analysis and joint code reviews with the supplier. Every finding carries an owner and a date. Containment comes first and the long-term improvement runs in parallel, prioritized by criticality, impact and recurrence risk. Progress is tracked, and the solving process is documented for the next program.

Conventional PMO steering on its own produces a well-maintained defect list. Combined with technical depth, the same list becomes a set of analyzed and closed problems. That requires analysis on two levels. The first is the technical root cause in the code, the second its actual origin, such as underspecified interfaces, unclear responsibilities, capability gaps or the sheer complexity of the product itself.

Our project experience at a global OEM illustrates the difference. A recurring cluster of software defects traced back to the complexity of its vehicle development, where variants, architectures and supplier scopes multiplied the number of software paths to be built and tested. Each fix in the code closed a defect, but not its cause. The program therefore mapped the full causal chain and resolved it at its origin. Defects from that source have largely stopped recurring, and fixes proved far more durable. Every defect is also classified against the seven origins from chapter 2, showing where the organization must get stronger. This is where defect resolution feeds prevention, and the next program faces fewer defects from the same source. Complexity is the most common origin we find, and the hardest to see from a single ticket.

On the smartphone feature, 8D and Ishikawa are applied to the P0 and P1 items, both of which are cross-domain between head unit, connectivity and the phone. Containment for the P1 dropout ships first, while the long-term rework of the connection handover follows SOP, prioritized on recurrence risk because the same handover serves every phone brand. The P2 artwork defect turns out to be a scaling error, closed in one release, with no root cause escalation needed.

Issue Prediction

Issue Prediction applies AI-based analytics in five steps: data input and cleansing, pattern identification, criticality prediction, measure definition and heuristic refinement. The model is built on defects and claims from earlier programs, where field occurrence and severity are already known. A new defect is matched against those patterns, and results from the current field are fed back into the model.

The output is a warranty probability and a criticality class per defect, together with countermeasures that carry an owner and a timeline, for example an OTA update instead of a parts replacement. In our projects to date, around 70 percent of defects were assigned to the correct class. The main use is early classification. A defect can be placed in a criticality class before it has field data of its own, precisely when the decision must be taken. Prediction adds severity and exposure to every defect, so that priorities rest on expected customer impact rather than on ticket status alone.

On the smartphone feature, connection failures after a phone operating system update are the most frequent claim pattern in earlier programs, so the P0 class is anticipated. Test capacity is reserved for new phone operating system versions, and the defect is found in testing and fixed before it reaches a customer. For the P1 dropout the model quantifies the expected warranty and dealer-visit rate, which is what makes the decision to mitigate defensible against a late attempt to fix now. The P3 name encoding sits below the claim threshold, so the model gives no signal.

> Where the model gives no signal, judgment replaces prediction.

6 Plan for what will escape. From launch project to standing capability.

You cannot prevent your way to zero.

Every program knows this before it starts, and yet most program plans are written as though it were not true. The missing discipline is Software Launch Control: preparing for the defects you already know will come, taking a deliberate decision on each one, and shipping features rather than a list of solved tickets.

The payoff extends beyond the launch at hand. Structured defect resolution also feeds prevention, as chapter 5 shows. Because every defect is traced to one of the seven origins, the program ends with a clear picture of where its defects came from. That picture shows which preventive measures to reinforce and which underlying origins to resolve, so the next program starts with fewer of the same defects.

The capability also outlasts the program. Because one software platform now serves several vehicle programs on a continuously evolving stack, the capability built for the first SOP becomes a standing capability rather than a project.

The same logic applies to suppliers. Tier-1 and software suppliers deliver stacks with their own release cycles and their own feature SOPs, often to several OEMs at once. That calls for a delivery model built on short iterations and continuous delivery over the entire lifecycle, rather than one aimed at a single SOP. Those who build the same discipline into their own organization become the partners OEMs rely on when defects escape, rather than the source of defects they escalate against. Software procurement is moving in the same direction. A supplier's ability to steer software projects, up to its own task forces at the highest management level when a project turns critical, is becoming a selection criterion.[22]

Launches will keep producing defects that prevention cannot catch. The difference between a program that ships and one that slips is not how many defects escape. It is whether anyone planned for them.

Where to start depends on the organization's current maturity across the six building blocks, which the Quick Strike establishes. The right first steps are therefore specific to each program.

Talk to our experts. If you would like to discuss where your organization stands today, or what Software Launch Control could look like for your next SOP, our experts at Berylls by AlixPartners look forward to the conversation.

CONTACT THE AUTHORS:



Christian Kaiser
Partner & Managing Director
Berylls by AlixPartners
christian.kaiser@berylls.com



Sebastian Bräuer
Partner
Berylls by AlixPartners
sebastian.braeuer@berylls.com



Andreas Maihöfner
Associate Partner
Berylls by AlixPartners
andreas.maihoefner@berylls.com



Max Bienlein
Project Manager
Berylls by AlixPartners
max.bienlein@berylls.com



Yannick Rüdebusch
Senior Consultant
Berylls by AlixPartners
yannick.ruedebusch@berylls.com

ABOUT ALIXPARTNERS

Expertise, execution strength, responsibility: AlixPartners stands for measurable results "when it really matters." As a global management consultancy, we help our clients respond quickly and decisively to their most important challenges. Our experienced consultants specialize in creating, protecting, and restoring enterprise value. In 2023, manager magazine and the Scientific Society for Management and Consulting (WGMB) named AlixPartners the consultancy with the strongest execution. For over 40 years, AlixPartners has supported its clients, and today it has around 3,500 employees across 26 offices worldwide.

www.alixpartners.com

ABOUT BERYLLS BY ALIXPARTNERS

Berylls by AlixPartners is a strategy consultancy specializing in the automotive industry. With experts in Germany, China, the United Kingdom, South Korea, North America, Austria, and Switzerland, Berylls by AlixPartners covers all future topics to help clients succeed, stay future-proof, and go digital within the automotive ecosystem. Our experts offer clients end-to-end support, from strategy through to final implementation.

www.berylls.com

The opinions expressed are those of the author(s) and do not necessarily reflect the views of AlixPartners, LLP, its affiliates, or any of its or their respective professionals or clients. This article "Software Launch Control" was prepared by AlixPartners, LLP ("AlixPartners") for general information and distribution on a strictly confidential and non-reliance basis. No one in possession of this Article may rely on any portion of this Article. This Article may be based, in whole or in part, on projections or forecasts of future events. A forecast, by its nature, is speculative and includes estimates and assumptions which may prove to be wrong. Actual results may, and frequently do, differ from those projected or forecast. The information in this Article reflects conditions and our views as of this date, all of which are subject to change. We undertake no obligation to update or provide any revisions to the Article. This Article is the property of AlixPartners, and neither the Article nor any of its contents may be copied, used, or distributed to any third party without the prior written consent of AlixPartners.

©2026 AlixPartners, LLP

SOURCES

1 NHTSA recall data; Berylls by AlixPartners analysis (keyword classification of recall texts, 20 major light-vehicle OEM groups), October 2026.

2 NHTSA recall data; Berylls by AlixPartners analysis (keyword classification of recall texts, 20 major light-vehicle OEM groups), October 2026. See also T. Warren, "2025 Automotive Software Recalls: Record Volume and the Rework Crisis", AutoSoftToday, 2026, https://autosofttoday.com/news/2025-automotive-software-recalls-record-volume-rework-crisis.

3 Berylls by AlixPartners, "Future Launch Management: Why the rules of the game have changed", July 2026, https://www.berylls.com/insights/future-launch-management-why-the-rules-of-the-game-have-changed/

4 See note 1.

5 Volvo Cars, press release of 11 May 2023 and Q2 2023 earnings call of 20 July 2023.

6 Chevrolet statement, December 2023, as reported by TechCrunch, 23 December 2023; IEEE Spectrum, "Snafu-Sidelined Chevy EV Hits the Road Again", 2024.

7 Volkswagen Group, Annual Reports 2023 and 2024 (CARIAD operating losses 2022–2024 of around €6.9 billion).

8 NHTSA recall filing, February 2026, as reported by Forbes, 26 February 2026.

9 NHTSA recall data, software-related recall campaigns 2021–2025 by stated remedy path; Berylls by AlixPartners analysis (October 2026).

10 Berylls by AlixPartners, complexity and software quality benchmarking of global OEMs, 2025.

11 See note 3.

12 S. Bräuer, "Automotive Software's Next Challenge Is Procurement – And It Needs to Be Addressed Today", Berylls by AlixPartners, November 2023, https://www.berylls.com/insights/automotive-softwares-next-challenge-is-procurement-and-it-needs-to-be-addressed-today/

13 See note 10.

14 See note 12.

15 See note 12.

16 See note 12.

17 J.D. Power, 2026 U.S. Vehicle Dependability Study.

18 HIL (hardware-in-the-loop): testing real control units against a simulated vehicle environment. G-HIL (full-vehicle HIL): HIL testing of the networked control units of the complete vehicle.

19 See note 12.

20 NHTSA, 2025 Annual Recalls Report.

21 NASA, Risk Management Handbook, NASA/SP-2011-3422, Version 1.0, November 2011 (criticality as the product of consequence and likelihood); applied to software defects by Berylls by AlixPartners.

22 See note 12.

©2026 AlixPartners, LLP