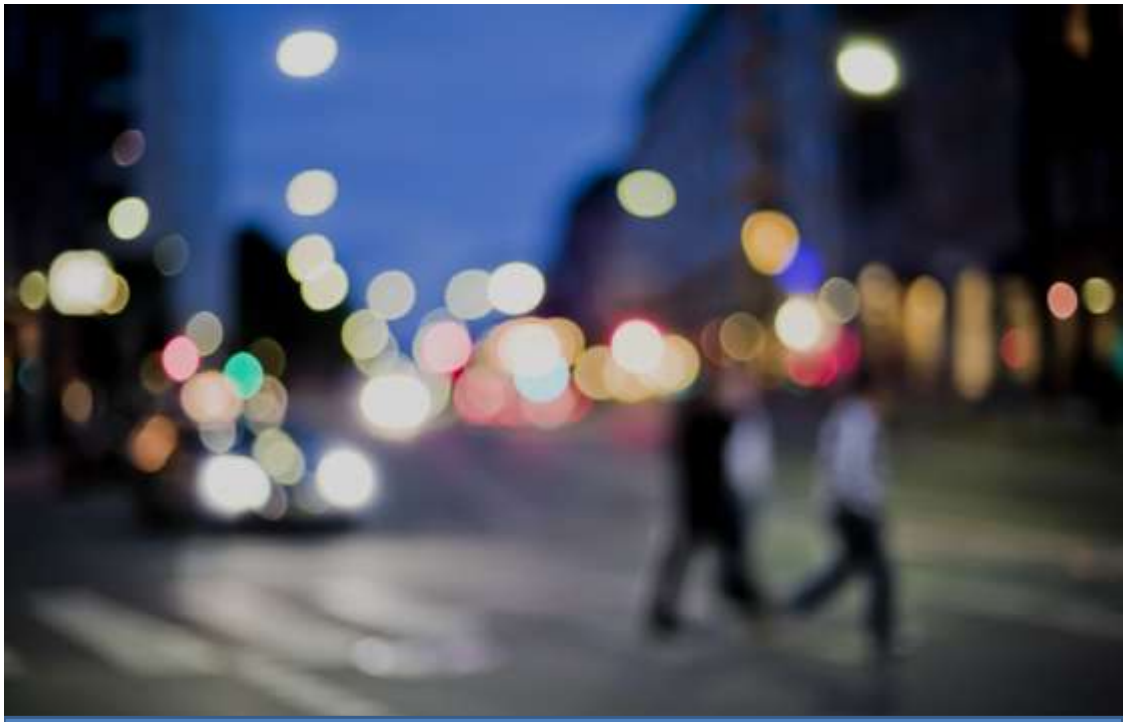
FFI

FORDONSSTRATEGISK
FORSKNING OCH INNOVATION

# Styrning av fordonsburna programvaruarkitekturer



Ulrik Eklund
2013-04-30
Delprogram: Fordonsutveckling

FFI
FORDONSSTRATEGISK
FORSKNING OCH INNOVATION

# Innehåll



**Kort om FFI**

FFI är ett samarbete mellan staten och fordonsindustrin om att gemensamt finansiera forsknings-, innovations- och utvecklingsaktiviteter med fokus på områdena Klimat & Miljö samt Säkerhet. Satsningen innebär verksamhet för ca 1 miljard kr per år varav de offentliga medlen utgör hälften.

För närvarande finns fem delprogram Energi & miljö, Fordons- och trafiksäkerhet, Fordonsutveckling, Hållbar produktionsteknik och Transporteffektivitet. Läs mer på www.vinnova.se/ffi

FFI
FORDONSSTRATEGISK FORSKNING OCH INNOVATION

# 1. Sammanfattning


Projektet har utforskat effektiv och hållbar utveckling av mjukvara för massproducerade inbyggda system, och vilka implikationer ett visst arbetssätt och en viss arkitektur har på utvecklingen av produkten och indirekt på affärsmöjligheterna.

Projektet satte upp tre mål för en tillverkare (OEM) som siktar mot att bli världsledande inom mjukvaruutveckling, vilket bidrar till Fordonsutvecklings-programmets mål inom *Utvecklingsmetoder* och *Inbyggda system & mjukvara*:

1. Minimera ledtiden från idé till implementation av nya mjukvarufunktioner. Detta stöds genom att maximera hastigheten hos enskilda team genom deras arbetssätt och genom att frikoppla teamen från varandra genom vald mjukvaruarkitektur.
2. Förmågan att regelbundet distribuera nya funktioner realiserade av mjukvara till slutkunden. The möjliggörs både på en team-nivå av samma anledning, förkortad ledtid, och genom innovationsexperimentsystem (IES).
3. Frikoppling av mjukvaruutveckling från hårdvaruutveckling, både tidsmässigt och konstruktionsmässigt. Detta möjliggörs genom att gå från en centralt synkroniserad utvecklingsprocess till självbestämmande team, och genom lämpliga gränssnitt i plattformen som används för applikations- och funktionsutveckling.

Resultaten som bidrar till dessa mål möjliggör nya affärsmöjligheter för en tillverkare av massproducerade inbyggda system i form av ny arbetssätt, ny mjukvaruarkitekturer och möjligen nya ekosystem.

Projektet identifierade fem arketypiska sätt att utveckla inbyggd mjukvara. Det vanligaste sättet är en grind- eller V-process baserad på kalendertid. Arkitekturen som används fokuserar på egenskaper som märks av slutkunden vid användning. En av affärsdrivkrafterna är att minimera risker med att investera i fel teknologi. Som en evolution från detta utforskade projektet agil utveckling kan användas av enstaka team inom stora industriprojekt.

Projektet har identifierat fem nyckelegenskaper hos mjukvaruarkitekturen för massproducerade system, till exempel för att kunna erbjuda mjukvaran som en tjänst istället för en produkt, eller för att kunna utveckla och distribuera mjukvaran i ett öppet ekosystem. En referensarkitektur togs fram som uppfyller dessa egenskaper, och som består av 20 konstruktionsbeslut tillsammans med fyra arkitekturmönster.

Projektet har tagit fram tre arkitekturmönster för *innovationsexperimentsystem* för massproducerade system med inbyggd programvara, tillsammans med nödvändig infrastrutkur för att samla in och analysera data. Detta möjliggör utveckling, distribution, och mätning av användandet av ny mjukvara i cykler som bestäms av hastigheten hos individuella mjukvaruteam snara än hur fabriksprocessen ser ut, med potentiell ledtidsreduktion från år till veckor.

Resultaten från projektet har presenterats i 11 granskade vetenskapliga papper och har resulterat i en teknologie doktor i data och informationsteknik för en av deltagarna.

# 2. Bakgrund

Mjukvara finns i en mäng produkter som tillverkas idag; bilar, tvättmaskiner, mobiltelefoner, flygplan och satelliter. Den inbyggda mjukvaran styr beteendet hos produkten och är ofta avgörande för hur väl den lyckas.

Typiskt utvecklas dessa produkter utvecklas typiskt i stora, och ibland väldigt komplexa, projekt där tillverkningen och leveransen av produkten är en större kostnad än hela budgeten för mjukvaruutveckling. I bilindustrin är inköps- och tillverkningskostnaden en tiopotens större än utvecklingskostnaden fördelad på antalet tillverkade produkter. Detta driver i sin tur hela utvecklingsprocessen och mjukvaran följer helt enkelt upplägget för tillverkning i fabrik.

Mjukvara skiljer sin från mekanik och elektronik, när den väl är konstruerad så tillkommer ingen kostnad om det behöver tillverkas fler produkter. På samma sätt är kostnaden för att uppgradera eller ersätta mjukvara i en redan tillverkad produkt en storleksordning billigare än att ersätta hårdvaran.

Den vanligaste affärsmodellen för massproducerade system är att sälja de som en produkt; tillverkaren får betalt en överenskommen summa för varje system som levereras. Från ett kundperspektiv kan detta ifrågasättas när det gäller funktioner som enbart bygger på mjukvara. Ett exempel från bilidnustrin skulle kunna vara:

> *Min granne köpte en ny Volvo några månader efter mig. Han fick Spotify i sin bil, men det fick inte jag. Känner jag mig ”lurad”?*

Ett annat exempel skulle kunna vara

> *Airplay*[1] *är nästa funktion som alla måste ha. Det tar Volvo 30 månader att inkludera en sådan funktion enligt nuvarande grindprocess. Är funktionen fortfarande konkurrenskraft när den kan släppas till kund?*

Ett möjligt framtida scenario skulle kunna vara att Airplay släpps till kund oberoende av när bilen byggts i fabrik, dvs. efter att bilen har levererats till kund. När fler och fler bilar blir uppkopplade är det möjligt att släppa nya mjukvaruutgåvor till kund hela tiden, tull en kostnad som är omöjlig att komma ner i för uppdateringar som kräver hårdvaruförändringar. Detta skulle bygga förtroende för varumärket att produkten förblir användbar och behåller sitt värde även efter dit ursprungliga köpet.

Teamet som utvecklar Airplay skulle kunna släppa det redan efter tre månaders utveckling, nästan jämförbart med dagens mobiltelefoner. Fler och fler tjänster används oavsett om man använder en telefon eller kör bil, och från ett kundperspektiv är det svårt att förstå varför tjänster som är tillgängliga i ett sammanhang inte kan användas I ett

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

annat. Detta synssätt är en drivkraft för att frikoppla mjukvaruutveckling forn mekanisk produktion I konsumentprodukter med inbygg mjukvara.

# 3. Syfte

Projektet har undersökt effektiv och uthållig utveckling av mjukvara I massproducerade system, och de följder ett valt arbetssätt och en vald arkitektur har på produktutveckling och indirekte på affärsmöjligheterna.

# 4. Genomförande

Huvuddelen av projektet var att konstruera nya lösningar på förutsedda problem; utveckla artefakter, processer och metoder för att möjliggöra nya affärsmöjligheter för inbyggd mjukvara i massproducerade system i allmänhet, och speciellt för bilburen mjukvara. Samtidigt var det ett ett forksningsprojekt som utfördes i samrabete mellan industrin och akademien med forskningsmålen styrda av industriella behov; både vilka problem oms utforskades och utvärderingen av lösningarna var baserade på veteskapligt ihopsamlade data från industrin. I ett sådant sammahang kan man använda sig av flera olika veteskapliga metoder; experiment, design science, design research, action research eller fallstudier.

För att adressera detta valdes en kombination av forksningsmetoder; Design research som den övergripande metoden för att sätta samman de olika delarna till en helhet, och deltagande fallstudier som metod i de olika undersökningarna för att stödja den efterfrågade närheten mellan akademisk forskning och industriella behov.

Fallstudierna gav insikt i kritiska frågeställningar för industriell utveckling av mjukvara eller utvärderade projektresultaten i en verklig miljö. De flesta resonemangen i projekten är kvalitativa, baserade på observationer och erfarenheter i industrin, snarare än kvantitativa experiment.

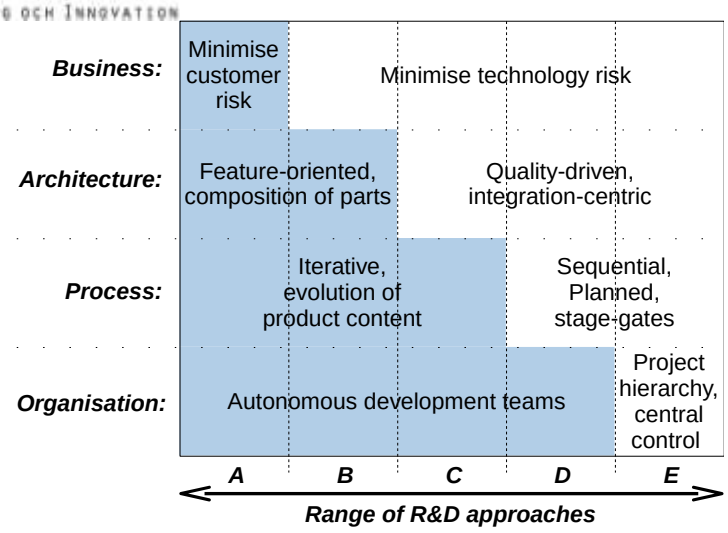
# 5. Resultat

## 5.1 Bidrag till FFI-mål

Projektet har bidragit till Fordonsutvecklingsprogrammets mål för forskning, innovaiotn och utveckling inom områdena Utvecklingsmetoder och Inbyggda system & mjukvara.

### 5.1.1. Utvecklingsmetoder

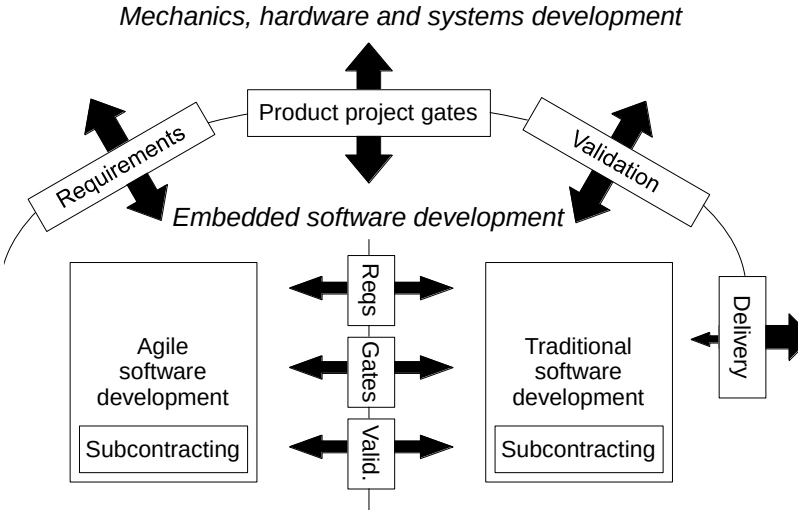
Projektet identifierade fem arketypiska sätt att utveckla inbyggd mjukvara (se figuren nedan), baserat på en litteraturstudie.




Den första observationen är att det högraste sättet (E) är det vanligaste sättet och kan anses som industristandard, speciellt i bilindustrin. Det näst vanligaste sättet är agil utveckling på team-nivå (D). Den andra observationen är att en organisation går från E till A. Inte något fall hittades där utvecklingen gick i motsatt riktning och slutsatsen är att evolutionen gällande utvecklingssätt är enkelriktad.

Dessa två observationer ger i vilken ordning en organisation bör anamma nya utvecklingssätt och föreskriver alltså i vilken ordning process- och arkitekturförändringar börgöras.

Projektet utforskade agil mjukvaruutveckling i bilindustrin, (angrepssätt D ovan). Agil utveckling är inte nytt, men är absolut inte det normala sättet att utveckla mjukvara i bil. Projektet tittade på hur enstaka team kan tillämpa agil utveckling i stora projekt för massproducerade system. Projektet identifierad 19 åtgärder baserade på enmodell (se endan) över nödvändiga interaktioner mellan agila team och resten av en stor projektorganisation.



Varje åtgärd är antingen en förutsättning för agil utveckling, dvs. måste definieras i pre-game-fasen av Scrum, eller är en aktivitet som måste göras iterativt under utvecklingen, dvs. i game-fasen av Scrum.

- Kravhanteringåtgärderna omfattar hur mjukvarukrav som implementeras av de agila teamen förhåller sig till produktkraven, typiskt omfattar de både funktionella och icke-funktionella krav som de upplevs av slutanvändaren.
- Projektgrindsåtgärderna fokuserar på gränssnittet mellan mjukvaruteamen och projektet som helhet. Det inkluderar den statiska organisationen, hur projektet styrs och rapporteras och vilka principer som gäller för styrning och uppföljning.
- *Valideringsåtgärderna* visar hur gränssnittet agil utveckling och valideringen produkten som helhet. Denna kategori inkluderar aktiviteter nödvändiga för att integrera mjukvarudelar med hårdvara till en helhet och hur denna helhet verifieras gentemot produktkraven.
- Leveranaskategorin beskriver principerna för hur den slutliga mjukvaran släpps och levereras. I massproducerade system levereras vanligtvis mjuk- och hårdvaran som en gemensam produkt och detta är den enda möjligheten om mjukvaran till exempel lagras i ROM.
- Interna åtgärder har ingen direkt koppling till hur andra utvecklingsteam arbetar, eller till resten av organisationen. Men de är ändå viktiga för framgångsrik agil utveckling av mass-producerade inbyggda system

Eftersom alltfler inbyggda system också blir uppkopplade är det möjligt att utveckla, distribuera och mäta användningen av ny mjukvara i iterationer, där iterationstiden bestäms av hastigheten hos individuella team snarare än av hur tillverkningsprocessen för den fysiska produkten ser ut, från år till veckor. Ett sådant innovationsexperimentsystem (IES) skulle använda sig av återkoppling från verkliga användare i en skala jämförbara med hela kundbasen. Iden med kontinuerlig innovation är inte ny, men den är hittills inte använd för inbyggda system.

Drivkraften för att använda IES ‘r att affärs- och konstruktionsbeslut bör baseras på data, inte tyckanden hos utvecklare, domänexperter eller chefer. Företaget som kan köra flest experiment till lägst kostnad per experiment kommer att konkurera ut andra genom at ha dataunderlaget för att gör konstruktioner med enastående kundupplevelse.

### 5.1.2. Inbyggda system & mjukvara

Projektet identifierade ett antal nyckelegenskaper för mjukvaruarkitekturen i massproducerade inbyggda system, som skulle tillåta alternative affärsmöjligheter. Till exempel skulle mjukvaran kunna erbjudas som en tjänst istället för som en produkt, eller det skulle vara möjligt att ta fram mjukvaran i ett öpet ekosystem. Följande egenskaper är nödvändiga för en plattform som möjliggör sammansättning av oberoende framtagna mjukvaror, med plattformen som en förutsättning för ett introducera ett öppet ekosystem för mjukvara:

- **Sammansätning:** Mjukvaruplattformen måste möjliggöra frikoppling av applikationer från varandra och eliminera behovet att olika utvecklare måste

synkroniseras. Arkitekturen skall tillåta utveckling, integration och validering av applikationer oberoende av varandra.

- **Distribuerbarhet:** Mjukvaruapplikationer måste vara möjliga att distribuera oberoende av varandra, och produktens beteende måste inte bero på vi vilken ordning applikationerna installeras. Det måste också finna en infrastruktur på plats som uppfyller nödvändiga säkerhets- och integritetskrav.
- **Unerhållsbarhet:** Plattformen måste vara stabil over tiden. Eftersom evolutionen av plattfromen och applikationerna är frikopplade från varandra, dvs. inga synkroniserade versioner, så är bakåtkompatabilitet ett måste.
- 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.
- **Konfigurerbarhet:** Plattformen måste stödja variationer i hårdvaran i form an sensorer och aktuatorer eftersom individuella produkter kan skilja sig från varandra.

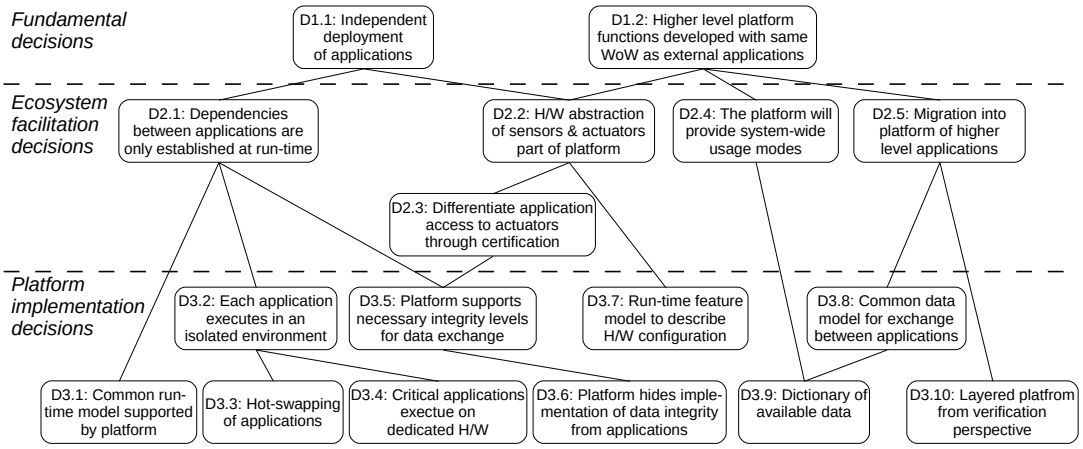
Utöver dessa egenskaper så finns det ytterligare två kvalitetsattribut som påverkar affärsvärdet för produkter med inbyggd programvara, men kvaliteterna är inte nödvändiga i alla affärsdomäner:

- **Konsekvent användargränssnitt:** Detta anses ofta viktigt av tillverkare av konsumentprodukter eftersom en stor del av märkesigenkänning och -lojalitet bygger på detta. Den andra viktiga aspekten är kvalitéerna som ges av hårdvaran (precision, tillförlitlighet etc.).
- **Pålitlighet:** Många områden med inbyggda system har stringenta pålitlighetskrav. Dessa områden är troligen inte de som först anammar ett ekosystem-baserat sätt att utveckla mjukvara. En säkerhetskritisk plattfrom måste tillgodose realtidskrav för exekvering av olika applikationer, integritetskrav, hög tillgänglighet, och mekanismer för att eliminera oönskad funktionsinteraktion om många applikationer vill använda sig av samma aktuator.

En referensarkitektur togs fram för att uppfylla egenskaperna ovan, bestående av 20 arkitekturella konstruktionsbeslut (se bild nedan) tillsammans med fyra arkitekturmönster för:

- Hårvaruabstraktioner
- Tillhandahållande av data och tjänster
- Sammansättning av information och hårdvaruabstraktioner
- Access för säkerhetskritiska certifierade och öppna applikationer

Slutsatsen är att det är fullt möjligt att ta fram en plattform för sammansättning av oberoende framtagna applikationer, som en förutsättning för ett öppen ekosystem för mjukvara.




Projektet har definierat tre varianter på en referensarkitektur för att stödja IES för massproducerade system med inbyggd mjukvara.

Prototypen som togs fram i samarbete med FFI-projektet DFEA 2020 implementerade de flesta av konstruktionsbesluten för en öppen plattfrom. Den implementerade också en av arkitekturerna för IES och körde ett experiment med A/V-testning baserat på sju användare. Slutsatsen är att det tekniskt möjligt att realisera IES och att uppmätta data kan stödja slutsatser om implementerad mjukvara.

De två referensarkitekturerna får anses som ligga i absoluta framkanten för inbyggda system och kommer troligen inte att vara norm i bilindustrin det nästkommande decenniet. Erfarenheterna från konstruktion och prototypimplementation antas vara av strategisk betydelse sett från ett internationellt perspektiv.

# 6. Spridning och publicering

## 6.1 Kunskaps- och resultatspridning

Inom Volvo Personvagnar kommer projektet att påverka mjukvaran i framtida bilprojekt, och förändringar i arbetssätt håller långsamt på att sprida sig inom produktutvecklings-organisationen. Volvo Personvagnar har ett flertal projekt där resultaten kommer att utvecklas vidare.

Resultaten och erfarenheterna från projektet har redan påverkat forskningen inom Software Center på Lindholmen i Göteborg.

## 6.2 Publikationer

Projektet har bidragit till följande granskade vetenskapliga artiklar:

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.
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.

Baserat på ovanstående publikationer har en doktorsavhandling försvarats vid Chalmers tekniska högskola med titeln: “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

Projetet har initierat följande examensarbeten:

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)

Projektet har också drivit en blogg under hela genomförandet: http://automotive-sw-architecture.blogspot.se/

# 7. Slutsatser och fortsatt forskning

Projektets bidrag gällande ledtidsförkortning har stötts av att möjliggöra maximal hastighet för individuella team i deras arbetssätt, och genom att frikoppla team från varandra genom arkitekturen.

Förmågan att frekvent leverera ny mjukvara åstadkoms på tema-nivå av samma skäl, stödjande av ledtidsreduktion från idé till införande, och också genom utveckling i innovationsexperimentsystem

Frikopplingen av mjukvaru- från hårdvaruutveckling åstadkoms både genom att gå från en centralt styrd process där alla team måste synkroniseras till självstyrande team, och genom lämpliga hårdvaruabstraktioner i plattformen som ligger som bas för applikations- och funktionsutveckling

### Fortsatt forskning

En framgångsrikt transformation till mer självstyre på en teamnivå verkar bygga på hängivna och entusiastiska utvecklare, en god domänkunskap, stabila gränssnitt tillandra grupperingar och en bra systemgrund i form av en arkitektur. Detta förutom lämpliga åtgärder gällande arbetssätt. Framtida studier av detta är av intresse både för industri och forskare.

Framtid forskning om hur man minimera problem med synkronisering och integration av manga team i stora projekt är nödvändigt, där agil utveckling bara är ett specialfall. Av särskilt intresse är om ansvaret för synkronisering kan flyttas från process till arkitekturen, vilket skulle skapa ett fullständigt frikopplat system där lycka integration till en komplett skulle vara säkerställd bara genom att följa arkitetkuren.

Fler studier där innovationsexperimentsystem som utvecklingssätt för inbyggd mjukvara valideras industriellt behövs.

FFI
FORDONSSTRATEGISK
FORSKNING OCH INNOVATION

# 8. Deltagande parter och kontaktpersoner


Volvo Personvagnar
Ulrik Eklund
Avd. 94121, PVD2:1
405 31 Göteborg
ulrik.eklund@volvocars.com

Chalmers tekniska högskola
Prof. Jan Bosch
INst. För data- och informationsteknik
412 96 Göteborg
jan.bosch@chalmers.com