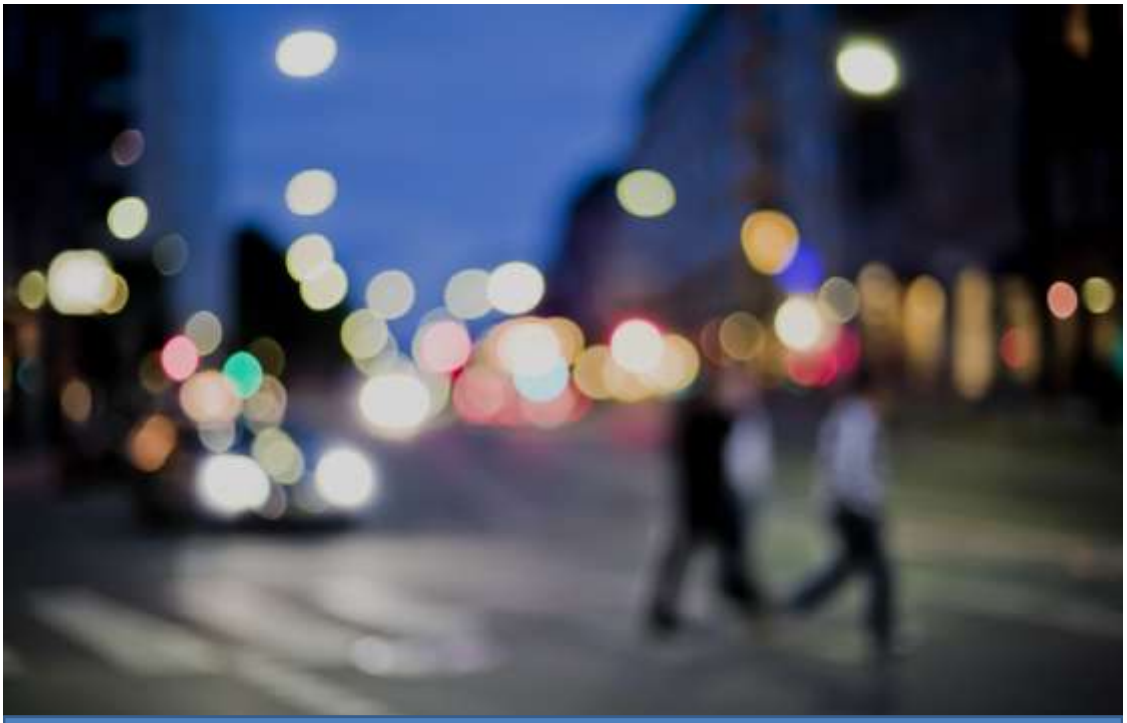
FFI

Managing Automotive Software Architectures



Project within the Vehicle Development programme

Ulrik Eklund, Volvo Car Group

2013-04-30

Content

FFI in short

FFI is a partnership between the Swedish government and automotive industry for joint funding of research, innovation and development concentrating on Climate & Environment and Safety. FFI has R&D activities worth approx. €100 million per year, of which half is governmental funding. The background to the investment is that development within road transportation and Swedish automotive industry has big impact for growth. FFI will contribute to the following main goals: Reducing the environmental impact of transport, reducing the number killed and injured in traffic and Strengthening international competitiveness. Currently there are five collaboration programs: **Vehicle Development, Transport Efficiency, Vehicle and Traffic Safety, Energy & Environment and Sustainable Production Technology.**

For more information: www.vinnova.se/ffi

1. Executive summary

The project investigated efficient and sustainable development of software for mass-produced embedded systems (MPES), and the implications a chosen way-of-working and software architecture have on research & development, and indirectly on the business.

The project defined three goals for an original equipment manufacturer aspiring to be world leading in software development, and thus contributed to the Vehicle Development programme targets in the areas of Embedded systems & software and Development methods:

1. Minimise leadtime from idea to implementation of new software features. This was supported by maximising the speed of the individual teams through their ways-of-working, and by decoupling the teams from each other through the composability of the architecture.
2. The ability to frequently deploy new software features to end customers. This was achieved both on a team level for the same reasons, supporting the leadtime reduction from idea to implementation, and also through the concept of innovation experiment systems.
3. Decoupling of software development from hardware and mechanical development, both in time and by the design dependencies. This was achieved both by moving from a centralised synchronised processes to more autonomous teams, and by providing suitable abstractions in the embedded platform underlying the application/feature software.

The deliverables contributing to these goals enabling new business models for OEMs delivering mass-produced embedded systems were in terms of new ways-of-working, new architectures, and possibly supporting new ecosystems.

The project identified five archetypical approaches of embedded software development. The common approach to develop embedded software is with a stage-gate or V process based on calendar time. The used architecture optimises qualities observable at run-time for the user/customer. The business goal is to minimise risks associated with technology investments. As an evolution from this, the project investigated how individual teams can use agile development as a way-of-working in the context of large MPES projects.

The project identified key properties of architectures for mass-produced embedded systems to create new business options, for example offering the embedded software as a service instead of a product, or developing embedded software in an open software ecosystem. A reference architecture was designed to enable this, consisting of 20 architectural decisions together with four architecture patterns.

The project defined three architecture patterns to support *innovation experiment systems* for mass-produced devices with embedded software, which together with an infrastructure capable of collecting and analysing the data. This would enable development, deployment and measurement of the usage of new software in iterations

which lengths are determined by the speed of the software development teams instead of the setup of the manufacturing process, going from years to weeks.

The results have been published in 11 peer-reviewed scientific papers and resulted in a PhD in Computer Science and Engineering for one of the project participants.

2. Background

Software is prevalent in many products manufactured today; cars, washing machines, mobile phones, airplanes and satellites. The embedded software controls the behaviour of the product and is often critical for the success of the product.

Typically these products are developed in large, and sometimes very complex, industrial projects where the manufacturing and delivery of the product in general is a heavier investment than the software budget. In the automotive domain the purchasing and manufacturing cost of the physical parts for each product is typically an order of magnitude higher than the R&D cost divided over the number of products built. This in turn tends to drive the entire R&D process, and software follows the process logic of the mechanical development and manufacturing setup.

Software differs compared to its mechanical and electronic counterparts; once the software is developed there is no additional cost if one increases the number of products manufactured. Likewise, the cost of updating or replacing software in an already manufactured product is orders of magnitude cheaper compared to replacing hardware.

The common business model of mass-produced systems is to sell the system as a product; the original equipment manufacturer (OEM) gets paid an agreed amount for each system delivered. From a customer perspective there are some issues with this business model when it comes to features purely relying on software. An example scenario in the automotive domain could be:

> *My neighbour bought a new Volvo four months after I did. He got Spotify, but I didn't. Do I feel "cheated"?*

Another example scenario would be

> *Airplay*[1] *turns out to be the next hot thing. It takes Volvo 30 months to include this feature according to the present stage gate process. Is the feature still "fresh" when delivered?*

A possible future scenario could be that Airplay is deployed to customers independently of when the car is built in the factory, i.e. after the customer has received the vehicle. With more and more vehicles becoming connected it is conceivable to have continuous updates with new software features to customers on a scale and to a cost that is impossible would the updates require hardware modifications. This builds trust in the

[1] http://www.apple.com/airplay/

brand to maintain a useful product and prolong the value for the customer after the purchase.

The team developing Airplay connectivity could deploy it after three months of development, making it “almost competitive” with smartphones. More and more services are used regardless if one is using the phone or driving in a car, and from a customer perspective it is difficult to understand why services in one context cannot be used in another. This perspective is a business driver for decoupling software development from mechanical development in embedded consumer products.

3. Objective

The topic of investigation in the project was efficient and sustainable development of software for mass-produced embedded systems (MPES), and the implications a chosen way-of-working and software architecture have on Research & Development, and indirectly on the business.

4. Project realization

The core of the project was about engineering new solutions to new or future problems; developing artefacts, processes and methods to create opportunities for business for embedded software in mass-produced systems in general, and automotive software in particular. At the same time it was a research project done in cooperation between industry and academia, with the research goals directed by industrial needs; both problem investigations and evaluations of prescriptive measures were based on scientifically collected empirical data from industrial contexts. In such a setting there are several established research methods that can be used; experiments, design science, design research, action research or participative case studies.

Therefore a research process using multiple methodologies was chosen: Design research was chosen as the overall method, scoping the complete project and putting the partial results into relation of each other. Case studies were chosen as the method in the individual investigations supporting the desired proximity between academic research and industrial needs

The studied cases provide either insight in critical issues in industrial development of software for MPES or evaluate developed artefacts in a real world context. Most of the reasoning and analysis in the cases are qualitative, based on industrial observations and experiences rather than experimentation and elaborate theories.

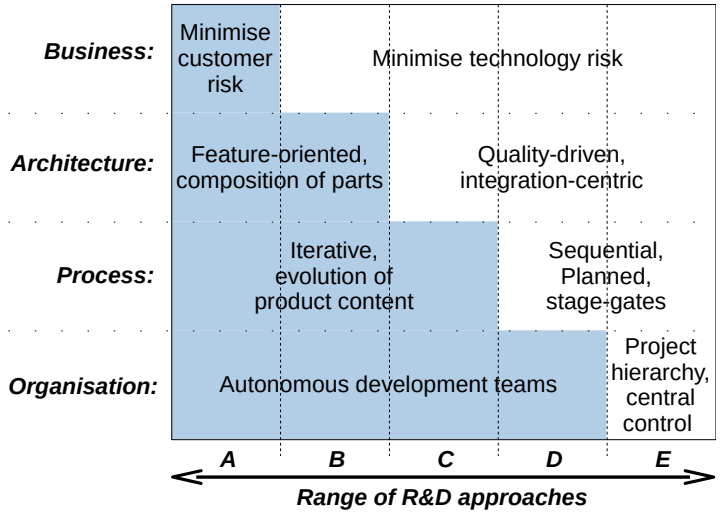
5. Results and deliverables

5.1 Delivery to FFI-goals

The project has contributed to the targets of the Vehicle Development programme targets with vehicle-related research, innovation and development activities in the areas of *Embedded systems & software* and *Development methods*.

5.1.1. Development methods

The project identified five archetypical approaches of embedded software development, based on a mapping study over published scientific literature.



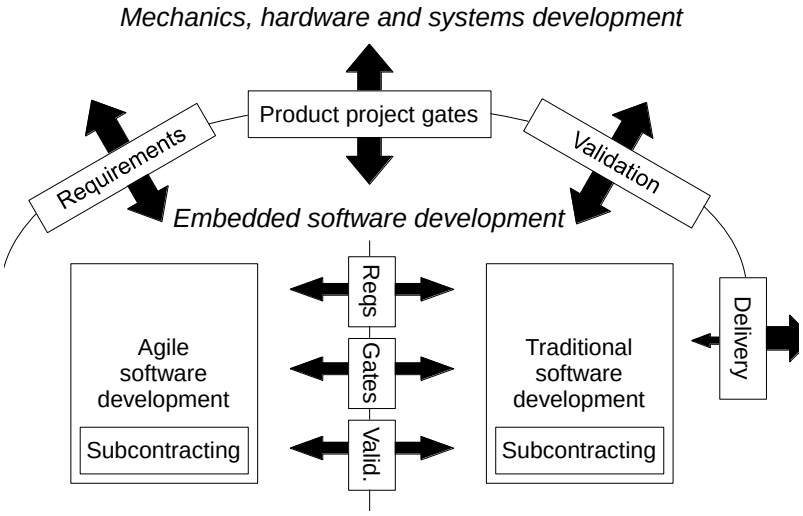
The first observation is that the rightmost (E) approach is the most common way to develop embedded software and can be considered the common practice, especially in the automotive industry. The second most common approach is using e.g. agile development on the team level (D). The second observation is that the evolution of an organisation goes from E to A, with the most common just being going from E to D. In no found case there an evolution in the opposite direction, and the conclusion is that evolution through the model is unidirectional.

These two observations together suggests an order in which organisation may adapt new practices and is therefore suggested as a prescription of process and architecture changes.

The project explored agile development in in the automotive industry, i.e. approach D from the model above. Agile is not new, but is nevertheless not the norm for automotive software development.

The project investigated how individual teams can use agile development as a way-of-working in the context of large MPES projects. The project identified a set of 19

measures, based on a model (see below) over the critical interactions a team doing agile software development has to the rest of a large project organisation.



Each measure is either a prerequisite for agile development, e.g. must be defined in the pre-game phase of Scrum, or an activity while doing the iterations, e.g. in the game phase of Scrum.

- The *requirements* measures encompasses how the software requirements implemented by the agile team relate to the requirements of the finished product, they typically contain both functional requirements experienced by the end-user, but also quality attributes, such as testability. The model includes methods and tools for capturing and transferring requirements in this category.
- The *project gate* measures focuses on the interface between the software development team and the full product project. This includes the static organisation of the project including governance and reporting, as well as basic principles for driving and measuring progress.
- *Validation* measures are concerned with the interface between agile software development and the validation of the product as a whole. This category includes activities necessary to integrate the various software and hardware parts to a whole, how this whole is verified against the requirements and the validation of the full product.
- The *delivery* category describes the principles for how the finished software is delivered to the end-user. In mass-produced embedded systems the software and the hardware is delivered as a single product and this is the only possibility if the software is stored in ROM
- The measures of *internal practices* have no direct relationship to other software development teams or the rest of the organisation, and are thus up to the agile teams. However, they are important for successful implementation of agile development in the context of mass-produced embedded systems.

Since more and more embedded products also are connected, it is conceivable to develop, deploy and measure usage on new software in iterations which lengths are determined by the speed of the software development teams instead of the setup of the manufacturing process, going from years to weeks. Such an innovation experiment system (IES) would utilise feedback from real users of the embedded products in a scale comparable to the entire customer base. The notion of continuous innovation is not new, but the concept is novel in the embedded domain.

The driver for having such an IES is that business and design decisions should be based on *data*, not opinions among developers, domain experts or managers. The company running the most experiments among the customer base against the lowest cost per experiment outcompetes the others by having the decision basis to engineer products with outstanding customer experience.

5.1.2. Embedded systems & software

The project identified key properties of architectures for mass-produced embedded systems to create business options, for example offering the embedded software as a service instead of a product, or developing embedded software in an open software ecosystem. The following list of quality attributes are necessary for a platform allowing for composition of independently developed software, the platform being a precursor for an open software ecosystem:

- **Composability:** The software platform must fulfil a set of properties to allow the decoupling of applications and eliminate the need for development synchronisation. The architecture should allow development, integration and validation of applications independent of other applications.
- **Deployability:** The applications must be possible to deploy independently of each other, and the product behaviour must not depend on the order in which applications are installed. There must also be a deployment infrastructure in place which fulfils necessary integrity requirements.
- **Maintainability:** The platform must be sufficiently stable over time. Since the evolution of the platform and applications is decoupled, i.e. no synchronised versioning, backwards compatibility is a key attribute.
- **Configurability:** The platform must support variability in the hardware configuration of sensors and actuators since individual products can vary within the product family.

In addition to these qualities, there are two more quality attributes that affect the business value for embedded products in various domains, but are not universal:

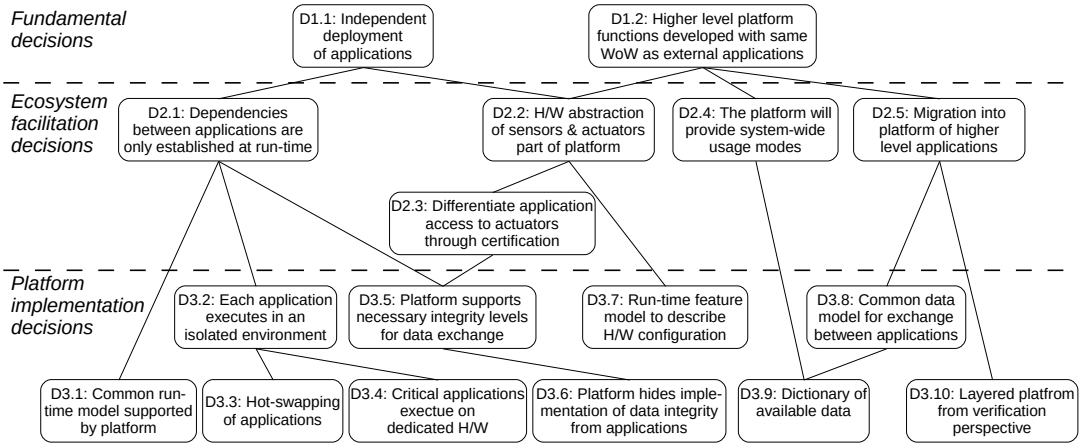
- **Consistent User Interface:** This is often considered important by manufacturers of embedded consumer products since much of brand recognition and willingness to stay with the brand lies here. The other major aspect for brand distinction is the qualities provided by the hardware (precision, reliability, etc.).
- **Dependability:** Many embedded domains have stringent dependability requirements. These domains are probably not the first adopters of an ecosystem-based approach to software development. A safety-critical embedded platform would need satisfy; real-time requirements for the execution of individual

applications, integrity requirements, high availability, and mechanisms to eliminate undesired feature interaction if several applications interact with the same actuators.

A reference architecture was designed to enable the qualities above, consisting of 20 architectural decisions (see figure below) together with four architecture patterns for:

- Device abstraction
- Data and service provision
- Device and information composition
- Safety-critical, certified and open application access

The conclusion is that it is conceivable to develop a platform suitable for compositional development, as a precursor for an open software ecosystem.



The project defined three architecture patterns to support IES for mass-produced devices with embedded software, which together with an infrastructure capable of collecting and analysing the data.

The prototype developed in cooperation with FFI-project DFEA 2020 implement most of the design decisions for an open platform. This is a novel reference architecture for composition of independently developed embedded software applications, suitable for using in an open software ecosystem. Open software ecosystems are not new, but no reference for implementation in the embedded domain was found in published literature.

The prototype also implemented one of the client architectures for IES and ran an experiment collecting data from seven users. The conclusion is that it is technically feasible to implement an IES with the defined architecture, and that the measured data can support conclusions about implemented designs.

The two reference architectures defined in the project is regarded as state-of-the art for embedded systems, and will probably no be common practice within the automotive industry within the next ten years. Experiences made from prototype design will be of strategic importance from an international perspective.

6. Dissemination and publications

6.1 Knowledge and results dissemination

Within Volvo Car Corporation the results and experiences from the project will influence the software in future car models, and changes in methodologies will propagate within the development organisation. Volvo Car Corporation has a number of projects, where the results of the project will be further developed and realized.

The results and experiences from the project already have and will continue to influence the research performed within the Software Center[2] at Lindholmen, Gothenburg.

6.2 Publications

The project has contributed to the following peer-reviewed scientific papers:

1. U. Eklund and C. M. Olsson, “A Case Study of the Architecture Business Cycle for an In-Vehicle Software Architecture,” in Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, Cambridge, UK, 2009, pp. 93–100.
2. U. Eklund and T. Arts, “A Classification of Value for Software Architecture Decisions,” in Proceedings of the European Conference on Software Architecture, Copenhagen, Denmark, 2010, vol. 6285, pp. 368–375.
3. H. Gustavsson and U. Eklund, “Architecting Automotive Product Lines: Industrial Practice,” in Proceedings of the Software Product Line Conference, Jeju, South Korea, 2010, vol. 6287, pp. 92–105.
4. R. A. McGee, U. Eklund, and M. Lundin, “Stakeholder identification and quality attribute prioritization for a global Vehicle Control System,” in Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, Copenhagen, Denmark, 2010, pp. 43–48.
5. J. Bosch and U. Eklund, “Eternal Embedded Software: Towards Innovation Experiment Systems,” in Proceedings of the International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, Heraclion, Crete, 2012, vol. 7609, pp. 19–31.
6. U. Eklund and J. Bosch, “Applying Agile Development in Mass-Produced Embedded Systems,” in Agile Processes in Software Engineering and Extreme Programming, Malmö, Sweden, 2012, vol. 111, pp. 31–46.
7. U. Eklund and J. Bosch, “Architecture for Large-Scale Innovation Experiment Systems,” in Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, Helsinki, Finland, 2012, pp. 244–248.

[2] http://www.lindholmen.se/sv/nyheter/mjukvara-framtiden-svensk-industri

8. U. Eklund and J. Bosch, “Introducing Software Ecosystems for Mass-Produced Embedded Systems,” in Proceedings of the International Conference on Software Business, Cambridge, MA, USA, 2012, pp. 248–254.
9. U. Eklund and J. Bosch, “Using Architecture for Multiple Levels of Access to an Ecosystem Platform,” in Proceedings of the ACM Sigsoft conference on Quality of Software Architectures, Bertinoro, Italy, 2012, pp. 143–148.
10. U. Eklund and H. Gustavsson, “Architecting Automotive Product Lines: Industrial Practice,” Science of Computer Programming, 2012.
11. U. Eklund, N. Jonsson, A. Eriksson, and J. Bosch, “A reference architecture template for software-intensive embedded systems,” in Proceedings of the WICSA/ECSA Companion Volume, Helsinki, Finland, 2012, pp. 104–111.

Based on the work in the above articles a PhD thesis was published at Chalmers University of Technology: “Engineering software for mass-produced embedded systems - Ways-of-working, architecture and ecosystems for innovation”, http://publications.lib.chalmers.se/publication/173642-engineering-software-for-mass-produced-embedded-systems-ways-of-working-architecture-and-ecosystems

The project has initiated the following bachelor theses:

1. “Exploring variation mechanisms in the automotive industry - A case study” by Emil Janitzek and Marcus Ljungblad, 2010 (http://hdl.handle.net/2077/23469)
2. “Value creation from an In-Vehicle Infotainment Perspective: A Case Study” by Robin Larsson and Maryam Zarrinjouei, 2011 (http://hdl.handle.net/2077/27850)
3. “Introducing the three tier model for app security and reliability in critical systems - An exploratory practical approach” by Per Lundin and Erik Kinding, 2011 (http://hdl.handle.net/2077/27849)

The project also maintained a blog during the duration of the project: http://automotive-sw-architecture.blogspot.se/

7. Conclusions and future research

The project contribution towards leadtime reduction is supported by maximising the speed of the individual teams through their way-of-working, and by decoupling the teams from each other through the composability of the architecture.

The ability to frequently deliver new software features is achieved both on a team level for the same reasons, supporting the leadtime reduction from idea to implementation, and also through the concept of innovation experiment systems.

The decoupling of software from hardware development is achieved both by moving from a centralised synchronised processes to more autonomous teams, and by providing suitable abstractions in the embedded platform underlying the application/feature software.

Future work

A successful transition to more autonomous development on a team level seem to depend on dedication and enthusiasm among developers, strong domain knowledge in the team(s), stable interfaces to other systems, and a good systems engineering foundation (in the form of a systems design or architecture), besides suitable process and architectural measures. Further studies of sufficient prerequisites should be of interest to practitioners and researchers alike.

Research on how to mitigate issues with synchronisation and integration of many teams in large projects, where scaling of agile practices is just a special case. Of special interest is if the responsibility of synchronising the teams can be moved from the process to the architecture, creating a completely composable system where successful integration is assured just by following the architecture.

More studies with industrial validation of innovation experiment systems for embedded systems are needed.

8. Participating parties and contact person

Volvo Car Group
Dr. Ulrik Eklund
Dept. 94121, PVD2:1
SE-405 31 Göteborg
Sweden
ulrik.eklund@volvocars.com

CHALMERS

Chalmers University of Technology
Prof. Jan Bosch
Dept. Computer Science and Engineering
SE-412 96 Göteborg
Sweden
jan.bosch@chalmers.com