AUTOMOTIVE FLOSS

State of Practice 2021. A survey on Free/Libre and Open-Source Software in the car industry

KUGLER MAAG CIE

Contents

['ɔʊ̯pn̩ sɔːɹs]

Op·en source

Open source software is source code that is made freely available for possible modification and redistribution.

Executive summary

Free/Libre and Open-Source Software (FLOSS) has completely transformed many industries for consumer goods. A prominent example being the mobile handheld industry where in a short amount of time, proprietary solutions were almost completely replaced by large FLOSS stacks including drivers and operating systems (e.g. Android OS).

In contrast, open-source projects in the automotive industry are seemingly not reaching critical mass – despite many announcements and initiatives.

To better understand this contradiction, we conducted this survey within the State of Practice industry barometer series. Through interviews with (open-source) software experts at manufacturers, Tier 1..n suppliers, as well as experienced industry outsiders we wanted to record the current status quo and take a glimpse into the future the respective interviewees see for FLOSS in automotive.

Given that automotive systems are increasingly defined and differentiated by software-driven functionalities and services, the question arises: Is the automotive industry next in line to be transformed by FLOSS?
Doesn't the ever-growing importance of software in the connected car call for the implementation of an accessible and reliable software base on which the differentiation of the respective vendor could then be built? FLOSS could provide the infrastructure enabling smooth connectivity with other vehicles and with back-end servers. On the other hand: The automotive industry has its own constraints, strict standards for functional safety (ISO 26262) and cybersecurity (ISO/SAE 21434) as well as expectations for process maturity (Automotive SPICE). At first glance, some FLOSS projects and methods seem to contradict these non-functional expectations. In addition, the automotive industry is highly constrained by compliance and warranty obligations, as well as the protection of each company's individual intellectual property. How would FLOSS challenge established collaboration and respective business models?

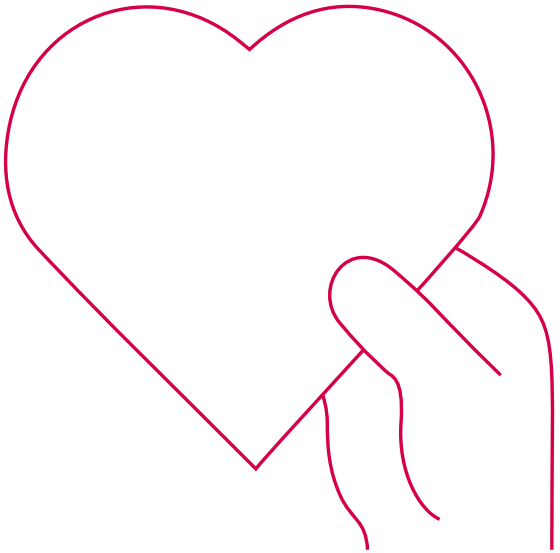
For the experts interviewed, the consensus was that FLOSS is currently already heavily used in the tool chain during product development and that it has already made its way into the actual vehicle and its embedded systems. It is currently mostly found in systems which require a lot of connectivity such as the info-tainment system. When it comes to any safety relevant system, there is a lot of hesitation in using FLOSS as its development often does not follow (or is believed not to follow) the systematic approach required by ISO 26262. However, the spread of FLOSS throughout the vehicle will continue and be accelerated through the trend of consolidating functionalities into more powerful centralized controllers. Abstracting the hardware of such complex server-like hardware will require large FLOSS stacks as the complexity cannot be handled and maintained without it.

Regarding the business case for using FLOSS, most interviewees responded that currently it is mostly limited to saving cost and time to market for individual development projects. The use of (and contribution to) FLOSS is not yet a strategic topic that could lead to different business models for most established automotive companies. As a result, the automotive industry is not a driving force in (m)any of the big FLOSS ecosystems. The industry is mostly “piggybacking” of the results that companies of other industries have created. There is a large internal resistance in established automotive companies to hand over the results of any internal effort to the public. Even if there is a promise of saving even more effort in return.

One topic that all companies focused on in our interviews is the talent pool that creates and maintains FLOSS. Software developers are in high demand and there is a shortage of qualified personnel. FLOSS projects and their documentation allow to see the technical expertise of a prospective employee and how they work (social skills) without even having had an interview yet. Furthermore, having a generous open-source policy attracts young and qualified talent. From their point of view being an expert for a successful FLOSS project increases their personal market value more than being a top expert for one proprietary software of one automotive company.

These were just some findings for topics we investigated for this survey. More details on the following topics can be found in the full survey results below.

- product portfolio, innovation, and maintenance
- business case and business model
- building ecosystems
- regional differences
- regulation and compliance
- governance and excellence
- talent pool and company culture
- enablers and hinderers



Acknowledgements

Without the experience of numerous experts and their willingness to share their knowledge, this State of Practice survey would not have come about. Therefore, we are indebted to all contributors for their openness and commitment.
First and foremost: Carl-Eric Mols – the former Head of Open-Source at Sony Mobile and now an open-source consultant at Addalot Consulting in Sweden. We got to know Carl-Eric through his and our (Kugler Maag Cie) engagement in the SCALARE project[1] and about a year ago he approached us with a comparison of the automotive industry now and the mobile handheld industry in the early 2000s. The similarities were plentiful and of course we all know the massive shift to open-source that happened in the handheld industry shortly afterwards. Proprietary solutions were almost completely replaced by large FLOSS stacks including drivers and operating systems (e.g. Android OS). Seeing that automotive systems are increasingly defined and differentiated through software functionalities, the question was born: Is the automotive industry next to be transformed?
Throughout designing this survey as well as executing it, Carl-Eric has continuously supported the authors with his open-source insights and continuous engagement in groups like the TODO group[2] establishing contacts and enabling interviews: Thank you very much, Calle!
We furthermore would like to thank our industry partners, that supported us with ideas and the survey design when we were merely in a brainstorming stage and served as an advisory board regarding the questionnaire. A big "THANK YOU!" to these partners:

- BMW Group – Simon Fuerst
- Bosch – Stefan Ferber, PhD
- Continental – Klaus Claas-Dieter
- Volkswagen – Dieter Mannigel
- Addalot Consulting – Carl-Eric Mols

1 The European SCALARE project: https://itea4.org/project/scalare.html
2 TODO group: https://todogroup.org/

Motivation

Our motivation for this survey grew from a perceived mismatch between many enthusiastic press releases by many automotive companies and the very limited observations of automotive companies contributing to FLOSS projects.
Such press releases talk about the importance and strategic value of FLOSS. That the complexity of a modern vehicle can only be managed and controlled by working together and bundling resources. That having generous open-source policies will make companies more attractive as employers for young and open minded talent. They sound like there is only one topic in the industry at the moment: FLOSS!
We mirrored this impression against our consulting projects with companies of all sizes and with vastly different product portfolios. Most of them reported that they have rarely witnessed their customers incorporating open-source into their embedded products and that even fewer have ever seen active contributions to the used project.
This impression was compounded by the fact, that no top-contributors statistics ever show (core) automotive companies among the top ten or even top fifty[3].
Of course, the automotive industry has its own boundary conditions. There are strict standards regarding functional safety (ISO 26262), cybersecurity (ISO/SAE 21434) and a general development state of the art (Automotive SPICE). At first glance, some FLOSS projects and methodologies seem to contradict these systematic expectations. Beyond that, the automotive industry is highly driven by compliance and warranty obligations as well as protecting each company's individual intellectual property. How would FLOSS challenge the corresponding working models and business cases?
We just had to find out and spent the next few months designing and conducting this state of practice survey regarding the use and management of FLOSS in the automotive industry.

3 [https://www.infoworld.com/article/3253948/who-really-contributes-to-open-source.html] [https://www.techrepublic.com/article/who-contributes-most-to-open-source-the-answers-will-definitely-surprise-you/] [https://opensourceindex.io/]

OSS vs. FOSS vs. FLOSS – and why we use the term FLOSS throughout this survey report

OSS

There are a lot of terms and abbreviations when it comes to non-proprietary software with open code sources.
The most commonly used term is surely "Open-Source Software" abbreviated as "OSS". However, that is not exactly the target of this survey as proprietary vendors can choose to sell their software with open sources for transparency while still insisting on licensing fees.

FOSS

Hence, the term "Free and Open-Source Software" abbreviated as "FOSS" is preferred in large parts of the community to immediately make clear that there shall be no commercial attachment to the source code.

FLOSS

However, even in this "FOSS space", there are certain licenses that limit the use or allow integrators to limit the use of the resulting software. The English term "free" is ambiguous as it can both mean "it does not cost anything" and "it guarantees freedom". Hence, the "free software movement" that opposes such limitations of use, introduced the term "libre" as it unequivocally determines the state of "freedom". The resulting term is "Free/Libre and Open-Source Software" abbreviated as "FLOSS".
The debate between the "open-source movement" and the "free software movement" is ongoing and the authors of this survey want to report on the entire space of non-proprietary software solutions without picking sides.
Hence, we use the most inclusive and neutral term "FLOSS"[4] throughout this report.

4 Neutral use of the term "FLOSS": https://www.gnu.org/philosophy/floss-and-foss.en.html

Methodology

We decided early on through brainstorming sessions with our industry partners and the advisory board that we would conduct a qualitative study as opposed to a quantitative investigation.
Through several iterations we designed a survey that would form the basis for our discussions with future expert interviewees.
The survey covers nine major topics and confronts interviewees with a thesis and an antithesis for each one. In addition, there is a set of open questions to challenge the interviewees with.

Product portfolio, innovation, and maintenance

THESIS	FLOSS will take over all aspects of automotive car software (embedded & back-end): FLOSS is a prerequisite for innovation and differentiation.
ANTHITHESIS	FLOSS will never be a significant part of any ECU inside the car.
RELATED QUESTIONS	• Where is FLOSS already used now and to which end? • Which automotive systems will be FLOSS dependent – never vs. certain? • How do current and future systems' architecture hinder or foster FLOSS usage?

FIGURE 1 Example of questionnaire structure

If you are interested or would like to facilitate a similar discussion within your company, you can download the questionnaire at KUGLERMAAG.COM/AOSS2021

Once the survey outline was finalized and agreed with all partners, the interview sessions were scheduled with selected (open-source) software experts.

The interview sessions included roles like …

- Heads of Open-Source
- Open-Source Program Officers
- Heads of Software Development
- Compliance Officers
- Software Development Engineers
- Researchers on software development strategies

… working at …

- OEMs / Vehicle manufacturers
- Tier 1 .. N / Suppliers
- Universities, Foundations

… in

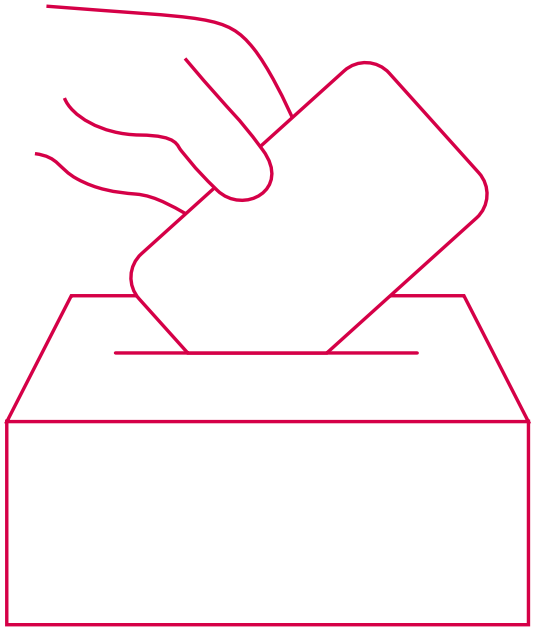
- Europe
- Asia
- North America

Given that the interview duration was scheduled to 1–2 hours each and one could spend much more than that on a single topic, the interviewees were asked to prioritize the topics. They did so based on either where they have the most insight or where they see the most risk for the industry.

The companies and institutions kind enough to participate include:

- BMW
- Bosch
- Continental
- Denso
- Elektrobit
- Scania
- Toyota
- Volkswagen
- ZF
- Airbus
- Addalot Consulting
- University of Erlangen-Nürnberg
- Lero – Irish SW Research Centre
- Eclipse Foundation

Others chose to participate in anonymity, and we respect their wishes.



Detailed results

Product portfolio, innovation, and maintenance

An unsurprising result was the unanimous statement of all interviewees that the respective development toolchains are already heavily incorporating FLOSS. All of them stated that benefits largely outweighed risks. They can avoid license cost and vendor lock-in while the effort overhead compared to proprietary solutions is minimal even in a safety relevant context.

Where the safety relevance plays a bigger role is for the embedded products of the companies. Many stated that there is no use of FLOSS for their functional safety relevant parts as the effort of qualifying this third-party development and monitoring its further evolution is often higher than a proprietary in-house or supplier development.

In contrast, there is already heavy use of FLOSS in all systems that incorporate a lot of connectivity but are not safety relevant (such as infotainment systems). The communication sector is very much built on FLOSS by now and integrating FLOSS reference implementations of communication protocols saves enormous efforts as well as time to market. Without functional safety constraints to worry about, it is an easy choice to make.

Another aspect the interviewees considered was the maturity of the product itself. For many of their systems FLOSS would actually be an option. However, since they are developing the >3rd generation of the same system and are only making evolutionary changes and not revolutionary ones, it makes more sense to re-use the existing proprietary solutions, rather than refactoring the entire system just to incorporate FLOSS.

What might trigger such a refactoring though and thus lead to more FLOSS is a well-known automotive megatrend: consolidated vehicle architectures (see FIGURE 2). This is the trend to reduce the overall number of electronic control units (ECUs) by consolidating functionality in fewer more powerful domain control units (DCUs) and finally converging in one or two server-like high performance vehicle control units (VCUs).

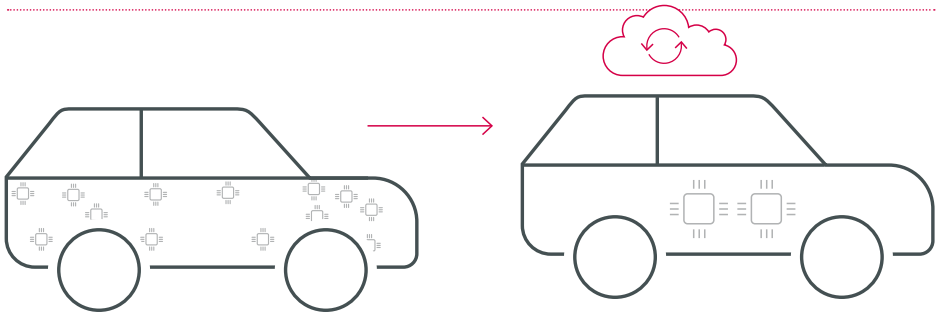


FIGURE 2 Towards a consolidated, server-like vehicle architecture with cloud connectivity

There are multiple reasons why this trend can and likely will accelerate the spread of FLOSS throughout the car:

- The more software drives the differentiation of functions and services, the more important it becomes that it is scalable, maintainable, and updatable. In turn, hardware is expected to provide a reliable, robust, and cost-effective server-like basis for software. This will lead to the use of standardized hardware components. The number of hardware variants, even between different vehicle manufacturers, will decrease. Fewer hardware variants will lead to more standardized hardware abstraction layers. This increases the need to collaborate on the development and maintenance of these abstraction layers, for example based on FLOSS. From a connectivity perspective, this standardization will support smooth communication and interaction between fleets of different manufacturers.
- A common approach to implement the different vehicle functions at software level on such a consolidated/centralized server is not to treat them as a single enormous software, but rather to separate them from each other using hypervisors. This means: Safety relevant software parts can reliably be separated from non-safety parts. Where non-safety parts must be refactored to run in this new environment, there is now an easy choice to use FLOSS components if they are available.
- Finally, even the safety relevant parts will need to be refactored and there are more and more FLOSS options that might be incorporated during this process. More on that in chapter FUNCTIONAL SAFETY (ISO 26262).

Virtualization by hypervisors

A "hypervisor" (as described in this chapter) is a software that typically runs directly on the hardware of a powerful server-like execution platform i.e. on the "host". It can abstract and allocate the hardware resources to multiple "guest" environments to run different operating systems (see FIGURE 3). This host-guest concept allows the parallel execution of multiple runtime environments on the same hardware. The hypervisor can ensure that those environments are entirely independent from each other and cannot e.g. block each other's execution as the hypervisor is in charge of the hardware resources. If the hypervisor itself is developed according to a sufficient "safety integrity level" ("SIL" – or more automotive specific "ASIL"), this can enable the parallel execution of e.g. a non-safety Linux environment and a safety relevant ASIL D real-time environment.

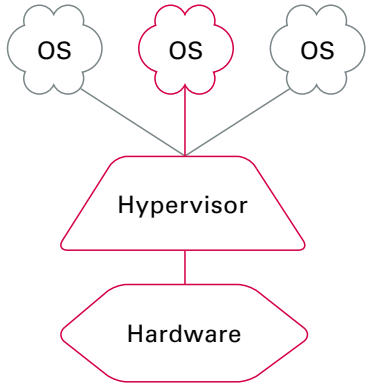


FIGURE 3 Example visualization of a hypervisor

Looking at FIGURE 4 the expectation is that FLOSS will spread from right to left i.e. from the non-safety infotainment parts towards the safety relevant vehicle control parts. All depicted software solutions or initiatives (e.g. Linux, AUTOSAR, GENIVI) are merely examples and not a recommendation for or against their use in this or other use cases.

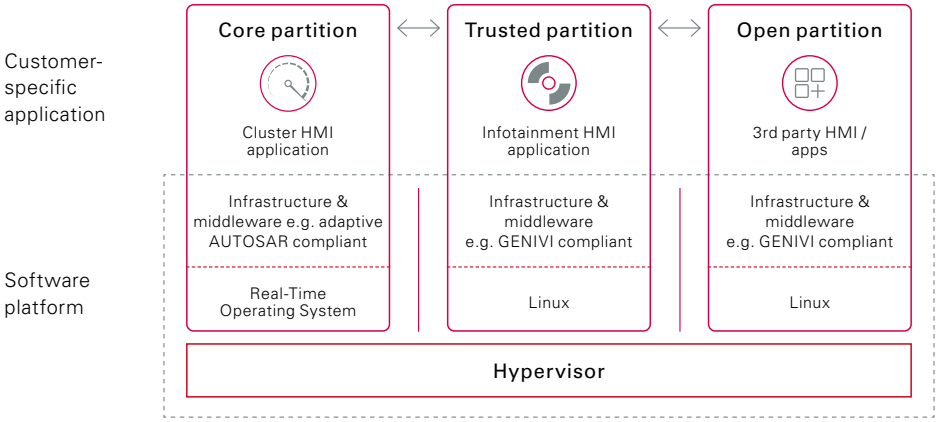


FIGURE 4 Example of a consolidated automotive software stack

In summary, at this time every interviewed company already has at least one embedded system that uses FLOSS and they all agreed that the use will only increase from here.

Business case and business model

Currently, the business case for FLOSS basically comes down to saving time to market and reducing the initial investment. All interviewees clearly stated that their companies understood that "free/libre" and open-source software still is not "free of cost". It comes with license obligations that can cause effort and their integration and maintenance definitely will cause significant effort and hence cost. However, since in many cases, there is already a huge software stack available the initial effort is reduced and a lot of time is saved by not "*writing requirements, finding a supplier, negotiating, developing, acceptance testing, …*". Beyond that however, FLOSS does not play a significant role in strategic planning in most of the interviewed companies. FLOSS is viewed as a solution to a specific engineering problem, but not considered a game changer e.g. for their own business model.

"If the automotive business becomes a software business, it will also become an Open Source Software (OSS) business. Unfortunately, as of today software is just part of the ECU. Hardware is what determines the product value and price. Hence, Bosch IoT Suite is based on and actively supporting Eclipse IoT for device and vehicle connectivity."

STEFAN FERBER, PHD, CO-CEO & CTO OF BOSCH.IO

One symptom of that is that few of the companies actually have rules in place that allow for active contributions to FLOSS projects and even fewer subsequently have evidence for their regular engagement in the projects they benefit from. One anonymous interviewee stated that the industry as a whole was "*piggy-backing of the results achieved by other industries*".

When pressed on why that was the case, multiple factors were identified:

- Many companies cite warranty and liability concerns whenever their engineers and developers request to participate actively in FLOSS projects.
- One culprit that was mentioned both by top level executives as well as project level engineers was the (infamous) middle management. This middle management supposedly still has not understood how investing their budget into the development of open software would benefit them and their projects. It is still not understood that investment into FLOSS can yield multiplied returns. Hence, the same patch is applied over and over again when integrating the new version of the FLOSS stack, rather than contributing the patch once and having it integrated into the next version after a free and thorough review by the community.

- A legitimate concern which still seems to be cited too often is "*giving away intellectual property*". The automotive industry has spent over a hundred years cultivating a protective environment. No details are ever discussed without a non-disclosure agreement and ownership of every single detail of a system is contractually mandated. One can imagine that it needs a big change in mindset to come around to FLOSS and giving away ideas and their implementation in the hope of getting similar ones back.

As it stands then, many established automotive companies mostly reduce FLOSS to helping them where opportunistic and ensuring that lack of license compliance does not become a business risk. They leave it to outsiders to generate most of the value. In turn, they also leave it to outsiders to reap the revenue that can come with understanding and steering FLOSS projects. Just look at examples like Red Hat or GitHub to see that there are billions of dollars to be made with a smart open-source centric business model.

Open-source business models and strategies

Open-source strategies can go far beyond simply solving an engineering problem and saving time to market in one particular project by consuming the available source code.

An initial step of strategic planning is to identify FLOSS components that are critical to the success of the company, then starting to influence the FLOSS development through regular contributions and getting a say in its leadership. Else, the FLOSS project entirely changing course and focus represents a business risk for the consuming company.

Similarly, company internal components should be regularly examined regarding their strategic value. A proprietary feature built on top of a FLOSS stack that was differentiating years ago, can be commodity now once all competitors have caught up. At this point it might be time to contribute said feature to the FLOSS stack to eliminate the in-house maintenance and enable the respective developers to work on new differentiating features. Of course, this might also reduce maintenance effort at competitors, but software engineering is not about having the most maintainers but rather about having the best innovators.

Beyond such still rather technical strategies, there are of course business strategies that go far beyond the value of the actual source code and its development and maintenance effort. Strategically open-sourcing a previously proprietary software can be one way to increase the adoption of one's solution in a competitive environment. With increased adoption will come increased need for integration and adaptation support of said software which can enable a new service-based business model. After all, the company who originally developed the software will likely be the best resource to support in its integration and adaptation.

Another strategic goal can be hurting competitors or preventing monopolies. The most famous example of this is surely giant corporations like IBM or Oracle supporting Linux development through code and monetary contributions as well as community support to prevent Microsoft from monopolizing the operating system market. Thanks to strategic decisions in the 1990s these companies have prevented vendor lock-in for themselves today.

Finally, there are asymmetric business models. We all know Android and its success story. Google realized that the value is not in the operating system itself but rather in the services and advertising running on top. By giving away a very useful piece of software (the operating system), they attracted the handheld companies. By providing a simple way to turn ideas into profit (the Play-Store) they attracted third party developers. This created an ecosystem that was so attractive that it disrupted its entire market and has Google earning billions in advertising and revenue cuts in its Play-Store.

Of course, this is not a comprehensive list of open-source strategies and business models. There are entire books and conference tracks regarding this topic. This section is mainly meant to highlight that open-source can be so much more than using ready-made software and making sure that one complies with all its licenses.

Building ecosystems

"I still meet way too many automotive managers that think that FLOSS is developed by a bunch of idealistic people in their basements."

ALEXANDER MUCH, CHIEF EXPERT AND HEAD OF SOFTWARE SYSTEMS ENGINEERING AT ELEKTROBIT

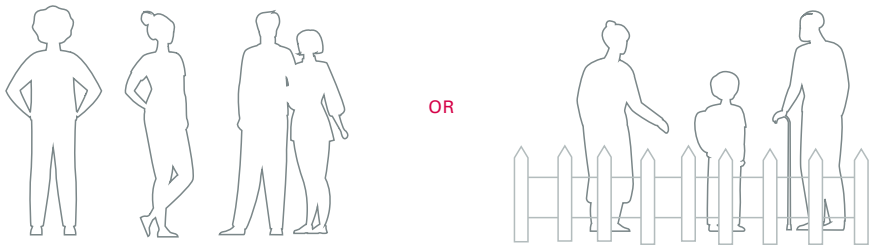


FIGURE 5 Open-source community or walled-garden?

Almost every interviewee agreed that the complexity of modern automotive systems was too big to be managed by a single company. They also agreed that there are giant parts in the automotive software stack that are non-differentiating, such as hardware abstraction layers. Maintaining those in a proprietary way at each company did not make sense to most interviewees.
A collective ecosystem to develop and maintain such software parts and to spread out the effort and cost would be very welcome, but:
Nobody reported that there are plans to start such an ecosystem e.g. by open sourcing their internal solution i.e. making the big initial investment in the hopes of others joining.
Few of the interviewees actually reported that their companies are actively taking part in initiatives like Automotive Grade Linux or ELISA.

"Comparing the number of Apple and Android devices to even the most successful OEMs it is ludicrous to think that any OEM on their own could create a competing ecosystem attracting outside developers. The OEMs need to band together to build a shared platform. And even then, the amount of money and developers that companies like Apple or Google have means that success is not guaranteed."

MIKE MILINKOVICH, EXECUTIVE DIRECTOR OF THE ECLIPSE FOUNDATION

Instead of collaborating on FLOSS, some interviewed companies preferred a walled-garden approach where contractually bound partners would cooperate and benefit from the results, but outsiders would not see the work. They envisioned community-source consortia similar to AUTOSAR.

The AUTOSAR initiative (and its limitations)

AUTOSAR (AUTomotive Open System ARchitecture) is a consortium between automotive manufacturers/OEMs, suppliers, and tool vendors. This partnership has developed a standardized software architecture for automotive ECUs. The standard describes the software architecture, application interfaces, and development methodology. The idea is to enable the development of independent software components that can be used on multiple hardware variants with minimal to no adaptation.

While AUTOSAR focuses on collaboration across the industry, it is not an open community. Membership is contractually regulated and involves financial engagement and licensing agreements. Furthermore, the standardization focuses on the specification of functionality. A FLOSS implementation of said specification is not foreseen and not allowed according to the license agreements.

In a community-source approach, the "COMASSO project"[5] was founded to collaborate on the AUTOSAR implementation across AUTOSAR partners.

Due to the license limitations explained above, the results of this project are only available to the AUTOSAR partners, that participate in the COMASSO project.

These limitations lead to the term walled-garden, that we use throughout this chapter of the survey report.

What this shows is that everybody is afraid of getting taken advantage of and contributing more than the others. This is understandable given the contract driven environment that has governed the automotive industry for decades: "*Who would ever do more than the contract states? – If there is no contract, why would anybody do anything?*" But doing more than is asked is one of the key foundations of FLOSS: "*I have developed this neat solution. I could keep it for myself, but I am making an extra effort, to document, explain, and upload it, so that others can benefit from it while I demonstrate my skills and earn recognition. Maybe one day I can learn and benefit from their skills and neat solutions.*"

5 COMASSO project: https://www.comasso.org/comasso_about

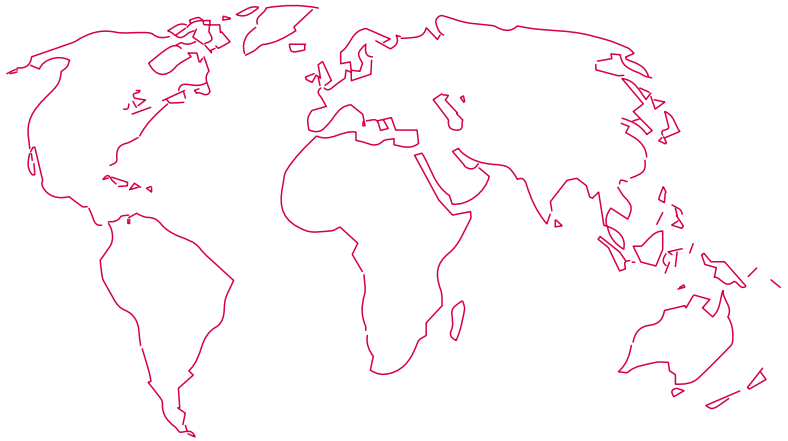
Those two mindsets are in direct contradiction, and it will take time to change minds.

One trend that a few interviewees stated should be highlighted here: Their companies understand the benefit of open-source principles but for whatever reason do not want to engage in FLOSS projects. Instead, they turn to inner-source where they apply similar principles but limit the participants to their internal employees and connecting them across business units.
Going back to the hardware abstraction layer example used above: If more than one business unit uses the same microcontroller in their system, they shall develop and maintain the hardware abstraction layers together as inner-source, thus eliminating redundancy and reducing overall cost. At the same time, the cost would be split between business units, potentially increasing each bottom line.

However: While we heard multiple inner-source success stories, even in such an inner-source environment some managers still have the same internal blockage asking, "*why them spending their budget should benefit another business unit and what they are getting in return*". Even in such a scenario, they would insist on a contractual governance of the activities slowing down progress and basically killing the idea. This is also a problem of the incentive systems that are often in place at large companies. They do not yet facilitate and reward selfless behavior and are not enforcing business-thinking across all levels but rather focus on meeting KPIs. Since these incentive systems are set up by the top management, the FLOSS understanding might not be as mature as those enthusiastic press releases mentioned before would have us believe.
In summary: At this time, it is unlikely that there will be large industry wide ecosystem initiatives very soon because the mindset of many in charge is not compatible enough with open-source principles yet.

Regional differences

All interviewees agreed that open-source affinity is more a question of company culture than of regional cultural differences. However, some did perceive some trends in the regions that they are active in.
In Europe there are many established/"old" companies that dominate the market. Said market is mostly divided up between players and few are interested in challenging or changing the status quo. The next generation of existing systems is developed with existing partners and iterated solutions. Why would people that can sell their software solutions over and over again contribute them to a FLOSS project? Why would a customer who has found a reliable supplier with a reasonable price try to integrate some FLOSS project by themselves for the next generation?
In contrast: In China there are a lot of new/"young" companies that are interested in short time to market and quick innovation to establish themselves.
If FLOSS is a tool to achieve that, they use it and actively contribute. There is no decades old supplier network with strategic relationships that needs to be maintained.
The interviewees saw North America somewhere in between those two extremes. On the one hand there is a very traditional and established automotive industry. On the other hand, there are a lot of young innovative start-ups (especially on the west coast) pushing into said automotive industry using among others FLOSS as an acceleration tool.



Regulation and compliance

The automotive industry has its own (in part unique) set of boundary conditions which govern and influence many aspects of product development and distribution. For this survey, we looked at three aspects in more detail and how they interact when it comes to the use and management of FLOSS.
Of course, there are dozens more de-facto standards such as e.g. Automotive SPICE and regulations for this industry and many of them will have interactions with FLOSS, but during our initial research, the following three topics seemed to lead to the most heated and controversial discussions.

- Legal and license compliance (ISO/IEC 5230 – OpenChain)
- Functional Safety (ISO 26262)
- Cybersecurity (ISO/SAE 21434)

Hence, we wanted to focus on these and give the interviewees space to voice their opinions on those.

Legal and license compliance (ISO/IEC 5230 – OpenChain)

„The OpenChain standard will become an obligatory standard in the automotive industry. One can save 20 pages of contract documents with one simple requirement to prove OpenChain compliance."

HANS MALTE KERN, HEAD OF CENTER OF COMPETENCE OPEN SOURCE AND OPEN SOURCE ENGINEERING EXPERT AT BOSCH

ISO/IEC 5230 – OpenChain

The OpenChain project was started in 2016 and aims "*to establish requirements to achieve effective management of open-source for software supply chain participants*"[6].
The idea is to create trust throughout the supply chain, by all supply chain participants conforming to a minimal set of requirements that enable reliable open-source management including license handling.
The OpenChain project does not only provide said requirements, but also training materials, policy templates, self-certification checklists, and more. All with the goal of enabling easy and wide-spread adoption of a single license compliance standard reducing overhead for all involved parties.
In 2020 the requirements were released as a joint standard by the ISO and IEC under ISO/IEC 5230. The requirements of said standard(s) are functionally identical to the OpenChain 2.1 specification which is still openly available on the OpenChain website.

6 The OpenChain project mission: https://www.openchainproject.org/about

Until recently every vehicle manufacturer worded their own requirements on handling FLOSS and the corresponding license obligations. From there, these requirements would propagate through the complete supply chain from Tier 1 to Tier 2 … to Tier n.
All these suppliers would need to maintain their license information in multiple systems and edited according to multiple sets of requirements to prove compliance, which created an enormous amount of overhead.
As one can imagine, the interviewees of such suppliers were delighted by the concept of OpenChain and that it turned into an ISO/IEC standard. They are hoping for a fast industry wide adoption of the standard and some have already had their customers asking for OpenChain compliance and certification. In fact, many interviewees are hoping for more international standardization surrounding FLOSS e.g. regarding the license information exchange format SPDX[7]. Optimistic interviewees even envisioned a future where the whole topic of license compliance could be implemented as a block-chain of micro-contracts across the entire supplier network.
The OEMs/manufacturers also saw the OpenChain standard as a benefit because their supplier selection process will become easier once the industry has widely embraced the standard. During said process they will no longer need to spend a lot of time evaluating the supplier maturity regarding FLOSS using their own questionnaires. They can instead rely on the supplier's OpenChain certification. If they do not trust the self-certification, they can insist on an independent third-party certification according to OpenChain's model.
All that sounds pretty promising, right? Unfortunately, there is a but:
Some of the interviewees (both at manufacturers and suppliers) had not even heard of OpenChain or ISO/IEC 5230 at all. Therefore, an industry wide adoption of the standard will take a significant amount of time.

7 SPDX project: https://spdx.dev/

ISO 26262

ISO 26262:2018 is the currently valid version of a series of standards describing objectives and requirements regarding functional safety for series production road vehicles (excluding mopeds).
It contains a functional safety lifecycle that shall be applied to achieve functional safety (as well as compliance to the standard):

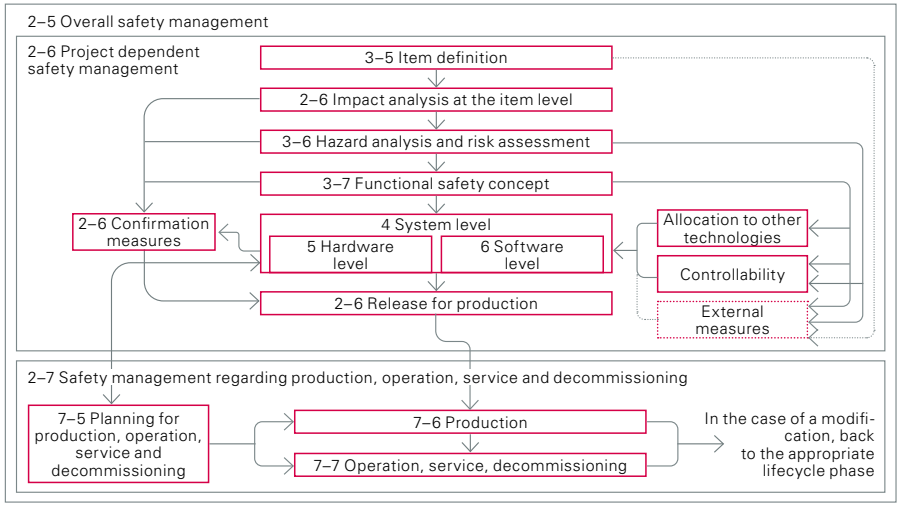


FIGURE 6 Functional Safety lifecycle according to ISO 26262:2018

FIGURE 6 illustrates that the safety lifecycle affects all parts of product (and hence software) development and even product distribution. For each aspect, the standard defines objectives to be achieved and extensive requirements on how to achieve those objectives. These requirements are not limited to what shall be done, but also touch upon how it shall be done e.g. through large method tables containing recommendations. Of course, there are also requirements, how this systematic development shall be documented and how to ensure consistency and correctness for every step along the way.

When it comes to safety relevant embedded systems (according to ISO 26262) many companies are quick to rule out the use of any FLOSS. Many engineers and managers have a story to tell where the supposed shortcut of using an open-source component led to astronomical effort in qualifying said component so it could be used in the safety context. For every new version of the FLOSS component, they would need to evaluate the changes and decide whether to integrate it. If it was integrated that astronomical effort had to be repeated. These are the horror stories that make functional safety seem like an immovable roadblock when it comes to using open-source in automotive where many systems happen to be safety relevant. However, that is only part of the truth.
Of course, many older and grass-roots FLOSS projects do not follow a development lifecycle that is inherently compatible with the functional safety lifecycle defined in ISO 26262, e.g. including the necessary upfront investment that goes with it. And of course, some of these projects are so big and complex that qualifying them and their code base according to ISO 26262 would be an insurmountable effort for any one company.
But: There are a few parameters in the statements above that can be modified to enable FLOSS usage even in safety relevant systems.

Old/grass-roots projects don't follow the functional safety lifecycle:

Set up a new project to develop the intended functionality and plan and steer it according to the ISO 26262 or any other suitable functional safety standard. There is nothing in those that prohibits shared development. Examples of such specifically functional safety-oriented projects include Zephyr[8] and Apex.OS[9].

8 Targeting SIL 3 according to IEC 61508 and ASIL D according to ISO 26262: https://zephyrproject.org/zephyr-project-rtos-first-functional-safety-certification-submission-for-an-open-source-real-time-operating-system/
9 Fork of ROS 2 which was successfully certified as a SEooC up to ASIL D: https://www.next-mobility.de/betriebssystem-fuer-autonomes-fahren-erhaelt-tuev-zertifizierung-a-1015627/

> Naturally, most grass-roots FLOSS projects would not follow such an elaborate approach especially if a software component is not originally intended for use in a safety relevant environment. This can of course lead to massive challenges when automotive companies try to integrate such components into their safety relevant application.

Too big and complex to qualify on your own: Don't do it alone. Since it is a big and complex project it obviously solves a problem that many people or companies have. Therefore, it is very likely that others in the automotive industry have the same interest in qualifying this software. Automotive Grade Linux[10] and ELISA[11] are examples of many industry players coming together to tackle the "beast known as the Linux Kernel" to enable a more widespread use throughout the vehicle. Similar initiatives could be set up for many other FLOSS projects.
Despite this, some interviewees have reported that even when they have successfully qualified a FLOSS component e.g. by developing a complete test suite to prove that it fulfils their safety requirements, they are still not allowed to contribute this test suite back to the project. If they did, the project would likely pick the test suite up as a great addition and maintain it for future releases, making the integration of future versions that much easier. Instead, the company has to spend enormous effort to evaluate each update, perform an impact analysis and if needed re-perform the qualification.
This is one example, where companies confuse effort with value and end up hurting themselves by not sharing the effort with others.
To summarize: Functional Safety is a hurdle for FLOSS, but not the roadblock it is often perceived as. With the right approach FLOSS can be used in safety relevant systems and interviewees agreed that recent success stories of Zephyr and Apex.OS will likely lead to accelerating this understanding.
One hope that some interviewees worded is that the future 3rd edition of ISO 26262 will explicitly address FLOSS and lay out best practices regarding its use and integration. The foundations for such a potential future update are currently evaluated in ISO/AWI PAS 8926.

10 Automotive Grade Linux: https://www.automotivelinux.org/
11 ELISA announcement: https://www.linuxfoundation.org/press-release/the-linux-foundation-launches-elisa-project-enabling-linux-in-safety-critical-systems/

Cybersecurity (ISO/SAE 21434)

„Cybersecurity becomes a lot more manageable through using FLOSS. By the way: It can also improve your relationship with the respective government agencies for type approval."

FALK LINDNER, PHD
LEAD ARCHITECT INDUSTRIAL CYBER SECURITY AT AIRBUS

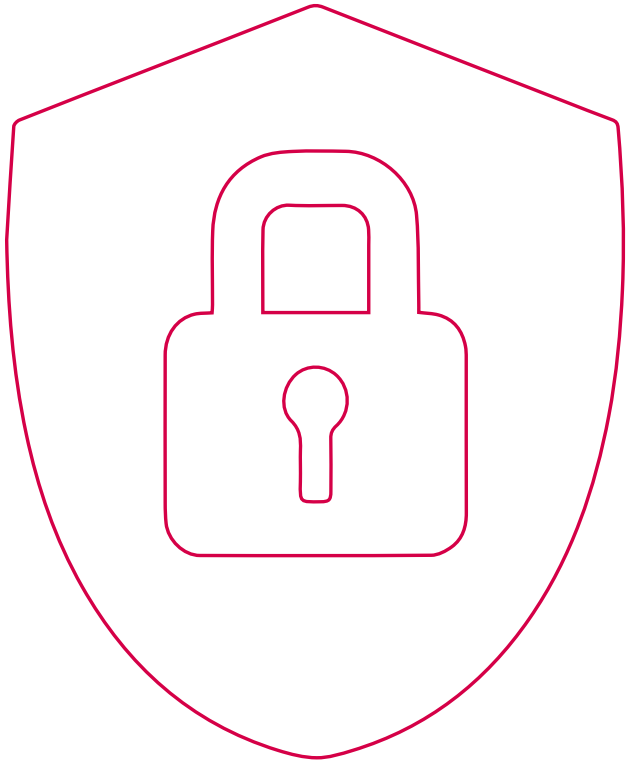
With more and more connectivity implemented in modern vehicles, there are more and more potential entry points for malicious attackers. As laid out above, especially in the connectivity space, there is heavy use of FLOSS reference implementations of communication protocols. How does this affect cybersecurity?
Most interviewees agreed that FLOSS is likely more secure than most proprietary software. The multiple independent eyes reviewing changes are a good measure to eliminate back doors. Also, in case any vulnerabilities are discovered, the turn-around time of the community is often very short, decreasing the attack window.

ISO/SAE 21434

ISO/SAE 21434:2021 was published in August of 2021 and is the joint ISO and SAE cybersecurity standard for automotive cybersecurity engineering. Similar to ISO 26262, it describes a systematic approach to ensure the cybersecurity of automotive systems. It defines requirements towards

- cybersecurity policies and processes
- management of cybersecurity risks
- fostering a cybersecurity culture

However, this transparency and high update speed was also considered a downside by some interviewees. The patch for a vulnerability is often basically a manual to build an exploit for unpatched software. Since there are no widespread FOTA (firmware over the air) update mechanisms yet, software updates for vehicles have a long lead time and are very expensive. From the cybersecurity perspective, within the regulations of the UNECE R155[12], this means, every vulnerability in a FLOSS component can potentially cost a lot of money and lead to a high risk in a rushed software update or in the worst case: negative consequences as a vulnerability remains unpatched for too long and gets exploited.
A proprietary software can be a bit more forgiving in this regard as a potential vulnerability might not be as well known, limiting the number of potential attackers, thus reducing the risk and allowing for a bit of a slower reaction time. Obviously, this issue should be somewhat mitigated with over the air updates becoming more commonplace.
It will be interesting to observe the impact of the recently released ISO/SAE 21434. It foresees a comparable systematic approach to (software) development as ISO 26262 when it comes to requirement specification, design, implementation, verification, and validation. As laid out above in chapter FUNCTIONAL SAFETY (ISO 26262), many FLOSS projects do not follow such an approach. Since much more software is cybersecurity relevant than is safety relevant, the issues laid out above for functional safety could be even more significant for cybersecurity. To a much greater extent, cybersecurity addresses the post-development phase until the end of service. In the case of FLOSS deployment, the measures required for this must be safeguarded in a dedicated management system. Similar mitigation measures as laid out for functional safety should offer potential relief i.e. specifically target cybersecurity during FLOSS-based development and maintenance, and servicing phases. And most importantly: don't do it alone.



12 UNECE R155 - International regulation "concerning the approval of vehicles with regards to cyber security and cyber security management system": https://unece.org/sites/default/files/2021-03/R155e.pdf

The SCALARE Open-Source Maturity Model

The maturity levels of FIGURE 7 can be briefly summarized as:

- LEVEL 1: Open-Source is an "under the radar" activity. Management position is often unclear, but developers use it based on own belief that development will be faster and cheaper this way.
- LEVEL 2: Partly understanding the potential of Open-Source and risks with not being compliant, the company wants a controlled repetitive framework. Focus is on intake with limited contributions.
- LEVEL 3: Collaboration with partners and competitors through Open-Source is understood. Focus is on efficient and innovative development, but still mainly for the engineering department.
- LEVEL 4: Understanding of Open-Source now spreads to encompass other domains like sales & business development. Focus is on business opportunities that can be harvested with the use of Open-Source.
- LEVEL 5: The company has developed a full-fledged Open-Source culture, with strategic support from management, to the extent that the company can disrupt markets by changing the market logic.

The SCALARE project[13] documented the following OS maturity model:

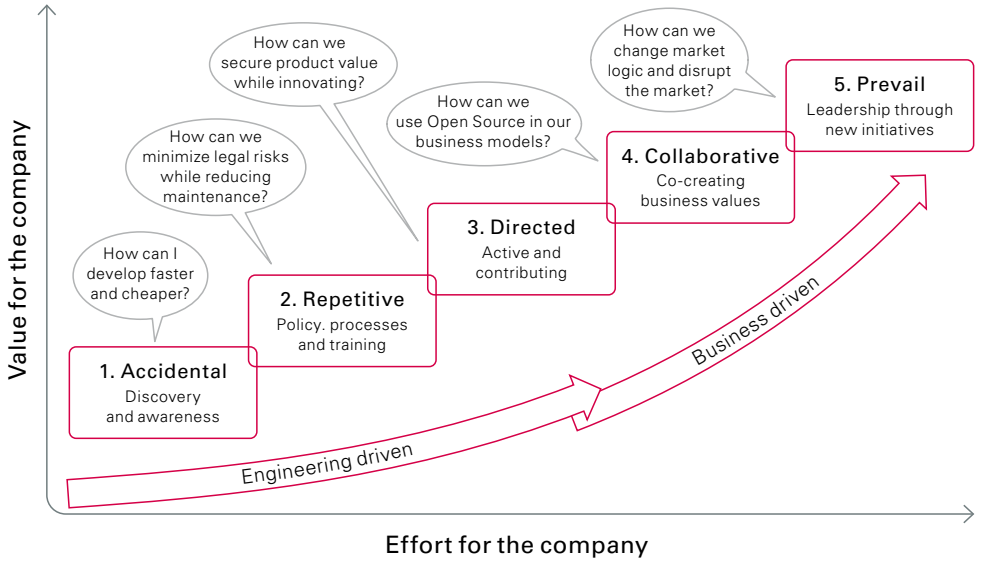


FIGURE 7 SCALARE Open-Source Maturity Model

We asked our interviewees where they place their own company in the SCALARE OS maturity model and where they put the automotive industry as a whole. Most saw themselves at about level 2. Many of the larger companies stated that their level of maturity is not uniform across their various business units. While some might actually be closer to level 3 or even higher (especially non-automotive centric business units), some are still struggling to avoid accidental use. For those units, systematic checks to ensure license compliance are supposedly in place, but the fact that those checks actually find accidental use shows that the systematic approach has not reached all developers, project leaders, and managers yet.

Analysing the responses for this and all other topics, we estimate that automotive companies are on average at level 2 or just about to reach it.

Interestingly, many interviewees estimated the over all automotive industry at a higher maturity level than their own company, fearing that they "had been a bit slow on the topic". Our hypothesis/assumption/interpretation is, that those enthusiastic press releases we described in our MOTIVATION section also caught out some of our interviewees.

13 "SCALARE – Scaling a Software Business": https://itea3.org/project/scalare.html

Talent pool and company culture

All our interviewees experience and struggle with the massive shortage of qualified software developers and engineers. Besides the obvious relief of needing less developers when using FLOSS, the interviewees also saw a generous open-source policy as a way to attract and retain talent, because:

- Being a top expert in a popular FLOSS project increases individual market value much more than being a top expert for one proprietary solution of one automotive company.
- Regularly contributing to various FLOSS projects lets developers build an online CV where other companies can see exactly how well they work and communicate, further compounding this effect.
- Additionally, such developers are passionate about open-source. Possibly they have worked on FLOSS projects during their studies as well. And letting people work on what they are passionate for is a great way to retain highly productive talent.

The downside for the company is of course, that their valuable talent is very visible to competitors. But: Out of all interviewed companies only one had already experienced this downside when headhunters poached some of their employees based on their FLOSS contributions.

Despite all interviewees agreeing that the positives outweigh the risk on this topic:

- Few companies make FLOSS subject of their job listings, highlighting their engagement to potential talent.
- None of the companies' HR departments actively search talent based on their FLOSS contributions, e.g. by scanning GitHub or similar platforms, as companies in other industries do.
- Many company guidelines make it difficult to employ talent once it has been found if they are not willing to move to the company site. But not everybody wants to live in Detroit or Stuttgart. Completely remote employment is still very difficult to accept for many companies, but not impossible as few examples show.

Enablers and hinderers

As mentioned in previous chapters, the mindset of many in the industry is a big hinderer for the use of and contribution to FLOSS.
One aspect stood out in the interviews, which shows that software is still treated as a cost factor and not as a strategic investment and asset: The origin of software plays an important role in the minds of many. If a piece of software was developed in-house and cost in-house budget, then it is valuable and absolutely can't be open-sourced or contributed. If the software was developed elsewhere and for free, then surely it can't be good and mustn't be used without extensive in-house qualification ("not made here"-syndrome). Neither assumption is true nor plays well with FLOSS. It is worth noting that a similar mindset dominated the handheld industry in the early 2000s. FLOSS did not take long to change those minds or punish the minds that did not change.

> "After building on top of a software stack for years, it must be re-evaluated regularly: Are there parts that used to be differentiating, but which are now a commodity and therefore should be open-sourced into a community to no longer require that much effort to maintain?"
>
> JONAS ÖBERG, OPEN SOURCE OFFICER AT SCANIA – VOLKSWAGEN GROUP

Another hindering aspect is the effort that comes with FLOSS even if a company is open minded. If you want to make your product open-source to further its development and market reach, you have to attract people. You can't just make your existing code open-source and hope that somebody will pick it up. We heard an example from one company that made their code open-source and had to write over 1000 pages of documentation just to enable outsiders to understand enough to be able to contribute. Previously that documentation did not exist because company internal it was "hive knowledge". You must anticipate such extra work when trying to leverage the community. And of course: Even after having spent the budget on the documentation, there is no guarantee there will be a vibrant community to make up for it. Such an outlook scares many managers.
Another big hinderer, which comes from the aforementioned decades of having every aspect of product development and distribution governed by contracts, is the focus on warranty and liability. The legal departments of our interviewees would regularly come back with the same questions:

- If we use FLOSS and have a liability case, there is no one we can hold accountable. Who should we sue?
- What if someone uses FLOSS we developed and has a liability case? What if they sue us?

While those are valid questions, other industries have found their answers and the interviewees felt like they are more rooted in lack of understanding than actual concerns.

On the other hand, legal concerns might also be a massive enabler for FLOSS:
As laid out in the section on BUILDING ECOSYSTEMS, many companies agree that there needs to be more collaboration between them to deal with the massive complexity of modern automotive systems. Yet, such cooperation (especially amongst competitors) will be under intense scrutiny of antitrust regulations and agencies if governed by commercial contracts. Performing a similar cooperation in the form of an open-source project under the governance of a foundation with a neutral chairperson could relieve such legal woes. According to our interviewees, the fewest legal departments have understood that though and could convince the respective managers of the benefits.

Conclusion

Open-source in automotive is coming but it will take some time.
Those enthusiastic press releases we all read regularly lead us to believe that at least part of the top management has understood the importance and strategic value of FLOSS. The engineers tell us that they want to use FLOSS and that technical hurdles like functional safety can be overcome. On the other hand, there are significant mindset problems and counterproductive incentive structures in many companies.
Because FLOSS seems too risky, many therefore turn to proprietary alliances with contractual partners, but in those they must deal with antitrust regulations and quitting partners. Both being major business risks. Many companies have yet to understand that open-source projects governed by a foundation can relieve such pains and enable close collaboration even among fierce competitors regarding non-differentiating parts. For that to work though, companies need to learn to trust each other and the community. Over one hundred years of having every business aspect governed by detailed contracts won't be overcome instantly. Everybody is afraid of paying more than the others and getting taken advantage of. In that sense, current small-scale engagements seem more like trust building exercises for the future than a strategic engagement. In fact, open-source in automotive is still very much engineering driven and not business strategy driven.
Accordingly, at this time, the automotive industry is not driving many large FLOSS projects but rather profiting of the efforts of others in an opportunistic fashion. Since all interviewees agreed that this is not a sustainable model, we conclude that large and successful open-source initiatives in the automotive industry are only a matter of time, the interesting question being: how much time?
The handheld industry was transformed from entirely proprietary to open-source driven in roughly 1.5–2 years. Given that the automotive industry has even more players with even more minds that need changing, more restrictive boundary conditions to overcome, and longer product lifecycles, we estimate that the transformation will of course take longer than that. A span of about five years seems most realistic until major parts of automotive features (besides infotainment systems) are developed based on FLOSS components and projects.

About the authors

Fabian Mueller is a Senior Consultant at Kugler Maag Cie. He holds a degree of Dipl.-Ing. in Cybernetic Engineering from the University of Stuttgart, Germany.
He started his career as a software developer for automotive functional safety systems at Bosch Engineering GmbH. This is where he had a brief first brush with FLOSS.
With each project he built his expertise at a higher level of abstraction and finally switched to the process side and Kugler Maag Cie. He now consults (automotive) companies in improving processes to reach their functional safety, Automotive SPICE, and business goals.
The topic of open-source and its communities always stuck with him though.
Prior to this survey, he evaluated the supposed automotive state of practice according to the press landscape. On the survey itself he acted as the survey management in designing the questionnaire, coordinating and conducting the expert interviews, as well as summarizing the results in this report.

Horst Hientz is a Principal Consultant, founder and partner of Kugler Maag Cie. He holds an M.Sc. in Computer Science, Dipl.-Inform., from the University of Kaiserslautern, Germany. He worked in Software Engineering since 1984 as programmer, consultant, manager, and entrepreneur. As CEO of Q-Labs Germany he has supported since mid- 1990's the automotive industry, multi-national tiers and OEMs world-wide, maturing their complex in-car software-determined systems methodically. 30 years later, his focus broadened with the automotive industry's need to emphasize software-determined business, too.
He lives in the heart of Europe, lovely Luxembourg.
On this survey, he not only contributed his industry overview and contacts to help in creating the survey questionnaire, but also acted as the lead interviewer and thus ensured comprehensive coverage of all topics from conceptualization until the summary in this report.

Contact

Public Relations
Dominik Strube
Kugler Maag Cie Study Program

Imprint

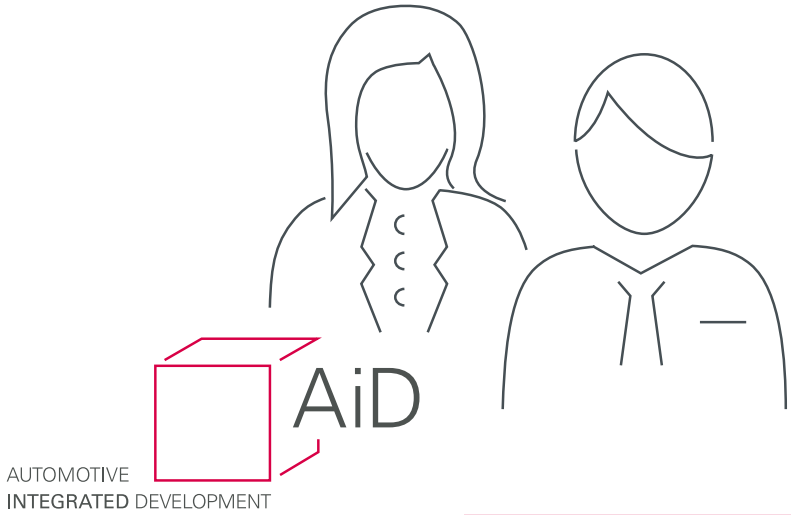
KUGLER MAAG CIE GmbH
Leibnizstr. 11
70806 Kornwestheim
Germany

kuglermaag.com

ISBN-13 978-3-945547-43-4

12|2021

Towards a **fully integrated** automotive E/E development.



For more about
Automotive integrated Development,
watch this video at
kuglermaag.com.

- Automotive SPICE®
- Functional Safety
- Automotive Security
- Agile Automotive

KUGLER MAAG CIE