Elektrobit

automotive.elektrobit.com

Lean Software Development for the Automotive Sector

A practical combination

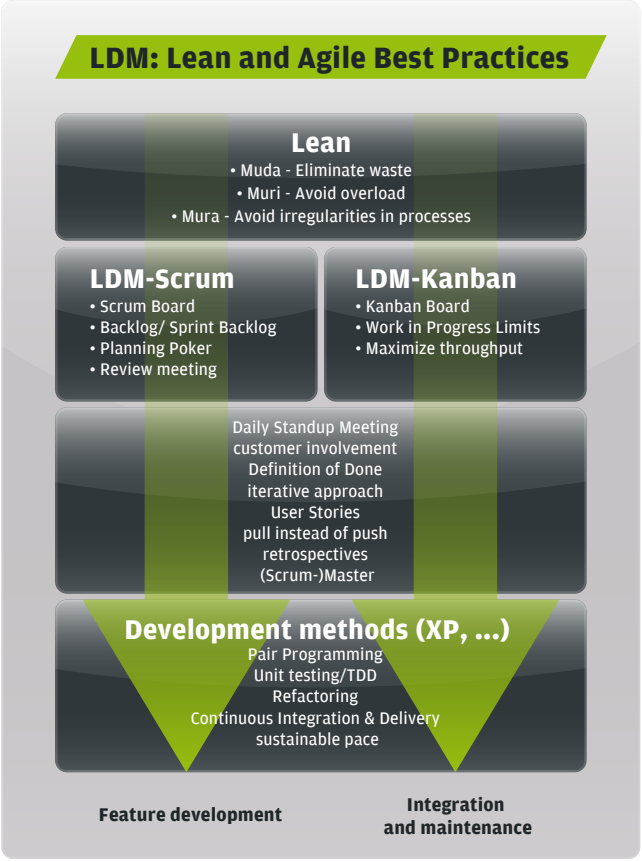
Agile and lean principles are firmly established in today's software development industry. However, despite having been one of the lean manufacturing pioneers, the automotive sector has been slow to apply the same principles to software development, where the delivery of defined work packages to a specific deadline is still the norm. As a result, it is becoming increasingly difficult to satisfy the automobile manufacturers' current software requirements. The Lean Development Model overcomes this challenge by combining various lean development methods.

The traditional areas of focus in the automotive sector such as design, power train, chassis and safety have been joined by a new element of differentiation: Customers nowadays expect the car's infotainment system to offer them the same features as their smartphones and other mobile devices. This is somewhat problematic because these technologies have a much shorter development cycle than the car development prototyping phases.
If a car manufacturer schedules 2 to 3 years for developing a car the hardware may also be defined this far in advance. This is different for the software requirements which will change drastically till the SOP (Start of Production), which is irrevocably fixed.

End customers also expect the car they purchase to have new functions and support the latest technologies throughout the car's lifecycle. So software requirements don't just change during the planning phase, but throughout the entire product lifecycle. At the same time, it is customary to offer different ranges of functions and country specific versions based on the same system. Sequential development processes such as the waterfall approach and the V-model are somewhat limited because they make it impossible to respond quickly and comprehensively enough to changes.

Continuous car software upgrades of the headunit also means that hardware requirements will be higher at SOP than they were at the outset of the design process. It is far more practical to develop software that is reduced to the specific requirements rather than a large generic framework to prevent an excessive burden on computing power. This makes code refactoring necessary, e.g. removing the code for obsolete requirements. Only software that is developed according to KISS (Keep It Simple and Stupid) and Clean Code principles will have long-term maintainability and be less susceptible to new errors. To avoid regression, it is essential to use automated test frameworks.

All these are aspects of agile software development which make it interesting for the automotive industry. The Lean Development Model is a combination of agile and lean principles that is specifically tailored to automotive software development. From a project and team management perspective, it mainly consists of Scrum and Kanban aspects. Additional methods from extreme programming (XP) are applied to support the software development process **(see Fig. 1).** So how did the Lean Development Model come about?



The Lean Development Model combines agile methods with lean software development principles **(Fig. 1)**

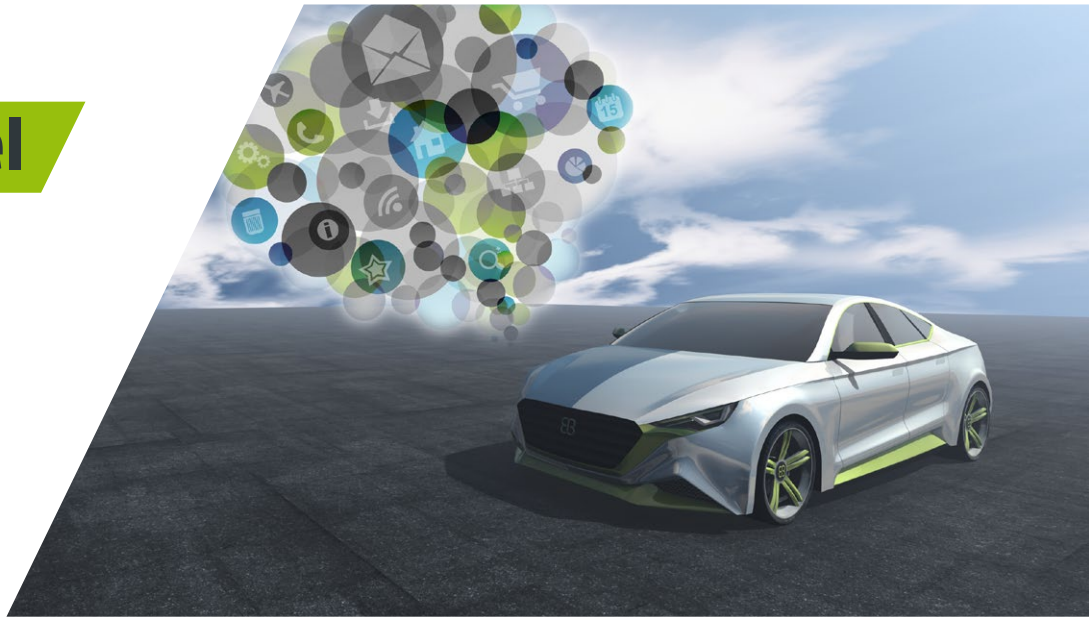
From Scrum and Kanban to the Lean Development Model

In 2008 Elektrobit Automotive introduced Scrum to organize the development of features for an infotainment system. This incremental development approach with fixed-length sprints delivered positive results because it made the timely and regular involvement of the client - a US automobile manufacturer - in the development process possible.

When the feature development was nearly finished, the focus shifted to the stabilization and optimization of the integrated system. However, new errors and the resulting change in priorities, plus the fact that the experts had different workloads, made reliable planning at the start of each sprint difficult. As a result of this, the team switched over to the Kanban software development method, which was far more suitable for stabilization and optimization work.

In a later phase additional features needed to be implemented. Instead of switching back from established Kanban to Scrum, the team decided to combine both methods. Task tickets were combined into stories representing one feature of the infotainment system. This granularity makes it possible to prioritize stories in the backlog in consultation with the customer. The team then work their way through the tickets until the story can be delivered to the customer.

This method was initially only used locally, because the team was predominantly based in Germany and the development process could be coordinated on a simple board. As the team expanded and became less localized, additional tools such as JIRA were used to permit the cross-site coordination of the entire project.



The reference project

A team of around 75 developers and testers uses LDM in a software development and integration project. They work at four sites in the USA, Germany and China and are responsible for the integration of supplied components and the development of features. Team members with more than 15 different nationalities were chosen because they are able to implement the requirements of the various international markets more efficiently. The use of native speakers is extremely valuable in the development and testing of text-to-speech as well as voice recognition functions because bugs can be prevented or at least identified faster.

Since this is a project with a duration of more than seven years and international assignment, many hardware variants have to be supported. This poses special challenges in the area of configuration and build management.

The Lean Development Model primarily follows seven principles:

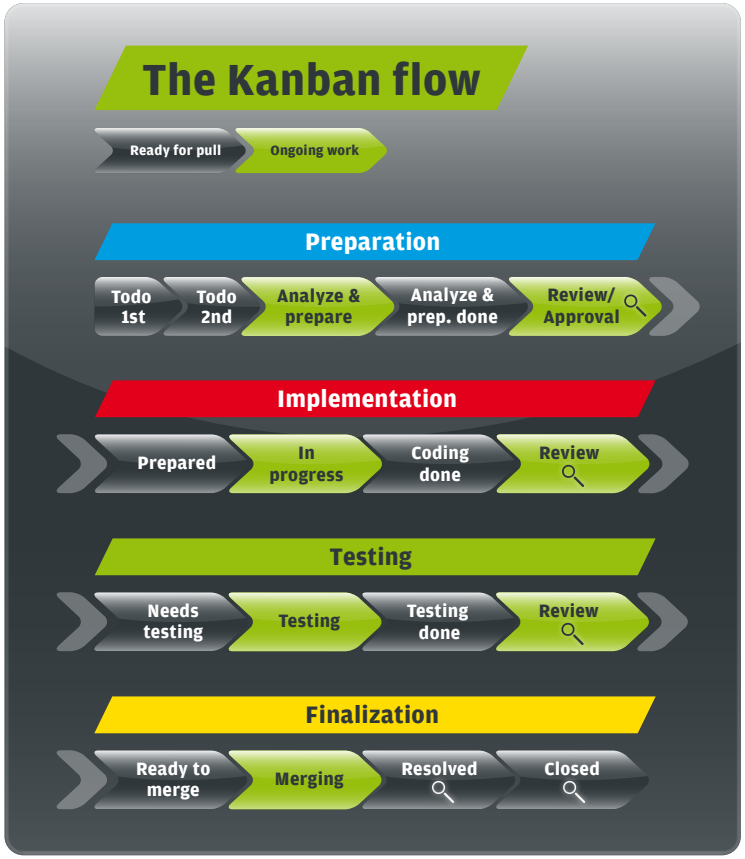
1. Deliver as fast as possible

In the Lean Development Model, the agile method of prioritizing a backlog replaces the traditional detailed planning of all resources. The tasks are not assigned in advance by the project manager. Instead the team pulls them into the next workflow step (pull rather than push) **(Fig. 2).** The focus is clearly on maximizing throughput.

The limitation of the number of work packages in each workflow step prevents the generation of more interim products in earlier steps than can be processed in later steps. Tasks that have commenced, i.e. tickets, have to be completed before new ones can be started. Every member of the team is responsible for ensuring that there is no stagnation.

For cycle time, which has to be optimized, breaks or pauses are non-deductible. For example, if a ticket is blocked because it is necessary to wait for customer feedback, the clock continues ticking. If a developer changes the code base, he is responsible for the ticket until the change has been validated via the continuous integration tool chain. No further changes are allowed to be implemented until a potential bug has been eliminated.

This procedure enables daily deliveries to the customer at an early stage of the development process. A selected version of the software is validated by a smoke test and delivered with the test result. As a result the software status is always transparent and tangible for the customer. This enables the customer to provide feedback for the continuous improvement of specifications and implementation.



The team independently pulls the tasks into the next workflow step **(Fig. 2)**

2. Build integrity in

The Definition of Done (DoD) is crucial for quality assurance and is defined by the team at the outset of the development process. The team decides which steps are necessary to implement a user story, for example when changes have to be integrated. These changes have to be implemented following all defined steps as well as being documented and verified by reviews. When a change is handed over to the test team, any new or modified code already had to pass numerous unit tests designed by the developers. This is done to ensure that the change has no errors at the time of hand over.

This also makes the timely identification of issues possible. The higher the degree of automation, the faster bugs and regressions can be found. The tests are scaled according to project phase and type of delivery:

- daily smoke tests for the daily delivery,
- focussed testing for engineering drops,
- integration tests for key milestone deliveries,
- long-term validation tests for the SOP and
- continuous unit tests as an aspect of continuous integration.

The automation is controlled by the Jenkins continuous integration tool. Integrated test frameworks allow the simulation of interactions and the proper presentation of results for the testers and developers. The developers get direct feedback on static code analysis and unit tests, as well as test coverage. Sonar evaluations (SonarQube) are a valuable source of support to architects, project managers and test managers. Metrics for Lint warnings, duplication and complexity, and the SIG Maintenance Model, (http://www.sig.eu/en/Research/690.html) make it easier to keep an overview.

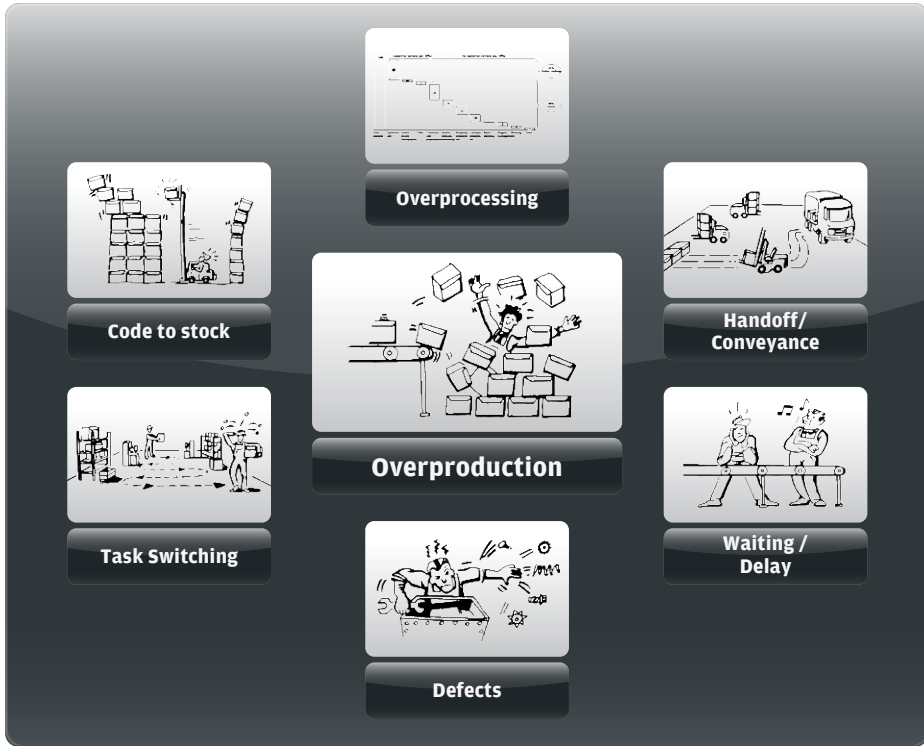
The course for successful testing is set right at the beginning. Design improvements and architecture stability are achieved by compliance with Design For Testability (DFT) and Test Driven Development (TDD) principles with refactoring.

A defined sequence of test steps guarantees the quality of new and modified code. **(Fig. 3)**

3. Decide as late as possible

In all long-term software projects, the initial plan is soon rendered obsolete by changed requirements and frameworks. It therefore has to be updated throughout the entire project. At the beginning of each iteration it is necessary to agree on all requirements in terms of content so that the story teams can plan it in detail. Requirements for future iterations are kept diffuse to ensure that the team can respond flexibly to changes. Assumptions made at the start of the project often have to be revised. Definitions for architecture and design should only be made if they are necessary for the next steps. The objective is to avoid expensive dead ends and keep options open so that the system can be adapted when precise information becomes available. Optimizations are implemented as late as possible. The software architecture and test coverage have to support regular refactoring.

Not all suppliers work iteratively. The integration of their updated components has to be followed by a stabilization phase. In this phase, all tickets are prioritized not in priorization levels but in a distinct order – the ticket backlog. The planned target for the next iteration, and therefore the number of tickets to be processed, is calculated on the basis of metrics such as cycle time and new ticket inflow. A high priority ticket may necessitate the adaptation of the content of an iteration.



The seven kinds of waste in software development **(Fig. 4)**

4. Eliminate waste

To optimally utilize team resources, the team always concentrates on the feature set which is required at any one time. Work-in-progress (WiP) limits for every team member and each workflow step prevent too many things being worked on simultaneously. This reduces change times, hand-over losses and the error rate. Semi-finished functionality, which is considered as waste at the time of delivery, can also be avoided in this way.

Errors can never be completely ruled out. Therefore the work process has to be geared to the earliest possible identification of bugs to minimize the development of erroneous functions and waste of time. It is equally important to discard features that are no longer required. Over engineering and rushed implementation based on assumptions shall be avoided. Besides reducing the development time, this also reduces the efforts for maintenance and testing.

5. Empower the team

Team work is based on self-determination, motivation and commitment to a common objective. The focus is on the individual and his competencies, not the human resource. The team continuously adapts the development process more or less independently, e.g. by holding retrospectives.

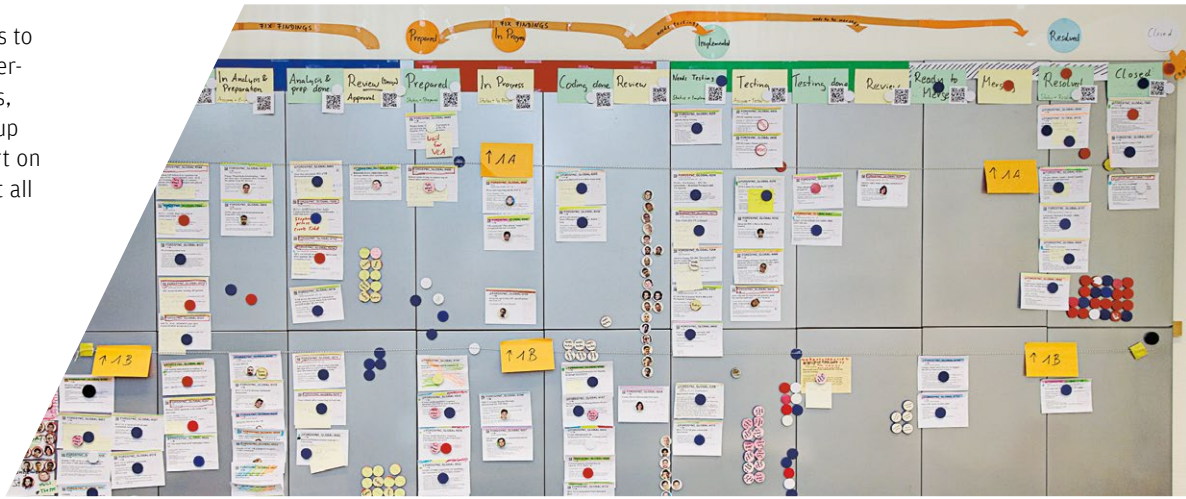
The team communication is supported by a visualization of the workflow on a magnetic wall board **(Fig. 5).**

When the team members work at different locations, they have local boards that are synchronized with an electronic overview. The project manager puts the high priority work packages in the to-do column. The next free member of a story team then 'pulls' the ticket. He or she tags it with one of his two magnets (personal WiP limit) on the board. If both personal magnets are already in use, the open tickets have to be finished first or passed on to a colleague **(Fig. 6).**

The physical board helps the local teams to keep an overview. If there are any unresolved issues or findings, the team member marks the ticket with a 'talk about me'-magnet. The team addresses the tagged tickets at the daily stand-up meeting. More detailed discussions may follow within smaller groups.

When features are being developed the story teams that perform all the tasks in a user story are dynamically formed. The stories, which are an aspect of Scrum, also help the teams to keep an overview in Kanban. Teams are interdisciplinary and include software architects, testers and developers. At the daily stand-up meeting team members take turns to report on the story's current status. This ensures that all team members are equally involved.

A 10 m² magnetic board supports team communication **(Fig. 5).**



Personalized magnets prevent team members from taking on too much work **(Fig. 6).**

6. Amplify learning

Kaizen (Japanese for "change for the best"), a concept of continuous improvement, is a central aspect of Lean Development. Problems are addressed at the daily stand-up meetings and then discussed and resolved in smaller groups. If necessary, the documented rules are adapted to prevent problems reoccurring. The team retrospectives are used to review the course of the project and identify best practices as well as improvement measures. This ensures a stable project progress while applying any necessary changes incrementally. Work processes, such as the “Definition of Done”, are always documented and prominently displayed for all team members.

Pair programming and reviews, plus an in-house Wiki, enable the regular exchange of project knowledge and technical know-how. Furthermore Elektrobit has an in-house academy to support knowledge transfer where anyone can be a participant as well as a trainer.

7. See the whole

In this project, Elektrobit is responsible for the entire system as integrator. However, even in projects where responsibility is only taken partially, it is important to keep an overview at all times. The interaction of all components is what creates the impression the end customer will have in the car. It is therefore important to test the software not only in a lab environment on development equipment but also in the car, both on a test track and in everyday driving situations. This not only identifies bugs but also shows possible usability optimizations for the carmaker. Usability is one of the most important customer satisfaction criteria [http://autos.jdpower.com/ratings/].

Conclusion

Over the course of the project it became evident that none of the lean methods alone adequately addressed all aspects necessary for a successful development of embedded software. However, this custom combination of suitable methods and their continuous adaptation to project requirements enabled the team to increase transparency during the development process and identify problems earlier.

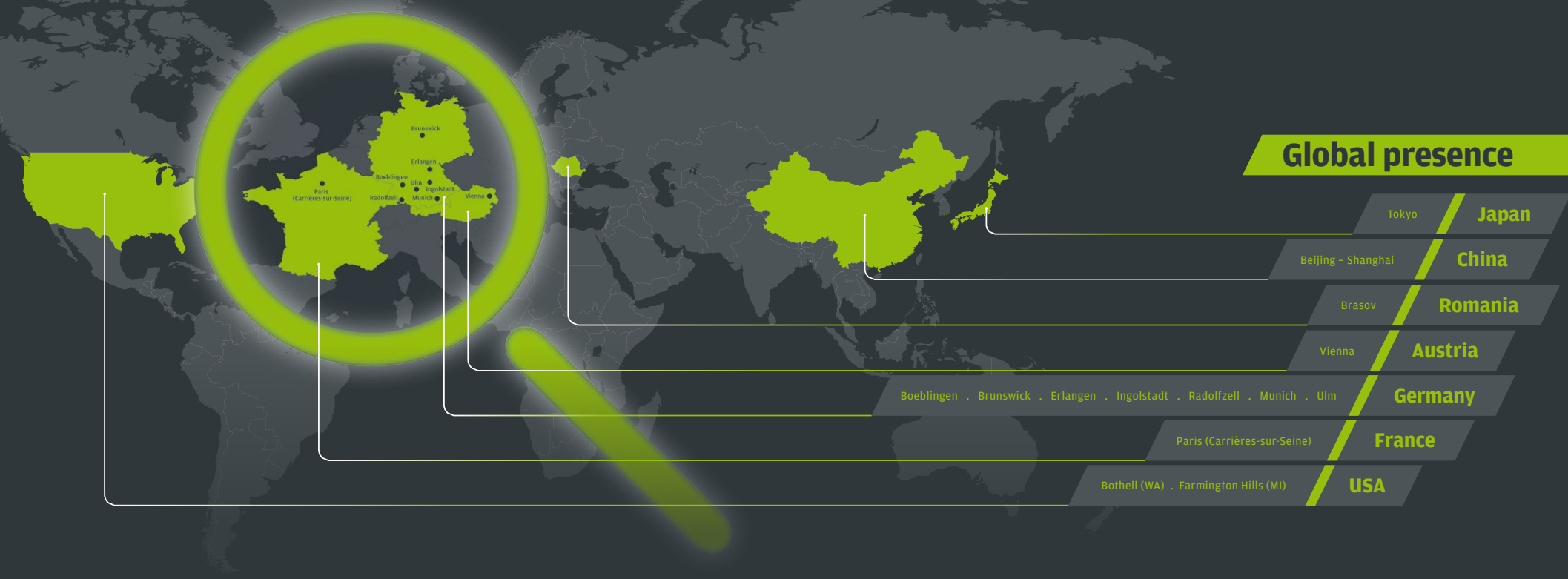
Using the Lean Development Model, the car manufacturer gets far more involved in the software development. Instead of receiving finished work packages on pre-defined dates, the customer gets daily insights into the development progress and may propose adaptations or suggest new ideas at any time. The resulting infotainment system satisfies the entertainment electronics requirements, not just the ones that were defined at the beginning of the project.

Christian Hausner

is Project Manager at Elektrobit Automotive GmbH. He is a certified Scrum Master and managed projects for various automotive manufacturers. He supports international projects in applying agile project management methods.

Martin Dusch

is Team Manager at Elektrobit Automotive GmbH. He has been a developer and project manager and worked in quality and knowledge management since 2002. As a certified Scrum Master, he has been involved in agile software development for more than five years.



Global presence

- Tokyo — Japan
- Beijing – Shanghai — China
- Brasov — Romania
- Vienna — Austria
- Boeblingen . Brunswick . Erlangen . Ingolstadt . Radolfzell . Munich . Ulm — Germany
- Paris (Carrières-sur-Seine) — France
- Bothell (WA) . Farmington Hills (MI) — USA

About EB Automotive

EB Automotive is recognized internationally as one of the most important suppliers of embedded software solutions in the automotive industry. In addition to the development of products, EB Automotive also specializes in services and consulting for the automotive industry, supplying implementations of software solutions for a broad range of AUTOSAR and FlexRay, functional safety, infotainment, navigation, HMI and driver assistance systems. EB continues to invest in feature integration and development tools ensuring in-vehicle devices ship in volume earlier and arrive quickly to market.

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
automotive.elektrobit.com