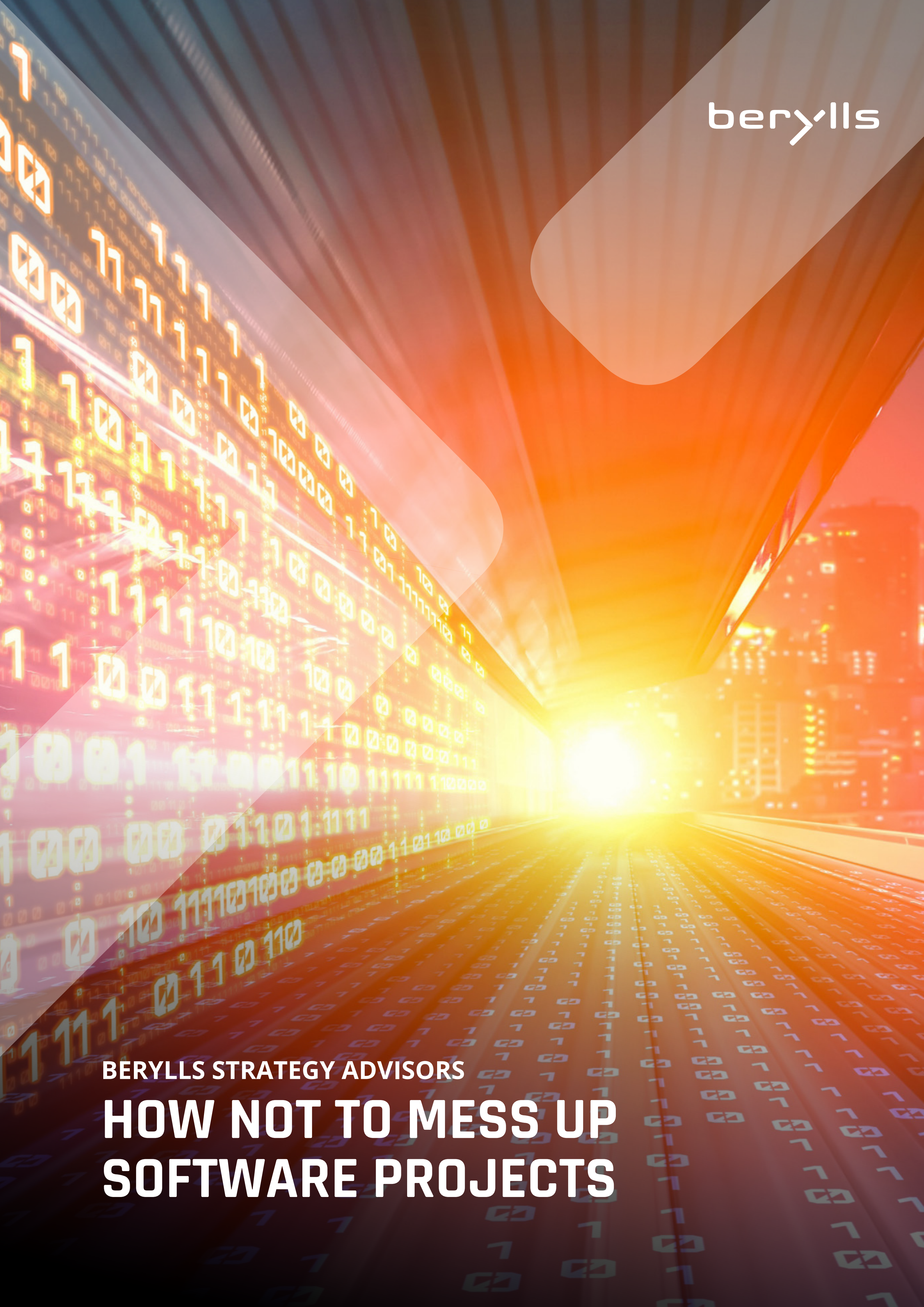


berylls

BERYLLS STRATEGY ADVISORS

HOW NOT TO MESS UP SOFTWARE PROJECTS

AGENDA

INTRODUCTION

Berylls' five-step software excellence framework enables OEMs and suppliers to combine their automotive expertise with best practices from Big Tech

Software is the key component driving innovation in modern cars, from electric engines to increasing levels of vehicle automation and infotainment systems. Yet while it is pulling the industry forward, software is also increasingly a source of problems for OEMs. Carmakers are being forced to recall vehicles already on the road, or postpone the start of production and sales, due to issues with new software.

Last year, one German OEM had to issue a recall affecting 1.3 million vehicles in the US and 2.6 million vehicles in China, due to defective emergency call software. A second German OEM had to stop production of one of its highest volume models for a few weeks due to software problems in 2019, again in the mandatory emergency call function.

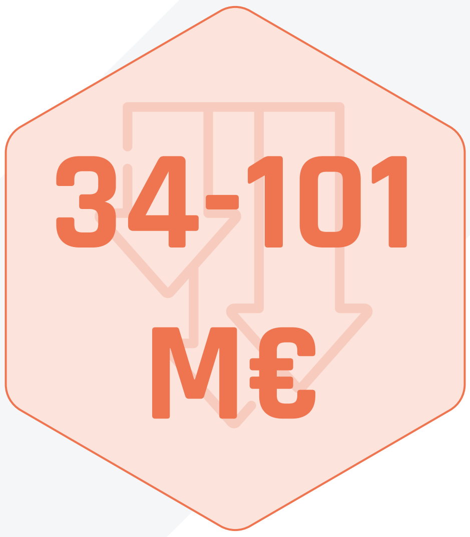


FIGURE 1

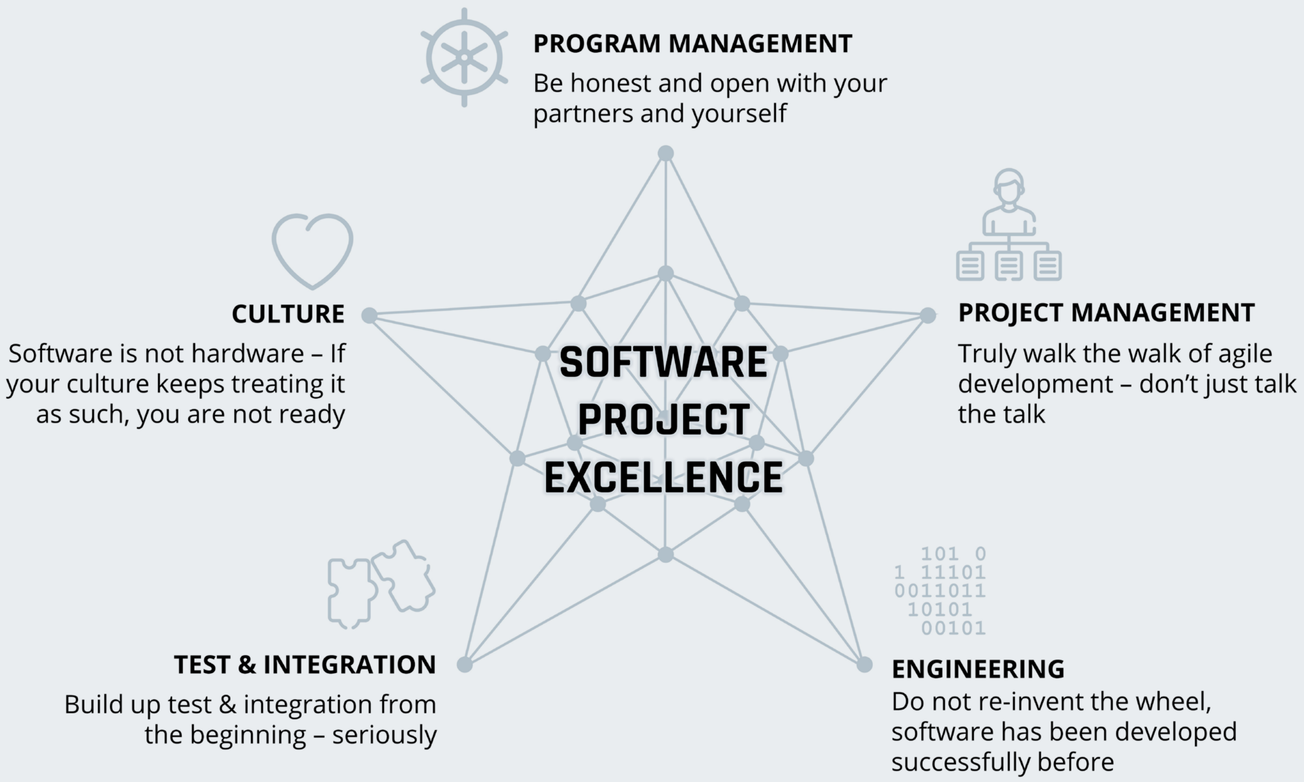
LOSS OF CONTRIBUTION MARGIN PER LOST PRODUCTION WEEK FOR PREMIUM VEHICLES

Such problems don't just impact carmakers' reputations for reliability and customer service - we estimate that for a premium model, the cost of a single week of lost production around the launch date could range from €34 million to as much as €101 million (Figure 1).

To avoid such additional costs, and to successfully tackle the challenges raised by future projects, we have developed a software project excellence framework around **five essential areas for action (Figure 2):**

FIGURE 2

FIVE ESSENTIAL AREAS FOR ACTION


PROGRAM MANAGEMENT
Be honest and open with your partners and yourself
CULTURE
Software is not hardware – If your culture keeps treating it as such, you are not ready
PROJECT MANAGEMENT
Truly walk the walk of agile development – don't just talk the talk
SOFTWARE PROJECT EXCELLENCE
TEST & INTEGRATION
Build up test & integration from the beginning – seriously
ENGINEERING
Do not re-invent the wheel, software has been developed successfully before

PROGRAM MANAGEMENT

Without aligning a clear goal within the organization and all partners even the best teams will not win the race

From Day One of a software development program, carmakers must communicate and align in a transparent way with all stakeholders involved. This includes suppliers, who should be treated as partners on the program rather than simply service or component providers.

The vital first step is to be clear from the outset about the aims (the 'what') of the program and each partner's contribution. We have noticed that this step is especially crucial in the development of innovative technologies, such as autonomous driving or augmented reality displays. Similarly, in joint projects between Big Tech and automotive players we have observed that too often the focus becomes the partnership itself, rather than the joint value proposition they are developing, and the contribution of each partner. Fix this by transparently defining every party's contribution and the joint target value proposition.

With clear goals in place, OEMs must then be realistic about the organization's capabilities in software development and integration. If the company wants to build something truly complex and new, it needs a clearly defined and well-functioning product development organization. If the OEM doesn't have one, it needs to start with a smaller development project first and build up the organization's capabilities – there is no point entering a Formula One race with a family car and then blaming the driver for not winning.

This doesn't mean thinking small – particularly in this period of industry transformation, OEMs should be visionary about what they want to achieve. However, it's important to be realistic about what tasks are better assigned to partners or outside suppliers, and the amount of time it will take to build up the organization's software program experience.

PROJECT MANAGEMENT

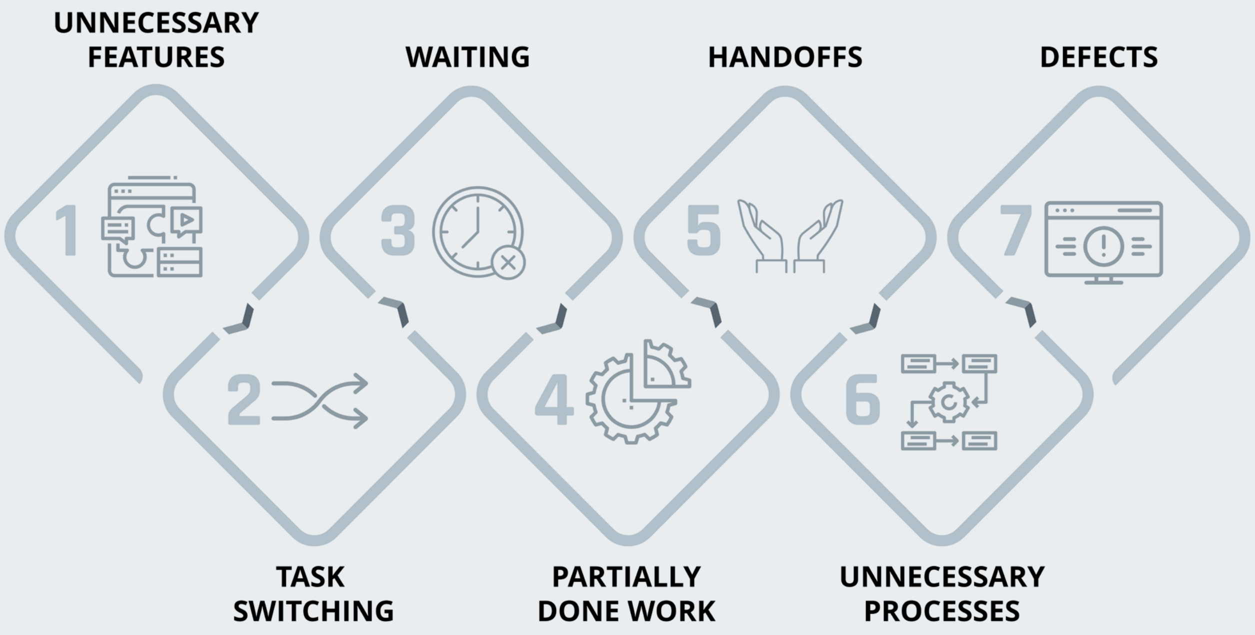
Applying lean principles to software development and tracing everything back to business requirements helps to avoid blind efforts and soaring expenses

The automotive industry has perfected lean management in production to avoid unnecessary waste. Happily, lean principles also apply to software development, where researchers have identified seven key areas of waste[1] **(Figure 3).**

Similar to the teachings of lean, these principles can appear obvious at first. However, based on our global experience working on more than 50 task force and project recovery assignments, we have seen that they are often neglected, especially in heated project situations. Focusing on these areas from the outset will help keep projects on track.

FIGURE 3

SEVEN AREAS OF WASTE IN SOFTWARE DEVELOPMENT



[1] Poppendieck, Lean Software Development: An Agile Toolkit (2006)

Unnecessary features | When it comes to software projects, carmakers should apply the lesson learned from building supercars: customers value premium quality over a huge array of features. To keep software projects lean, project managers and stakeholders should set a clear scope and complete each element before advancing to the next stage. They must not let scope creep happen by adding unnecessary features or start changing the task list as the team works through it.

Task switching | To maximize the time software engineers spend on value-adding activities, OEMs should automate processes wherever possible. Time is also lost to task switching when people have more than one role; for example, a combined lead developer and product owner (PO). The PO has the best overarching understanding of what the product should look like, and is responsible for connecting with all stakeholders. They should focus on that exclusively.

Waiting | To foster a quick decision culture, rigid hierarchies must go. Instead, teams should be empowered to take decisions, test the results, and pivot accordingly. Unlike hardware, software development is iterative and benefits from early and continuous feedback. When changing tack, data beats opinions.

Partially done work | Maintain a clean backlog with a clear focus and keep the details of the backlog items up to date. Items no longer needed should be removed or deprioritized – do not create zombie tasks.

Hand-offs | Fewer unnecessary hand-offs between suppliers, partners, and the OEM will cut waiting times and increase product quality due to the prevention of know-how loss in transmission. To achieve this outcome, trust suppliers and partners and hand over some control. Give them room to innovate and deliver meaningful increments of change.

Unnecessary processes | The automotive industry loves processes (and we will see why that is good in some cases later in this report). However, to make software projects lean, companies must adopt a crucial part of the Agile Manifesto: Individuals and interactions before processes[2]. This means trusting their employees and fostering open communication.

Beyond these key areas where time, effort, and money are wasted, there are **two other common mistakes** that OEMs need to guard against in their software programs:

❯

[2] Beck et. al. 2001, Agile Manifesto (2001)

Firstly, spending years of manpower working on the perfect set of requirements for the software before getting started. Instead, companies should focus on the most valuable capabilities that the feature must include to meet customer needs and regulatory compliance. From there, continuously refine the product by breaking down the requirements from the overall capability level into elements that can be developed by software teams.

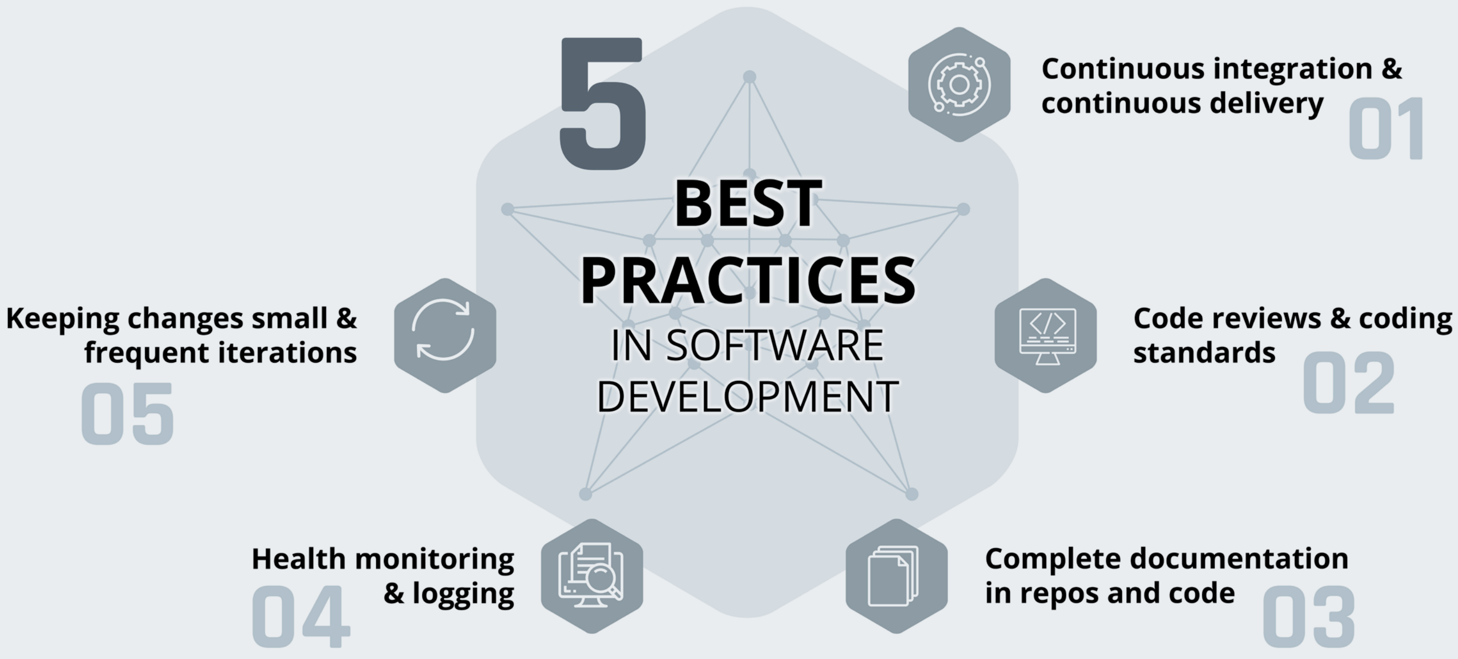
Secondly, as business requirements are turned into technical requirements and then features are developed and tested, testing results should be linked back to the original business requirement from the earliest stages. This ensures features are fulfilling the brief. Maintaining this level of traceability might seem daunting, particularly when working in an agile way – but it ensures that the correct product is being developed.

ENGINEERING

Don't be too proud to learn from the best in software engineering but maintain what made you the best in automotive engineering

FIGURE 4

BEST PRACTICES


5 BEST PRACTICES IN SOFTWARE DEVELOPMENT
Continuous integration & continuous delivery 01
Code reviews & coding standards 02
Complete documentation in repos and code 03
Health monitoring & logging 04
Keeping changes small & frequent iterations 05

5 BEST PRACTICES IN SOFTWARE DEVELOPMENT

While program or project management standards are important, many automotive software projects also fall short simply because of poor engineering quality. To improve performance in this area, there is no need to reinvent the wheel. There are well-established best practices to draw on from both software and automotive engineering.

In software, these include **(Figure 4):**

» **Establish an automated, reliable continuous integration/continuous delivery (CI/CD) pipeline from the start of a program, to ensure high-quality code and fast feedback to developers, allowing them to work at speed.**

» **Agree on a common process and tool for code reviews. Focus not only on the design, functionality, and complexity of the code but also on readability, by adhering to coding and naming conventions and the amount of comments needed. This creates quality code that will be reused and maintained long into the future.**

» **Across all stages, from source repository down to code, documentation should be treated as being as important as the code itself. This is because a clear 'paper trail' greatly affects reusability, comprehensibility, and resilience when there are staff changes.**

These high-quality practices for developing code are paramount. By looking at Big Tech companies, for example Google[3] or Microsoft[4], we noticed how deliberate these firms are about measures such as code reviews or testing and design guidelines. By comparison, in the automotive sector, we see that some engineers follow best practices, while others muddle around in the code without reviewing their changes with peers. It is worth spending time ensuring the organization has such strong quality measures for software in place.

Automotive engineering standards matter too – software teams should not set them aside just because they are "old". The basic concepts are still needed today:

» **Use ASPICE (Automotive Software Performance Improvement and Capability Determination) standards to break down requirements and build up the solution while integrating and testing along the way, although the change or the feature that is run through the process might be smaller compared to classic software development V-cycles.**

» **Safety standards such as ISO 26262 are still paramount – not only for certification but also for society and customers.**

» **Pay attention from the beginning to compliance in general (legal, environmental, safety, security) to ensure the product has built-in compliance, rather than looking the other way and hoping any problems will magically solve themselves.**

[3] https://google.github.io/eng-practices/review/
[4] https://microsoft.github.io/code-with-engineering-playbook/

TESTING AND INTEGRATION

Classic gated approaches designed for hardware will not work with software. Reliable risk reduction only comes with early end-to-end design validations and continuous testing and integrating from the very beginning

Testing and integration are essential for the success of software products – they are an inherent part of the development and not something to be added on late in the process. As a result, procedures need to be set up right at the beginning of the project. This is well known now, but few projects stick to the rule in reality. Buck the trend and just do it.

The reasons are clear: testing needs to be carried out end-to-end to get feedback on every iteration of the solution. That means there must be measurable acceptance criteria for each incremental change and a test pyramid with proper foundations. Proper testing starts at the detailed level, with each test level including more and more parts of the system, until system-level validation is carried out.

Working to agile principles is no reason for omitting any of the necessary testing levels – in fact, built-in quality is one of the intrinsic values of agile. Acceptance criteria for test results must be concrete and measurable, and tasks should not be initiated without them.

An incremental, agile approach allows for quick validation of design decisions - essential for the development of a complex system like car software. But it comes at a cost. Only parts of test routines and test data are reusable, so each new test requires resources. On the other hand, waiting for the system as a whole to be finished does not work either, since incremental development and test cycles are essential to reach a feasible product design in the first place.

```
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

    #selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
    #mirror_ob.select = 0
```

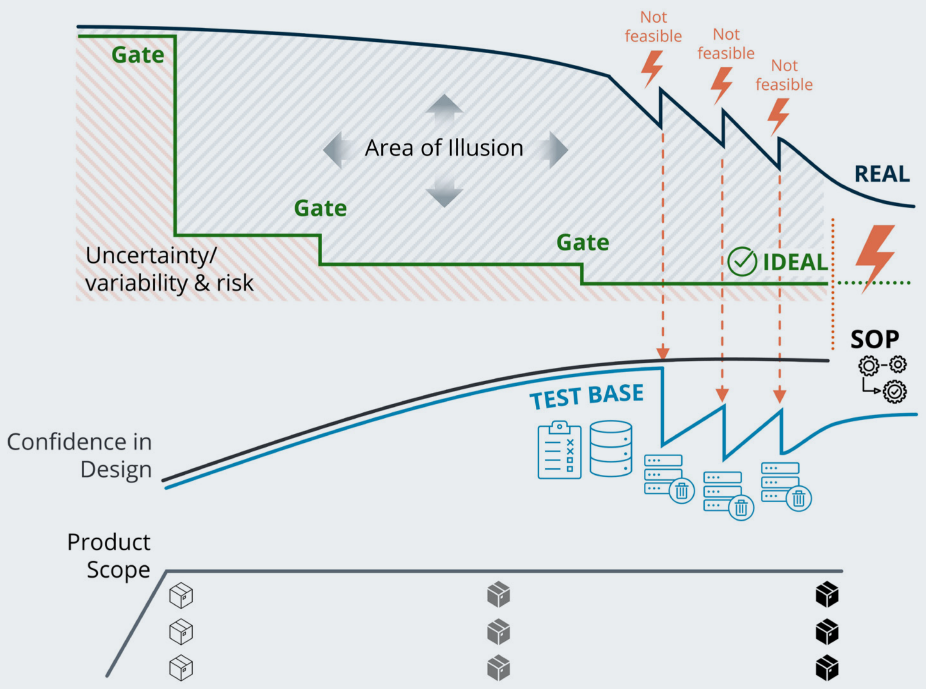
In the case of automotive software, the "classic" waterfall software testing approach with fixed maturity gates and architecture freezes does not apply. The same holds true for the selective approach to agile taken by some automotive companies, which only adopt parts of the agile framework. To work out, elements of both approaches must be included and managed, particularly with safety-critical software.

Take an autonomous vehicle as an example. The idea that an OEM could design a complete system from scratch simply doesn't work when there are no pre-existing system designs to draw on, as there are for example with engines, and without testing as they go along. If the organization postpones real design validations until late in the process, failures will be prohibitively costly and will endanger production (SOP) deadlines. The best option is to build proofs of concept (POCs) and fail early, then learn and move on to a better solution.

On the other hand, building an autonomous vehicle without thinking of safety and compliance early in the process,

FIGURE 5

GATED APPROACH TO DEVELOPMENT & TEST



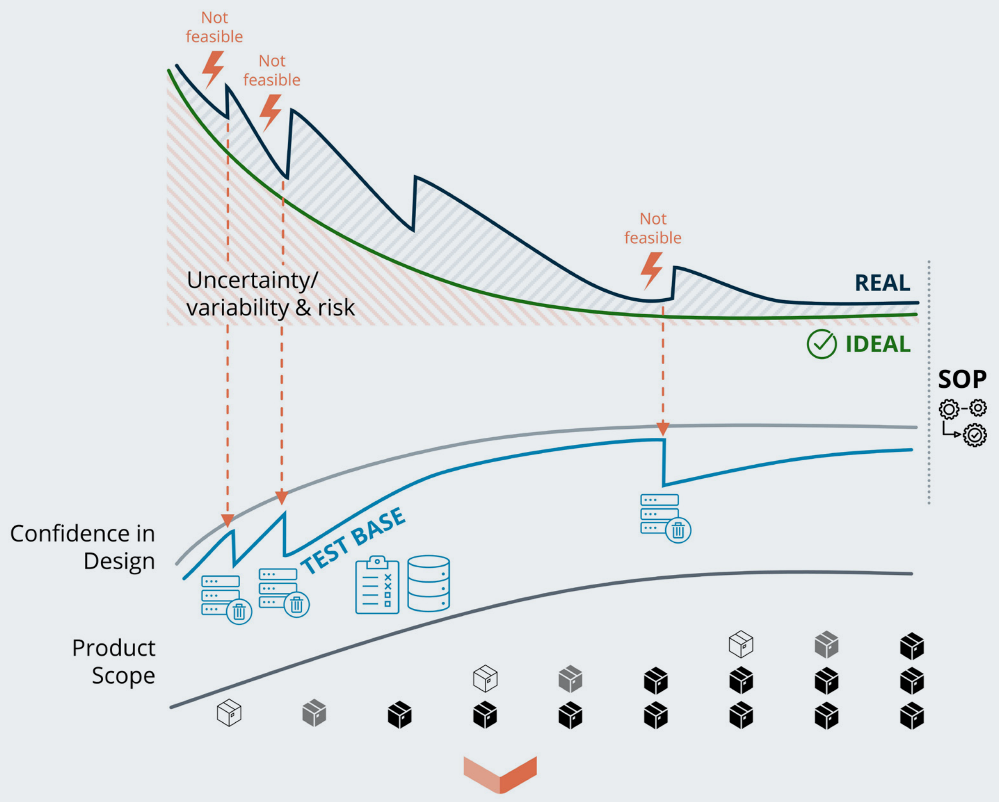
A GATED APPROACH SIMULATES RISK REDUCTION WHILE LEADING TO LATE INFEASIBILITY ISSUES WITH HEAVY LOSS OF TEST BASIS ENDANGERING SOP

and the time needed to achieve it, will lead to fiddling around with the software indefinitely without it ever reaching the maturity needed to get a car on the road.

Successfully managing these two different approaches means using them together to narrow down the amount of uncertainty in the project, as Figure 5 below shows. During product development, decisions on the shape and features of the product are taken iteratively and sometimes reversed, starting with the biggest ones. As the design becomes more clearly defined, confidence in the quality of the product increases, as a more extensive base of test data is available.

It's possible to make big changes late in product development, rendering parts of the testing process obsolete. However, the cost of doing so can and must be weighed against the possible benefit of making the change. Deciding on changes for a maturing product becomes increasingly tough and needs full management buy-in.

FUNNEL APPROACH TO DEVELOPMENT & TEST



A FUNNEL APPROACH REDUCES RISK EARLY IN DEVELOPMENT AND MINIMIZES LATE LOSS OF TEST BASIS ALLOWING FOR TIMELY SOP

CULTURE

Becoming a software company is an arduous process of cultural change including letting go of selected core believes – but the gains outweigh the pain manifold

Software is quickly becoming the driving force for automotive innovation, but long-established carmakers were not designed to manage software projects. As a result, important cultural and practical obstacles stand in the way of doing so successfully. Culturally, automotive companies tend to have very hierarchical structures with only a few senior managers taking most of the im-

portant decisions. Those managers and their management practices are in general the product of years of experience developing automotive hardware.

In practice, this means a small group of people with only limited knowledge of the actual subject will take most of the important decisions on a software program. This is a recipe that is doomed to fail, often because they simply lack the necessary understanding of how software is different.

Instead, successful software development happens in a culture of problem-solving through collaboration, rather than penalizing people for failure. An essential part of creating such a culture is the belief that team members are only falling short on projects due to the inherent complications of an uncertain process, rather than because of personal unwillingness or lack of motivation.

If problems occur, it is essential not to heap blame on teams and turn up the pressure through tight control and reporting. Instead, software program and project leaders should focus on helping to solve the problem with all relevant resources. Pulling struggling team members into never-ending processes of reporting and top-down control only decreases their problem-solving ability.

OEM software task force managers brought in to turn around struggling programs tell us that the efficiency of their teams increased significantly after bringing an external party on board to manage the day-to-day work of the task force, and let developers focus on fixing errors.

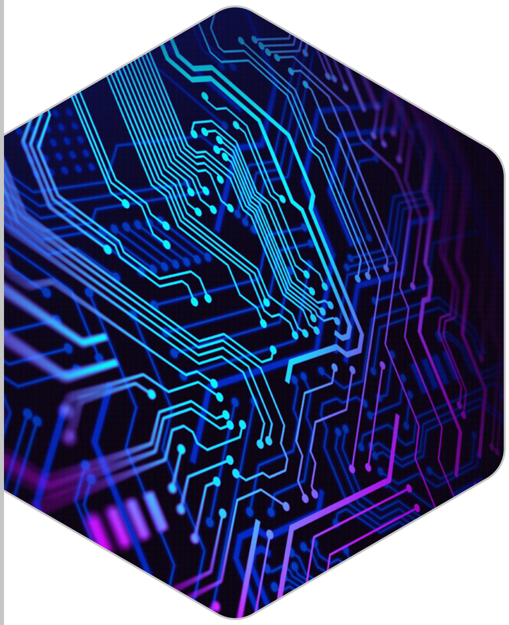
Managing complexity

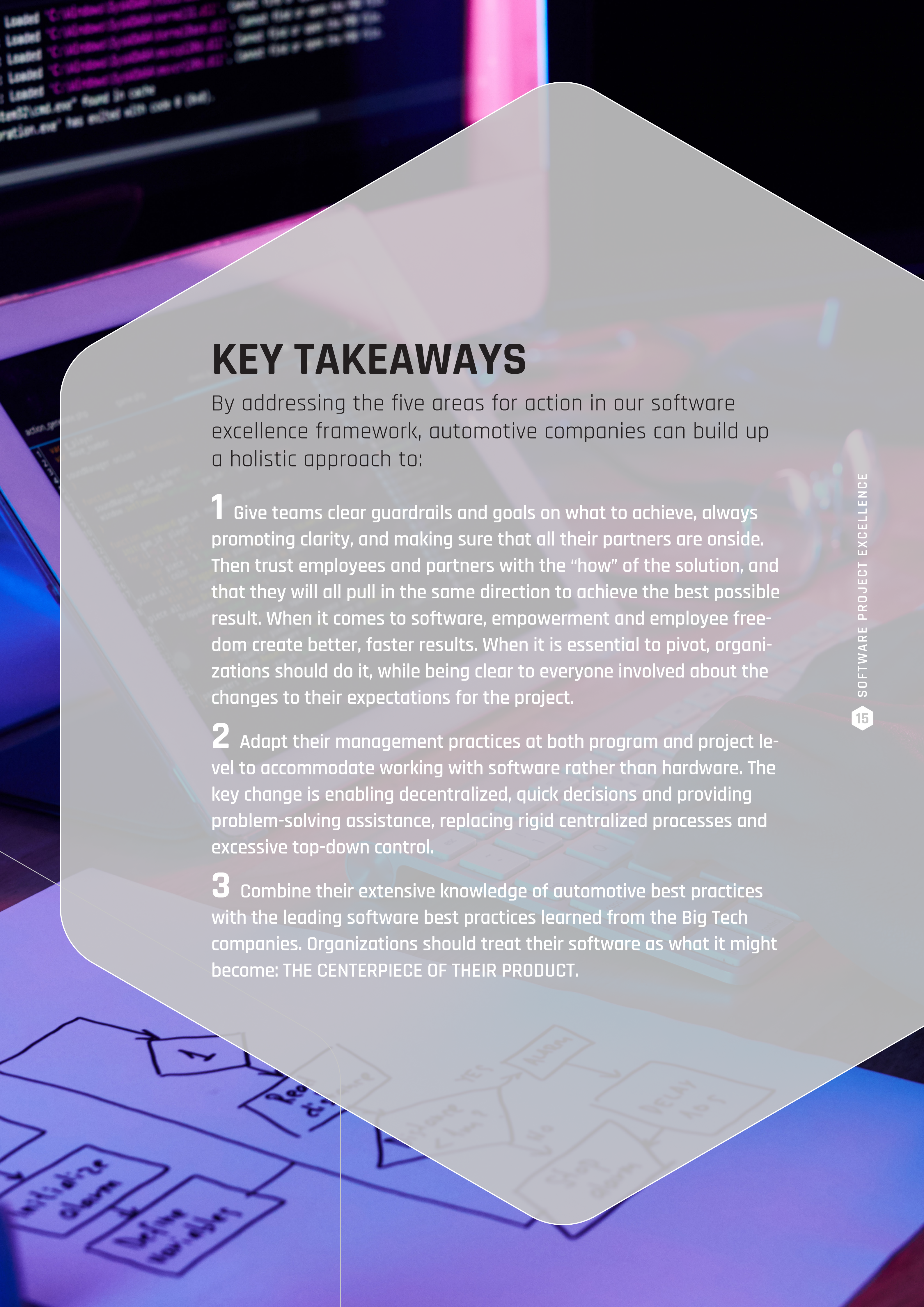
In addition, in a more complex and volatile automotive market, projects often need teams with different work approaches, different cultures, and different languages to work together across several time zones. This is not an easy task and requires substantial knowledge and experience of inter-cultural working to make it a success. OEMs should not ignore this fact in their project and organizational setup, because without careful consideration and management from the outset, a lack of mutual cultural understanding within teams will slow projects down significantly and increase tensions and frustration.

We were able to overcome the issues in software development in one example involving a task force made up of two European software players, by co-locating the development teams for two months to conquer their cultural difficulties. After this period of team-building and co-development, the teams maintained their newly found increase in productivity, even after returning to work at their home locations.

Overall, the culture and people in an organization must match the desired way of working. An all-out agile approach with teams made up of automotive old hands trained on hardware is likely to cause some friction. In many cases, a radical approach just does not work with the organization. Making statements like “from today on, we act as one team and do everything differently” will not make it happen at once.

Instead, OEMs should be realistic about where the organization’s people and culture are currently, and factor in substantial time for change if the current state does not yet match their ambitions.





KEY TAKEAWAYS

By addressing the five areas for action in our software excellence framework, automotive companies can build up a holistic approach to:

1 Give teams clear guardrails and goals on what to achieve, always promoting clarity, and making sure that all their partners are onside. Then trust employees and partners with the "how" of the solution, and that they will all pull in the same direction to achieve the best possible result. When it comes to software, empowerment and employee freedom create better, faster results. When it is essential to pivot, organizations should do it, while being clear to everyone involved about the changes to their expectations for the project.

2 Adapt their management practices at both program and project level to accommodate working with software rather than hardware. The key change is enabling decentralized, quick decisions and providing problem-solving assistance, replacing rigid centralized processes and excessive top-down control.

3 Combine their extensive knowledge of automotive best practices with the leading software best practices learned from the Big Tech companies. Organizations should treat their software as what it might become: THE CENTERPIECE OF THEIR PRODUCT.

BERYLLS STRATEGY ADVISORS

YOUR CONTACT PERSONS

T +49-89-710 410 40-0

info@berylls.com

Timo Kronen
timo.kronen@berylls.com
T +49 170 22 38 992

Sebastian Böswald
sebastian.boeswald@berylls.com
T +49 151 41 88 73 18

Martin Ruchti
martin.ruchti@berylls.com
T +49 160 96 24 34 52